Is Decentralized LLM Agent RL Robust to Heterogeneity? An Asymmetric Tale

Canyu Chen[1*] **Kangyu Zhu**[3*] **Zhaorun Chen**[4] **Zhanhui Zhou**[2] **Shizhe Diao**[5]
Yiping Lu[1] **Tian Li**[4] **Manling Li**[1†] **Dawn Song**[2†]
[1]Northwestern University [2]University of California, Berkeley [3]Brown University
[4]University of Chicago [5]NVIDIA Research
fed-agent.github.io github.com/sunblaze-ucb/FedAgent

Abstract

Training AI agents powered by Large Language Models (LLMs) requires large volumes of user data. Privacy concerns limit *centralized* collection, and an individual user may have insufficient data. We explore **FEDAGENT (Federated Agent Reinforcement Learning)**, a *decentralized* reinforcement learning paradigm that trains LLM agents collaboratively across distributed clients without sharing local data. We ask whether FEDAGENT is effective under a *uniform* client distribution and, more importantly, whether it is robust to client *heterogeneity*. For the first question, we provide the first empirical evidence that FEDAGENT matches Centralized Agent Training and outperforms Local Agent Training. For the second, we formalize Agent Heterogeneity at two structurally distinct levels: task-level differences in what clients ask the agent to do, and environment-level differences in the dynamics in which the agent acts. This distinction is grounded in the **Input-Dynamics Asymmetry** of task-augmented Markov Decision Processes (MDPs): tasks enter the policy as input, while environments do not. We then theoretically establish an **Asymmetric Robustness Mechanism**: FEDAGENT is robust to task-level heterogeneity but non-robust to environment-level heterogeneity. We identify three sufficient conditions under which FEDAGENT recovers robustness despite environment-level heterogeneity and describe four possible training-curve patterns. Experiments on WebShop and ALFWorld verify that FEDAGENT remains robust under extreme task-level heterogeneity, while under environment-level heterogeneity its outcomes range from stability to degradation and collapse.

1 Introduction

LLM-based AI agents have shown remarkable capabilities in web navigation, embodied tasks, and tool use [34, 77, 175]. Training these agents requires large volumes of user task queries and trajectories. Privacy concerns limit centralized collection of such data, and an individual user may have insufficient data to train a capable agent. We study **FEDAGENT** (**Federated Agent Reinforcement Learning**, Figure 1), a decentralized paradigm for LLM agent RL. Each client optimizes its policy locally on its own data. The server averages the client models into a global model and broadcasts it to the clients for the next round. Only model parameters are exchanged, so local data is never shared.

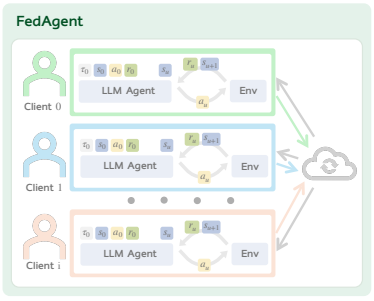


Figure 1: **FEDAGENT Framework.**

*Equal contribution. †Equal advising.

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

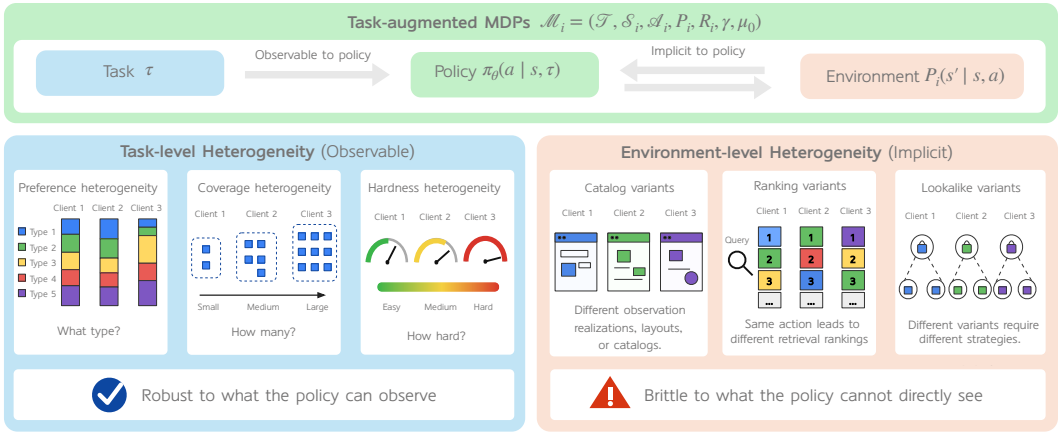


Figure 2: **Asymmetric Robustness Mechanism of FEDAGENT under Agent Heterogeneity. Top:** the task τ is *observable* to the policy through its input channel, while the transition kernel P_i is *implicit*, visible only through successor states. This contrast is the Input-Dynamics Asymmetry (§3.1). **Left:** task-level heterogeneity (preference/coverage/hardness sub-types, §3.2) lies on the observable side and is provably benign (Theorems 1/2). **Right:** environment-level heterogeneity (*e.g.*, catalog, ranking, and lookalike variants, §3.3) perturbs the implicit dynamics, and FEDAGENT is worst-case non-robust to it (Theorems 3/4) unless one of the recovery conditions (C1)–(C3) holds (§4.3).

Two questions about FEDAGENT are *structurally new* relative to two conventional settings: supervised federated learning (FL) on static datasets and federated reinforcement learning (FRL) on simple MDPs with low-dimensional states and actions.

(Q1) Is FEDAGENT *effective* under a uniform (i.i.d.) client distribution, i.e., does it match centralized agent training? Supervised FL has long established that FedAvg matches centralized SGD on input–label pairs [120], and FRL [54, 145] extends this result to simple MDPs. LLM agent RL involves *natural-language state and action spaces, diverse tasks, and complex multi-step environment interactions*, so whether federated aggregation preserves training quality here remains under-explored.

(Q2) Is FEDAGENT *robust* when clients are *heterogeneous*? Conventional FL and FRL treat statistical client heterogeneity as a *data-distribution skew*: differences in labels, features, or data quantities [51, 64, 168] or in the transitions or rewards of simple MDPs [50, 125]. In LLM agent RL, heterogeneity has an intrinsic *two-level MDP structure* that neither paradigm models jointly: clients differ in *what task* they ask the agent to perform and *which environment* the agent acts in, and these two levels have structurally different effects on an instruction-following LLM policy. For example, two web-shopping clients can differ in their requests (task level) or in their product catalogs (environment level).

We formalize Agent Heterogeneity at these two structurally distinct levels (Figure 2): *task-level* over per-client task distributions, with three sub-types (preference, coverage, and hardness), and *environment-level* over per-client transition kernels. Both levels are grounded in a single architectural fact about LLM agents that we call the **Input-Dynamics Asymmetry** of task-augmented MDPs: the task descriptor enters the policy through its *input channel* (the prompt), while the transition kernel does not. The kernel manifests only through successor states the agent observes after acting. Conventional FL and FRL analyses do not exploit this asymmetry.

The Input-Dynamics Asymmetry has a direct theoretical consequence that we call the **Asymmetric Robustness Mechanism**. FEDAGENT is *provably robust* to task-level heterogeneity (Theorems 1/2): any per-client gap vanishes as LLM capacity and optimization improve. By contrast, FEDAGENT is *worst-case non-robust* to environment-level heterogeneity (Theorems 3/4): an irreducible $\Omega(R_{\max} H \delta)$ worst-client gap remains, and additional capacity, samples, or training cannot remove it. Here $R_{\max}$ is the reward scale, H the episode horizon, and δ the degree of environment divergence (§2–3.3). Any of three sufficient conditions (C1)–(C3) recovers robustness: an optimal policy of the highest-value client avoids the region where the environments differ, the environments share optimal actions, or the agent can infer its environment from its history. The mechanism predicts four training-curve patterns, from robust (A, B) through degrade-but-stable (C) to collapse (D) (§4.4). Unlike conventional FL/FRL heterogeneity theory, which mostly analyzes convergence rates, our analysis concerns the quality of the resulting policy. Pre-training makes the mechanism

Algorithm 1 FEDAGENT with Client training and Server aggregation.

Require: Total clients N, rounds T, clients-per-round M, local epochs E, learning rate η_{lr}
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 $S_t \subset [N]$ with $|S_t| = M$ (uniform without replacement)
  Server: broadcast θ_t to all $i \in S_t$
  for each $i \in S_t$ **in parallel do**
    Set local iterate $\theta_{i,t,0} \leftarrow \theta_t$
    for $e = 0$ **to** $E - 1$ **do**
      Collect a minibatch of trajectories $B_{i,t,e}$ using policy $\pi_{\theta_{i,t,e}}$ in $\mathcal{M}_i$
      Estimate policy gradient $g_{i,t,e}$ from $B_{i,t,e}$ (*e.g.*, GRPO or PPO)
      Local update: $\theta_{i,t,e+1} \leftarrow \theta_{i,t,e} + \eta_{\text{lr}}\, g_{i,t,e}$
    end for
    Client returns local model $\theta_{i,t,E}$ ▷ equivalently $\Delta\theta_{i,t} = \theta_{i,t,E} - \theta_t$
  end for
  Server: model averaging $\theta_{t+1} \leftarrow \frac{1}{M} \sum_{i \in S_t} \theta_{i,t,E}$ ▷ PPO also averages critic ϕ
end for
return $\theta_{\text{final}} \leftarrow \theta_T$

more pronounced for LLM agents (Appendix P). We validate the theory on WebShop [165] and ALFWorld [114] with Qwen and Llama agents (1.5B–7B parameters) under GRPO and PPO. Our main contributions are summarized as follows:

- We make an early attempt to explore decentralized LLM agent RL and formalize Agent Heterogeneity at two levels grounded in the Input-Dynamics Asymmetry: *task-level* (preference, coverage, hardness) and *environment-level*, with partition strategies that isolate each sub-type.
- We theoretically establish the Asymmetric Robustness Mechanism: FEDAGENT is robust to task-level heterogeneity (Theorems 1 and 2), worst-case non-robust to environment-level heterogeneity (Theorems 3 and 4), and recovers robustness under three sufficient conditions (C1)–(C3), yielding four training-curve patterns (A–D).
- We empirically verify the effectiveness and mechanism: FEDAGENT matches centralized training and remains robust under extreme task-level heterogeneities, while its outcomes under environment-level heterogeneities span a stable-degrade-collapse spectrum.

2 FEDAGENT: Federated Agent Reinforcement Learning

Each client i has its own task-augmented MDP $\mathcal{M}_i = (\mathcal{T}, \mathcal{S}_i, \mathcal{A}_i, P_i, R_i, \gamma, \mu_0)$. The task $\tau \in \mathcal{T}$ conditions the initial state, $s_0 \sim \mu_0(\cdot \mid \tau)$, and the reward $R_i(s, a, s'; \tau)$, while the transition kernel $P_i(\cdot \mid s, a)$ describes the environment and does not depend on τ. The client holds a finite local task pool $X_i \subset \mathcal{T}$ and samples each rollout's task uniformly from it during training. We write $\mathcal{D}_{\tau_i}$ for this per-client task distribution, so both the composition and the size of the pool enter through $\mathcal{D}_{\tau_i}$. Throughout our constructions, the action space and reward function are shared, $\mathcal{A}_i = \mathcal{A}$ and $R_i = R$. We retain the subscripts for generality.

The client acts under an LLM-induced policy $\pi_\theta(a \mid s, \tau)$, whose action a may span hundreds of tokens, including intermediate reasoning and an environment-facing tool call. Per-step rewards $r_u = R_i(s_u, a_u, s_{u+1}; \tau)$ are non-negative and bounded, $r_u \in [0, R_{\max}]$. We use the episodic-undiscounted setting ($\gamma = 1$), where H is the finite episode horizon: the maximum number of agent steps per episode. Steps are indexed by u, and communication rounds below by t. The discounted analogue uses $H = 1/(1-\gamma)$ (Appendix N.3). Rolling out a trajectory $h_H = (s_0, a_0, r_0, \ldots, s_H)$ in $\mathcal{M}_i$ on task τ yields the per-task expected return $\mathcal{J}(\pi_\theta; \tau, \mathcal{M}_i) := \mathbb{E}_{h_H \sim (\pi_\theta, \mathcal{M}_i, \tau)}[\sum_u \gamma^u r_u]$ and the per-client expected return $\mathcal{J}_i(\theta) = \mathbb{E}_{\tau \sim \mathcal{D}_{\tau_i}}[\mathcal{J}(\pi_\theta; \tau, \mathcal{M}_i)]$.

Algorithm 1 states the protocol. A population of N clients runs T communication rounds. In round t, the server samples $S_t \subset [N]$ with $|S_t| = M$ uniformly without replacement and broadcasts θ_t. Every client $i \in S_t$ runs E local epochs of policy optimization on its own rollouts and returns $\theta_{i,t,E}$. The

server averages the returned models, $\theta_{t+1} = \frac{1}{M}\sum_{i\in S_t}\theta_{i,t,E}$, to optimize the federated objective $\mathcal{J}_{\text{fed}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{J}_i(\theta)$. FEDAGENT is algorithm-agnostic, and we instantiate GRPO [107] and PPO [106]. Under PPO, the critic ϕ is averaged alongside θ. GRPO has no critic. Appendix I describes the algorithm line by line and discusses privacy and communication costs.

3 Two-Level Agent Heterogeneity

3.1 Input-Dynamics Asymmetry

We define the *federated mixture task distribution* as $\bar{\mathcal{D}}_\tau := \frac{1}{N}\sum_i \mathcal{D}_{\tau_i}$. Federated training effectively samples its tasks from this mixture.

Input-Dynamics Asymmetry. Inspecting how each MDP component enters the policy reveals an architectural fact that underlies the rest of our analysis:

> The task descriptor τ enters the policy through its input channel as part of the prompt. The transition kernel P does not: the policy never directly observes P and only observes its effects through the successor states $s_{u+1} \sim P(\cdot \mid s_u, a_u)$ that arise after acting.

We call this Input-Dynamics Asymmetry (I-D Asymmetry hereafter). Because τ is observable as input, a single θ can encode a task-conditional decision rule $(s, \tau) \mapsto a$. Instruction-tuned LLMs already condition their behavior on the task stated in the prompt. Because the policy does not observe P directly, a single θ cannot encode an environment-conditional rule unless environment identity is revealed through the policy's input or observation history. When P_1 and P_2 imply different optimal actions at the same (s, τ), θ must choose one behavior, which is wrong in at least one environment. For example, if two shops' search engines rank results differently, the same query returns the target at the top of the results for one client and near the bottom for the other, so clicking the top result is optimal only for the first client.

Decomposition along I-D Asymmetry. I-D Asymmetry suggests a natural two-level decomposition of Agent Heterogeneity: task-level (§3.2) when $\mathcal{D}_{\tau_i}$ varies across clients while the environment tuple is shared, and environment-level (§3.3) when P_i varies across clients over a shared $(\mathcal{S}, \mathcal{A})$. In the latter case, P_i induces the client-specific reachable subsets $\mathcal{S}_i \subseteq \mathcal{S}$.

3.2 Task-Level Heterogeneity

Definition 1 (Task-Level Heterogeneity). Clients share a single environment tuple $\mathcal{M}_{\text{env}} := (\mathcal{S}, \mathcal{A}, P, R, \gamma, \mu_0)$, including the task-conditional reward and initial state, but differ in their per-client task distributions: $\mathcal{D}_{\tau_i} \neq \mathcal{D}_{\tau_j}$ for some $i \neq j$. By I-D Asymmetry, task-level heterogeneity is *observable* to the policy through its input channel. Examples include cross-domain user cohorts (a coding agent fixing different bugs across users, a writing assistant for novelists vs. technical writers).

Three operationally separable sub-types. We split task-level heterogeneity along three properties of $\mathcal{D}_{\tau_i}$, each of which affects federated agent RL through a distinct mechanism (Appendix F.1):

- **Preference Heterogeneity** (*What type?*). Clients differ in *which types of tasks* they mostly encounter (*e.g.*, one ALFWorld client's tasks may be mostly kitchen tasks while another's are mostly bedroom tasks). This skew shifts *which task-conditional behaviors* receive training signal under the federated mixture. We measure each client's preference by the type marginal $p_i = \ell_* \mathcal{D}_{\tau_i} \in \Delta^{C-1}$ under a category map $\ell : \mathcal{T} \to [C]$, and inter-client dispersion by $\Delta^2_{\text{pref}} := \frac{1}{N}\sum_i \|p_i - \bar{p}\|_2^2$.
- **Coverage Heterogeneity** (*How many?*). Clients differ in *how many tasks* their local pools contain. Because each client samples rollout tasks from its pool with replacement [31], the pool size limits the client's *overall exploration breadth*, and a small pool makes the client revisit the same tasks. We measure coverage by the pool size $n_i = |X_i|$ (the support size of $\mathcal{D}_{\tau_i}$) and inter-client dispersion by $\Delta^2_{\text{cov}} := \widehat{\text{CV}}^2(\{n_i\}) = \frac{1}{N}\sum_i \big((n_i - \bar{n})/\bar{n}\big)^2$.
- **Hardness Heterogeneity** (*How hard?*). Clients differ in *how difficult their tasks are for the model*. Difficulty affects the *policy-gradient advantage signal*: tasks the policy almost never solves yield few positive samples, while tasks it almost always solves leave little room for improvement [171]. We measure difficulty relative to the model by the success rate $\rho(\tau)$ of a frozen reference checkpoint

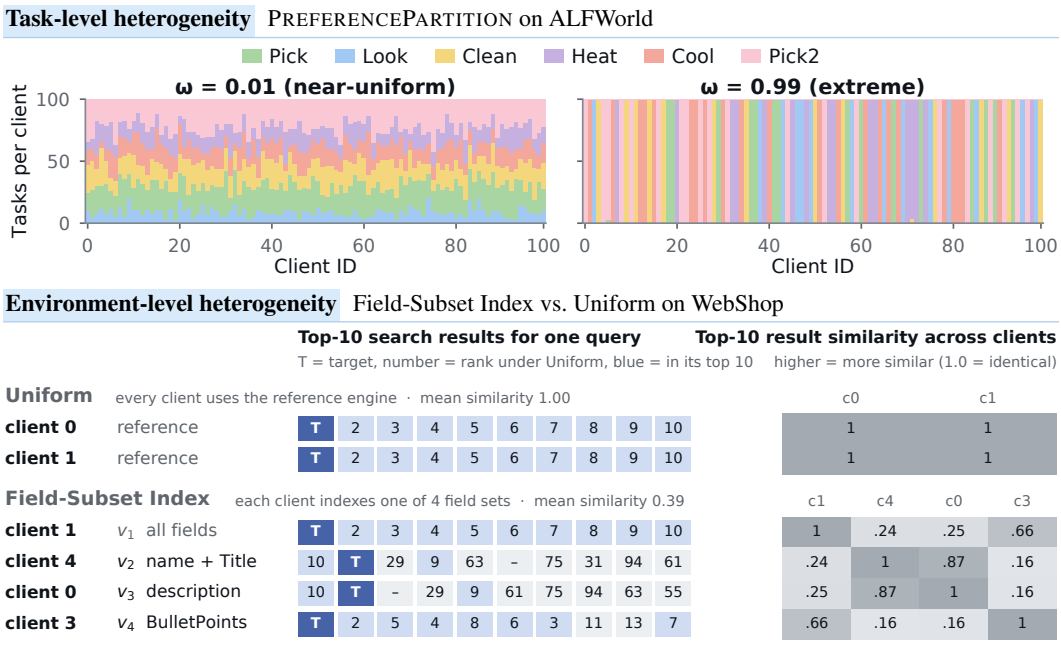


Figure 3: **Constructing the Two Heterogeneity Levels.** Each task-level bar is one client's task pool, colored by task type, at two values of the hyperparameter ω. Each environment-level row shows one client's top-10 search results for the same query, and the matrix shows how similar these results are across clients, averaged over 500 held-out queries. Figures 14 and 19 show the full versions.

π_{ref} on task τ (higher means easier, Appendix J.6), the client statistic $\rho_i = \Pr_{\tau \sim \mathcal{D}_{\tau_i}}[\rho(\tau) \geq \eta]$ for a fixed threshold $\eta \in (0, 1)$, and the dispersion $\Delta^2_{\text{hard}} := \widehat{\text{Var}}_i(\rho_i)$.

Partition algorithms and design properties. We design PREFERENCEPARTITION(ω), COVERAGEPARTITION(ξ), and HARDNESSPARTITION(ξ'), each instantiating one sub-type under a single dispersion hyperparameter: a Dirichlet with concentration $\propto (1-\omega)/\omega$ over per-client category mixes for preference, and Beta-shaped distributions over pool sizes and per-client success-rate quotas for coverage and hardness (Appendices L.1 and F.2). Small ω and large ξ, ξ' give near-uniform splits, while $\omega \to 1$ and $\xi = \xi' = 1$ give the most heterogeneous splits in our experiments (all splits in Appendix L.3). At $\omega = 0.99$ almost every client holds a single task type (Figure 3, top), yet all clients act in the same environment, so the task in the prompt still determines which behavior is needed. Each hyperparameter monotonically controls its target measure and, in expectation, leaves the means $(\bar{p}, \bar{n}, \bar{\rho})$ and the other two measures unchanged (the latter approximately on ALFWorld), so §5.2 can vary each sub-type independently (Theorems 9 and 10, Appendix L.2).

3.3 Environment-Level Heterogeneity

Definition 2 (Environment-Level Heterogeneity). Clients share the task distribution $\mathcal{D}_\tau$, the state/action spaces $(\mathcal{S}, \mathcal{A})$, the reward function R, the discount γ, and the initial-state distribution μ_0, but differ in their transition kernels: $P_i \neq P_j$ for some $i \neq j$. We measure the worst-case inter-environment divergence by $\delta := \sup_{i \neq j,\,(s,a)} D_{\text{TV}}(P_i(\cdot \mid s,a), P_j(\cdot \mid s,a)) \in [0,1]$. By I-D Asymmetry, environment-level heterogeneity is *not* directly observable to the policy. Concrete instances include the same shopping agent deployed across retailers whose search engines rank results differently, or the same computer-use agent across hosts with different OS conventions.

Construction on WebShop. We perturb the kernel along the four stages of WebShop's transition pipeline: *content/catalog*, *encoding/index*, *matching/score*, and *rendering* (Appendices M and F.3). Catalog Split (content) assigns each client a different product subset and leaves the optimal "search → click → buy" policy unchanged. Field-Subset Index (encoding) indexes a client-specific subset of doc-text fields, so identical queries yield different rankings: in Figure 3 (bottom), the clients' top-10 results overlap only 0.39 on average, versus 1.00 with one shared index (Uniform). BM25

Reweighting (matching) sets per-client BM25 parameters (k_1, b). Lookalike Injection (content and matching) injects per-client lookalike products that tie with the target in search ranking but differ from it in one or two attributes (*e.g.*, price). Each lookalike ID carries a visible `LK_` prefix. Rank Wrapper (rendering) reorders BM25 outputs, so the top-ranked result is no longer reliable.

4 Asymmetric Robustness Mechanism

We now derive the central theoretical consequence of I-D Asymmetry. Each theorem compares the federated solution with a task- or environment-conditional optimum and therefore assumes that the policy class $\Pi_\Theta = \{\pi_\theta : \theta \in \Theta\}$ can express it: (R) and (R′) posit exact and ϵ-approximate realizability of task-conditional optima, ($\mathrm{R}_{\mathrm{env}}$) that a policy optimal in every environment is realizable, ($\mathrm{R}_{\mathrm{aug}}$) the same for the history-augmented policies of (C3), and (LR) that each client's own (local) optimum is realizable (formal statements in Appendix N.1, discussion in Appendix F.4).

4.1 Mechanism 1: Task-Level Robustness

Under task-level heterogeneity (Definition 1), federated agent training is *structurally equivalent* to centralized training on the mixture $\bar{\mathcal{D}}_\tau$. Here *robustness* means that for every client i, $\mathcal{J}_i(\hat{\theta}_{\mathrm{fed}})$ matches $\sup_\theta \mathcal{J}_i(\theta)$ up to a slack that vanishes as the LLM grows and training proceeds.

Theorem 1 (Task-Level Robustness, idealized)**.** *Under Definition 1 and (R), (i) the federated objective reduces to the mixture expectation, $\mathcal{J}_{fed}(\theta) = \mathbb{E}_{\tau\sim\bar{\mathcal{D}}_\tau}[\mathcal{J}(\pi_\theta; \tau, \mathcal{M}_{env})]$; and (ii) any $\theta^\star$ satisfying (R) is simultaneously optimal for every client: $\mathcal{J}_i(\theta^\star) = \sup_\theta \mathcal{J}_i(\theta)$ for all i. In particular, no client–federation tradeoff exists at the global optimum.*

Part (i), the *mixture-equivalence identity*, says the federated and centralized objectives coincide pointwise in θ, so extreme one-hot splits and i.i.d. splits pose the same population optimization problem (their FedAvg trajectories can still differ through client drift).

Theorem 2 (Task-Level Robustness, approximate, black-box optimizer)**.** *Under Definition 1 and (R′) (task-conditional optima realizable up to capacity slack ϵ_{approx}), let $\hat{\theta}_{fed}$ be any parameter satisfying $\mathcal{J}_{fed}(\hat{\theta}_{fed}) \geq \sup_\theta \mathcal{J}_{fed}(\theta) - \epsilon_{opt}$. Then for every client i,*

$$\sup_\theta \mathcal{J}_i(\theta) - \mathcal{J}_i(\hat{\theta}_{fed}) \;\leq\; \sqrt{\big(1 + \chi^2(\mathcal{D}_{\tau_i} \,\|\, \bar{\mathcal{D}}_\tau)\big) \cdot R_{\max} H \cdot (\epsilon_{approx} + \epsilon_{opt})}.$$

The bound of Theorem 2 contains *no irreducible δ-term*: both slacks vanish under conditions standard in centralized training, and $\chi^2(\mathcal{D}_{\tau_i} \| \bar{\mathcal{D}}_\tau) \leq N - 1$ under uniform weights no matter how extreme the split is. A linear bound $N(\epsilon_{\mathrm{approx}} + \epsilon_{\mathrm{opt}})$ also holds and is the sharper one as the slacks vanish (Appendix N.2). The mechanism is I-D Asymmetry: because τ is part of the prompt, a single θ can implement different behaviors on different tasks, so different clients' gradients train complementary parts of one task-conditional optimal behavior. For example, if one client holds only kitchen tasks and another only bedroom tasks, their updates improve kitchen and bedroom behavior, respectively, and improving the policy on one client's tasks need not lower the other client's return. Pre-training helps further by making (R′) plausible for instruction-tuned LLMs (Appendix P.1).

4.2 Mechanism 2: Environment-Level Non-Robustness

At the environment level, *non-robustness* means that for some constructions, the worst-off client's return stays below its own optimum by a gap that no larger LLM or longer training removes.

Theorem 3 (Environment-Level Non-Robustness, idealized)**.** *Under Definition 2 and (LR), there exist environment constructions $\{\mathcal{M}_i\}$ such that any $\theta^\star_{fed} \in \arg\max_\theta \mathcal{J}_{fed}(\theta)$ admits a strictly positive worst-client gap $\sup_i[\sup_{\theta_i} \mathcal{J}_i(\theta_i) - \mathcal{J}_i(\theta^\star_{fed})] > 0$.*

The canonical construction is a two-client *transition-swap bandit* (Figure 20, Appendix N.3). The clients share states, actions, rewards, and observations, but action a_1 reaches the rewarding state under P_1 and action a_2 does under P_2. A shared policy that picks a_1 with probability p obtains return $pR_{\max}$ for client 1 and $(1-p)R_{\max}$ for client 2, so $\mathcal{J}_{\mathrm{fed}} = R_{\max}/2$ for every p and the worse-off client always stays at least $R_{\max}/2$ below its own optimum. This tradeoff holds for every θ, however expressive the policy class is, because the policy receives identical inputs in the two clients.

Theorem 4 (Environment-Level Non-Robustness, quantitative)**.** *Under Definition 2 (with divergence δ) and (LR), define the policy-disagreement gap $\Delta_{pol} := \frac{1}{N}\sum_{i=1}^N \sup_{\theta_i\in\Theta} \mathcal{J}_i(\theta_i) -$*

$\sup_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} \mathcal{J}_i(\theta)$. *Then there exist multi-step constructions and a corresponding policy class* Π_Θ *satisfying (LR), such that the observations reveal neither past rewards nor outcomes (i.e., (C3) below is unavailable), yielding* $\Delta_{pol} \geq \Omega(R_{\max} H\delta)$. *The worst-client gap satisfies* $\sup_i[\sup_{\theta_i} \mathcal{J}_i(\theta_i) - \mathcal{J}_i(\theta^\star_{fed})] \geq \Delta_{pol}$ *for any* $\theta^\star_{fed} \in \arg\max_\theta \mathcal{J}_{fed}(\theta)$.

Δ_{pol} is the average value that clients lose by sharing one policy instead of each using its own optimum. The quantitative construction makes the swap partial: each action reaches the rewarding state with probability $(1+\delta)/2$ in one client and $(1-\delta)/2$ in the other, so the per-step gap shrinks to $R_{\max}\delta/2$. Repeating this step H times with observations that never reveal the client adds these gaps up to $\Omega(R_{\max}H\delta)$. This bound is a *floor*, an irreducible lower bound on the worst-client gap, and it characterizes the federated *solution* independently of the optimization trajectory. The mechanism is again I-D Asymmetry, operating in the opposite direction: environment identity is *not* part of the prompt, so at the same observed (s,τ) the policy must emit the same action distribution in every $\mathcal{M}_i$. When per-client optimal actions disagree, client gradients pull the *same* parameters in *opposite* directions, and FedAvg encodes a compromise that is wrong for at least one client. For LLM agents, these conflicting updates can also overwrite the pre-training prior (Appendix P.2).

4.3 Three Sufficient Conditions to Recover Robustness

Theorem 4's construction requires that (a) the optimal trajectory *visits* the region where the kernels disagree, (b) optimal actions there *disagree* across environments, and (c) environment identity is *not* recoverable as policy input. We expect real-world environment heterogeneity to break at least one of these, and identify one condition for each:

- **(C1) Common optimal off-support** (breaks requirement (a)). Some client's optimal policy $\pi^\star$ has an occupancy that avoids the disagreement region $\{(s,a) : P_i(\cdot \mid s,a) \neq P_j(\cdot \mid s,a)\}$, and no client's optimal value exceeds that of $\pi^\star$: the kernels differ only outside the optimal trajectory, so $\pi^\star$ is optimal in every $\mathcal{M}_i$. For example, clients' homes differ in room layout, but the optimal route passes only through rooms that are identical across homes.
- **(C2) Action-preserving with shared** $\pi^\star$ (breaks requirement (b)). On every reachable (s,τ), some action lies in every client's optimal set $\arg\max_a Q^\star_i(s,\tau,a)$, even if the underlying per-client optimal Q-functions $Q^\star_i$ differ. For example, robots differ in friction, but the same torque is optimal at each state for every robot.
- **(C3) Self-revealing environment via observation history** (breaks requirement (c)). Environment identity is inferable from the trajectory prefix $h_u := (s_0, a_0, r_0, \ldots, s_u)$: after a fixed default policy runs for $u^\star \leq H$ steps, a classifier identifies the environment with error at most η_{cls}, per-client kernel models are η_{ker}-accurate in TV, and these steps avoid low-value states (formal conditions in Appendix F.5). For example, a web agent conditions on a "united.com" banner.

Theorem 5 (Recovery via (C1)–(C3))**.** *Under (C1) or (C2) (with* (R_{env})*), the policy-disagreement gap* $\Delta_{pol} = 0$. *Under (C3) and* (R_{aug}), $\Delta_{pol} \leq O(R_{\max}u^\star + R_{\max}H\eta_{cls} + R_{\max}H^2\eta_{ker})$, *so* $\Delta_{pol}/(R_{\max}H) \to 0$ *as* $(u^\star/H, \eta_{cls}, H\eta_{ker}) \to 0$, *for any TV divergence* δ. *Here* (R_{aug}) *ensures that* Θ *realizes the history-augmented policies, so both suprema in* Δ_{pol} *remain over* Θ.

Thus (C1) and (C2) return the environment-level case to Mechanism 1's regime *exactly*, and (C3) *approximately*. The conditions are sufficient but not necessary (Appendix O.8.4). For LLM agents, **(C3) is particularly important**: pre-training can enable *in-context posterior inference* about the environment from branding, error messages, and structural cues, and (R_{aug}) captures this inference formally. Our synthetic variants suppress such environment-identity cues. Appendices P.3 and P.4 discuss the explicit (environment-aware prompting, Corollary 1) and implicit forms of (C3).

4.4 Four Empirical Patterns Explained by the Mechanism

The mechanism predicts four training-curve patterns (full statements in Appendix F.6, extended discussion in Appendix O). Two orthogonal factors set the pattern: the achievable plateau, and the optimizer slack ϵ_{opt}, which determines whether the plateau is attained. In the task-level regime (Pattern A), the plateau is the same for every split. In the environment-level regime (Patterns B/C/D), the plateau depends on how closely (C1)–(C3) hold.

- **Pattern A (Task-level robust).** Splits sharing a mixture share the population objective (Theorem 1(i)) and, under (R), a common optimum (Theorem 1(ii)), so the federated curve reaches the near-uniform baseline's plateau up to training noise unless the split raises ϵ_{opt}.

Method	ALFWorld							WebShop	
	Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Qwen2.5-1.5B-Instruct									
Local (Client 21)	14.3	25.0	16.7	22.2	27.3	20.0	20.3	71.8	37.5
Local (Client 42)	7.1	12.5	8.3	33.3	9.1	10.0	12.5	62.7	32.8
Local (Client 84)	42.9	50.0	33.3	11.1	45.5	30.0	35.9	60.3	26.6
Centralized	**81.0**$_{\pm 10.9}$	79.2$_{\pm 14.4}$	44.4$_{\pm 4.8}$	**81.5**$_{\pm 6.4}$	27.3$_{\pm 9.1}$	26.7$_{\pm 11.5}$	56.2$_{\pm 5.6}$	76.5$_{\pm 2.4}$	60.4$_{\pm 3.3}$
FEDAGENT	50.0$_{\pm 12.4}$	**83.3**$_{\pm 7.2}$	**50.0**$_{\pm 14.4}$	63.0$_{\pm 12.8}$	**60.6**$_{\pm 10.5}$	**83.3**$_{\pm 5.8}$	**63.0**$_{\pm 2.4}$	**76.6**$_{\pm 4.7}$	**61.5**$_{\pm 3.9}$
Qwen2.5-3B-Instruct									
Local (Client 21)	14.3	12.5	8.3	11.1	9.1	10.0	10.9	64.3	43.8
Local (Client 42)	21.4	50.0	16.7	33.3	36.4	30.0	29.7	71.8	37.5
Local (Client 84)	35.7	25.0	50.0	44.4	45.5	50.0	42.2	60.8	31.2
Centralized	**90.5**$_{\pm 4.1}$	**75.0**$_{\pm 12.5}$	**86.1**$_{\pm 9.6}$	44.4$_{\pm 11.1}$	**75.8**$_{\pm 13.9}$	20.0$_{\pm 10.0}$	67.7$_{\pm 7.9}$	81.4$_{\pm 7.0}$	**64.6**$_{\pm 4.8}$
FEDAGENT	73.8$_{\pm 8.2}$	62.5$_{\pm 12.5}$	61.1$_{\pm 12.7}$	**85.2**$_{\pm 6.4}$	51.5$_{\pm 5.2}$	**76.7**$_{\pm 11.5}$	**68.2**$_{\pm 3.6}$	**83.6**$_{\pm 5.5}$	63.0$_{\pm 7.0}$
Qwen2.5-7B-Instruct									
Local (Client 21)	42.9	50.0	41.7	33.3	27.3	30.0	37.5	75.1	45.3
Local (Client 42)	50.0	25.0	50.0	44.4	36.4	40.0	42.2	70.4	48.4
Local (Client 84)	7.1	12.5	16.7	22.2	9.1	20.0	14.1	62.2	37.5
Centralized	59.5$_{\pm 14.9}$	**79.2**$_{\pm 7.2}$	83.3$_{\pm 8.3}$	**81.5**$_{\pm 12.8}$	**87.9**$_{\pm 5.2}$	**66.7**$_{\pm 5.8}$	75.5$_{\pm 3.3}$	**89.3**$_{\pm 4.1}$	**71.4**$_{\pm 2.4}$
FEDAGENT	**88.1**$_{\pm 4.1}$	58.3$_{\pm 14.4}$	**88.9**$_{\pm 4.8}$	77.8$_{\pm 11.1}$	84.8$_{\pm 10.5}$	53.3$_{\pm 11.5}$	**77.1**$_{\pm 5.0}$	88.1$_{\pm 3.3}$	68.8$_{\pm 3.1}$
Llama-3.2-3B-Instruct									
Local (Client 21)	7.1	12.5	8.3	11.1	9.1	10.0	9.4	65.7	26.6
Local (Client 42)	28.6	37.5	41.7	**33.3**	18.2	50.0	34.4	60.1	34.4
Local (Client 84)	14.3	25.0	16.7	22.2	**45.5**	20.0	23.4	71.0	39.1
Centralized	78.6$_{\pm 7.1}$	**70.8**$_{\pm 7.2}$	**80.6**$_{\pm 12.7}$	18.5$_{\pm 12.8}$	21.2$_{\pm 13.9}$	50.0$_{\pm 10.0}$	55.2$_{\pm 6.5}$	77.2$_{\pm 6.3}$	56.8$_{\pm 5.5}$
FEDAGENT	**83.3**$_{\pm 10.9}$	45.8$_{\pm 14.4}$	63.9$_{\pm 9.6}$	29.6$_{\pm 6.4}$	36.4$_{\pm 9.1}$	**76.7**$_{\pm 5.8}$	**58.3**$_{\pm 1.8}$	**77.7**$_{\pm 3.9}$	**57.8**$_{\pm 6.2}$

Table 1: **Performance Comparison on ALFWorld and WebShop**. The policy optimization algorithm is GRPO. The PPO counterpart is in Table 3 (Appendix G.2). We report the mean and standard deviation over three random seeds for Centralized and FEDAGENT. For ALFWorld, we report the Success Rate (%). For WebShop, we report both the Task Score (0–100, the WebShop convention of $100\times$ the mean graded goal-match score $r \in [0, 1]$) and the Success Rate (%).

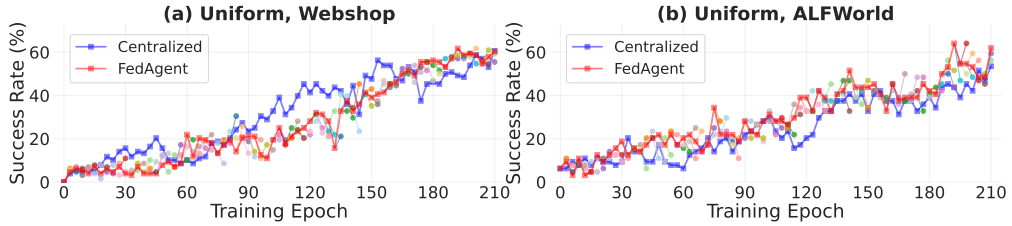


Figure 4: **Training Dynamics of FEDAGENT (GRPO) and Centralized Training** (validation success rate, `Qwen2.5-1.5B-Instruct`). Red is FEDAGENT's *aggregated server model*, blue the centralized baseline, and colored circles the *client models* after local training in each round.

- **Pattern B (Environment-level robust).** One of (C1)–(C3) holds and the optimizer attains the shared optimum: the curve nearly matches the single-environment baseline (Theorem 5).
- **Pattern C (Environment-level degrade-but-stable).** (C1)–(C3) hold at best partially, or one holds fully but the optimizer leaves a nonzero ϵ_{opt}. Performance falls by a finite margin that depends on these slacks, and training stays stable.
- **Pattern D (Environment-level collapse).** All of (C1)–(C3) fail on an adversarial construction whose observations never reveal the client, so Theorem 4's floor can apply. Performance degrades severely, possibly with oscillation, seed divergence, or forgetting of pre-trained capabilities.

5 Empirical Results

We empirically validate the Asymmetric Robustness Mechanism on two agent benchmarks: WebShop [165], a simulated e-commerce environment where the agent searches, browses, and purchases products against natural-language shopping instructions, and ALFWorld [114], a text-based embodied household benchmark with six task types (Pick, Look, Clean, Heat, Cool, Pick2). §5.1 answers (Q1) by comparing FEDAGENT with centralized and single-client local training under

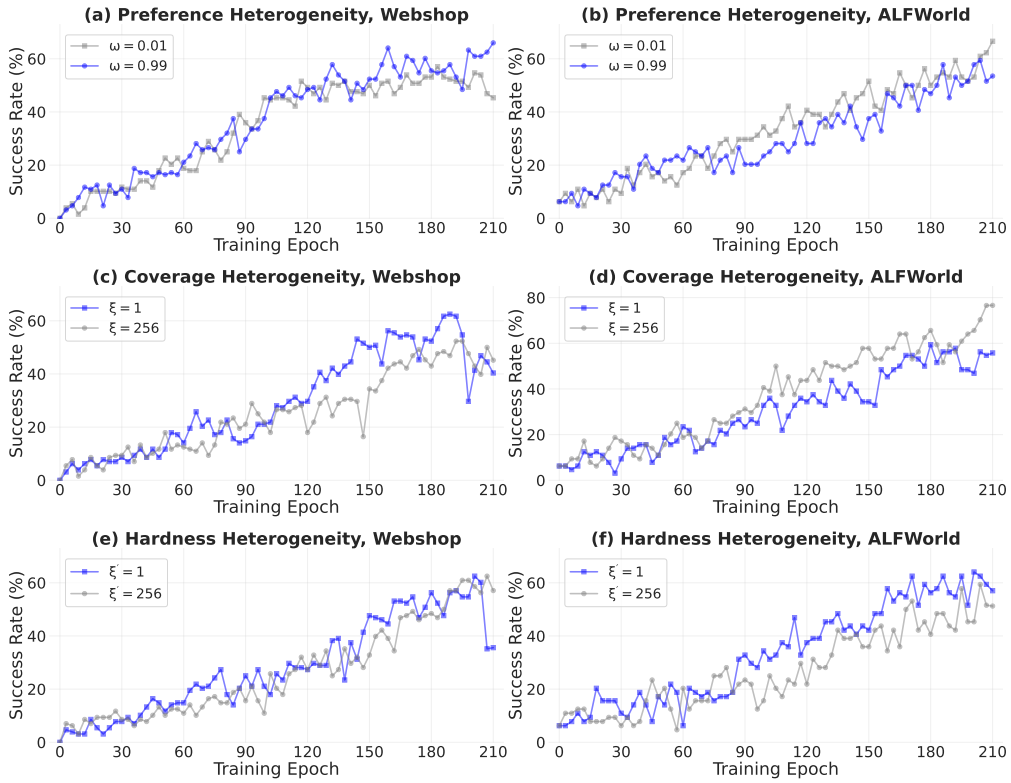


Figure 5: **Training Dynamics of FEDAGENT (GRPO) under Task-Level Heterogeneity** (PPO in Appendix G.3). Panels (a,b) show preference, (c,d) coverage, and (e,f) hardness. The extreme-heterogeneity curve (blue) closely tracks the near-uniform baseline (gray), consistent with Pattern A.

uniform client distributions. §5.2 and §5.3 answer (Q2) by testing whether Pattern A holds across the three task-level sub-types and whether environment-level heterogeneity elicits Patterns B/C/D. We use `Qwen2.5-{1.5,3,7}B-Instruct` and `Llama-3.2-3B-Instruct` as agent backbones, and instantiate FEDAGENT with GRPO and PPO. Tables report three-seed means and standard deviations for FEDAGENT and centralized training, and training curves are single-seed runs.

5.1 FEDAGENT under Uniform Client Distribution

Setup. We partition each dataset into 100 clients with 100 task instructions each (pools may overlap), select $M = 2$ clients per round, train each for $E = 3$ local epochs, and run $T = 70$ rounds, for $T \cdot E = 210$ sequential local epochs (the x-axis of our training curves). Appendix G.1 ablates this configuration. Each local epoch trains on one batch of 64 rollout trajectories, with tasks sampled with replacement (GRPO uses 8 tasks with $G = 8$ rollouts each, and PPO uses 64 tasks with one rollout each). We compare against Centralized Agent Training (full dataset, 210 epochs with the same per-epoch batch) and Local Agent Training (one client's dataset, for clients 21, 42, and 84).

Results. Table 1 and Figure 4 answer (Q1) affirmatively: **FEDAGENT matches centralized agent training and outperforms single-client local training, often substantially**, across model scales, benchmarks, and client indices. On ALFWorld with `Qwen2.5-1.5B-Instruct`, FEDAGENT reaches 63.0% success rate on `All` versus 12.5–35.9% for local training and 56.2% for centralized. On WebShop with `Llama-3.2-3B-Instruct`, the corresponding success rates are 57.8%, 26.6–39.1%, and 56.8%. Across backbones, FEDAGENT minus centralized is within ± 2.6 pp on both WebShop metrics and $+0.5$ to $+6.8$ pp on ALFWorld `All`, comparable to the seed-level standard deviations. Local training is weaker and, depending on the backbone, varies by 23–31 pp across client indices on ALFWorld `All`, whereas every FEDAGENT participant receives the same aggregated model. The validation curves (Figure 4) show both paradigms reaching $\sim$55–60% on both benchmarks, with centralized training leading by up to $\sim$15 points mid-training on WebShop before FEDAGENT closes the gap. This match is consistent with Theorem 1(i): the federated objective equals the centralized objective on the mixture of client data, so the remaining differences come from optimization, namely client drift over 3 local epochs and partial participation of 2 clients per round.

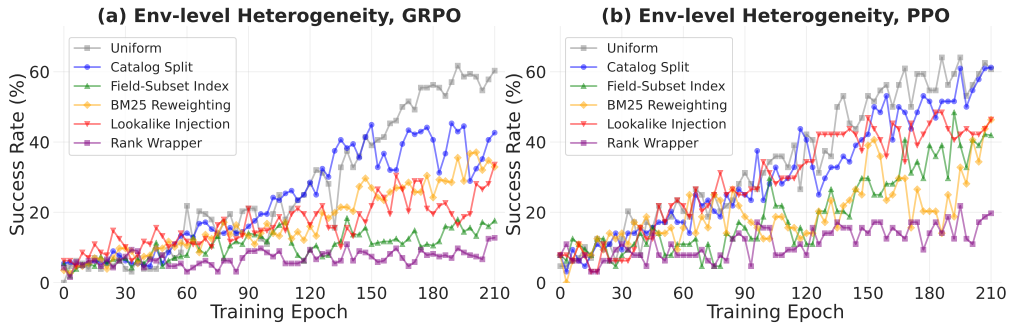


Figure 6: **Training Dynamics of FEDAGENT on WebShop under Environment-Level Heterogeneity**, with GRPO (left) and PPO (right). The gray curve is the single-environment baseline, and the five variants span the **stable** → **degrade** → **collapse** spectrum of Patterns B/C/D (§5.3).

5.2 Impact of Task-Level Heterogeneity

Setup. We vary each sub-type independently with the partitions of §3.2: the preference and hardness partitions keep $|X_i| = L = 100$, while COVERAGEPARTITION(ξ) varies n_i within $[100, 1000]$ (mean 500). For each hyperparameter, we compare a near-uniform setting (gray: $\omega = 0.01$, $\xi = 256$, $\xi' = 256$) with an extreme setting (blue: $\omega = 0.99$, $\xi = 1$, $\xi' = 1$). Other settings (100 clients, 210 total epochs, `Qwen2.5-1.5B-Instruct`) match §5.1.

Results. Figure 5 shows that FEDAGENT **remains robust under extreme task-level heterogeneity** in all three sub-types. Moving each sub-type from near-uniform (gray) to its extreme (blue) shifts the converged plateau only marginally, sometimes even upward. The largest separation (Coverage/ALFWorld, panel d) is within the 10–20-point round-to-round variability of LLM-agent RL under sparse rewards. This split-invariance is Pattern A, consistent with mixture equivalence (Theorem 1).

5.3 Impact of Environment-Level Heterogeneity

Setup. We train on each WebShop variant (§3.3) under both GRPO and PPO. Clients differ along that variant's perturbation axis and use a *uniform* task partition, with all other settings as in §5.1 (search-engine differences in Appendix J.6). Except for Catalog Split, each variant assigns clients near-uniformly to four configurations (Appendix M.3). Validation uses the unperturbed WebShop.

Results. In contrast to §5.2, Figure 6 shows that FEDAGENT **is often non-robust under environment-level heterogeneity** (Appendix F.7). Catalog Split leaves the "search → click → buy" optimal policy unchanged, so (C2) holds. PPO attains the baseline plateau at ∼60% (Pattern B), while GRPO plateaus stably at ∼40% (Pattern C at its high end, due to optimizer slack ϵ_{opt}). Field-Subset Index and BM25 Reweighting change how the same query maps to ranked products, so (C1) fails and (C2) holds only partially. Under GRPO they plateau at ∼17% and ∼33%, respectively (Pattern C). PPO raises Field-Subset Index to ∼42%, but BM25 Reweighting only fluctuates (∼14–46% late in training, mean ∼30%). Lookalike Injection and Rank Wrapper break (C1) but differ on (C2). The `LK_` prefix makes one client-independent rule (never buy a lookalike) optimal for every client, so (C2) holds, whereas Rank Wrapper's reordering admits no shared rule and leaves no usable (C3) cue. Lookalike Injection ends at ∼34% under GRPO and ∼46% under PPO (Pattern C, optimizer slack). Rank Wrapper stays flat below 13% under GRPO and reaches only ∼20% under PPO, the lowest curve under each optimizer (Pattern D, partly because clients whose wrapper inverts the ranking cannot succeed, Appendix M.8). **The optimizer thus affects only whether a variant reaches Pattern B or C, and the variant that breaks all of (C1)–(C3) stays in Pattern D**.

6 Conclusion

We explored FEDAGENT, a decentralized RL paradigm for LLM agents. The Input-Dynamics Asymmetry of task-augmented MDPs yields an Asymmetric Robustness Mechanism: FEDAGENT is robust to task-level heterogeneity (Pattern A) but worst-case non-robust to environment-level heterogeneity (Pattern D), and (C1)–(C3) recover full (Pattern B) or partial (Pattern C) robustness. Experiments on WebShop and ALFWorld support this mechanism and identify FEDAGENT as a promising way to train LLM agents without centralizing user data.

Acknowledgments and Disclosure of Funding

Canyu Chen is partially supported by the Bloomberg Data Science Ph.D. Fellowship. The authors declare no competing interests.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama. Federated learning based on dynamic regularization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=B7v4QMR6Z9w.

[3] A. Ahmadian, C. Cremer, M. Gallé, M. Fadaee, J. Kreutzer, O. Pietquin, A. Üstün, and S. Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.662.

[4] D. Alistarh, D. Grubic, J. Z. Li, R. Tomioka, and M. Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[5] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary. Federated learning with personalization layers. *CoRR*, abs/1912.00818, 2019. URL http://arxiv.org/abs/1912.00818.

[6] H. Bai, Y. Zhou, M. Cemri, J. Pan, A. Suhr, S. Levine, and A. Kumar. DigiRL: Training in-the-wild device-control agents with autonomous reinforcement learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/1704ddd0bb89f159dfe609b32c889995-Abstract-Conference.html.

[7] J. Bai, D. Chen, B. Qian, L. Yao, and Y. Li. Federated fine-tuning of large language models under heterogeneous tasks and client resources. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/1a134b50202088aa8c595cc99b310e5a-Abstract-Conference.html.

[8] M. Balunović, D. I. Dimitrov, N. Jovanović, and M. Vechev. LAMP: Extracting text from gradients with language model priors. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/323752600904f907ceae19f3564a7e-Abstract-Conference.html.

[9] J. Bian, L. Wang, L. Zhang, and J. Xu. FedALT: Federated fine-tuning through adaptive local training with rest-of-world LoRA. In *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pages 19728–19736. Association for the Advancement of Artificial Intelligence (AAAI), 2026. doi: 10.1609/aaai.v40i24.39054.

[10] K. A. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1175–1191. ACM, 2017. doi: 10.1145/3133956.3133982. URL https://doi.org/10.1145/3133956.3133982.

[11] B. Chen, C. Shu, E. Shareghi, N. Collier, K. Narasimhan, and S. Yao. FireAct: Toward language agent fine-tuning. *CoRR*, abs/2310.05915, 2023. doi: 10.48550/ARXIV.2310.05915. URL https://doi.org/10.48550/arXiv.2310.05915.

[12] K. Chen, M. Cusumano-Towner, B. Huval, A. Petrenko, J. Hamburger, V. Koltun, and P. Krähenbühl. Reinforcement learning for long-horizon interactive LLM agents. *CoRR*, abs/2502.01600, 2025. doi: 10.48550/ARXIV.2502.01600. URL https://doi.org/10.48550/arXiv.2502.01600.

[13] P. Chen, S. Li, K. Wang, Y. Yuan, K. Han, J. Luo, and F. Li. FGRPO: Federated GRPO with adaptive aggregation on non-IID data. *CoRR*, abs/2606.03094, 2026. URL https://arxiv.org/abs/2606.03094.

[14] S. Chen, T. Zhu, Z. Wang, J. Zhang, K. Wang, R. Zhou, S. Gao, T. Xiao, Y. W. Teh, J. He, and M. Li. Why do LLM agents fail in exploring new environments? A world-modeling perspective. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 21612–21626. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.findings-emnlp.1023/.

[15] X. Chen, Y. Shi, Q. Lan, Y. Qiu, M. Wang, X. Gu, and Y. Yan. Fed-SE: Federated self-evolution for privacy-constrained multi-environment LLM agents. *CoRR*, abs/2512.08870, 2025. URL https://doi.org/10.48550/arXiv.2512.08870.

[16] Z. Chen, Z. Zhao, K. Zhang, B. Liu, Q. Qi, Y. Wu, T. Kalluri, S. Cao, Y. Xiong, H. Tong, H. Yao, H. Li, J. Zhu, X. Li, D. Song, B. Li, J. Weston, and D. Huynh. Scaling agent learning via experience synthesis. In *The Fourteenth International Conference on Learning Representations, ICLR 2026*, 2026. URL https://openreview.net/forum?id=cf7qpBwttr.

[17] M. Cheng, S. Yu, D. Wang, Q. Li, X. Tao, J. Ouyang, Y. Luo, Y. Zhou, Q. Liu, and E. Chen. Agent-R1: A unified and modular framework for agentic reinforcement learning. In *Proceedings of the 35th ACM International Conference on Information and Knowledge Management*, pages 9092–9096. ACM, 2026. doi: 10.1145/3799682.3840255.

[18] K. Cheruiyot, N. Kiprotich, V. Kungurtsev, K. Mugo, V. Mwirigi, and M. Ngesa. A survey of multi agent reinforcement learning: Federated learning and cooperative and noncooperative decentralized regimes. *CoRR*, abs/2507.06278, 2025. doi: 10.48550/ARXIV.2507.06278. URL https://doi.org/10.48550/arXiv.2507.06278.

[19] Y. J. Cho, L. Liu, Z. Xu, A. Fahrezi, and G. Joshi. Heterogeneous LoRA for federated fine-tuning of on-device foundation models. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 12903–12913. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.717. URL https://doi.org/10.18653/v1/2024.emnlp-main.717.

[20] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai. Exploiting shared representations for personalized federated learning. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 2089–2099. PMLR, 2021. URL http://proceedings.mlr.press/v139/collins21a.html.

[21] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[22] Y. Deng, M. M. Kamani, and M. Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

[23] H. Ding, P. Liu, J. Wang, Z. Ji, M. Cao, R. Zhang, L. Ai, E. Yang, T. Shi, and L. Yu. DynaWeb: Model-based reinforcement learning of web agents. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, pages 54628–54642. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.emnlp-main.2404/.

[24] C. T. Dinh, N. H. Tran, and T. D. Nguyen. Personalized federated learning with Moreau envelopes. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f4f1f13c8289ac1b1ee0ff176b56fc60-Abstract.html.

[25] Y. Du, R. Ye, F. Yuchi, W. Zhao, J. Qu, Y. Wang, and S. Chen. FedDQC: Data quality control in federated instruction-tuning of large language models. In W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, editors, *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 15267–15291. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.791/.

[26] A. Fallah, A. Mokhtari, and A. Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/24389bfe4fe2eba8bf9aa9203a44cdad-Abstract.html.

[27] F. X. Fan, Y. Ma, Z. Dai, W. Jing, C. Tan, and B. K. H. Low. Fault-tolerant federated reinforcement learning with theoretical guarantee. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 1007–1021, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/080acdcce72c06873a773c4311c2e464-Abstract.html.

[28] F. X. Fan, C. Tan, Y.-S. Ong, R. Wattenhofer, and W.-T. Ooi. FedRLHF: A convergence-guaranteed federated framework for privacy-preserving and personalized RLHF. In *Proceedings of the 2025 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025*, pages 713–721. International Foundation for Autonomous Agents and Multiagent Systems, 2025.

[29] T. Fan, G. Ma, Y. Song, L. Fan, K. Chen, and Q. Yang. FedProxy: Federated fine-tuning of LLMs via proxy SLMs and heterogeneity-aware fusion. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17482–17497. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.acl-long.794/.

[30] W. Fang, D. Han, L. Yuan, S. Hosseinalipour, and C. G. Brinton. Federated sketching LoRA: A flexible framework for heterogeneous collaborative fine-tuning of LLMs. In T. Zhang, M. Dudik, M. Jaggi, A. Agarwal, S. Li, D. Schuurmans, J. Zhu, F. Berkenkamp, H. Dong, and A. Bietti, editors, *Forty-third International Conference on Machine Learning, ICML 2026, Seoul, South Korea, July 6-11, 2026*, volume 306 of *Proceedings of Machine Learning Research*, pages 29060–29092. PMLR, 2026. URL https://proceedings.mlr.press/v306/fang26e.html.

[31] L. Feng, Z. Xue, T. Liu, and B. An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=QXEhBMNrCW.

[32] D. Fu, K. He, Y. Wang, W. Hong, Z. Gongque, W. Zeng, W. Wang, J. Wang, X. Cai, and W. Xu. AgentRefine: Enhancing agent generalization through refinement tuning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=FDimWzmcWn.

[33] D. Gao, X. Yao, and Q. Yang. A survey on heterogeneous federated learning. *CoRR*, abs/2210.04505, 2022. doi: 10.48550/ARXIV.2210.04505. URL https://doi.org/10.48550/arXiv.2210.04505.

[34] H. Gao, J. Geng, W. Hua, M. Hu, X. Juan, H. Liu, S. Liu, J. Qiu, X. Qi, Q. Ren, Y. Wu, H. Wang, H. Xiao, Y. Zhou, S. Zhang, J. Zhang, J. Xiang, Y. Fang, Q. Zhao, D. Liu, C. Qian, Z. Wang, M. Hu, H. Wang, Q. Wu, H. Ji, and M. Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=CTr3bovS5F.

[35] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c4ede56bbd98819ae6112b20ac6bf145-Abstract.html.

[36] R. C. Geyer, T. Klein, and M. Nabi. Differentially private federated learning: A client level perspective. *CoRR*, abs/1712.07557, 2017. URL http://arxiv.org/abs/1712.07557.

[37] A. Golubev, M. Trofimova, S. Polezhaev, I. Badertdinov, M. Nekrashevich, A. Shevtsov, S. Karasik, S. Abramov, A. Andriushchenko, F. Fisin, S. Skvortsov, and B. Yangel. Training long-context, multi-turn software engineering agents with reinforcement learning. *CoRR*, abs/2508.03501, 2025. doi: 10.48550/ARXIV.2508.03501. URL https://doi.org/10.48550/arXiv.2508.03501.

[38] P. Guo, S. Zeng, Y. Wang, H. Fan, F. Wang, and L. Qu. Selective aggregation for low-rank adaptation in federated learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=iX3uESGdsO.

[39] P. Guo, S. Zeng, Z. Li, W. Zheng, M. Liu, and L. Qu. Fed-GRPO: Reward-signal-driven federated group relative policy optimization. *CoRR*, abs/2610.11502, 2026. URL https://arxiv.org/abs/2610.11502.

[40] Y. Guo, X. Tang, and T. Lin. Enhancing clustered federated learning: Integration of strategies and improved methodologies. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=zPDpdk3V8L.

[41] S. He. Gradient inversion in federated reinforcement learning. *CoRR*, abs/2512.00303, 2025. doi: 10.48550/ARXIV.2512.00303. URL https://doi.org/10.48550/arXiv.2512.00303.

[42] Z. He and G. Joshi. FedGuide: Diffusion prior alignment and value baseline guidance for heterogeneous federated reinforcement learning. In *10th Conference on Robot Learning, CoRL 2026*, 2026. URL https://openreview.net/forum?id=QBAA8svAIK.

[43] T. H. Hsu, H. Qi, and M. Brown. Measuring the effects of non-identical data distribution for federated visual classification. *CoRR*, abs/1909.06335, 2019. URL http://arxiv.org/abs/1909.06335.

[44] G. Huang and T. Shu. Federated oriented learning: A practical one-shot personalized federated learning framework. In A. Singh, M. Fazel, D. Hsu, S. Lacoste-Julien, F. Berkenkamp, T. Maharaj, K. Wagstaff, and J. Zhu, editors, *Forty-second International Conference on*

Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025, volume 267 of *Proceedings of Machine Learning Research*, pages 25762–25790. PMLR, 2025. URL https://proceedings.mlr.press/v267/huang25ae.html.

[45] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora. Evaluating gradient inversion attacks and defenses in federated learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 7232–7241, 2021.

[46] U. Hwang and S. Hong. Federated reinforcement learning in heterogeneous environments. *CoRR*, abs/2507.14487, 2025. doi: 10.48550/ARXIV.2507.14487. URL https://doi.org/10.48550/arXiv.2507.14487.

[47] N. Hyeon-Woo, M. Ye-Bin, and T.-H. Oh. FedPara: Low-rank Hadamard product for communication-efficient federated learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[48] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan. SWE-bench: Can language models resolve real-world Github issues? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

[49] B. Jin, H. Zeng, Z. Yue, J. Yoon, S. Ö. Arık, D. Wang, H. Zamani, and J. Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling, COLM 2025*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

[50] H. Jin, Y. Peng, W. Yang, S. Wang, and Z. Zhang. Federated reinforcement learning with environment heterogeneity. In G. Camps-Valls, F. J. R. Ruiz, and I. Valera, editors, *International Conference on Artificial Intelligence and Statistics, AISTATS 2022, 28-30 March 2022, Virtual Event*, volume 151 of *Proceedings of Machine Learning Research*, pages 18–37. PMLR, 2022. URL https://proceedings.mlr.press/v151/jin22a.html.

[51] P. Kairouz, H. B. McMahan, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1-2):1–210, 2021.

[52] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.

[53] M. J. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002. doi: 10.1023/A:1017984413808. URL https://doi.org/10.1023/A:1017984413808.

[54] S. Khodadadian, P. Sharma, G. Joshi, and S. T. Maguluri. Federated reinforcement learning: Linear speedup under Markovian sampling. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 10997–11057. PMLR, 2022. URL https://proceedings.mlr.press/v162/khodadadian22a.html.

[55] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. *CoRR*, abs/1610.05492, 2016. URL http://arxiv.org/abs/1610.05492.

[56] J. Koo, H. Kim, M. Jang, and J. Ok. Federated variational preference alignment with Gumbel-Softmax prior for personalized user preferences. In T. Zhang, M. Dudik, M. Jaggi, A. Agarwal, S. Li, D. Schuurmans, J. Zhu, F. Berkenkamp, H. Dong, and A. Bietti, editors, *Forty-third International Conference on Machine Learning, ICML 2026, Seoul, South Korea, July 6-11, 2026*, volume 306 of *Proceedings of Machine Learning Research*, pages 60215–60235. PMLR, 2026. URL https://proceedings.mlr.press/v306/koo26a.html.

[57] W. Kuang, B. Qian, Z. Li, D. Chen, D. Gao, X. Pan, Y. Xie, Y. Li, B. Ding, and J. Zhou. FederatedScope-LLM: A comprehensive package for fine-tuning large language models in federated learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 5260–5271. ACM, 2024. doi: 10.1145/3637528.3671573. URL https://doi.org/10.1145/3637528.3671573.

[58] S. Labbi, D. Tiapkin, L. Mancini, P. Mangold, and E. Moulines. Federated UCBVI: Communication-efficient federated regret minimization with heterogeneous agents. In Y. Li, S. Mandt, S. Agrawal, and E. Khan, editors, *International Conference on Artificial Intelligence and Statistics, AISTATS 2025, Mai Khao, Thailand, 3-5 May 2025*, volume 258 of *Proceedings of Machine Learning Research*, pages 1315–1323. PMLR, 2025. URL https://proceedings.mlr.press/v258/labbi25a.html.

[59] G. Lan, D. Han, A. Hashemi, V. Aggarwal, and C. G. Brinton. Asynchronous federated reinforcement learning with policy gradient updates: Algorithm design and convergence analysis. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=5DUekOKWcS.

[60] S. Lee, W. Fang, D. Han, S. Hosseinalipour, and C. G. Brinton. Self-play enhancement via advantage-weighted refinement in online federated LLM fine-tuning. *CoRR*, abs/2605.07977, 2026. URL https://arxiv.org/abs/2605.07977.

[61] C. Li, Q. Zhao, F. Mo, and C. Chen. FedCoT: Communication-efficient federated reasoning enhancement for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 2586–2605. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.findings-emnlp.127/.

[62] D. Li and J. Wang. FedMD: Heterogenous federated learning via model distillation. *CoRR*, abs/1910.03581, 2019. URL http://arxiv.org/abs/1910.03581.

[63] Q. Li, B. He, and D. Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10713–10722, 2021.

[64] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[65] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems*, volume 2, pages 429–450, 2020.

[66] T. Li, S. Hu, A. Beirami, and V. Smith. Ditto: Fair and robust federated learning through personalization. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6357–6368. PMLR, 2021. URL http://proceedings.mlr.press/v139/li21h.html.

[67] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang. On the convergence of FedAvg on non-IID data. In *8th International Conference on Learning Representations, ICLR 2020, Virtual Event, April 26 - May 1, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJxNAnVtDS.

[68] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou. FedBN: Federated learning on non-IID features via local batch normalization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=6YEQUn0QICG.

[69] X. Liang, Y. Liu, T. Chen, M. Liu, and Q. Yang. Federated transfer reinforcement learning for autonomous driving. In *Federated and Transfer Learning*, Adaptation, Learning, and Optimization, pages 357–371. Springer International Publishing, 2022. doi: 10.1007/978-3-031-11748-0_15.

[70] Z. Liang, H. Zhang, and L. Xue. Provably efficient federated reinforcement learning with linear function approximation and logarithmic communication cost. *CoRR*, abs/2609.00193, 2026. URL https://arxiv.org/abs/2609.00193.

[71] Y. Liao, W. Huang, G. Wan, J. Liang, B. Yang, and M. Ye. Splitting with importance-aware updating for heterogeneous federated learning with large language models. In A. Singh, M. Fazel, D. Hsu, S. Lacoste-Julien, F. Berkenkamp, T. Maharaj, K. Wagstaff, and J. Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*, pages 37495–37510. PMLR, 2025. URL https://proceedings.mlr.press/v267/liao25c.html.

[72] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[73] T. Lin, L. Kong, S. U. Stich, and M. Jaggi. Ensemble distillation for robust model fusion in federated learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/18df51b97ccd68128e994804f3eccc87-Abstract.html.

[74] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=SkhQHMW0W.

[75] Z. Ling, D. Chen, L. Yao, Y. Li, and Y. Shen. On the convergence of zeroth-order federated tuning for large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1827–1838. ACM, 2024. doi: 10.1145/3637528.3671865.

[76] B. Liu, L. Wang, and M. Liu. Lifelong federated reinforcement learning: A learning architecture for navigation in cloud robotic systems. *IEEE Robotics and Automation Letters*, 4(4): 4555–4562, 2019.

[77] B. Liu, X. Li, J. Zhang, J. Wang, T. He, S. Hong, H. Liu, S. Zhang, K. Song, K. 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.

[78] H. Liu, R. Wen, S. Nair, J. Liu, W. Lou, C. Zhang, W. Yeoh, Y. Vorobeychik, and N. Zhang. EcoLoRA: Communication-efficient federated fine-tuning of large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20732–20746. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1046.

[79] S. Liu, Z. Liang, X. Lyu, and C. Amato. LLM collaboration with multi-agent reinforcement learning. In S. Koenig, C. Jenkins, and M. E. Taylor, editors, *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pages 32150–32158. Association for the Advancement of Artificial Intelligence (AAAI), 2026. doi: 10.1609/AAAI.V40I38.40487. URL https://doi.org/10.1609/aaai.v40i38.40487.

[80] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, S. Zhang, X. Deng, A. Zeng, Z. Du, C. Zhang, S. Shen, T. Zhang, Y. Su, H. Sun, M. Huang, Y. Dong, and J. Tang. AgentBench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

[81] X. Liu, K. Wang, Y. Wu, F. Huang, Y. Li, J. Jiao, and J. Zhang. Agentic reinforcement learning with implicit step rewards. In *The Fourteenth International Conference on Learning Representations, ICLR 2026*, 2026. URL https://openreview.net/forum?id=ooROvpmxMV.

[82] Z. Liu, J. Chai, X. Zhu, S. Tang, R. Ye, W. Ma, B. Zhang, L. Bai, and S. Chen. ML-Agent: Reinforcing LLM agents for autonomous machine learning engineering. In T. Zhang, M. Dudik, M. Jaggi, A. Agarwal, S. Li, D. Schuurmans, J. Zhu, F. Berkenkamp, H. Dong, and A. Bietti, editors, *Forty-third International Conference on Machine Learning, ICML 2026, Seoul, South Korea, July 6-11, 2026*, volume 306 of *Proceedings of Machine Learning Research*, pages 75549–75579. PMLR, 2026. URL https://proceedings.mlr.press/v306/liu26i.html.

[83] Y. Luan, Z. Zhao, Y. Zhang, W. Wang, C. Sun, L. Li, H. Peng, C. Zhang, and L. Xie. SemFed: Semantic federated reinforcement learning for heterogeneous LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 24294–24308. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.findings-emnlp.1147/.

[84] S. Maity and A. Mitra. Learning from unreliable trajectories: Adversarially-robust federated Q-learning. *CoRR*, abs/2610.06918, 2026. URL https://arxiv.org/abs/2610.06918.

[85] S. Maity and A. Mitra. Robust federated Q-learning with almost no communication. In *2026 American Control Conference (ACC)*, pages 462–469, New Orleans, LA, USA, 2026.

[86] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 2017. URL http://proceedings.mlr.press/v54/mcmahan17a.html.

[87] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=BJ0hF1Z0b.

[88] G. Mialon, C. Fourrier, T. Wolf, Y. LeCun, and T. Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

[89] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 739–753. IEEE, 2019. doi: 10.1109/SP.2019.00065. URL https://doi.org/10.1109/SP.2019.00065.

[90] J. Oh, S. Kim, and S. Yun. FedBABU: Toward enhanced representation for federated image classification. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=HuaYQfggn5u.

[91] J. Pan, X. Wang, G. Neubig, N. Jaitly, H. Ji, A. Suhr, and Y. Zhang. Training software engineering agents and verifiers with SWE-Gym. In A. Singh, M. Fazel, D. Hsu, S. Lacoste-Julien, F. Berkenkamp, T. Maharaj, K. Wagstaff, and J. Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*, pages 47717–47737. PMLR, 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

[92] H. Peng, Y. Qi, X. Wang, Z. Yao, B. Xu, L. Hou, and J. Li. Agentic reward modeling: Integrating human preferences with verifiable correctness signals for reliable reward systems. In W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 15934–15949. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.775/.

[93] J. Peng, Y. Liu, R. Zhou, C. Fleming, Z. Wang, A. Garcia, and M. Hong. HiPER: Hierarchical plan–execute RL for multi-turn LLM agents. In T. Zhang, M. Dudik, M. Jaggi, A. Agarwal, S. Li, D. Schuurmans, J. Zhu, F. Berkenkamp, H. Dong, and A. Bietti, editors, *Forty-third International Conference on Machine Learning, ICML 2026, Seoul, South Korea, July 6-11, 2026*, volume 306 of *Proceedings of Machine Learning Research*, pages 98057–98083. PMLR, 2026. URL https://proceedings.mlr.press/v306/peng26h.html.

[94] I. Petrov, D. I. Dimitrov, M. Baader, M. N. Müller, and M. Vechev. DAGER: Exact gradient inversion for large language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/9ff1577a1f8308df1ccea6b4f64a103f-Abstract-Conference.html.

[95] P. Putta, E. Mills, N. Garg, S. Motwani, C. Finn, D. Garg, and R. Rafailov. Agent Q: Advanced reasoning and learning for autonomous AI agents. *arXiv preprint arXiv:2408.07199*, 2024.

[96] J. Qi, Q. Zhou, L. Lei, and K. Zheng. Federated reinforcement learning: Techniques, applications, and open challenges. *Intelligence & Robotics*, 1(1):18–57, 2021. doi: 10.20517/ir.2021.02.

[97] Z. Qi, X. Liu, I. L. Iong, H. Lai, X. Sun, J. Sun, X. Yang, Y. Yang, S. Yao, W. Xu, J. Tang, and Y. Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=oVKEAFjEqv.

[98] Z. Qin, D. Chen, B. Qian, B. Ding, Y. Li, and S. Deng. Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pages 41473–41497. PMLR, 2024. URL https://proceedings.mlr.press/v235/qin24a.html.

[99] Z. Qin, Z. Wu, B. He, and S. Deng. Federated data-efficient instruction tuning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15550–15568, 2025.

[100] C. Rawles, A. Li, D. Rodriguez, O. Riva, and T. Lillicrap. AndroidInTheWild: A large-scale dataset for Android device control. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/bbbb6308b402fe909c39dd29950c32e0-Abstract-Datasets_and_Benchmarks.html.

[101] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan. Adaptive federated optimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, May 3-7, 2021*. OpenReview.net, 2021.

[102] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora. FetchSGD: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*, pages 8253–8265. PMLR, 2020.

[103] L. Sani, A. Iacob, Z. Cao, R. Lee, B. Marino, Y. Gao, D. Cai, Z. Li, W. Zhao, X. Qiu, and N. D. Lane. Photon: Federated LLM pre-training. In M. Zaharia, G. Joshi, and Y. Lin, editors, *Proceedings of the Eighth Conference on Machine Learning and Systems, MLSys 2025, Santa Clara, CA, USA, May 12-15, 2025*. MLSys, 2025. URL https://openreview.net/forum?id=AQgYcfg5EI.

[104] L. Sani, Z. Cao, M. Kurmanji, A. Iacob, A. Jovanović, Y. Gao, W. Zhao, and N. D. Lane. FoMoE: Breaking the full-replica barrier with a federation of MoEs. *CoRR*, abs/2606.19025, 2026. URL https://arxiv.org/abs/2606.19025.

[105] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

[106] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[107] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[108] L. Shen, Z. Tang, L. Wu, Y. Zhang, X. Chu, T. Qin, and B. Han. Hot-pluggable federated learning: Bridging general and personalized FL via dynamic selection. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=B8akWa62Da.

[109] Z. Shen, T. Xu, H. Wang, J. Li, and M. Pan. pFedGPT: Hierarchically optimizing LoRA aggregation weights for personalized federated GPT models. In C. Christodoulopoulos, T. Chakraborty, C. Rose, and V. Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 4766–4778. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.239. URL https://doi.org/10.18653/v1/2025.emnlp-main.239.

[110] T. Shi, S. Chen, B. Jiang, L. Song, L. Yang, and J. Zhao. Experiential reinforcement learning. *CoRR*, abs/2602.13949, 2026. doi: 10.48550/ARXIV.2602.13949. URL https://doi.org/10.48550/arXiv.2602.13949.

[111] Z. Shi, G. Wan, W. Huang, G. Zhang, J. Shao, M. Ye, and C. Yang. Privacy-enhancing paradigms within federated multi-agent systems. *CoRR*, abs/2503.08175, 2025. doi: 10.48550/ARXIV.2503.08175. URL https://doi.org/10.48550/arXiv.2503.08175.

[112] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

[113] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[114] M. Shridhar, X. Yuan, M. Côté, Y. Bisk, A. Trischler, and M. Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

[115] J. Singh, R. Magazine, Y. Pandya, and A. Nambi. Agentic reasoning and tool integration for LLMs via reinforcement learning. *arXiv preprint arXiv:2505.01441*, 2025.

[116] R. Singhal, K. Ponkshe, and P. Vepakomma. FedEx-LoRA: Exact aggregation for federated and efficient fine-tuning of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1316–1336. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.67.

[117] Y. Song, D. Yin, X. Yue, J. Huang, S. Li, and B. Y. Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 7584–7600. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.409. URL https://doi.org/10.18653/v1/2024.acl-long.409.

[118] F. Spadea and O. Seneviratne. Federated fine-tuning of large language models: Kahneman-Tversky vs. direct preference optimization. In *Companion Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025 - 2 May 2025*, pages 1757–1760. ACM, 2025. doi: 10.1145/3701716.3717647. URL https://doi.org/10.1145/3701716.3717647.

[119] M. Srewa, T. Zhao, and S. Elmalaki. PluralLLM: Pluralistic alignment in LLMs via federated learning. In *Proceedings of the 3rd International Workshop on Human-Centered Sensing, Modeling, and Intelligent Systems, HumanSys 2025, Irvine, CA, USA, May 6-9, 2025*, pages 64–69. ACM, 2025. doi: 10.1145/3722570.3726898. URL https://doi.org/10.1145/3722570.3726898.

[120] S. U. Stich. Local SGD converges fast and communicates little. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[121] Y. Sun, Z. Li, Y. Li, and B. Ding. Improving LoRA in privacy-preserving federated learning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=NLPzL6HWNl.

[122] M. Tao, Y. Tian, W. Tu, Y. Yang, X. Yang, and X. Tang. Safe-FedLLM: Delving into the safety of federated large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24405–24420. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.acl-long.1120/.

[123] Z. Wan, Y. Li, X. Wen, Y. Song, H. Wang, L. Yang, M. Schmidt, J. Wang, W. Zhang, S. Hu, and Y. Wen. ReMA: Learning to meta-think for LLMs with multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=ur295YVtmt.

[124] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni. Federated learning with matched averaging. In *8th International Conference on Learning Representations, ICLR 2020, Virtual Event, April 26 - May 1, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=BkluqlSFDS.

[125] H. Wang, S. He, Z. Zhang, F. Miao, and J. Anderson. Momentum for the win: Collaborative federated reinforcement learning across heterogeneous environments. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pages 50530–50560. PMLR, 2024. URL https://proceedings.mlr.press/v235/wang24v.html.

[126] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. In H. Larochelle, M. Ranzato, R. Hadsell,

M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/564127c03caab942e503ee6f810f54fd-Abstract.html.

[127] K. Wang, P. Zhang, Z. Wang, Y. Gao, L. Li, Q. Wang, H. Chen, C. Wan, Y. Lu, Z. Yang, L. Wang, R. Krishna, J. Wu, L. Fei-Fei, Y. Choi, and M. Li. VAGEN: Reinforcing world model reasoning for multi-turn VLM agents. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=xpjWEgf8zi.

[128] L. Wang, J. Bian, L. Zhang, and J. Xu. Adaptive LoRA experts allocation and selection for federated fine-tuning. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=es4TTVGJ9x.

[129] L. Wang, J. Bian, L. Zhang, and J. Xu. Breaking the structural identity: Personalized federated LoRA fine-tuning under rank heterogeneity. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 12816–12832. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.findings-emnlp.608/.

[130] P. Wang, L. Li, Z. Shao, R. Xu, D. Dai, Y. Li, D. Chen, Y. Wu, and Z. Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 9426–9439. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.510. URL https://doi.org/10.18653/v1/2024.acl-long.510.

[131] R. Wang and P. Ammanabrolu. A practitioner's guide to multi-turn agentic reinforcement learning. *arXiv preprint arXiv:2510.01132*, 2025.

[132] R. Wang, C. Huang, Z. Wang, J. Wu, R. Wang, T. Yu, J. McAuley, L. Yao, and D. Zhou. FERA: Uncertainty-aware federated reasoning for large language models. In *Third Conference on Language Modeling, COLM 2026*, 2026. URL https://openreview.net/forum?id=EZ2O9py7RO.

[133] W. Wang, H. Shi, M. Yuan, Y. Lin, P. Tong, H. Zhou, G. Liu, P. Zhao, Y. Wang, and S. Chen. FedGUI: Benchmarking federated GUI agents across heterogeneous platforms, devices, and operating systems. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 28747–28767. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.findings-acl.1436/.

[134] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb. Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching. *IEEE Internet Things J.*, 7(10):9441–9455, 2020. doi: 10.1109/JIOT.2020.2986803. URL https://doi.org/10.1109/JIOT.2020.2986803.

[135] Z. Wang, Z. Shen, Y. He, G. Sun, H. Wang, L. Lyu, and A. Li. FLoRA: Federated fine-tuning large language models with heterogeneous low-rank adaptations. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/28312c9491d60ed0c77f7fff4ad86dd1-Abstract-Conference.html.

[136] Z. Wang, K. Wang, Q. Wang, P. Zhang, L. Li, Z. Yang, X. Jin, K. Yu, M. N. Nguyen, L. Liu, et al. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

[137] Z. Wang, C. Xu, B. Liu, Y. Wang, S. Han, Z. Yao, H. Yao, and Y. He. Agent World Model: Infinity synthetic environments for agentic reinforcement learning. In T. Zhang, M. Dudik,

M. Jaggi, A. Agarwal, S. Li, D. Schuurmans, J. Zhu, F. Berkenkamp, H. Dong, and A. Bietti, editors, *Forty-third International Conference on Machine Learning, ICML 2026, Seoul, South Korea, July 6-11, 2026*, volume 306 of *Proceedings of Machine Learning Research*, pages 131469–131511. PMLR, 2026. URL https://proceedings.mlr.press/v306/wang26jh.html.

[138] M. Waseem, N. Tastan, A. Jovanovic, N. D. Lane, N. Lukas, K. Nandakumar, and S. Horváth. PreLort: Prefix-nested LoRA for federated fine-tuning under rank heterogeneity. *CoRR*, abs/2606.15963, 2026. URL https://arxiv.org/abs/2606.15963.

[139] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837, 2022.

[140] Q. Wei, S. Zeng, C. Li, Z. Wang, W. Brown, O. Frunza, W. Deng, A. Schneider, Y. Nevmyvaka, Y. K. Zhao, A. Garcia, and M. Hong. Reinforcing multi-turn reasoning in LLM agents via fine-grained reward structure and credit assignment. *arXiv preprint arXiv:2505.11821*, 2025.

[141] S. Wei, Y. Tong, Z. Zhou, Y. Xu, J. Gao, T. Wei, T. He, and W. Lv. Federated reasoning LLMs: A survey. *Front. Comput. Sci.*, 19(12):1912613, 2025. doi: 10.1007/S11704-025-50480-3. URL https://doi.org/10.1007/s11704-025-50480-3.

[142] Y. Wei, O. Duchenne, J. Copet, Q. Carbonneaux, L. Zhang, D. Fried, G. Synnaeve, R. Singh, and S. I. Wang. SWE-RL: Advancing LLM reasoning via reinforcement learning on open software evolution. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=ULblO61XZ0.

[143] Y. Wei, Z. Sun, E. McMilin, J. Gehring, D. Zhang, G. Synnaeve, D. Fried, L. Zhang, and S. Wang. Toward training superintelligent software agents through self-play SWE-RL. In T. Zhang, M. Dudik, M. Jaggi, A. Agarwal, S. Li, D. Schuurmans, J. Zhu, F. Berkenkamp, H. Dong, and A. Bietti, editors, *Forty-third International Conference on Machine Learning, ICML 2026, Seoul, South Korea, July 6-11, 2026*, volume 306 of *Proceedings of Machine Learning Research*, pages 134075–134095. PMLR, 2026. URL https://proceedings.mlr.press/v306/wei26x.html.

[144] Z. Wei, W. Yao, Y. Liu, W. Zhang, Q. Lu, L. Qiu, C. Yu, P. Xu, C. Zhang, B. Yin, H. Yun, and L. Li. WebAgent-R1: Training web agents via end-to-end multi-turn reinforcement learning. In C. Christodoulopoulos, T. Chakraborty, C. Rose, and V. Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 7909–7928. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.401. URL https://doi.org/10.18653/v1/2025.emnlp-main.401.

[145] J. Woo, G. Joshi, and Y. Chi. The blessing of heterogeneity in federated Q-learning: Linear speedup and beyond. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 37157–37216. PMLR, 2023. URL https://proceedings.mlr.press/v202/woo23a.html.

[146] B. Woodworth, K. K. Patel, and N. Srebro. Minibatch vs local SGD for heterogeneous distributed learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 6281–6292, 2020.

[147] F. Wu, Z. Li, Y. Li, B. Ding, and J. Gao. FedBiOT: LLM local fine-tuning in federated learning without full model. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 3345–3355. ACM, 2024. doi: 10.1145/3637528.3671897. URL https://doi.org/10.1145/3637528.3671897.

[148] F. Wu, X. Liu, H. Wang, X. Wang, L. Su, and J. Gao. Towards federated RLHF with aggregated client preference for LLMs. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=mqNKiEB6pd.

[149] P. Wu, K. Li, J. Nan, and F. Wang. Federated in-context LLM agent learning. *CoRR*, abs/2412.08054, 2024. doi: 10.48550/ARXIV.2412.08054. URL https://doi.org/10.48550/arXiv.2412.08054.

[150] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling, COLM 2024*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

[151] Y. Wu, C. Tian, J. Li, H. Sun, K. Tam, Z. Zhou, H. Liao, J. Xiong, Z. Guo, L. Li, and C. Xu. A survey on federated fine-tuning of large language models. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=rnCqbuIWnn.

[152] Z. Xi, Y. Ding, W. Chen, B. Hong, H. Guo, J. Wang, X. Guo, D. Yang, C. Liao, W. He, S. Gao, L. Chen, R. Zheng, Y. Zou, T. Gui, Q. Zhang, X. Qiu, X. Huang, Z. Wu, and Y. Jiang. AgentGym: Evaluating and training large language model-based agents across diverse environments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27914–27961. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1355.

[153] Z. Xi, J. Huang, C. Liao, B. Huang, H. Guo, J. Liu, R. Zheng, J. Ye, J. Zhang, W. Chen, et al. AgentGym-RL: Training LLM agents for long-horizon decision making through multi-turn reinforcement learning. *arXiv preprint arXiv:2509.08755*, 2025.

[154] P. Xia, J. Chen, H. Wang, J. Liu, K. Zeng, Y. Wang, S. Han, Y. Zhou, X. Zhao, H. Chen, Z. Zheng, C. Xie, and H. Yao. SkillRL: Evolving agents via recursive skill-augmented reinforcement learning. *CoRR*, abs/2602.08234, 2026. doi: 10.48550/ARXIV.2602.08234. URL https://doi.org/10.48550/arXiv.2602.08234.

[155] Z. Xiao, J. Tu, C. Zou, Y. Zuo, Z. Li, P. Wang, B. Yu, F. Huang, J. Lin, and Z. Liu. WebWorld: A large-scale world model for web agent training. In T. Zhang, M. Dudik, M. Jaggi, A. Agarwal, S. Li, D. Schuurmans, J. Zhu, F. Berkenkamp, H. Dong, and A. Bietti, editors, *Forty-third International Conference on Machine Learning, ICML 2026, Seoul, South Korea, July 6-11, 2026*, volume 306 of *Proceedings of Machine Learning Research*, pages 138952–138977. PMLR, 2026. URL https://proceedings.mlr.press/v306/xiao26o.html.

[156] B. Xiong, X. Yang, Y. Song, Y. Wang, and C. Xu. Pilot: Building the federated multimodal instruction tuning framework. In *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 21716–21724. Association for the Advancement of Artificial Intelligence (AAAI), 2025. doi: 10.1609/AAAI.V39I20.35476. URL https://doi.org/10.1609/aaai.v39i20.35476.

[157] G. Xiong, S. Wang, D. Jiang, and J. Li. On the linear speedup of personalized federated reinforcement learning with shared representations. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=BfUDZGqCAu.

[158] F. F. Xu, Y. Song, B. Li, Y. Tang, K. Jain, M. Bao, Z. Z. Wang, X. Zhou, Z. Guo, M. Cao, M. Yang, H. Y. Lu, A. Martin, Z. Su, L. M. Maben, R. Mehta, W. Chi, L. Jang, Y. Xie, S. Zhou, and G. Neubig. TheAgentCompany: Benchmarking LLM agents on consequential real world tasks. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=LZnKNApvhG.

[159] Y. Yan, C. Feng, W. Zuo, R. S. M. Goh, Y. Liu, and L. Zhu. Federated residual low-rank adaptation of large language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=e0rQRMUhs7.

[160] J. Yang, G. Yao, Y. Zhang, R. R. Kompella, G. Liu, and S. Chang. FederatedSkill: Federated learning for agentic skill evolution. *CoRR*, abs/2606.03143, 2026. URL https://arxiv.org/abs/2606.03143.

[161] Q. Yang, Z. Kou, X. Zhang, D. Wu, H. Gu, J. Guo, Y. Liu, D. Jiang, and Q. Xu. Federated agent optimization. *CoRR*, abs/2610.01195, 2026. URL https://arxiv.org/abs/2610.01195.

[162] T. Yang, S. Cen, Y. Wei, Y. Chen, and Y. Chi. Federated natural policy gradient and actor critic methods for multi-task reinforcement learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/dbdea7859f1d2fc10f2c9e79b8f5ae54-Abstract-Conference.html.

[163] Y. Yang, G. Long, Q. Lu, L. Zhu, J. Jiang, and C. Zhang. Federated low-rank adaptation for foundation models: A survey. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pages 10779–10787. International Joint Conferences on Artificial Intelligence Organization, 2025. doi: 10.24963/IJCAI.2025/1196. URL https://doi.org/10.24963/ijcai.2025/1196.

[164] D. Yao, T. Rabbani, and T. Li. Federation over harness. *CoRR*, abs/2610.24007, 2026. URL https://arxiv.org/abs/2610.24007.

[165] S. Yao, H. Chen, J. Yang, and K. Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

[166] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of Thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html.

[167] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[168] M. Ye, X. Fang, B. Du, P. C. Yuen, and D. Tao. Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Comput. Surv.*, 56(3):1–44, 2024. doi: 10.1145/3625558. URL https://doi.org/10.1145/3625558.

[169] R. Ye, W. Wang, J. Chai, D. Li, Z. Li, Y. Xu, Y. Du, Y. Wang, and S. Chen. OpenFedLLM: Training large language models on decentralized private data via federated learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 6137–6147. ACM, 2024. doi: 10.1145/3637528.3671582. URL https://doi.org/10.1145/3637528.3671582.

[170] X. Yi, Y. Liu, B. Yang, and J. Zhang. Federated continual learning via orchestrating multi-scale expertise. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=AlSHcopwHi.

[171] Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, W. Dai, T. Fan, G. Liu, J. Liu, L. Liu, X. Liu, H. Lin, Z. Lin, B. Ma, G. Sheng, Y. Tong, C. Zhang, M. Zhang, R. Zhang, W. Zhang, H. Zhu, J. Zhu, J. Chen, J. Chen, C. Wang, H. Yu, Y. Song, X. Wei, H. Zhou, J. Liu, W. Ma, Y. Zhang, L. Yan, Y. Wu, and M. Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

[172] X. Yu, B. Peng, R. Xu, M. Galley, H. Cheng, S. Nath, J. Gao, and Z. Yu. Dyna-Think: Synergizing reasoning, acting, and world model simulation in AI agents. *CoRR*, abs/2506.00320, 2025. doi: 10.48550/ARXIV.2506.00320. URL https://doi.org/10.48550/arXiv.2506.00320.

[173] X. Yu, B. Peng, R. Xu, Y. Shen, P. He, S. Nath, N. Singh, J. Gao, and Z. Yu. Reinforcement world model learning for LLM-based agents. *CoRR*, abs/2602.05842, 2026. doi: 10.48550/ARXIV.2602.05842. URL https://doi.org/10.48550/arXiv.2602.05842.

[174] A. Zeng, M. Liu, R. Lu, B. Wang, X. Liu, Y. Dong, and J. Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077, 2024.

[175] G. Zhang, H. Geng, X. Yu, Z. Yin, Z. Zhang, Z. Tan, H. Zhou, Z. Li, X. Xue, Y. Li, Y. Zhou, Y. Chen, C. Zhang, Y. Fan, Z. Wang, S. Huang, F. Piedrahita-Velez, Y. Liao, H. Wang, M. Yang, H. Ji, J. Wang, S. Yan, P. Torr, and L. Bai. The landscape of agentic reinforcement learning for LLMs: A survey. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=RY19y2RI1O.

[176] H. Zhang, X. Liu, B. Lv, X. Sun, B. Jing, I. L. Iong, Z. Hou, Z. Qi, H. Lai, Y. Xu, R. Lu, H. Wang, J. Tang, and Y. Dong. AgentRL: Scaling agentic reinforcement learning with a multi-turn, multi-task framework. *CoRR*, abs/2510.04206, 2025. doi: 10.48550/ARXIV.2510.04206. URL https://doi.org/10.48550/arXiv.2510.04206.

[177] J. Zhang, S. Vahidian, M. Kuo, C. Li, R. Zhang, T. Yu, G. Wang, and Y. Chen. Towards building the FederatedGPT: Federated instruction tuning. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2024, Seoul, Republic of Korea, April 14-19, 2024*, pages 6915–6919. IEEE, 2024. doi: 10.1109/ICASSP48485.2024.10447454. URL https://doi.org/10.1109/ICASSP48485.2024.10447454.

[178] K. Zhang, X. Chen, B. Liu, T. Xue, Z. Liao, Z. Liu, X. Wang, Y. Ning, Z. Chen, X. Fu, J. Xie, Y. Sun, B. Gou, Q. Qi, Z. Meng, J. Yang, N. Zhang, X. Li, A. Shah, D. Huynh, H. Li, Z. Yang, S. Cao, L. Jang, S. Zhou, J. Zhu, H. Sun, J. Weston, Y. Su, and Y. Wu. Agent learning via early experience. In T. Zhang, M. Dudik, M. Jaggi, A. Agarwal, S. Li, D. Schuurmans, J. Zhu, F. Berkenkamp, H. Dong, and A. Bietti, editors, *Forty-third International Conference on Machine Learning, ICML 2026, Seoul, South Korea, July 6-11, 2026*, volume 306 of *Proceedings of Machine Learning Research*, pages 154443–154475. PMLR, 2026. URL https://proceedings.mlr.press/v306/zhang26m.html.

[179] K. Zhang, K. Tian, R. Liu, S. Zeng, X. Zhu, G. Jia, Y. Fan, X. Lv, Y. Zuo, C. Jiang, Y. Wang, J. Wang, E. Hua, X. Long, J. Gao, Y. Sun, Z. Ma, G. Cui, N. Ding, B. Qi, and B. Zhou. MARTI: A framework for multi-agent LLM systems reinforced training and inference. In *The Fourteenth International Conference on Learning Representations, ICLR 2026*, 2026. URL https://openreview.net/forum?id=E7jZqo0A50.

[180] Y. Zhang, Z. Qin, Z. Wu, J. Hou, and S. Deng. Personalized federated fine-tuning for LLMs via data-driven heterogeneous model architectures. In *Proceedings of the ACM Web Conference 2026*, pages 5099–5110. ACM, 2026. doi: 10.1145/3774904.3792147.

[181] Z. Zhang, J. Zhang, J. Huang, H. Wang, Y. Yu, H. Zhang, X. Zhou, L. Qu, and Z. Xu. PPFedIT: Towards privacy-preserving federated instruction tuning with few-shot local examples. *ACM Transactions on Intelligent Systems and Technology*, 17(5), 2026. doi: 10.1145/3806196.

[182] B. Zhao, K. R. Mopuri, and H. Bilen. iDLG: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[183] J. Zhao, H. Xie, Y. Lei, X. Song, Z. Shi, L. Li, S. Liu, and H. Zhang. Weaving in the clouds: Achieving synergistic collaboration among LLM agents via federated learning. In T. Zhang, M. Dudik, M. Jaggi, A. Agarwal, S. Li, D. Schuurmans, J. Zhu, F. Berkenkamp, H. Dong, and A. Bietti, editors, *Forty-third International Conference on Machine Learning, ICML 2026, Seoul, South Korea, July 6-11, 2026*, volume 306 of *Proceedings of Machine Learning Research*, pages 162414–162432. PMLR, 2026. URL https://proceedings.mlr.press/v306/zhao26bd.html.

[184] Z. Zheng, F. Gao, L. Xue, and J. Yang. Federated Q-learning: Linear regret speedup with low communication cost. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=fe6ANBxcKM.

[185] Z. Zheng, H. Zhang, and L. Xue. Federated Q-learning with reference-advantage decomposition: Almost optimal regret and logarithmic communication cost. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=FoUpv84hMw.

[186] A. Zhou, K. Yan, M. Shlapentokh-Rothman, H. Wang, and Y. Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pages 62138–62160. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhou24r.html.

[187] D. Zhou, Y. Zhang, A. Sonabend-W, Z. Wang, J. Lu, and T. Cai. Federated offline reinforcement learning. *Journal of the American Statistical Association*, 119(548):3152–3163, 2024. doi: 10.1080/01621459.2024.2310287.

[188] S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, T. Ou, Y. Bisk, D. Fried, U. Alon, and G. Neubig. WebArena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

[189] Y. Zhou, A. Zanette, J. Pan, S. Levine, and A. Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pages 62178–62209. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhou24t.html.

[190] Y. Zhou, S. Jiang, Y. Tian, J. Weston, S. Levine, S. Sukhbaatar, and X. Li. SWEET-RL: Training multi-turn LLM agents on collaborative reasoning tasks. *arXiv preprint arXiv:2503.15478*, 2025.

[191] K. Zhu, H. Du, Z. Hong, X. Yang, S. Guo, Z. Wang, Z. Wang, C. Qian, X. Tang, H. Ji, and J. You. MultiAgentBench : Evaluating the collaboration and competition of LLM agents. In W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 8580–8622. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.421. URL https://doi.org/10.18653/v1/2025.acl-long.421.

[192] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[193] Y. Zhu and X. Gong. Single-loop federated actor-critic across heterogeneous environments. In *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 23054–23062. Association for the Advancement of Artificial Intelligence (AAAI), 2025. doi: 10.1609/AAAI.V39I21.34469. URL https://doi.org/10.1609/aaai.v39i21.34469.

[194] H. H. Zhuo, W. Feng, Y. Lin, Q. Xu, and Q. Yang. Federated deep reinforcement learning. *CoRR*, abs/1901.08277, 2019. URL http://arxiv.org/abs/1901.08277.

Appendix

A Author Contributions **32**

B Notation **33**

C Reproducibility Statement **35**

D Broader Impact **36**

E Limitations **38**

F Extended Discussion of Main-Text Sections **39**

F.1 Task-Level Heterogeneity: Sub-Types and Measurement 39

F.2 Partition Algorithms 40

F.3 Environment-Level Heterogeneity: Construction on WebShop 40

F.4 Asymmetric Robustness Mechanism: Extended Discussion of the Theorems 41

F.5 Formal Statement of Condition (C3) and Discussion of the Recovery Theorem . . . 43

F.6 Four Empirical Patterns: Full Statements 44

F.7 Empirical Results: Extended Discussion 45

G More Experimental Results **47**

G.1 Hyperparameter Sensitivity of FEDAGENT's Federated Configuration 47

G.2 PPO Results: Effectiveness under Uniform Client Distribution 49

G.3 PPO Results: Pattern A under Task-Level Heterogeneity 49

H Extended Related Work **52**

H.1 Supervised Federated Learning 52

H.2 Federated Reinforcement Learning 53

H.3 Federated Learning for LLMs 54

H.4 LLM Agent Reinforcement Learning 56

I FEDAGENT Algorithm **60**

I.1 Line-by-Line Description 60

I.2 Intrinsic Privacy Properties 61

I.3 Communication Cost and Deployment Considerations 62

J Experimental Setup Details **64**

J.1 Hardware and Software 64

J.2 Models and Tokenization 64

J.3 Federation Protocol 64

J.4 Policy Optimization Hyperparameters . . . 65
J.5 Rollout Protocol . . . 66
J.6 Heterogeneity Construction Protocol . . . 67
J.7 Evaluation Protocol . . . 68
J.8 Reproducibility Checklist . . . 71

K Task-Level Heterogeneity Characterization **72**
K.1 Inter-Client Dispersion Measures . . . 72
K.2 Primary Properties of the Three Projections . . . 73
K.3 Definitional Orthogonality and First-Order Sufficiency . . . 73

L Task-Level Partition Algorithms **75**
L.1 Pseudocode . . . 75
L.1.1 PREFERENCEPARTITION . . . 75
L.1.2 COVERAGEPARTITION . . . 76
L.1.3 HARDNESSPARTITION . . . 76
L.2 Design Properties (D1)–(D4) . . . 76
L.2.1 Single-Hyperparameter Monotone Control (D1) . . . 77
L.2.2 Cross-Measure Isolation (D2) . . . 78
L.2.3 Factor Invariance (D3) . . . 79
L.2.4 Joint Configurability (D4) . . . 80
L.3 Per-Client Distribution Figures . . . 81
L.3.1 Preference Heterogeneity . . . 81
L.3.2 Coverage Heterogeneity . . . 82
L.3.3 Hardness Heterogeneity . . . 83

M Environment-Level Heterogeneity Variants **84**
M.1 A Four-Stage Transition Pipeline . . . 84
M.2 The WebShop Pipeline as a Concrete Instantiation . . . 85
M.3 Deterministic Per-Client Environment Assignment . . . 85
M.4 Variant 1: Catalog Split (Stage 1) . . . 87
M.5 Variant 2: Field-Subset Index (Stage 2) . . . 88
M.6 Variant 3: BM25 Reweighting (Stage 3) . . . 90
M.7 Variant 4: Lookalike Injection (Stages 1+3) . . . 91
M.8 Variant 5: Rank Wrapper (Stage 4) . . . 92
M.9 Measuring the Induced Environment-Level Heterogeneity . . . 93

N Asymmetric Robustness Mechanism: Theory and Proofs **97**
N.1 Realizability Assumptions for LLM Agents . . . 97
N.2 Proofs of Theorems 1 and 2 (Task-Level Robustness) . . . 98
N.3 Proofs of Theorems 3 and 4 (Environment-Level Non-Robustness) . . . 100

N.4 Proof of Theorem 5 (Recovery via (C1)–(C3)) . . . 104
N.5 Gradient Heterogeneity ζ^2 at the Federated Solution . . . 107

O Empirical Patterns: Extended Discussion **109**
O.1 The Unified Causal Chain . . . 109
O.1.1 The Single Architectural Fact: Input-Dynamics Asymmetry . . . 109
O.1.2 The Two Sides of Asymmetry . . . 110
O.1.3 The Full Causal Chain . . . 110
O.2 Pattern A: Plateau Identity vs. Trajectory Drift . . . 111
O.3 Pattern B vs. Pattern A: Same Plateau, Different Mechanism . . . 112
O.4 The B/C/D Spectrum and Its Slack Parameters . . . 112
O.5 Pattern D Mechanism: Decomposing Oscillation and Forgetting . . . 113
O.6 ζ^2 Connects the Mechanism to Optimization: Rate vs. Quality . . . 114
O.7 Diagnostic Decision Tree . . . 114
O.8 Operational Mitigation Protocol . . . 115
O.8.1 Pre-Deployment Tests for (C1), (C2), (C3) . . . 116
O.8.2 The Quantitative Policy-Disagreement Gap Bound . . . 117
O.8.3 Mitigations Indexed by the Dominant Slack . . . 118
O.8.4 The Uncertain Regime: When All (C) Tests Fail but Collapse Does Not Occur 119
O.9 Predictive and Falsifiable Use of the Mechanism . . . 119
O.9.1 Connecting Theory and Experiment . . . 120
O.9.2 Falsifiability per Pattern . . . 120
O.9.3 Predictive Use Given Deployment Characteristics . . . 121

P Why the Mechanism Is Sharper on LLM Agents **123**
P.1 Task-as-Input: Pre-Training as Implicit Realizability . . . 123
P.2 Environment-as-Implicit-Knowledge: Pre-Training Prior Degradation (Informal Claim) 124
P.3 (C3) Explicit Form: Environment-Aware Prompting and Corollary 1 . . . 125
P.4 (C3) Implicit Form: In-Context Posterior Inference . . . 127

A Author Contributions

Ideation. Canyu Chen contributed all of the core ideas and concepts of this work, including (i) the FEDAGENT paradigm of decentralized (federated) agent reinforcement learning and the central question of whether it is robust to client heterogeneity; (ii) the formalization of Agent Heterogeneity into two structurally distinct levels, task-level and environment-level, together with the three task-level sub-types (preference, coverage, and hardness heterogeneity); (iii) the Input-Dynamics Asymmetry that grounds this distinction; and (iv) the Asymmetric Robustness Mechanism as the central thesis, namely robustness to task-level but worst-case non-robustness to environment-level heterogeneity. Kangyu Zhu was heavily involved in all of the discussions, and all authors provided feedback.

Theory. Canyu Chen contributed all of the theoretical development, including the formalization of the federated agent RL objective, the Asymmetric Robustness Mechanism, the main upper and lower bounds, the three sufficient conditions under which FEDAGENT recovers robustness despite environment-level heterogeneity, and the four predicted empirical patterns. Tian Li and Yiping Lu provided suggestions, feedback, and proof-checking.

Writing. Canyu Chen wrote the manuscript, with suggestions and feedback from all authors.

Coding & Infrastructure. Canyu Chen and Kangyu Zhu were the core contributors to the FEDAGENT library for federated RL training of LLM agents. The library implements a federated training server with FedAvg aggregation (plus an optional client-side FedProx term), federated PPO and GRPO trainers, the two-level heterogeneity suite (task-level partitions over preference, coverage, and hardness, and five environment-level WebShop transition variants), and a fully configurable federation protocol (N clients, M clients per round, E local epochs, T rounds, $|X_i|$ tasks per client), with support for the WebShop and ALFWorld benchmarks under FSDP sharding from single-GPU to multi-node setups.

Experiments & Analysis. Canyu Chen and Kangyu Zhu conducted the majority of the experiments and led the empirical analysis, with additional experimental help from Zhaorun Chen and Shizhe Diao.

Supervision. Dawn Song and Manling Li supervised the entire project from start to finish and were heavily involved in all of the discussions.

Funding & Compute Support. Dawn Song and Manling Li provided the funding and computational resources for this work.

Acknowledgments. We additionally thank Jie Zhang (ETH), Chulin Xie (UIUC), and Wenxuan Bao (UIUC) for insightful discussions and feedback.

B Notation

Table 2 summarizes the symbols used throughout the paper.

Symbol	Description
Federation setup	
N	total number of clients
M	clients selected per round
T	total communication rounds
E	local epochs per round (one local epoch is one batch of 64 rollout trajectories)
G	GRPO group size (rollouts per task)
$\lvert X_i\rvert$	per-client task pool size (number of distinct tasks the client holds)
θ	policy parameter (LLM weights)
θ_t	global parameter at round t
$\theta_{i,t,e}$	client i's parameter at round t, local epoch e
S_t	subset of clients selected at round t, $\lvert S_t\rvert = M$
η_{lr}	local learning rate (Algorithm 1)
ϕ	PPO critic parameters, averaged alongside θ
Task-augmented MDP	
$\mathcal{M}_i$	client i's task-augmented MDP $(\mathcal{T}, \mathcal{S}_i, \mathcal{A}_i, P_i, R_i, \gamma, \mu_0)$
$\mathcal{M}_{\text{env}}$	shared environment under task-level heterogeneity
$\mathcal{T}$	task descriptor space
$\mathcal{S}_i, \mathcal{A}_i$	state and action spaces of $\mathcal{M}_i$ (under Definition 2, $\mathcal{S}_i \subseteq \mathcal{S}$ is the client's reachable subset)
P_i	client i's transition kernel $P_i(s' \mid s, a)$ (independent of the task)
R_i or R	task-conditional reward $R_i(s, a, s'; \tau)$ (shared under Definitions 1–2)
γ	discount factor
H	episode horizon (max steps when $\gamma = 1$; $1/(1-\gamma)$ when $\gamma < 1$)
$R_{\max}$	per-step reward bound, $r_u \in [0, R_{\max}]$
$\mu_0(\cdot \mid \tau)$	task-conditional initial-state distribution
τ	task descriptor (prompt input)
s_u, a_u, r_u	state, action, reward at step u (communication rounds are indexed by t)
h_u	trajectory prefix (history) $(s_0, a_0, r_0, \ldots, s_u)$; h_H is a full trajectory
$\mathcal{H}$	history (trajectory-prefix) space, $h_u \in \mathcal{H}$
Distributions and policies	
$\mathcal{D}_{\tau_i}$	per-client task distribution (uniform on the pool X_i)
$\mathcal{D}_\tau$	shared task distribution under environment-level heterogeneity (Definition 2)
$\bar{\mathcal{D}}_\tau := \frac{1}{N}\sum_i \mathcal{D}_{\tau_i}$	federated mixture task distribution
$\pi_\theta(a \mid s, \tau)$	LLM-induced policy
Π_Θ	parameterized policy class $\{\pi_\theta : \theta \in \Theta\}$
Π_Θ^{aug}	history-augmented policy class $\{\pi_\theta(a \mid s, \tau, h_u)\}$
$\pi^\star$	shared optimal policy (when one exists)
$\pi_i^\star$	per-client optimal policy in $\mathcal{M}_i$
$b(\tau)$	task-conditional optimal behavior on task τ
$Q_i^\star$	per-client optimal Q-function $Q_i^\star(s, \tau, a)$; τ omitted when the task is shared
$V_i^\star,\ V_{\max}$	per-client optimal value function (step index suppressed); $V_{\max} := R_{\max} H$
$d^\pi_{\mathcal{M}_i}$	normalized state-action occupancy of π in $\mathcal{M}_i$
Objectives	
$\mathcal{J}(\pi; \tau, \mathcal{M})$	per-task expected return of π on task τ in MDP $\mathcal{M}$
$\mathcal{J}_\tau(\theta),\ \mathcal{J}_\tau^\star$	$\mathcal{J}(\pi_\theta; \tau, \mathcal{M}_{\text{env}})$ and its optimum over all policies (task-level proofs)
$\mathcal{J}_i(\theta)$	per-client expected return $\mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}} \mathbb{E}_{h_H}[\sum_u \gamma^u r_u]$
$\mathcal{J}_{\text{fed}}(\theta) = \frac{1}{N}\sum_i \mathcal{J}_i(\theta)$	federated objective
$\theta^\star$	realizing parameter under (R) or (R′)
$\theta^\star_{\text{fed}}$	federated optimum, $\theta^\star_{\text{fed}} \in \arg\max_\theta \mathcal{J}_{\text{fed}}(\theta)$
$\hat{\theta}_{\text{fed}}$	ϵ_{opt}-approximate federated maximizer
Heterogeneity measures (task-level)	
Δ^2_{pref}	preference dispersion, $\frac{1}{N}\sum_i \lVert p_i - \bar{p}\rVert_2^2$
Δ^2_{cov}	coverage dispersion, $\widehat{\text{CV}}^2(\{n_i\})$
Δ^2_{hard}	hardness dispersion, $\widehat{\text{Var}}_i(\rho_i)$

continued on next page

(Table 2 continued)

Symbol	Description
C	number of task categories (types)
$\ell : \mathcal{T} \to [C]$	category map; ℓ_* its pushforward, $p_i = \ell_* \mathcal{D}_{\tau_i}$
p_i	client i's type marginal on $[C]$
p_0	global (population) type marginal on $[C]$
n_i	client i's local pool size $\lvert X_i \rvert$ (the support size of $\mathcal{D}_{\tau_i}$)
$L;\ L_{\min}, L_{\text{avg}}, L_{\max}$	fixed per-client pool size; bounds and mean of n_i under CoveragePartition
r	global overlap: average number of clients holding each task (distinct from the reward r_u)
π_{ref}	reference checkpoint used to pre-label task hardness
$\rho(\tau)$	success rate of π_{ref} on task τ
η	hardness success threshold, $\eta \in (0, 1)$
ρ_i	client i's success quota (fraction of its tasks with $\rho(\tau) \geq \eta$), $\Pr_{\tau \sim \mathcal{D}_{\tau_i}}[\rho(\tau) \geq \eta]$
$\bar{p}, \bar{n}, \bar{\rho}$	client averages of p_i, n_i, ρ_i ($\bar{p}$ is the sample counterpart of p_0)
ω, ξ, ξ'	partition hyperparameters (preference, coverage, hardness)
Heterogeneity measures (environment-level)	
δ	worst-case environment divergence, $\sup_{i \neq j, (s,a)} D_{\text{TV}}(P_i, P_j)$
$\delta_{\text{eff}}(\pi; i, j)$	occupancy-weighted (effective) divergence between $\mathcal{M}_i, \mathcal{M}_j$
D_{TV}	total variation distance
$\mathcal{V}$	configuration pool of one variant, containing $\lvert \mathcal{V} \rvert$ configurations (Appendix M)
`env_div`	Catalog Split diversity hyperparameter (how differently clients filter distractors)
`keep_ratio`	Catalog Split retention-fraction hyperparameter
Theory slacks	
ϵ_{approx}	capacity/realizability slack on the policy class
ϵ_0	suboptimality of the initialization θ_0 on the mixture, an upper bound on ϵ_{approx}
ϵ_{opt}	federated optimization slack of $\hat{\theta}_{\text{fed}}$
ϵ_{LR}	realizability slack of (LR) in the lower-bound constructions
$\chi^2(P \,\|\, Q)$	chi-squared divergence between P and Q
$\zeta^2(\theta)$	cross-client gradient variance at θ; $\bar{\zeta}^2$ denotes its uniform bound
L_{sm}	smoothness constant of the objectives (Appendix N.5)
Δ_{pol}	policy-disagreement gap, $\frac{1}{N}\sum_i \sup_{\theta_i} \mathcal{J}_i(\theta_i) - \sup_\theta \frac{1}{N}\sum_i \mathcal{J}_i(\theta)$
$\widehat{(\cdot)}$	empirical estimate of the corresponding population quantity
Self-revealing environment (C3) parameters	
$u^\star$	environment-reveal step (integer), after which the classifier is reliable
π_0	default prefix policy generating $h_{u^\star}$ (followed for $u < u^\star$)
$\hat{\imath}$	environment-identity classifier $\hat{\imath} : \mathcal{H} \to [N]$
$\hat{P}_i$	kernel model of client i's environment, used by the post-reveal policy
η_{cls}	classifier slack, post-reveal misclassification rate
η_{ker}	kernel-modeling slack, $\max_i \sup_{s,a} D_{\text{TV}}(\hat{P}_i, P_i)$
c_0	absolute constant in the recoverability condition (C3-rec)
$e_i,\ \hat{e},\ \hat{P}_e$	environment descriptor, its classifier, and kernel model (Corollary 1)

Table 2: This table lists the notation used throughout the paper. Symbols local to the partition-construction appendices (*e.g.* N_{pool}, $\mathcal{Y}, \mathcal{U}$, $\kappa_{\min}, \kappa_{\text{avg}}, \kappa_{\max}$, μ, μ', q_i) are defined where they first appear in Appendices K–L. The construction constants C_n, C_h are defined at the start of Appendix L.2.1 and used in Propositions 2 and 3.

C Reproducibility Statement

We have made every effort to make FEDAGENT fully reproducible. The following paragraphs identify the appendix supporting each contribution.

Code and data release. The full implementation of FEDAGENT, dataset preprocessing scripts for WebShop and ALFWorld, and all per-experiment configuration files are available in a public repository at `https://github.com/sunblaze-ucb/FedAgent`. Its `paper-reproduce-verl-agent` branch holds the code of the reported runs, built on HuggingFace `transformers` and the `verl-agent` framework [31], and its `main` branch re-implements the same protocol on `verl` 0.8 (Appendix J). The repository does not include large benchmark data, trained checkpoints, or training logs. Instead, it provides setup, data-acquisition, training, and evaluation scripts that allow the reported curves and tables to be regenerated from the released configurations.

Theoretical claims. The five realizability assumptions (R), (R′), (R_{env}), (R_{aug}), and (LR) are formally stated in Appendix N.1. Theorem 1 and Theorem 2 (task-level robustness, idealized and approximate) are proved in Appendix N.2. Theorem 3 and Theorem 4 (environment-level non-robustness, existence and quantitative $\Omega(R_{\max} H\delta)$ lower bound) are proved in Appendix N.3. Theorem 5 (recovery via (C1)–(C3)) is proved in Appendix N.4. The cross-MDP simulation lemma (Lemma 2) is in Appendix N.3, and the solution-level bounds on the gradient-heterogeneity quantity ζ^2 are in Appendix N.5. The formal foundation for the partitions of §3.2 and the experiments of §5.2 is in Appendices K and L: the inter-client dispersion measures $\Delta^2_{\text{pref}}, \Delta^2_{\text{cov}}, \Delta^2_{\text{hard}}$, the primary properties (P1)–(P3), and the definitional orthogonality / first-order sufficiency results (Lemma 1 and Theorems 6 to 8) are in Appendix K. The three partition algorithms and the four design properties (D1)–(D4) are in Appendix L. Propositions 1 to 3 establish (D1), single-hyperparameter monotone control, and Theorems 9 to 11 establish (D2)–(D4).

Heterogeneity construction. Pseudocode for the three task-level partition procedures (PREFERENCEPARTITION, COVERAGEPARTITION, HARDNESSPARTITION) is in Appendix L.1. Per-client distribution figures verifying that each hyperparameter cleanly controls its dispersion measure are in Appendix L.3. The five WebShop environment-level variants (Catalog Split, Field-Subset Index, BM25 Reweighting, Lookalike Injection, Rank Wrapper) are formalized at the MDP level in Appendix M, including the four-stage transition pipeline taxonomy that classifies them.

Experimental setup and hyperparameters. The full experimental configuration (hardware, software stack, model backbones, federation protocol, GRPO/PPO hyperparameters, rollout protocol, and evaluation protocol) is in Appendix J. The justification for the federated-configuration choices ($M = 2$, $E = 3$, $|X_i| = 100$) used throughout the main text is provided by the hyperparameter sensitivity ablation in Appendix G.1. PPO counterparts of the main-text GRPO results are in Appendices G.2 and G.3 and in Figure 6(b).

Compute and statistical reporting. All experiments run on NVIDIA H100 (80 GB) GPUs. All federated and centralized rows in Table 1 and Table 3 report the mean and sample standard deviation over 3 runs, shown as a subscript (seeds in Appendix J). The training-dynamics figures (Figure 4, Figure 5, Figure 6) show the aggregated server curve, with per-client checkpoints overlaid as colored circles where reported (Figure 4).

Notation. A consolidated notation table is provided in Table 2 (Appendix B) for quick reference of all symbols used throughout the paper.

D Broader Impact

Privacy-preserving agent training. FEDAGENT addresses a fundamental tension in modern AI agent deployment: training high-quality LLM agents typically requires centralizing user task queries and interaction trajectories, but these data are often sensitive (medical conversations, personal browsing histories, financial workflows, on-device assistant logs) and subject to privacy regulation such as the EU GDPR, HIPAA in the U.S., the California CCPA, and emerging AI-specific statutes like the EU AI Act. By enabling collaborative agent training across distributed clients without sharing local data, FEDAGENT extends federated learning beyond the single-turn preference setting of federated RLHF to the multi-step trajectory data (state-action sequences, intermediate observations, tool-use traces) that are often more identifying than preference labels alone. FEDAGENT thereby aligns agent training with the data minimization and purpose limitation principles that underlie modern privacy law. Appendix I.2 discusses these privacy properties in detail, arguing structurally that the gradient-inversion and membership-inference attacks designed for supervised classification transfer poorly to long-horizon LLM agent rollouts, and that orthogonal defenses (differential privacy, secure aggregation, parameter-efficient transmission) compose with the base FEDAGENT protocol for hardened deployments.

Enabling high-stakes, cross-institutional applications. Beyond compliance, FEDAGENT *enables* agent applications in domains where centralization is entirely forbidden. Centralized agent training is structurally impossible in many settings: multi-hospital studies of clinical decision-support agents, cross-bank training of financial-fraud agents, cross-jurisdiction (EU, U.S., APAC) federated training where data residency laws prohibit cross-border movement, and IRB-constrained behavioral research. FEDAGENT makes collaborative agent training possible in these settings by replacing the data-sharing requirement with a parameter-aggregation requirement, and our structural results (Theorems 1, 4, and 5) give operators principled predictions, from checks of (C1)–(C3) before training and from the training curve during it, about which collaborations are likely to preserve agent quality and which are not.

Personalized agents. The same federated protocol that protects user data also produces agents that adapt to each client's own task distribution (and, when one of (C1)–(C3) holds, to its environment) without the clients' data ever leaving local storage. The Asymmetric Robustness Mechanism additionally gives personalization a *principled* basis: under task-level heterogeneity and assumption (R), Theorem 1(ii) implies that federated averaging need not average away the preferences of small or minority groups (rare task intents, low-resource languages, atypical user needs). A single global agent can accommodate per-client variation (Pattern A). When environment-level heterogeneity breaks all of (C1)–(C3), Pattern D signals that plain federated averaging should be replaced, for example by environment-aware prompting, per-client personalization, or hierarchical aggregation. The failed conditions give downstream personalization methods a precise structural target.

Scalability and accessibility. The modular client structure of FEDAGENT means that contributors with heterogeneous compute capabilities can still meaningfully participate, since each client only runs local rollouts and policy updates rather than the full population-scale training. This modular structure lowers the entry cost for individuals, hospitals, schools, small businesses, and public-sector organizations to participate in agent training previously feasible only with centralized data and infrastructure access. The modular structure also broadens who shapes modern agent capabilities, reaching application domains (rare diseases, low-resource languages, regional public-sector workflows) that rarely attract centralized investment. When FEDAGENT is combined with parameter-efficient fine-tuning (LoRA, adapters) or zeroth-order optimization, the per-client cost can be reduced further to fit edge and on-device settings.

A structural diagnostic for deployment. The Asymmetric Robustness Mechanism (§4) gives practitioners a structural diagnostic that goes beyond a generic "federated learning works" claim. In our analysis, federated LLM agent training is robust to task-level heterogeneity under realizability (Pattern A), robust under environment-level heterogeneity when one of (C1)–(C3) holds and the optimizer attains the shared optimum (Pattern B), degrade-but-stable when the conditions hold only partially or optimizer slack remains (Pattern C), and worst-case non-robust when all three fail (Pattern D). Pattern D's flat or collapsing training curve is itself an actionable signal. It indicates that the clients' environments demand conflicting behavior that one shared policy cannot reconcile,

so plain federated averaging should be replaced by one of the alternatives named in the paragraph on personalized agents. Operators can thus make the deployment decision from the training curve, before incurring the cost of large-scale deployment.

Foundation for future federated agent RL research. Beyond practitioners, our analysis equips the research community with explicit design targets for the next generation of federated agent algorithms. The Input-Dynamics Asymmetry (§3.1) and the three sufficient conditions (C1), (C2), (C3) play a role analogous to the bounded-gradient-heterogeneity assumptions underlying supervised federated learning algorithm design (FEDPROX, SCAFFOLD, FEDNOVA): future federated agent RL methods can be designed either to satisfy (C1), (C2), (C3) by construction (for instance, via observation-augmented policy classes or hierarchical action spaces), or to relax them in controlled ways. Our open-source release of code, configuration files, preprocessing utilities, and data-setup instructions further lowers the entry cost for follow-up research, supporting reproducibility and red-teaming of federated agent systems at the same level of rigor as centralized agent training.

E Limitations

We describe the scope of FEDAGENT in four areas.

Scope of the heterogeneity abstraction. The two-level decomposition of §3 (task vs. environment, under Input-Dynamics Asymmetry) is, to our knowledge, the first principled framework for organizing Agent Heterogeneity. The decomposition already captures two regimes we expect practitioners to encounter frequently: cross-user task cohorts and cross-deployment environment drift. Richer regimes such as heterogeneous reward functions, asynchronous communication, or Byzantine clients are not modeled here and are promising directions for follow-up work.

Theoretical assumptions. The Asymmetric Robustness Mechanism is built on a deliberately minimal set of assumptions, stated alongside each theorem in §4 and Appendix N: bounded reward $R_{\max}$, finite horizon H, and the LLM-tailored realizability assumptions (R), (R′), (R_{env}), (R_{aug}), (LR) on the policy class. The boundedness assumptions are standard. The realizability assumptions are designed to be plausible for modern instruction-tuned LLMs operating on bounded-reward, episodic agent benchmarks (Appendix N.1 discusses when and how tightly they hold). The gradient-heterogeneity quantity ζ^2 appears only in the discussions of Appendices N.5 and O.6. The solution-level bound on ζ^2 in Appendix N.5 additionally assumes smoothness. None of Theorems 1–5 depends on ζ^2. The sufficient conditions (C1), (C2), (C3) provide a checklist for ruling out Theorem 4's worst-case $\Omega(R_{\max} H \delta)$ floor in a deployment. We expect (C3) in particular to be the condition that applies for LLM agents in many real deployments, because of natural-language environment cues.

Gap between the theory and the experimental setup. Six simplifications separate the theorems from our runs. (i) The bounds are stated for per-step rewards in $[0, R_{\max}]$, so $R_{\max} H$ is the range of the return. The reward R in our runs is a sparse terminal success signal whose return lies in $[0, 1]$, and training adds a small invalid-action penalty that is not part of R (Appendix J.4). The upper bounds of Theorems 2 and 5 remain valid with the range $R_{\max} H$ replaced by this maximal return, whereas the H-amplification of Theorem 4, which needs rewards that accumulate over steps, cannot arise in our runs. (ii) Assumption (R_{aug}) and condition (C3) assume that the policy conditions on the full trajectory prefix, while our rollouts keep only the two most recent steps in the prompt (Appendix J.5). This two-step history limits in-context identification of the environment. (iii) The environment-level runs are validated in the unperturbed reference environment. They therefore measure how well the federated model transfers to that environment, whereas Theorems 4 and 5 concern the policy-disagreement gap Δ_{pol}. (iv) Every WebShop variant has $\delta = 1$ (Appendix M.9), so the experiments test which of (C1)–(C3) a variant preserves while δ stays fixed. (v) The task-level runs are likewise validated on one held-out set shared by all clients. Their curves therefore compare the global model across splits. Theorem 1(i) addresses this comparison, whereas Theorem 2 concerns the per-client gap. For two splits to reach the same plateau, the split must also not raise ϵ_{opt}, and no theorem guarantees this condition (Appendix O.2). (vi) Definition 2 assumes that clients share the task distribution, whereas each environment-level client samples its tasks from its own pool of 100 goals produced by the uniform partition of Appendix J.3, so the clients' task distributions are similar but not identical.

Empirical scope. Our empirical evaluation in §5 spans a substantive cross-section of modern LLM agent training: two widely used benchmarks (WebShop, ALFWorld), four backbones across two model families (`Qwen2.5-{1.5,3,7}B-Instruct`, `Llama-3.2-3B-Instruct`) covering the 1.5B–7B regime in which most agent-RL practice today operates (the heterogeneity experiments use `Qwen2.5-1.5B-Instruct`), and the two dominant policy optimizers (GRPO, PPO). The formal results are stated at the MDP level and therefore extend beyond this scope. Whether the empirical patterns persist for multi-modal or physically embodied agents, other model scales, and additional RL algorithms remains to be verified. Each of these settings is a natural next direction for the two-level framework.

F Extended Discussion of Main-Text Sections

To fit the page limit, §3–§5 of the main text are condensed. This appendix collects the full discussion of those sections, organized in the order of the main text.

F.1 Task-Level Heterogeneity: Sub-Types and Measurement

Three operationally separable sub-types. Three first-order properties of $\mathcal{D}_{\tau_i}$ each affect federated agent RL through a distinct mechanism, motivating the decomposition. For each sub-type we state what varies across clients, why it matters for training, and the statistic we use to measure it.

- **Preference Heterogeneity** (*What type?*). Clients differ in *which types of tasks* they mostly encounter: the tasks from one ALFWorld client's household may be mostly kitchen tasks ("*put the apple in the fridge*") while another's are mostly bedroom tasks ("*examine the lamp*"). One WebShop client searches mostly for electronics while another searches mostly for clothing. This skew matters for training because it shifts *which task-conditional behaviors* receive training signal under the federated mixture, and thus how much of the LLM's instruction-tuned pre-training is used on each category. We measure each client's preference by the type marginal $p_i = \ell_* \mathcal{D}_{\tau_i} \in \Delta^{C-1}$, obtained by mapping $\mathcal{D}_{\tau_i}$ through a category map $\ell : \mathcal{T} \to [C]$ (*e.g.*, ALFWorld's six household task types or WebShop's product categories). Because p_i lies on the simplex, we summarize inter-client dispersion by the natural squared L^2 deviation from the client mean $\bar{p}$, $\Delta^2_{\text{pref}} := \frac{1}{N}\sum_i \|p_i - \bar{p}\|_2^2$.
- **Coverage Heterogeneity** (*How many?*). Clients differ in *how many tasks* their local pools contain: on WebShop, one client has an extensive browsing history of hundreds of product queries while another has only a few completed sessions. Pool size matters for training because, under iterative-with-replacement RL sampling [31], the pool size limits the client's *overall exploration breadth*: a small pool makes the client revisit the same tasks. This effect is specific to RL training and is not equivalent to supervised quantity-imbalance (where every example is visited once per pass over the data). We measure this sub-type by the per-client local pool size $n_i = |X_i|$. Since pool sizes can differ by an order of magnitude or more across clients, we use the scale-invariant squared coefficient of variation $\Delta^2_{\text{cov}} := \widehat{\text{CV}}^2(\{n_i\}) = \frac{1}{N}\sum_i \big((n_i - \bar{n})/\bar{n}\big)^2$ rather than raw variance.
- **Hardness Heterogeneity** (*How hard?*). Clients differ in *how difficult their tasks are for the model*: on ALFWorld, one client's pool may consist mostly of tasks that a reference model already solves, while another's is dominated by tasks of the same types that the reference model mostly fails. Difficulty matters for training because it affects the *policy-gradient advantage signal*. A client whose tasks the policy almost never solves yields few positive samples and hence little learning signal. A client whose tasks are almost always solved has little room for improvement. Group-relative methods such as GRPO [107] normalize advantages by within-group reward statistics, so a group whose rollouts all succeed or all fail yields zero advantage. Our measurement of difficulty is model-relative by design: what matters for policy-gradient training is whether the policy can collect reward on a task, so the operationally relevant notion of difficulty is a checkpoint's measured success rate on the task. Let $\rho(\tau)$ be the empirical success rate of a fixed reference checkpoint π_{ref} rolled out on task τ. In our experiments, π_{ref} is a benchmark-fine-tuned reference checkpoint frozen before any federated training (Appendix J.6), and higher $\rho(\tau)$ means an *easier* task. We then measure client i's difficulty by the thresholded success probability $\rho_i = \Pr_{\tau \sim \mathcal{D}_{\tau_i}}[\rho(\tau) \geq \eta]$, where $\eta \in (0, 1)$ is a fixed success threshold. The base symbol ρ is shared between the per-task rate $\rho(\tau)$ and the client-level statistic ρ_i. A hard client's pool concentrates at low $\rho(\tau)$, giving a small ρ_i. Since ρ_i already lies in the bounded interval $[0, 1]$, we summarize inter-client dispersion by the standard sample variance $\Delta^2_{\text{hard}} := \widehat{\text{Var}}_i(\rho_i)$.

Our experiments in §5.2 aim to attribute any observed effect to a single task-level sub-type, which requires the three dispersion measures $(\Delta^2_{\text{pref}}, \Delta^2_{\text{cov}}, \Delta^2_{\text{hard}})$ to be *operationally separable*: we must be able to vary each measure independently of the other two while the first-order means stay fixed (properties (D1)–(D4) of Appendix F.2). The triple (p_i, n_i, ρ_i) gives exactly this granularity. The triple is a *first-order task-distribution descriptor* of client i: each component is a first-order projection of $\mathcal{D}_{\tau_i}$ (its pushforward along the category map ℓ, its support size, and its mass on the success set $\{\rho(\tau) \geq \eta\}$, respectively). The descriptor deliberately contains only first-order information about $\mathcal{D}_{\tau_i}$: it does not capture higher-order structure such as the within-client difficulty profile or the type–difficulty joint distribution, and two clients with identical (p_i, n_i, ρ_i) can still differ in those

respects (Appendix K proves that the triple determines the three dispersion measures and gives a counterexample to the converse).

F.2 Partition Algorithms

Partition algorithms and design properties. We design three sampling procedures, each instantiating one task-level sub-type under a single dispersion hyperparameter (pseudocode in Appendix L.1):

- PREFERENCEPARTITION(ω) samples each client's category distribution from $\mathrm{Dirichlet}(p_0 \cdot (1-\omega)/\omega)$ centered at the global marginal p_0 (the population-level type distribution, of which $\bar{p}$ is the client-sample mean) while keeping a fixed per-client pool size $|X_i| = L$. $\omega \to 0$ *is the near-uniform setting, and* $\omega \to 1$ *is the most heterogeneous, with each client collapsing onto a single category (one-hot vertices).* Figure 14 visualizes the two endpoints on both benchmarks.
- COVERAGEPARTITION(ξ) samples each client's pool size n_i from a Beta-shaped distribution on the admissible range $[L_{\min}, L_{\max}]$ with mean $\bar{n} = L_{\text{avg}}$, holding the global overlap r (the average number of clients holding each task) fixed. This construction controls the spread of $\{n_i\}$ without changing the mean pool size $\bar{n}$: the expected dispersion decays as $C_n/(\xi+1)$, where C_n is a constant of the construction (Proposition 2). *Large ξ is the near-uniform setting, and small ξ is the most heterogeneous ($\xi = 1$ in our experiments). Because ξ parameterizes Beta concentration whereas ω parameterizes Dirichlet dispersion, this hyperparameter direction is opposite to ω.*
- HARDNESSPARTITION(ξ') first measures hardness operationally using the reference model, without human annotation. The procedure rolls out the frozen reference checkpoint π_{ref} on every task and labels tasks with measured success rate $\rho(\tau) \geq \eta$ as "successful" and the rest as "unsuccessful." The labels are measured once before federated training and used only to construct the partition. Difficulty relative to the evolving policy, by contrast, changes during training. The procedure then samples per-client success-rate quotas from a Beta-shaped distribution governed by ξ', controlling Δ^2_{hard} while keeping the pool size $|X_i| = L$ identical across clients. *The direction matches ξ: larger ξ' is near-uniform, and $\xi' = 1$ is the most dispersed setting in our experiments.*

The three partition algorithms satisfy four design properties (proofs in Appendix L.2). **(D1) Single-hyperparameter monotone control:** each hyperparameter monotonically controls its target measure (Propositions 1–3). **(D2) Cross-measure isolation:** varying one hyperparameter leaves the other two measures invariant in expectation under the assumption that type and difficulty are threshold-independent. A chi-squared test does not reject this assumption on WebShop but rejects it on ALFWorld, where the resulting leakage stays small (Appendix L.2). For example, $\partial_\omega \mathbb{E}[\Delta^2_{\text{cov}}] = \partial_\omega \mathbb{E}[\Delta^2_{\text{hard}}] = 0$ (Theorem 9). **(D3) Factor invariance:** the first-order means $(\bar{p}, \bar{n}, \bar{\rho})$ are preserved as the hyperparameters vary (Theorem 10). **(D4) Joint configurability:** a composite of the three partitions reaches any feasible target triple under threshold independence (Theorem 11). Together, these properties let us separate and vary each task-level sub-type independently in the experiments of §5.2.

F.3 Environment-Level Heterogeneity: Construction on WebShop

Construction on WebShop. Environment-level heterogeneity cannot be captured by a single type of perturbation, because the transition kernel $P(s' \mid s, a)$ governs every coupling between an agent's action and the next observation. Perturbing the kernel can therefore target four non-equivalent aspects: *what* content the environment exposes, *how* that content is indexed, *how* an action is matched and scored against the indexed content, and *how* the resulting match is presented as the next observation. Perturbing each aspect changes the transition kernel in a different way, and our construction covers all four.

We make this four-aspect decomposition concrete on WebShop, whose transition pipeline naturally factors into four stages aligned with these aspects: *content/catalog*, *encoding/index*, *matching/score*, and *rendering*. We then instantiate one or more variants per stage, yielding five WebShop perturbations spanning all four stages (Appendix M). The simplest, **Catalog Split**, perturbs the *content* stage by assigning each client a different product subset. Because the optimal policy "search → click → buy" transfers across subsets, $\pi^\star$ stays invariant. **Field-Subset Index** perturbs *encoding* by indexing only a subset of doc-text fields per client (*e.g.*, `name+Title` vs. `BulletPoints`), so identical queries yield different rankings and force client-specific query formulation. **BM25 Reweighting** perturbs

matching by setting per-client extreme (k_1, b) combinations, changing term-frequency saturation and length normalization at the score level. **Lookalike Injection**, the most elaborate construction of the five, jointly perturbs *content* and *matching* by injecting per-client lookalike products that tie with the target in BM25 ranking but fail one or two reward components. Each lookalike ID carries a visible `LK_` prefix. Finally, **Rank Wrapper** perturbs *rendering* by post-processing BM25 outputs (with the identity, a shuffle, an inversion, or, on about half of the queries, a random list), invalidating a heuristic that the policy might otherwise learn, namely that the top-ranked result is reliable. Together, these variants elicit the full spectrum of **Pattern B** (robust) / **C** (degrade) / **D** (collapse) training behaviors defined by the Asymmetric Robustness Mechanism (§4.4). See §5.3 for results.

F.4 Asymmetric Robustness Mechanism: Extended Discussion of the Theorems

Overview. This subsection discusses the central theoretical consequence of I-D Asymmetry: FEDAGENT's robustness is *asymmetric* across the two heterogeneity levels. FEDAGENT is robust to task-level heterogeneity but worst-case non-robust to environment-level heterogeneity. Three sufficient conditions recover robustness under environment-level heterogeneity. The mechanism relies on five LLM-tailored realizability assumptions about the policy class $\Pi_\Theta = \{\pi_\theta : \theta \in \Theta\}$. Each theorem compares the federated solution with a task- or environment-conditional optimum and therefore assumes that Π_Θ can express the relevant optimum. Assumptions (R) and (R′) posit that Π_Θ realizes exact and ϵ-approximate task-conditional optima, respectively. Assumptions (R_{env}) and (R_{aug}) are the environment-level and history-augmented analogues. Assumption (LR) posits that Π_Θ realizes each client's own (local) optimum in the lower-bound constructions. Appendix N.1 states all five assumptions formally. The mechanism yields four observable training-curve patterns that connect theory and experiments (§4.4).

Mechanism 1: task-level robustness. We first show that under task-level heterogeneity (Definition 1), federated agent training is *structurally equivalent* to centralized training on the mixture task distribution $\bar{\mathcal{D}}_\tau$. Concretely, *robustness* here means that for every client i, the federated per-client return $\mathcal{J}_i(\hat{\theta}_{\text{fed}})$ matches the per-client optimum $\sup_\theta \mathcal{J}_i(\theta)$ up to a slack that vanishes as the LLM grows and training proceeds, with no irreducible cost from the federated structure. We establish this robustness in two steps, an idealized statement (Theorem 1) and an approximate statement that handles imperfect realizability and optimizer slack (Theorem 2).

Theorem 1's two parts state what task-level robustness means in the idealized limit. Part (i), the *mixture-equivalence identity*, says the federated objective *equals* the centralized objective on $\bar{\mathcal{D}}_\tau$ pointwise in θ: the population optimization problem is the same regardless of how task mass is distributed across clients, so extreme one-hot splits and i.i.d. uniform splits share the same optima (their FedAvg trajectories can still differ through client drift, Appendix O.2). Part (ii) adds that the federated optimum is simultaneously optimal for every client: there is no tradeoff between client and federation performance at the global optimum. In this idealized limit, federated and centralized training therefore coincide in both objective and optimum.

Theorem 2, the practical form of Theorem 1, relaxes (R) to its ϵ-version (R′): the LLM may realize the task-conditional optimum only up to capacity slack ϵ_{approx}, and the optimizer may reach the federated optimum only up to ϵ_{opt}. The bound contains no irreducible δ-term. There is no structural floor (an irreducible lower bound on the per-client gap) that optimization cannot remove. The slack terms inside the square root vanish under conditions already standard in centralized training: $\epsilon_{\text{approx}} \to 0$ as the LLM grows and $\epsilon_{\text{opt}} \to 0$ as training proceeds. The divergence $\chi^2(\mathcal{D}_{\tau_i} \| \bar{\mathcal{D}}_\tau)$ need not vanish, but it is at most $N - 1$ in the standard uniform-weight federated setup, regardless of how extreme the per-client split is, and the per-client gap also admits the linear bound $N(\epsilon_{\text{approx}} + \epsilon_{\text{opt}})$ (Appendix N.2).

Theorem 2 thus establishes the task-level robustness of FEDAGENT: under Definition 1 and the realizability assumption (R′), the guarantee holds no matter how tasks are split across clients. The mechanism is I-D Asymmetry: because τ enters the policy as part of the prompt, a single θ can implement different behaviors conditioned on different τ (the same LLM that summarizes for one prompt writes code for another). Different clients' gradients therefore train θ on different parts of the task-conditional optimal behavior $\tau \mapsto b(\tau)$ (the behavior an optimal policy implements on task τ). These parts are complementary and share a common optimum. Away from that optimum, gradients from different clients can still disagree. This disagreement is the transient gradient heterogeneity of Appendix N.5, and it leaves the objective unchanged: the federated population objective still coincides

with the centralized one (Theorem 1(i)). Pre-training makes this mechanism sharper than the bound alone suggests: pre-training and instruction tuning typically place θ_0 near a task-conditional optimum on a wide mixture, so the initial suboptimality ϵ_0 of θ_0, which upper-bounds ϵ_{approx}, is typically small, and federated fine-tuning extends an existing capability rather than learning $\tau \mapsto b(\tau)$ from scratch (Appendix P.1).

Mechanism 2: environment-level non-robustness. Under environment-level heterogeneity (Definition 2), federated agent training is *fundamentally different*: there are environment constructions in which FEDAGENT is worst-case non-robust. Here, *non-robustness* means that for some constructions, the worst-off client's return at any federated optimum $\theta^\star_{\text{fed}}$ stays below its own optimum $\sup_\theta \mathcal{J}_i(\theta)$ by a gap that no larger LLM, longer training, or choice of optimizer removes. This gap has a structural origin. We establish an existence claim in Theorem 3 and a quantitative lower bound in Theorem 4.

Theorem 3 states that we can construct environments in which every federated optimum is strictly suboptimal for at least one client. A canonical construction is a two-client transition-swap bandit (Appendix N.3): client 1's optimal action is client 2's worst action, and conversely. Any single θ mixes the two actions in a fixed proportion, so at least one client incurs strictly positive suboptimality. In the task-level case (Theorem 1), the federated optimum is simultaneously optimal for *every* client. In this construction, no single θ can be optimal for both clients.

Theorem 4 is the quantitative form of Theorem 3: it characterizes the magnitude of the policy-disagreement gap Δ_{pol} as a function of the inter-environment divergence δ. The bound has an *irreducible floor* $\Omega(R_{\max} H\delta)$: on such worst-case constructions, where the environment does not reveal its identity through observations, no amount of LLM capacity, samples, or training, and no choice of optimizer, can remove this floor. (The single-step $\Omega(R_{\max}\delta)$ gap holds for any policy class satisfying (LR). The H-amplification additionally requires that (C3) be unavailable.) Given the transition heterogeneity in Definition 2, the gap characterizes the federated *solution* independently of the optimization *trajectory* that reaches it. This trajectory-independent floor is the central contrast with Theorem 2, whose bound consists entirely of slacks that vanish under standard centralized-training conditions.

The mechanism is again I-D Asymmetry, operating in the opposite direction: environment identity i is *not* part of the prompt, so a single θ cannot switch behavior across environments at the same observed (s, τ). At such an input, the policy must emit the same action distribution regardless of which $\mathcal{M}_i$ the rollout is in. When per-client optimal actions at (s, τ) disagree across environments, different clients' gradients pull θ in *opposite* directions on the *same* parameters, and FedAvg averages them toward zero, encoding a compromise that is wrong for at least one client. The contrast with task-level heterogeneity is clear: there, different clients' gradients provide different pieces of the function $\tau \mapsto b(\tau)$ and add complementarily. Here, different clients' gradients act at the same (s, τ) input but pull θ's action in opposite directions, with a structural floor $\Omega(R_{\max} H\delta)$ on the resulting worst-client gap. For LLM agents specifically, this non-robustness can incur an additional cost not captured by the bound: when conflicting per-client $\{P_i\}$ force θ to encode contradictory world models in the same parameters, the pre-training prior in θ can be overwritten, which would appear as catastrophic forgetting of capabilities the base model originally possessed (an informal claim that our runs do not measure, Appendix P.2).

Both mechanisms on one bandit. Figure 7 illustrates the two mechanisms and the recovery conditions with one two-client bandit, in which client 1 is paid for a_1 and client 2 for a_2. Each panel plots the client returns $(\mathcal{J}_1, \mathcal{J}_2)$ that policies attain. The federated optimum maximizes $\mathcal{J}_1 + \mathcal{J}_2$, so it is the attained point furthest toward the upper right, whereas each client's own optimum is the corner $(R_{\max}, R_{\max})$. The worst-client gap is therefore zero exactly when the attained region contains the corner. When the payoffs come from the clients' tasks, the policy reads the task from the prompt, the region is the whole square, and the federated optimum is the corner (Theorem 1). When the same payoffs come from the transition kernels, the policy cannot distinguish the clients, and the region collapses to the segment $\mathcal{J}_1 + \mathcal{J}_2 = R_{\max}$. Every point of the segment is federation-optimal, the two clients' gradients cancel there, and each point leaves some client at least $R_{\max}/2$ below its optimum (Theorem 3). An action that pays in both environments satisfies (C1) and (C2) and restores the corner. A banner that names the client but is wrong with probability η_{cls} brings the federated optimum to within $\eta_{\text{cls}} R_{\max}$ of the corner. The resulting worst-client gap $\eta_{\text{cls}} R_{\max}$ matches the $R_{\max} H \eta_{\text{cls}}$ term of the (C3) bound of Theorem 5 at $H = 1$ and $u^\star = 0$. Because the banner is visible before the first

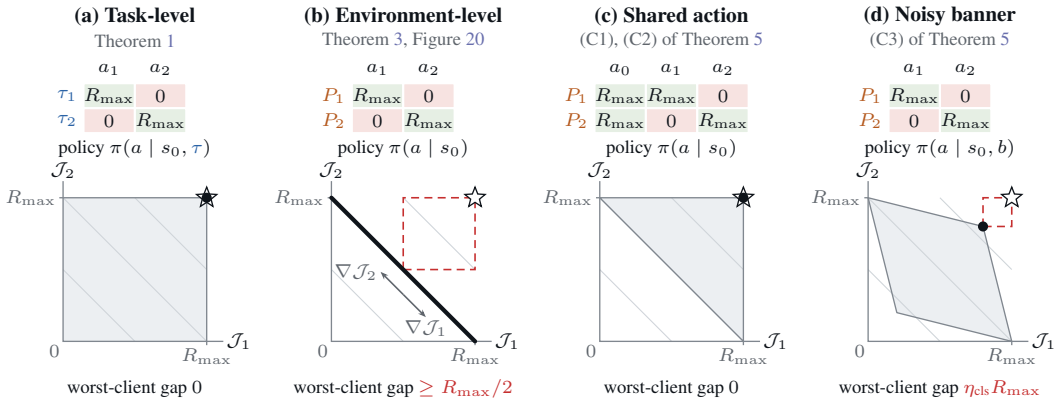


Figure 7: **The same payoffs give a zero worst-client gap when they come from the task and a gap of at least $R_{\max}/2$ when they come from the transitions.** Each panel is a one-step, two-client bandit. The grid gives each client's reward for each action, with one row per client labeled by what differs between the clients, and the line below it gives the policy's input. The plot shows the client returns $(\mathcal{J}_1, \mathcal{J}_2)$ that policies attain (gray), with level sets of $\mathcal{J}_{\text{fed}}$ as hairlines. The federated optimum (•) is the attained point furthest toward the upper right, and in (b) every point of the black segment is one. The star is each client's own optimum, filled where the federated optimum reaches it, and the side of the red square is the worst-client gap. Panel (b) is the bandit of Figure 20, and its arrows are the two clients' gradients, which cancel. Panel (c) adds an action a_0 that pays in both environments. In panel (d), a banner b names the client but is wrong with probability η_{cls} (plotted for $\eta_{\text{cls}} = 0.2$).

action, the initial state here depends on the client. Definition 2 excludes such a dependence, but the (C3) case of the proof of Theorem 5 never uses a shared μ_0 (Appendix N.4). At $\eta_{\text{cls}} = \frac{1}{2}$ the banner carries no information, and the region is again the segment.

F.5 Formal Statement of Condition (C3) and Discussion of the Recovery Theorem

- **(C3) Self-revealing environment via observation history** (breaks requirement (c) of §4.3). Environment identity is inferable from the trajectory prefix $h_u := (s_0, a_0, r_0, \ldots, s_u) \in \mathcal{H}$ that the policy has observed so far. Formally, there exist a classifier $\hat{\imath} : \mathcal{H} \to [N]$, kernel models $\{\hat{P}_i\}_{i \in [N]}$, a reveal step $u^\star \leq H$, and a fixed default prefix policy π_0 (followed for $u < u^\star$) such that:
  - **(C3-cls)** $\Pr_{h_{u^\star} \sim (\mathcal{M}_i, \pi_0)}[\hat{\imath}(h_{u^\star}) \neq i] \leq \eta_{\text{cls}}$ for every i, where $h_{u^\star}$ is the prefix generated by rolling out π_0 in $\mathcal{M}_i$ (classifier slack: the environment label inferred at the reveal step $u^\star$ is kept fixed thereafter);
  - **(C3-ker)** $\sup_{s,a} D_{\text{TV}}(\hat{P}_i(\cdot \mid s, a), P_i(\cdot \mid s, a)) \leq \eta_{\text{ker}}$ for every i, where $\hat{P}_i$ is the kernel model with respect to which the post-reveal policy for environment i is optimal (Appendix N.4, Step (c)) (kernel-modeling slack; $\eta_{\text{ker}} = 0$ when that policy is optimal for the true kernel);
  - **(C3-rec)** the prefix is *recoverable*: $\mathbb{E}_{(\mathcal{M}_i, \pi_0)}[V_i^\star(s_{u^\star})] \geq V_i^\star - c_0 R_{\max} u^\star$ for every i and an absolute constant c_0, where $V_i^\star(\cdot)$ is the optimal value function of $\mathcal{M}_i$ for the remaining horizon and, slightly abusing notation, $V_i^\star$ is its expectation over the task and initial state. In words, following π_0 for $u^\star$ steps does not drive the state into a low-value region. The condition holds automatically when π_0 only gathers information without changing the reward-relevant state (*e.g.*, reading a banner), or when a back or home action restores the initial state within the episode, as WebShop's back-to-search action does. Resets between episodes do not imply it, because they act only after the episode ends.

  Under (R_{aug}), the history-augmented policy class Π_Θ^{aug} (the original class with the trajectory prefix h_u appended to the input) represents the resulting environment-conditional policy. For example, a web agent reads "united.com" from a banner and conditions subsequent actions accordingly.

Theorem 5 says each condition, when satisfied, returns the environment-level case to Mechanism 1's structurally equivalent regime. Conditions (C1) and (C2) achieve this *exactly* ($\Delta_{\text{pol}} = 0$), while (C3) achieves it *approximately* with a slack-bounded residual that vanishes as the classifier and kernel models become accurate. In each case, one of requirements (a)/(b)/(c) of §4.3 is structurally

excluded, so Theorem 4's floor no longer applies and robustness is recovered. These three conditions are sufficient but not necessary: a deployment where all three appear to fail empirically can still avoid Pattern D and may even be robust (Appendix O.8.4). For LLM agents specifically, **(C3) is particularly important**: pre-training can equip the LLM with natural-language understanding sufficient to perform *in-context posterior inference* about its environment from branding, error messages, prompt cues, and structural signals that synthetic benchmarks intentionally suppress. Assumption (R_{aug}) captures this capability formally: the LLM can implement the classifier $\hat{\imath}$ in context as a side effect of next-token prediction. The in-context form of (C3) predicts that federated fine-tuning across heterogeneous environments can work without modification on real web/tool deployments with informative cues, where branding alone can make (C3-cls) trivial. Our synthetic variants, by contrast, deliberately suppress such cues. The prediction still admits failure on adversarial constructions like the observation-silent transition-swap chain of Theorem 4 (Appendix N.3), whose observations carry no information about the environment identity. Appendix P.3 formalizes the explicit form of (C3) via environment-aware prompting (Corollary 1), and Appendix P.4 discusses the implicit (in-context inference) form.

F.6 Four Empirical Patterns: Full Statements

The Asymmetric Robustness Mechanism (Theorems 2, 4, 5) predicts four possible training-curve patterns for federated agent runs. Pattern A describes the task-level regime (Mechanism 1), where mixture-equivalence makes the population objective split-invariant, so no task split creates a structural floor (the per-client guarantee requires (R) or (R′)), although attaining the plateau still depends on the optimizer slack ϵ_{opt}, as for every pattern. Patterns B/C/D describe the environment-level regime (Mechanism 2), ranging from full robustness recovery to complete collapse. Which pattern a variant exhibits depends on two factors. The first is how closely (C1)/(C2)/(C3) hold, which determines the achievable plateau. The second is the optimizer's optimization-level slack ϵ_{opt}, which determines whether that plateau is attained. The second factor is orthogonal to the first, so the same variant can exhibit Pattern B under one optimizer and Pattern C under another (§5.3). Each pattern has a specific theorem and empirical signature. These associations make the mechanism a diagnostic tool: an observed curve shape indicates which regime is likely active and which mitigation applies.

- **Pattern A (Task-level robust).** The federated curve converges to the same plateau as the near-uniform baseline (the i.i.d.-style split of §5.2) up to training-level noise, irrespective of how task mass is split across clients, unless the split raises the optimizer slack ϵ_{opt}. The mixture-equivalence identity (Theorem 1(i), underlying Theorem 2) shows that the population objective is the same for any split sharing the same mixture. The population optimization problem is therefore that of centralized training on $\bar{\mathcal{D}}_\tau$, so the achievable plateau is the same, and the per-client gap vanishes with the slacks of Theorem 2. Neither theorem states that FedAvg attains this plateau equally well under every split. The common optimum of Theorem 1(ii) makes this outcome plausible: under (R), $\Delta_{\text{pol}} = 0$, so by Equation (2) client gradients nearly agree near the solution, leaving little split-dependent client drift there. Away from the optimum, one-hot splits can still produce conflicting gradients and transient drift (Appendix O.2). The empirical signature is this split-invariance of the converged curve, up to the run-to-run variability of agent RL.
- **Pattern B (Environment-level robust: one of (C1)–(C3) holds and the optimizer attains the shared optimum).** The federated curve nearly matches the homogeneous-environment (single-environment) baseline despite $P_i \neq P_j$. Under (C1) or (C2) (with (R_{env})), Theorem 5 guarantees a single $\theta^\star$ that is simultaneously optimal for every $\mathcal{M}_i$ (under (C3), up to its slack bound), so FedAvg can reach the per-client optima without a structural compromise. For LLM agents in real deployments, we expect (C3) to be the condition that applies: environmental cues (*e.g.*, branding, DOM) can enable in-context identification. Environment identity then becomes part of the input, reducing the case to task-level heterogeneity. In our synthetic WebShop variants, which suppress such cues, Pattern B instead arises via (C2) (§5.3).
- **Pattern C (Environment-level degrade-but-stable: (C1)–(C3) hold at best partially, or one holds fully but the optimizer leaves a nonzero ϵ_{opt}).** Performance falls by a finite margin while training stays stable and converges. The conditions hold *approximately*: environment identity is inferable from history with classifier slack $\eta_{\text{cls}} > 0$, the clients share an optimal action on most but not all reachable states, or Π_Θ contains only an approximate common optimum. Alternatively, one condition holds in full but the optimizer's ϵ_{opt} keeps the plateau below the baseline (the Catalog

Split/GRPO and Lookalike Injection cases of §5.3). The resulting per-client gap depends on these slacks (Theorem 5 bounds the (C3) terms), and Theorem 4's worst-case $\Omega(R_{\max} H\delta)$ floor does not determine it. Partial satisfaction of (C1)–(C3) incurs a finite structural cost for the federation, but training does not collapse.

- **Pattern D (Environment-level collapse: all of (C1)–(C3) fail and the heterogeneity is of the adversarial, observation-silent type constructed in the proof of Theorem 4).** Performance degrades severely, possibly with oscillation, divergence across seeds, or catastrophic forgetting of pre-trained world knowledge. On such constructions Theorem 4's $\Omega(R_{\max} H\delta)$ floor can apply: the clients' optimal actions $\arg\max_a Q_i^\star$ disagree, so their gradients systematically conflict and no single θ can simultaneously satisfy all clients' environment-conditional optima. For LLM agents specifically, these conflicting updates can overwrite the pre-training prior, which would appear as degradation of capabilities the base model originally possessed (not measured in our runs, Appendix O.5).

F.7 Empirical Results: Extended Discussion

Uniform client distribution. Table 1 and Figure 4 answer (Q1) affirmatively: **FEDAGENT matches centralized agent training and outperforms single-client local training, often substantially**, across model scales (`Qwen2.5-{1.5, 3, 7}B-Instruct`, `Llama-3.2-3B-Instruct`), benchmarks (WebShop, ALFWorld), and random client indices (21, 42, 84). On WebShop, we report both the graded task score (`Score`, 0–100 scale) and binary success rate (`Succ.`, %), while on ALFWorld we report per-category and aggregate (`All`) success rates (Appendix J.7). On ALFWorld with `Qwen2.5-1.5B-Instruct`, FEDAGENT reaches 63.0% success rate on `All` versus 12.5–35.9% for local training and 56.2% for centralized. On WebShop with `Llama-3.2-3B-Instruct`, the corresponding success rates are 57.8%, 26.6–39.1%, and 56.8%. Across the twelve backbone×metric cells of Table 1, the federated-vs.-centralized gap is small and mixed in sign, on the order of the seed-level standard deviations: within ± 2.6 pp on both WebShop columns, and between $+0.5$ and $+6.8$ pp on the noisier ALFWorld `All` column, with the largest deviations in FEDAGENT's favor. Individual ALFWorld subtask columns (8–14 test episodes each) vary in opposite directions across runs because sparse-reward RL converges to different per-category skill profiles, whereas on the aggregate `All` column (64 episodes) the federated-vs.-centralized gap stays within 7.3 pp on every backbone, at $+0.5$ to $+6.8$ pp under GRPO and -5.2 to $+7.3$ pp under PPO (Appendix G.2). Local training, in contrast, is not only weaker on average but also depends strongly on the client index: its ALFWorld `All` column varies by 23–31 pp across the three client indices (*e.g.*, 10.9–42.2% for `Qwen2.5-3B-Instruct`), whereas every participant in FEDAGENT receives the same aggregated model, removing this source of variance entirely.

The training-dynamics view (Figure 4, `Qwen2.5-1.5B-Instruct`) also shows the two paradigms reaching end-of-training success rates of ∼55–60% on both benchmarks. On WebShop, centralized leads by up to ∼15 points mid-training on a five-checkpoint moving average (up to 28 points at a single checkpoint) before FEDAGENT closes the gap. On ALFWorld, which shows the higher run-to-run variability typical of agent RL, FEDAGENT matches or slightly exceeds centralized. These results resolve (Q1) in our setting: federated aggregation preserves training quality in this LLM agent RL regime, even though the regime lies outside supervised FL and simple-MDP FRL, the settings where FedAvg was previously shown to match centralized training.

Environment-level heterogeneity. In sharp contrast to §5.2, Figure 6 shows that FEDAGENT **is often non-robust under environment-level heterogeneity**: the five environment variants form a stable → degrade → collapse spectrum. **Catalog Split** gives each client a different subset of products, but the "search → click → buy" optimal policy is the same for every client, so (C2) holds and the structural conditions place the achievable plateau at the homogeneous-environment baseline. Condition (C1) also holds once states are abstracted to their reward-relevant content (Appendix M). PPO attains this plateau at ∼60% (Pattern B), while GRPO plateaus stably at ∼40% (Pattern C, high end). The residual gap is optimization-level slack (ϵ_{opt}, the second, optimizer-dependent factor of §4.4), and no structural floor applies: the optimal action sequence is shared across clients, so the gap reflects how the two optimizers converge when each client's catalog holds different distractor products. **Field-Subset Index and BM25 Reweighting** change how the same query maps to ranked products across clients (different doc-text fields indexed or different BM25 hyperparameters), so the search results differ across clients on states the optimal trajectory visits. (C1) therefore fails, and (C2)

holds only partially because the optimal query and click order differ across clients on some states. Under GRPO, Field-Subset Index and BM25 Reweighting plateau at $\sim$17% and $\sim$33%, respectively (Pattern C). PPO raises the Field-Subset Index plateau to $\sim$42%. For BM25 Reweighting, PPO does not raise the level, and the PPO curve fluctuates more. Over the last ten checkpoints, the BM25 Reweighting mean is 30% under PPO and 33% under GRPO, while its range is 14–46% under PPO and 29–37% under GRPO. **Lookalike Injection and Rank Wrapper** change the clients' top-10 search results the most (mean between-client overlaps of 0.36 and 0.10, Appendix M.9): Lookalike Injection jointly perturbs content and matching by injecting per-client lookalike products, while Rank Wrapper post-processes search results (with the identity, a shuffle, an inversion, or, on about half of the queries, a random list) so the same query returns different orderings across clients. Both break (C1), but they differ on (C2). The rendered `LK_` prefix (and, for the price configurations, the \$9999 attack price of Appendix M.7) makes one client-independent rule (never buy a lookalike) optimal for every client, so (C2) holds for Lookalike Injection, and its residual degradation is optimization slack in learning to inspect before buying. Rank Wrapper admits no shared rule, and its (C3) effectively fails: identifying the client's reordering would require cross-query in-context inference that the 1.5B backbone does not consistently realize, especially within the two-step history window of Appendix J.5. This difference is one reason why only Rank Wrapper behaves like Theorem 4's worst-case construction. Under GRPO, **Rank Wrapper** fails to learn (Pattern D). Its success rate never exceeds 13% and stays at 5–10% for most of training, close to its starting value, and its checkpoint-to-checkpoint changes are the smallest of the twelve curves in Figure 6. Part of this low success rate comes from its invert clients, which cannot reach the goal product within 15 steps (Appendix M.8). **Lookalike Injection**, despite being the more elaborate construction, leaves a cue the agent can exploit: performance degrades without collapsing and ends at $\sim$34% (Pattern C).

Under PPO, Lookalike Injection rises to $\sim$46% and stays in Pattern C, while Rank Wrapper rises only to $\sim$20%, the lowest of the six PPO curves and about one third of the $\sim$60% baseline, so it stays in Pattern D. **The pattern of each variant thus follows (C1)–(C3) under both optimizers, and the optimizer changes only whether a variant with a satisfied condition reaches Pattern B or stays in Pattern C.** This division of roles follows the two factors of §4.4: (C1)–(C3) set the achievable plateau and do not depend on the optimizer, whereas the optimizer slack ϵ_{opt} determines whether that plateau is attained. PPO raises the attained plateaus of Catalog Split, Field-Subset Index, Lookalike Injection, and, slightly, Rank Wrapper, but leaves the BM25 Reweighting level unchanged (see above).

G More Experimental Results

This appendix collects experimental extensions of §5: a hyperparameter sensitivity analysis of FEDAGENT's federated configuration (Appendix G.1), and PPO counterparts of the GRPO results in the main text, organized into effectiveness under a uniform client distribution (Appendix G.2) and Pattern A under task-level heterogeneity (Appendix G.3).

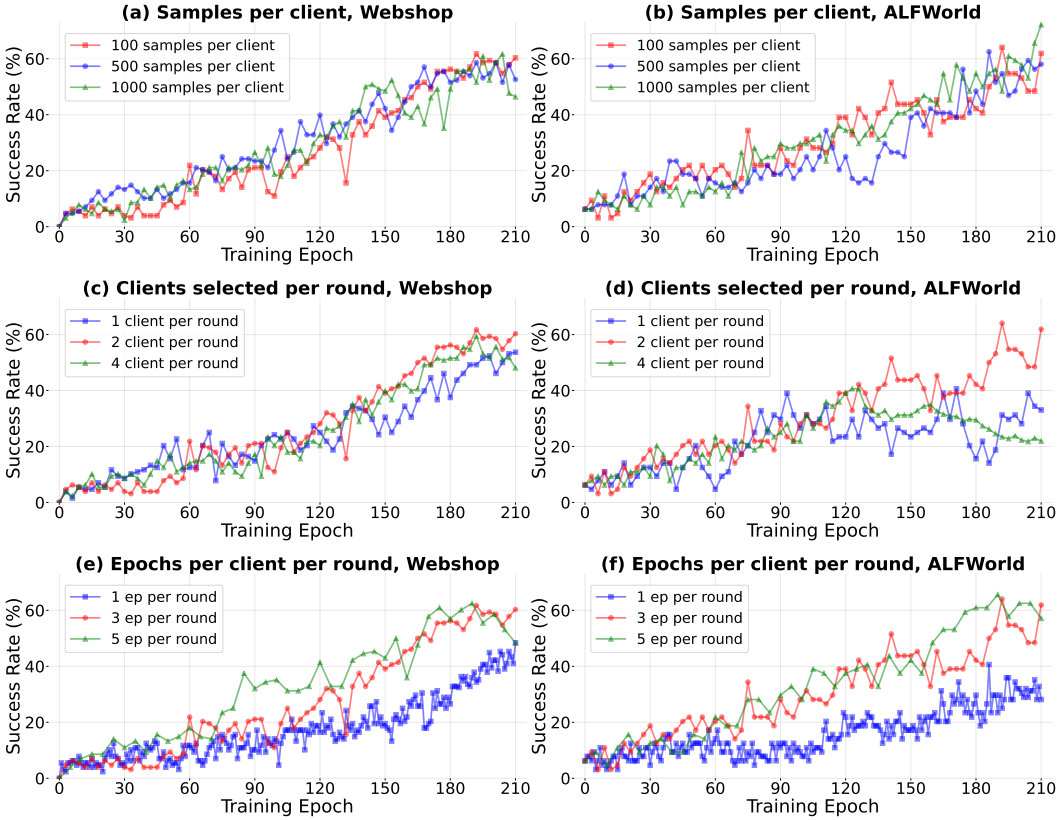


Figure 8: **Sensitivity of FEDAGENT to Federated Configuration Hyperparameters** on WebShop and ALFWorld. Panels (a,b) show the effect of tasks per client $|X_i|$, (c,d) of clients per round M, and (e,f) of local epochs per round E. Here $|X_i|$ is the number of tasks in each client's pool. The red curves use the setting of the main text ($|X_i| = 100$, $M = 2$, $E = 3$). Every run trains for 210 local epochs, so $E = 1$, 3, and 5 correspond to 210, 70, and 42 rounds. Across the tested values, E and M affect final performance most, while $|X_i|$ has a smaller, benchmark-dependent effect.

G.1 Hyperparameter Sensitivity of FEDAGENT's Federated Configuration

This subsection examines how the performance of FEDAGENT depends on the three standard federated optimization hyperparameters that govern its behavior and are *not part of the heterogeneity analysis* of §3–§4: tasks per client $|X_i|$, clients per round M, and local epochs per round E. Coverage heterogeneity varies $|X_i|$ across clients, whereas here all clients share one value of $|X_i|$. The goal is twofold: (i) to justify the $M = 2$, $E = 3$, $|X_i| = 100$ configuration used throughout the main text, and (ii) to provide design guidance for practitioners deploying FEDAGENT on new agent benchmarks.

Setup. We vary three dimensions: (i) **tasks per client** $|X_i| \in \{100, 500, 1000\}$, controlling each client's overall exploration breadth; (ii) **clients per round** $M \in \{1, 2, 4\}$, controlling federation parallelism and the gradient-averaging signal; and (iii) **local epochs per round** $E \in \{1, 3, 5\}$, controlling the depth of local optimization between aggregations. We hold the total local-epoch budget fixed at 210 across all configurations (so $E \in \{1, 3, 5\}$ corresponds to $T \in \{210, 70, 42\}$ communication rounds, respectively). All experiments use `Qwen2.5-1.5B-Instruct` and

Method	ALFWorld							WebShop	
	Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Qwen2.5-1.5B-Instruct									
Local (Client 21)	35.7	50.0	8.3	44.4	36.4	40.0	34.4	62.3	26.6
Local (Client 42)	21.4	37.5	16.7	22.2	27.3	30.0	25.0	70.8	34.4
Local (Client 84)	7.1	12.5	33.3	11.1	9.1	10.0	14.1	60.1	40.6
Centralized	$54.8_{\pm 4.1}$	**83.3**$_{\pm 7.2}$	**86.1**$_{\pm 4.8}$	$18.5_{\pm 12.8}$	$15.2_{\pm 10.5}$	**86.7**$_{\pm 5.8}$	$57.3_{\pm 5.0}$	$77.3_{\pm 3.9}$	$55.2_{\pm 2.9}$
FEDAGENT	**83.3**$_{\pm 10.9}$	$45.8_{\pm 14.4}$	$44.4_{\pm 9.6}$	**74.1**$_{\pm 6.4}$	**84.8**$_{\pm 5.2}$	$46.7_{\pm 11.5}$	**64.6**$_{\pm 1.8}$	**77.5**$_{\pm 3.3}$	**59.9**$_{\pm 2.4}$
Qwen2.5-3B-Instruct									
Local (Client 21)	7.1	12.5	16.7	11.1	18.2	10.0	12.5	70.9	39.1
Local (Client 42)	42.9	25.0	41.7	44.4	45.5	40.0	40.6	60.1	29.7
Local (Client 84)	21.4	37.5	25.0	22.2	36.4	30.0	28.1	73.2	45.3
Centralized	**85.7**$_{\pm 7.1}$	$50.0_{\pm 12.5}$	**75.0**$_{\pm 14.4}$	**81.5**$_{\pm 6.4}$	$33.3_{\pm 13.9}$	$70.0_{\pm 10.0}$	$67.2_{\pm 6.2}$	**82.7**$_{\pm 4.7}$	**60.9**$_{\pm 3.3}$
FEDAGENT	$78.6_{\pm 12.4}$	**79.2**$_{\pm 7.2}$	$41.7_{\pm 8.3}$	$51.9_{\pm 12.8}$	**72.7**$_{\pm 9.1}$	**83.3**$_{\pm 5.8}$	**67.7**$_{\pm 0.9}$	$82.2_{\pm 7.0}$	$58.9_{\pm 5.1}$
Qwen2.5-7B-Instruct									
Local (Client 21)	42.9	25.0	50.0	22.2	27.3	40.0	35.9	70.4	42.2
Local (Client 42)	50.0	37.5	16.7	44.4	45.5	50.0	40.6	60.3	48.4
Local (Client 84)	21.4	12.5	41.7	11.1	9.1	10.0	18.8	75.3	37.5
Centralized	$88.1_{\pm 8.2}$	**79.2**$_{\pm 14.4}$	**77.8**$_{\pm 9.6}$	**77.8**$_{\pm 11.1}$	**87.9**$_{\pm 5.2}$	$73.3_{\pm 11.5}$	**81.2**$_{\pm 6.8}$	**89.6**$_{\pm 3.1}$	$69.8_{\pm 3.3}$
FEDAGENT	**90.5**$_{\pm 4.1}$	$75.0_{\pm 12.5}$	$55.6_{\pm 4.8}$	$74.1_{\pm 6.4}$	$78.8_{\pm 10.5}$	**80.0**$_{\pm 10.0}$	$76.0_{\pm 5.5}$	$88.2_{\pm 5.5}$	**70.8**$_{\pm 6.1}$
Llama-3.2-3B-Instruct									
Local (Client 21)	35.7	37.5	41.7	**44.4**	9.1	50.0	35.9	68.3	40.6
Local (Client 42)	7.1	12.5	8.3	11.1	36.4	10.0	14.1	67.5	25.0
Local (Client 84)	14.3	25.0	16.7	22.2	18.2	20.0	18.8	60.4	32.8
Centralized	$59.5_{\pm 14.9}$	**45.8**$_{\pm 14.4}$	**80.6**$_{\pm 12.7}$	$22.2_{\pm 11.1}$	**63.6**$_{\pm 9.1}$	$26.7_{\pm 5.8}$	$52.1_{\pm 2.4}$	$78.8_{\pm 7.8}$	**55.7**$_{\pm 8.6}$
FEDAGENT	**71.4**$_{\pm 14.3}$	$20.8_{\pm 7.2}$	$58.3_{\pm 14.4}$	$37.0_{\pm 12.8}$	$60.6_{\pm 13.9}$	**83.3**$_{\pm 11.5}$	**57.8**$_{\pm 4.1}$	**78.9**$_{\pm 4.8}$	$54.2_{\pm 5.6}$

Table 3: **Performance Comparison on ALFWorld and WebShop under PPO**. This table is the PPO counterpart of Table 1. We report the mean and the sample standard deviation over three runs for Centralized Training and FEDAGENT. The three FEDAGENT runs use training-goal shuffle seeds 42, 21, and 84. For ALFWorld, we report the Success Rate (%). For WebShop, we report both the Task Score (0–100, the WebShop convention of $100\times$ the mean graded goal-match score $r \in [0, 1]$) and the Success Rate (%).

GRPO as the policy optimizer, with a uniform task partition (no heterogeneity), so any sensitivity is attributable to the federated configuration alone.

Findings. Figure 8 shows three sensitivity trends. **(1) The local-epoch count E matters substantially.** Moving from $E = 1$ to $E \in \{3, 5\}$ yields large performance gains, particularly after epoch $\sim$100. These gains indicate that shallow local updates do not fully exploit FEDAGENT's local-SGD-style optimization and that more local epochs amortize the aggregation cost well. $E = 3$ and $E = 5$ end within a few pp on WebShop (panel e), whereas on ALFWorld (panel f) $E = 5$ stays about 8 pp higher over the last ten checkpoints of this single run. Because every run uses the same 210 local epochs, $E = 5$ needs fewer communication rounds at equal total compute. We keep $E = 3$ as the default, and $E = 5$ is a reasonable choice when communication is costly. **(2) Clients-per-round M performs best at an intermediate value.** On ALFWorld, $M = 2$ outperforms both $M = 1$ (a single client per round may give too little diversity in the aggregated update) and $M = 4$ (averaging more divergent client updates may dilute the global signal). These runs isolate neither cause. The effect is milder on WebShop. **(3) The number of tasks per client $|X_i|$ has the smallest, benchmark-dependent effect.** On WebShop the three pool sizes $|X_i| \in \{100, 500, 1000\}$ stay within $\sim$ 3–5 pp of each other (with 100 best), whereas on ALFWorld larger pools help somewhat (1000 best by $\sim$ 5–8 pp). About 100 tasks already provide substantial per-client exploration on both benchmarks. We default to $|X_i| = 100$ because it performs well at low cost, although performance has not strictly saturated at this value. These findings support the $M = 2$, $E = 3$, $|X_i| = 100$ configuration used in the main text, and we recommend it as a default starting point for new FEDAGENT deployments.

G.2 PPO Results: Effectiveness under Uniform Client Distribution

This subsection reports the PPO counterpart of §5.1 and addresses (Q1) with PPO [106] as the local policy optimizer. PPO estimates advantages with a learned value baseline (critic) and GAE. GRPO [107] uses group-relative advantages without a critic. Both use the same clipped surrogate actor loss in our runs. The goal is to verify that the central finding of the main text extends from GRPO's group-relative advantages to PPO's critic-based advantages: FEDAGENT matches centralized agent training and substantially outperforms single-client local training.

Setup. We use the configuration of §5.1 unchanged except for the local optimizer. The dataset (WebShop / ALFWorld) is partitioned among 100 clients with $|X_i| = 100$ task instructions per client and possible overlap between clients. In each communication round, the server selects $M = 2$ clients, and each selected client trains for $E = 3$ local epochs. The protocol runs for $T = 70$ communication rounds, giving 210 total local epochs. The GRPO sensitivity ablation in Appendix G.1 motivates this $M = 2$, $E = 3$, $|X_i| = 100$ configuration. In each local epoch, the client iteratively samples a minibatch of 64 tasks with replacement from local data. PPO collects one rollout per task ($= 64$ rollout trajectories), matching GRPO's per-epoch trajectory budget of §5.1. The value model is federated alongside the actor: each selected client uploads its critic with its actor, and the server averages and re-broadcasts both. We test four agent backbones, `Qwen2.5-1.5B-Instruct`, `Qwen2.5-3B-Instruct`, `Qwen2.5-7B-Instruct`, and `Llama-3.2-3B-Instruct`, on both WebShop and ALFWorld. We compare against two baselines: **Centralized Agent Training** (full dataset, same total epochs, PPO optimizer) and **Local Agent Training** (one client's dataset, with results reported separately for the three randomly selected client indices $21, 42, 84$). Federated and centralized rows report the mean and the sample standard deviation (subscripts) over three runs. The three federated runs use training-goal shuffle seeds 42, 21, and 84, which change which tasks each client holds. Table 3 reports per-subtask success rate on ALFWorld (Pick, Look, Clean, Heat, Cool, Pick2, plus the All-task aggregate) and Task Score / Success Rate on WebShop.

Findings. Across all four backbones and both benchmarks, Table 3 shows that **FEDAGENT with PPO is never more than** 5.2 **pp below centralized PPO on any aggregate metric and outperforms single-client local PPO on every aggregate metric**, in agreement with the GRPO conclusion of §5.1. The aggregate metrics are the ALFWorld All column and the two WebShop metrics. For example, on ALFWorld with `Qwen2.5-1.5B-Instruct`, FEDAGENT reaches 64.6% All-task success versus 57.3% for Centralized and 14.1–34.4% across the three Local clients. With `Qwen2.5-7B-Instruct`, the corresponding numbers are 76.0%, 81.2%, and 18.8–40.6% (here Centralized has its largest lead on any aggregate metric, which lies within one seed-level standard deviation). On WebShop with `Llama-3.2-3B-Instruct`, FEDAGENT reaches 54.2% Success Rate versus 55.7% for Centralized and 25.0–40.6% for Local clients. As under GRPO, the Local rows also depend strongly on the client index, varying by 20–28 pp on the ALFWorld All column. Two additional observations follow. **(1) The federated-vs.-centralized gap is consistently within a few points on the WebShop Success Rate column** ($+4.7$, -2.0, $+1.0$, -1.5 pp on Qwen-1.5B/3B/7B and Llama-3B respectively). This result empirically resolves (Q1) for PPO with the same conclusion as GRPO: model averaging over per-client local PPO updates recovers the centralized-PPO learning signal under partial participation. **(2) The per-subtask ALFWorld columns vary in both directions, and by far more than their seed-level standard deviations** (*e.g.*, under Qwen2.5-1.5B FEDAGENT leads by 69.6 pp on Cool yet lags behind by 41.7 pp on Clean): the two paradigms converge to visibly different per-type skill profiles, a familiar outcome in sparse-reward agent RL where distinct skill mixes reach similar overall success. We therefore interpret Table 3 only at the aggregate level, where the two paradigms differ by at most 7.3 pp, and draw no per-skill conclusions. The effectiveness result of §5.1 is thus not specific to GRPO.

G.3 PPO Results: Pattern A under Task-Level Heterogeneity

This subsection reports the PPO counterpart of §5.2, addressing whether **Pattern A** (the federated curve under extreme task-level heterogeneity tracks the near-uniform baseline at convergence) holds when the local optimizer is PPO instead of GRPO. Recall that the population objective behind Pattern A does not depend on the optimizer: by Theorem 1(i) it reduces to a single mixture expectation over $\bar{\mathcal{D}}_\tau = \frac{1}{N}\sum_i \mathcal{D}_{\tau_i}$, so the objective that every policy-gradient estimator targets is invariant to how tasks are split across clients, as long as the split keeps $\bar{\mathcal{D}}_\tau$ fixed. Whether a given optimizer

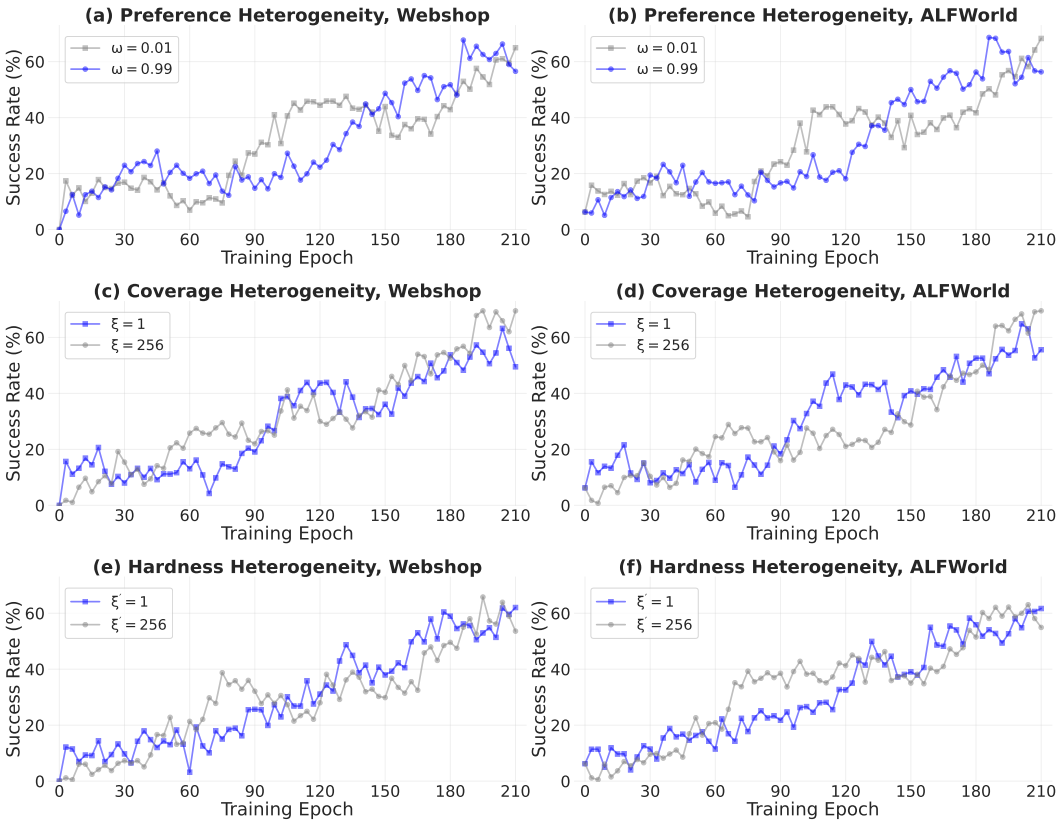


Figure 9: **Training Dynamics of FEDAGENT (PPO) under Task-Level Heterogeneity**. This figure is the PPO counterpart of Figure 5. The federated curve under extreme task-level heterogeneity (blue) converges close to the near-uniform baseline (gray) across all three sub-types (preference, coverage, hardness) and both benchmarks (WebShop, ALFWorld). The largest residual gap is on coverage. In all three sub-types, the extreme curve dips transiently below the baseline mid-training (around epochs 60–120). It rejoins the baseline on preference and hardness and stays about 7–10 points lower on average on coverage over the last ten checkpoints (single runs). These results are consistent with **Pattern A** under PPO.

reaches the optimum of this objective equally well under every split is an empirical question, which the present experiment answers for PPO, complementing the GRPO results in Figure 5. Together with the environment-level PPO results already reported in panel (b) of Figure 6, this experiment completes the PPO coverage of the four-pattern spectrum.

Setup. We repeat the setup of §5.2 except for the local optimizer. PPO replaces GRPO as the local policy-gradient estimator, and with it the batch of each local epoch changes from 8 tasks with 8 rollouts each to 64 tasks with one rollout each. The three task-level sub-types from §3.2 are varied independently via PREFERENCEPARTITION(ω), COVERAGEPARTITION(ξ), and HARDNESSPARTITION(ξ'), with each hyperparameter varied from a near-uniform setting (gray: $\omega = 0.01$, $\xi = 256$, $\xi' = 256$) to an extremely heterogeneous setting (blue: $\omega = 0.99$, $\xi = 1$, $\xi' = 1$). See Appendices L.3.1, L.3.2, and L.3.3 for the resulting per-client distributions on WebShop and ALFWorld. The federated configuration is otherwise identical to that of the previous subsection: 100 clients, $M = 2$ per round, $E = 3$ local epochs, $T = 70$ rounds, $|X_i| = 100$ for preference and hardness (and $[L_{\min}, L_{\text{avg}}, L_{\max}] = [100, 500, 1000]$ for coverage), batch of 64 tasks per local epoch, `Qwen2.5-1.5B-Instruct` backbone. The heterogeneity construction and partition algorithms are unchanged, so any difference relative to the GRPO curves of Figure 5 comes from the local-optimizer choice or from run-to-run variability.

Findings. Figure 9 shows that **Pattern A holds under PPO across all three sub-types and both benchmarks**: at convergence, the blue curve (extreme heterogeneity) tracks the gray curve (near-uniform) closely on the preference and hardness panels and stays about 7–10 points lower on average over the last ten checkpoints on coverage. On WebShop (panels a, c, e), both curves plateau around 55–65% success rate (the extreme-hardness curve in panel e reaches $\sim$ 62% by the final epochs). On ALFWorld (panels b, d, f), both converge to around 55–65% All-task success. The coverage separation is the largest residual gap of the three sub-types. It comes from single runs, so we cannot tell how much of it reflects run-to-run variability. Beyond the convergence-level match, two further features of the PPO curves are apparent. **(1) All three sub-types show a transient mid-training dip.** The extreme-heterogeneity curves do not start slowly. Averaged over 30-epoch windows, they stay within 4 points of the near-uniform curves, or ahead of them, during the first 60 epochs on every panel, and they lead on both hardness panels. Each sub-type then dips by 11–18 points and recovers within 30–60 epochs: preference over epochs 90–120 (-18 on WebShop, -16 on ALFWorld), coverage over epochs 60–90 (-13, -11), and hardness over epochs 60–90 (-14, -12, lasting through epoch 120 on ALFWorld). Because the dip has a similar size in all three sub-types, we do not attribute it to the sparse early rewards of hard clients. The dip is the kind of transient drift that (A-ii) in Appendix O.2 allows, and with one run per curve we cannot separate it from run-to-run variability. After the dip, the extreme curves rejoin the baseline on preference and hardness, whereas on coverage we cannot tell from single runs whether the dip contributes to the 7–10 point gap of the converged plateau. The drift also has the opposite sign at times: after recovering, the extreme curves lead the baseline by up to about 20 points for a while (*e.g.* panel d, epochs 100–140). **(2) The PPO curves vary less than the GRPO curves.** Compared to the GRPO curves of Figure 5, the PPO curves are visibly smoother, especially late in training, where the GRPO curves exhibit sharp drops between adjacent checkpoints (*e.g.* the late-training dips in the WebShop coverage and hardness panels of Figure 5). Both optimizers use the same clipped actor loss in our runs (clip ratio 0.2, dual-clip 3.0), so clipping cannot explain the difference. The PPO-specific components are the learned critic with GAE, value-loss clipping, and a batch of 64 tasks with one rollout each instead of GRPO's 8 tasks with 8 rollouts each. A plausible cause is that the critic baseline and the larger task batch give lower-variance advantages than GRPO's per-group normalization, under which a task whose rollouts all receive the same reward contributes zero advantage. Our runs do not isolate which PPO component is responsible.

Combined with the environment-level PPO results in panel (b) of Figure 6, the present experiment is consistent with a pattern structure that does not depend on the optimizer. **Across both optimizers tested, the four-pattern spectrum is structural: the heterogeneity level (task vs. environment) determines the active regime (Pattern A vs. the B/C/D family), and the choice between GRPO and PPO moves variants only *within* the B/C/D family**. Within the environment-level regime, the optimizer slack ϵ_{opt} determines only whether a variant with a satisfied condition reaches Pattern B or stays in Pattern C. As discussed in §5.3 and Appendix F.7, PPO moves Catalog Split from Pattern C to Pattern B and raises most Pattern C plateaus (*e.g.* Lookalike Injection). BM25 Reweighting is the exception, with no gain in level under PPO (Appendix F.7). Rank Wrapper, which breaks all of (C1)–(C3), reaches $\sim$20% under PPO but stays in Pattern D. The underlying B/C/D structure and the task-level Pattern A hold under both optimizers.

H Extended Related Work

H.1 Supervised Federated Learning

Federated learning (FL) was introduced by McMahan et al. [86] with the Federated Averaging (FEDAVG) algorithm, which enables multiple clients to collaboratively train a shared model by communicating only model updates, so raw data is never shared. In the canonical FL protocol, a central server coordinates N clients over T communication rounds. In each round, the server broadcasts the current global model $\boldsymbol{w}^t$ to selected clients. Each selected client k performs E epochs of stochastic gradient descent on its local dataset $\mathcal{D}_k$ to obtain an updated model $\boldsymbol{w}_k^{t+1}$. The server then aggregates these local models via a weighted average $\boldsymbol{w}^{t+1} = \sum_k p_k \boldsymbol{w}_k^{t+1}$, where $p_k = |\mathcal{D}_k| / \sum_j |\mathcal{D}_j|$. This paradigm preserves data privacy by never transmitting raw data, and has since motivated a large body of research addressing its key challenges: *statistical heterogeneity* arising from non-i.i.d. client data distributions, *systems heterogeneity* due to varying client computation and communication capabilities, and *privacy* concerns requiring formal guarantees beyond the implicit protection of keeping data local [51, 64]. The majority of FL research has focused on supervised classification tasks, establishing both theoretical foundations and practical algorithms.

Data Heterogeneity. A central challenge in supervised FL is *data heterogeneity*, where clients possess non-identically distributed (non-i.i.d.) data. This heterogeneity has been characterized along several axes, including label distribution skew [43], feature distribution shift [68], and quantity imbalance [33, 168]. Convergence analyses have been developed for local SGD under i.i.d. data [120] and for FedAvg under non-i.i.d. data [67, 146], establishing how heterogeneity slows convergence and motivating algorithm improvements. To address client drift, various methods have been proposed. FEDPROX [65] adds a proximal regularization term to the local objective. SCAFFOLD [52] corrects for client drift using control variates. FEDNOVA [126] normalizes local updates to handle heterogeneous local computation. FEDDYN [2] augments each client's objective with dynamic linear and quadratic penalty terms. MOON [63] uses model-contrastive learning to align local and global representations.

Server-Side Optimization and Aggregation. Beyond local corrections, server-side optimization has also received attention. Reddi et al. [101] proposed FEDOPT, a general framework applying adaptive server-side optimizers (FedAdam, FedAdagrad, FedYogi) to federated learning. FEDMA [124] aligns neural network layers via matched averaging, while knowledge distillation-based approaches such as FEDDF [73] aggregate client models via ensemble distillation on the server. FEDMD [62] enables FL with heterogeneous model architectures by distilling averaged logits on a shared public dataset.

Personalization. Personalized federated learning methods further address heterogeneity by learning client-specific models while still benefiting from collaborative training. Per-FedAvg [26] formulates personalized FL as a MAML-style meta-learning problem. pFedMe [24] decouples personalized model optimization from global model learning using Moreau envelopes. FedRep [20] learns a shared representation (body) with client-specific heads, and FedBABU [90] trains only the shared body under a fixed random head and fine-tunes the head for each client afterward. Ditto [66] balances global and local objectives for fair and robust personalization. Other approaches include FedPer [5] with personalization layers and APFL [22] with adaptive mixture weights. More recently, Hot-Pluggable FL [108] proposes a plug-in module framework that bridges general and personalized FL via dynamic module selection from a shared modular store, and FOL [44] introduces a practical one-shot personalized FL framework in which each pair of neighboring clients exchanges models only once. Clustered FL has also advanced with HCFL [40], which proposes a four-tier framework that systematically integrates and extends clustering strategies. MultiFCL [170] addresses federated continual learning by orchestrating multi-scale expertise with lightweight frozen adapters to prevent catastrophic forgetting across sequential tasks.

Communication Efficiency and Privacy. Reducing communication overhead has also been extensively studied. Gradient compression [55, 74], quantization [4], and sketching [102] can substantially reduce communication cost. Privacy-preserving techniques such as differential privacy [36, 87] have been integrated into FL to provide formal privacy guarantees.

Positioning of Our Work. Supervised FL primarily addresses heterogeneity along label/feature/quantity-skew axes over static datasets, with methods that modify the optimizer (client-drift correction, normalization, contrastive alignment) or the model (personalization). Our work differs structurally in two ways. First, we operate on *LLM agents* performing multi-step decision-making with natural-language state and action spaces, so heterogeneity concerns *which task the agent is asked to perform* and *which environment it acts in*. Second, we organize Agent Heterogeneity along a *two-level* decomposition (task vs. environment) grounded in the *Input-Dynamics Asymmetry* of LLM agent MDPs, and we prove an asymmetric robustness result characteristic of this asymmetry. Existing supervised-FL heterogeneity tools do not directly target this asymmetry.

H.2 Federated Reinforcement Learning

Federated reinforcement learning (FRL) extends federated learning to sequential decision-making problems, where multiple agents learn a shared policy by training in their own environments without sharing raw trajectories [96]. In contrast to supervised FL, which operates on static datasets with well-defined input–label pairs, FRL must contend with the unique challenges of RL: agents interact with potentially different Markov Decision Processes (MDPs) $\mathcal{M}_k = (\mathcal{S}_k, \mathcal{A}_k, P_k, R_k, \gamma)$, generating non-stationary trajectory data that depend on the evolving policy. This setting introduces several distinctive difficulties beyond standard FL, including (i) *non-stationarity* of the data distribution as the policy changes during training, (ii) *client heterogeneity* in transition dynamics P_k (environment heterogeneity in our terms) or in reward functions R_k, (iii) the need for *exploration-exploitation tradeoffs* that must be coordinated across agents, and (iv) *credit assignment* over temporally extended trajectories. These challenges have motivated a growing body of theoretical and algorithmic work.

Foundations and Theory. Early work by Zhuo et al. [194] proposed FedRL, in which agents build deep RL models in a federated manner without directly transferring their data or models, adding Gaussian noise to the shared information. Jin et al. [50] proposed QAvg and PAvg for FRL with heterogeneous transition dynamics and proved that they converge to suboptimal solutions whose gap depends on the degree of environment heterogeneity. Khodadadian et al. [54] analyzed the convergence of federated temporal-difference learning with linear speedup under Markovian sampling. Woo et al. [145] established sample complexity guarantees with linear speedup for federated Q-learning and showed that heterogeneity across agents relaxes the single-agent coverage requirement: the agents only need to cover the state-action space collectively. More recently, Zheng et al. [184] proposed the FedQ-Hoeffding and FedQ-Bernstein algorithms, which achieve linear regret speedup with logarithmic communication cost in tabular episodic MDPs, and Zheng et al. [185] further achieved almost minimax-optimal regret via reference-advantage decomposition. Labbi et al. [58] extended UCBVI to the federated setting with communication-efficient regret minimization. Beyond the tabular setting, Liang et al. [70] proposed Fed-LSVI, the first provably efficient federated algorithm for online RL with linear function approximation. Fed-LSVI exchanges only compressed sufficient statistics and incurs logarithmic communication cost. Maity and Mitra [84, 85] further showed that federated Q-learning can retain the statistical gains of collaboration even when a fraction of the agents are adversarial.

Federated Policy Gradient Methods. Several works have developed federated policy optimization algorithms. Yang et al. [162] developed federated natural policy gradient (NPG) and natural actor-critic (NAC) methods with the first near dimension-free global convergence guarantees among policy optimization methods for federated multi-task RL. Lan et al. [59] proposed AFedPG, an asynchronous FRL framework that achieves linear speedup under heterogeneous computing. Zhu and Gong [193] introduced a single-loop federated actor-critic method that jointly considers both federated policy evaluation and improvement. Fan et al. [27] proposed the first Byzantine fault-tolerant FRL framework, whose convergence is guaranteed when fewer than half of the agents are faulty or adversarial, and proved that its sample efficiency improves with the number of agents.

Heterogeneous Environments. Handling environment heterogeneity across clients is a key challenge. Wang et al. [125] proposed momentum-based federated policy gradient methods that converge to a stationary point regardless of the magnitude of environment heterogeneity. Xiong et al. [157] introduced personalized FRL with shared representations, proving the first linear speedup result in the personalized FRL setting. Liu et al. [76] explored lifelong federated RL for cloud robotic systems,

where heterogeneous robots share knowledge to accelerate learning. More recently, FedGuide [42] aggregates diffusion priors over local behavior with an optimal-transport mixture of experts (OT-MoE). The OT-MoE aggregation preserves heterogeneous behavior modes across clients.

Applications. FRL has been applied to diverse domains including autonomous driving [69], edge caching for the Internet of Things [134], and healthcare through federated offline RL [187]. Hwang and Hong [46] proposed FedRQ for tabular FRL with proven asymptotic convergence under environment heterogeneity and extended it to continuous state spaces via expectile loss. A comprehensive survey by Cheruiyot et al. [18] provides a unified perspective on the connections between federated RL, cooperative decentralized RL, and noncooperative multi-agent RL paradigms.

Positioning of Our Work. Beyond the LLM-specific challenges that prior FRL work does not face (long natural-language action sequences, sparse episode-level rewards, token-level credit assignment), our work differs from prior FRL work in two structural ways. First, we distinguish two levels of client heterogeneity: task-level variation is observable through the policy's input channel, and environment-level variation is implicit in the dynamics. This distinction replaces prior FRL's uniform treatment of all non-i.i.d. variation. Second, prior FRL heterogeneity analyses either target the *convergence rate* of the federated optimizer [125] or bound the suboptimality of specific averaging schemes in terms of transition heterogeneity [50]. Our analysis characterizes the *quality of the federated solution* as a structural property of the two heterogeneity levels: task-level heterogeneity admits a vanishing $\sqrt{\cdot}$ slack, whereas worst-case environment-level heterogeneity admits an irreducible $\Omega(R_{\max} H \delta)$ gap that no fast-rate optimizer can close. The two kinds of analysis are orthogonal and can be combined.

H.3 Federated Learning for LLMs

The intersection of federated learning and large language models (LLMs) has recently attracted growing attention [57, 169, 177]. As LLMs become increasingly central to real-world applications, the need to fine-tune them on distributed private data, such as enterprise documents, clinical records, and user interaction logs, makes federated learning a natural paradigm. However, applying FL to LLMs introduces unique challenges beyond classical FL settings: (i) the *massive parameter count* of modern LLMs (7B–70B+ parameters) makes communicating full model updates costly, motivating parameter-efficient methods; (ii) *heterogeneous client capabilities* mean that some clients can only fine-tune small adapters while others may handle larger updates; (iii) LLM training objectives span *supervised instruction tuning, preference optimization, and reinforcement learning from human feedback (RLHF)*, each requiring different federated protocols; and (iv) *privacy requirements* are often stricter for LLMs due to the sensitivity of natural language data and the risk of memorization. These challenges have motivated rapid development of federated LLM training methods [141, 151].

Federated Instruction Tuning. A natural first step is to apply FL to supervised instruction tuning. Zhang et al. [177] proposed FedIT for federated instruction tuning with LoRA adapters on heterogeneous instruction datasets. Ye et al. [169] introduced OpenFedLLM, a training pipeline supporting federated supervised fine-tuning (SFT) and instruction tuning across diverse datasets. A key challenge in federated instruction tuning is *data quality and selection*: FedHDS [99] selects representative subsets at both intra- and inter-client levels without sharing raw data, achieving a 10.72% improvement in ROUGE-L on unseen tasks while using less than 1.5% of the data. FedDQC [25] introduces an instruction-response alignment metric for efficient client-side quality evaluation and designs a quality-aware hierarchical training framework that progressively fine-tunes from high- to low-quality data. To address *data scarcity*, PPFedIT [181] uses LLMs to generate synthetic data that enriches scarce local data, trains the shared global model on synthetic data and a local model on private data (parameter isolation), and mixes global and local parameters before upload to mitigate data extraction attacks. For *communication efficiency* beyond LoRA-based methods, FedKSeed [98] employs zeroth-order optimization with a finite set of random seeds, reducing communication to under 18 kilobytes per round while enabling full-parameter tuning of billion-sized LLMs. FedMeZO [75] provides the first convergence analysis of memory-efficient zeroth-order optimization in federated LLM tuning, demonstrating faster convergence than first-order FedAvg with substantially reduced GPU memory requirements. FedAMoLE [180] proposes a personalized FL framework featuring a heterogeneous mixture-of-LoRA-experts module with a reverse-selection-based expert assignment strategy that allows domain experts to select aligned clients. Pilot [156] extends federated instruction tuning to the multimodal setting, building a framework for federated vision-language model fine-tuning

that handles heterogeneous multimodal data across clients. FedProxy [29] replaces the lightweight adapters of offsite-tuning with a proxy small language model compressed from the proprietary LLM, combining interference-mitigating aggregation with training-free fusion back into the LLM to narrow the performance gap between offsite-tuning and centralized training while keeping model IP and client data protected.

Parameter-Efficient Federated Fine-Tuning. Since communicating full LLM parameters is prohibitively expensive, parameter-efficient methods have become central. Sun et al. [121] proposed FFA-LoRA, which freezes the randomly initialized matrix A and only fine-tunes the zero-initialized matrix B, halving communication and computation costs while working well under differential privacy. Wang et al. [135] introduced FLoRA, a stacking-based aggregation method that achieves noise-free aggregation and supports heterogeneous LoRA ranks. Bai et al. [7] proposed FlexLoRA, which dynamically adjusts local LoRA ranks to use diverse client resources via SVD-based weight redistribution. Guo et al. [38] proposed FedSA-LoRA, which shares only the A matrices for server aggregation while keeping B matrices local, and Yan et al. [159] introduced FRLoRA, performing global updates in a higher-rank parameter space via residual low-rank adaptation. Cho et al. [19] proposed HetLoRA, which allows clients to use different LoRA ranks that match their individual system resources. Hyeon-Woo et al. [47] re-parameterize weight layers using low-rank weights followed by the Hadamard product, achieving 3–10$\times$ lower communication cost. More recently, FedEx-LoRA [116] achieves exact LoRA aggregation by adding a residual error term to frozen weights, eliminating the inexactness of traditional federated averaging of LoRA adapters. FedALT [9] departs from FedAvg by maintaining individual LoRA adapters alongside a shared Rest-of-World LoRA component with an adaptive MoE-style mixer. FedLEASE [128] adaptively clusters clients based on representation similarity to allocate domain-specific LoRA experts with a top-M MoE mechanism. pFedGPT [109] hierarchically optimizes LoRA aggregation weights for personalized federated GPT models. EcoLoRA [78] uses round-robin segment sharing where each client uploads only a complementary LoRA segment per round, reducing communication time by up to 79%. FedICU [71] splits client updates into consensus and divergence components, balances them dynamically based on their contribution to global model performance, and uses importance-aware updating to prevent catastrophic forgetting and domain overfitting. Several recent works target rank heterogeneity: FSLoRA [30] lets clients update sketched submatrices of global LoRA modules with client-specific sketching ratios and analyzes how these ratios affect convergence, PreLort [138] organizes adapter dimensions into a prefix-nested hierarchy with segment-wise aggregation, and FedRoRA [129] personalizes rank-heterogeneous federated LoRA by decoupling shared global directions from personalized rank-wise magnitudes. A comprehensive survey on federated LoRA methods was provided by Yang et al. [163].

Federated Preference Optimization and Alignment. For alignment, FedRLHF [28] introduced a federated framework for privacy-preserving RLHF, demonstrating convergence guarantees and personalization capabilities across clients. However, it focuses on *single-turn preference alignment* on small-scale tasks (*e.g.*, movie recommendations) and does not address multi-step agent training. Wu et al. [148] proposed FedBiscuit, which encodes each client's preferences into binary selectors and aggregates them to capture common preferences, establishing the first federated RLHF benchmark with heterogeneous human preference data. Spadea and Seneviratne [118] evaluated Kahneman-Tversky Optimization (KTO) against Direct Preference Optimization (DPO) in federated settings, finding KTO more robust to data heterogeneity. Srewa et al. [119] proposed PluralLLM, using federated learning for pluralistic LLM alignment that preserves group fairness. More recently, FedVPA-GP [56] adapts variational preference learning to FL with a federated mixture prior, so that conflicting user preferences are not averaged out by a single reward model. SPEAR [60] performs online federated LLM fine-tuning from interaction-level binary satisfaction feedback via self-play with advantage-weighted refinement. On the security side, Safe-FedLLM [122] detects malicious clients by probing their local LoRA updates with lightweight classifiers.

Federated LLM Pre-Training and Knowledge Distillation. Sani et al. [103] presented Photon, the first complete system for federated end-to-end LLM pre-training from scratch, training models of up to 7B parameters. FoMoE [104] extends low-communication federated pre-training to mixture-of-experts models by partitioning expert layers across workers, removing the need for a full model replica at every site. Wu et al. [147] proposed FedBiOT, where the server generates a compressed LLM via knowledge distillation and distributes it to clients who fine-tune only a lightweight adapter.

A comprehensive survey by Wei et al. [141] organizes federated learning for reasoning LLMs by the type of training signal, namely signals derived from raw data, learned representations, and preference feedback.

Federated RL Post-Training and Reasoning. Following the success of RL post-training for reasoning [21, 107], recent works apply GRPO in the federated setting. Fed-GRPO [39] uses the reward statistics produced during GRPO training to weight client aggregation, calibrate local objectives, and allocate communication, while FGRPO [13] weights clients by their performance gain relative to their own historical baselines to cope with divergent reward scales on non-i.i.d. data. Both are evaluated only on reasoning benchmarks. Other works improve federated reasoning without RL: FERA [132] is a training-free framework that aggregates client-generated reasoning traces with query-dependent trust weights derived from uncertainty signals, and FedCoT [61] combines chain-of-thought resampling with client-aware LoRA stacking.

Federated Agent Systems. Very recently, a few works have begun to explore the intersection of federated learning and LLM agents. Fed-SE [15] proposes a federated self-evolution framework where agents perform parameter-efficient fine-tuning on filtered high-return trajectories locally and aggregate updates globally within a low-rank subspace, improving average task success rates by 10% over FedIT across five heterogeneous environments. FICAL [149] takes a different approach based on in-context learning. FICAL transmits knowledge compendiums instead of model parameters and adds RAG-based tool learning. EPEAgent [111] introduces embedded privacy-enhancing agents that integrate into the RAG phase of federated multi-agent systems, minimizing data flows by ensuring only task-relevant, agent-specific information is shared. Several works from 2026 also study federated learning for LLM agents. SemFed [83] trains LLM agents with federated RL on ALFWorld and WebShop across heterogeneous LLM backbones. Its clients communicate only semantic policy summaries, which the server aggregates into shared, elite, and anti-patterns and injects back into local training. FedWave [183] enables workflow-aware collaboration among federated LLM experts by combining federated LoRA, a shared mixture-of-experts router, and a DPO stage. FedGUI [133] is a federated GUI-agent benchmark that covers cross-platform, cross-device, cross-OS, and cross-source heterogeneity. In its experiments, platform and OS are the most influential factors. Two other works exchange knowledge derived from agent experience: FederatedSkill [160] exchanges structured skill patches for personalized skill evolution, and Federation over Harness [164] automatically evolves both the client and server harnesses of a distributed agent reasoning pipeline. Yang et al. [161] formulate Federated Agent Optimization as a multi-objective problem that balances agent utility, privacy leakage, and communication cost across policies, memory, tools, rewards, and skills.

Positioning of Our Work. Federated LLM training is mostly limited to supervised fine-tuning, preference optimization, single-turn RLHF, or GRPO-based reasoning [13, 39]. Existing federated LLM-agent methods fine-tune agents on filtered trajectories or with preference objectives [15, 183], exchange semantic policy summaries, knowledge, skills, or harnesses without exchanging model updates [83, 149, 160, 164], protect privacy in the retrieval stage of multi-agent systems [111], or characterize client heterogeneity empirically [133]. None of them provides a *structural* account of when federated agent training is robust to client heterogeneity and when it is not. We close this gap with a federated agent RL framework supporting both GRPO and PPO, a *two-level decomposition* of Agent Heterogeneity grounded in the Input-Dynamics Asymmetry, and the *Asymmetric Robustness Mechanism*. The mechanism maps four observable training-curve patterns to their underlying realizability and structural conditions.

H.4 LLM Agent Reinforcement Learning

LLM-based agents can perform complex interactive tasks requiring multi-step reasoning, tool use, and environment interaction [152]. Unlike standard LLM applications that generate single-turn responses, LLM agents operate in a *closed-loop* fashion: they observe an environment state, generate an action (typically a natural language command or API call), receive feedback from the environment, and iterate over multiple turns until a task is completed or a horizon limit is reached. This agentic paradigm has been studied in diverse domains including web navigation [97, 188], software engineering [48], embodied household tasks [114], and mobile device control [6, 100]. Training LLM agents presents unique challenges compared to standard LLM fine-tuning: (i) the *action space is exponentially large*, as each action is a variable-length sequence of natural language tokens; (ii) rewards are typically

sparse and delayed, provided only at the end of multi-step episodes; (iii) *credit assignment* must operate across both the token level within a single action and the turn level across the full interaction trajectory; and (iv) the agent must learn to balance *exploration* of new strategies with *exploitation* of known successful patterns. Two main paradigms have emerged for agent training: supervised fine-tuning on expert demonstrations and reinforcement learning from environment interaction.

Agent Frameworks. Prompting-based methods such as ReAct [167], Reflexion [112], Chain-of-Thought [139], and Tree of Thoughts [166] improve the reasoning of LLMs and LLM agents without parameter updates, but their performance is bounded by the underlying model's capabilities. Schick et al. [105] demonstrated that LLMs can learn to use external tool APIs in a self-supervised manner. Zhou et al. [186] unified reasoning, acting, and planning via Monte Carlo Tree Search with LM-powered value functions. Multi-agent systems such as AutoGen [150] enable complex workflows through conversable agents.

Agent Training via Supervised Fine-Tuning. SFT methods use expert demonstrations or successful trajectories to train agents via imitation learning [11, 174]. Song et al. [117] proposed ETO, which learns from contrastive pairs of successful expert and failed agent trajectories using DPO. Zhang et al. [178] propose "early experience" as a middle ground between expert demonstrations and full RL, using interaction data generated by the agent's own actions, where the resulting future states serve as supervision without reward signals. However, SFT-based methods are fundamentally limited by the quality and coverage of demonstrations and cannot improve beyond the expert's behavior.

RL Policy Optimization for LLM Agents. RL offers a principled framework for agents to improve through trial-and-error interaction with environments. Traditional RL methods such as PPO [106] have been adapted for LLM agent training, but they face challenges with the vast token-level action space and sparse rewards in multi-step tasks. DeepSeek-R1 [21] showed that RL can incentivize complex reasoning in LLMs and motivated rapid follow-up work. GRPO [107] eliminates the need for a separate critic model by using group-relative advantages. DAPO [171] introduces clip-higher and dynamic sampling techniques that achieve strong reasoning at scale. GiGPO [31] further improves upon GRPO with group-in-group advantage estimation for finer-grained credit assignment. Ahmadian et al. [3] revisit the REINFORCE Leave-One-Out (RLOO) estimator for RLHF and show that it outperforms PPO at a lower computational cost. Zhou et al. [189] proposed ArCHer, a hierarchical multi-turn RL framework, and Putta et al. [95] introduced Agent Q with advanced reasoning for autonomous agents. Wang et al. [136] introduced StarPO, a general trajectory-level agent RL framework, identifying the "Echo Trap" failure mode. Agent-R1 [17] provides a unified and modular agentic RL framework that treats each interaction step as the basic RL transition, supporting both token-level and step-level credit assignment.

Reward Design and Credit Assignment. The design of reward signals and credit assignment strategies is crucial for effective agent RL, given the sparse and delayed nature of episode-level feedback. Process reward models [72, 130] provide step-level supervision that outperforms outcome supervision on math reasoning. iStar [81] jointly optimizes an implicit process reward model with the policy model via multi-turn DPO to obtain implicit step rewards, achieving state-of-the-art performance on WebShop and SOTOPIA. Agentic Reward Modeling [92] combines human preference rewards with verifiable correctness signals from factuality and instruction following. For multi-turn credit assignment, Wei et al. [140] study dense turn-level reward structures, deriving GRPO and PPO versions for terminal, delayed, and per-turn rewards and showing that dense per-turn rewards yield better training dynamics. Wang and Ammanabrolu [131] offer a practitioner's guide that systematically analyzes the design space across environment, reward, and policy dimensions. SWEET-RL [190] uses a critic model with access to training-time information to provide step-level rewards for collaborative reasoning tasks. HiPER [93] introduces hierarchical RL that separates high-level planning from low-level execution, achieving 97.4% on ALFWorld and 83.3% on WebShop.

World Models and Experience Efficiency. A key bottleneck of model-free agent RL is the reliance on expensive real-environment rollouts. An emerging line of work addresses this by integrating *world models* into the agent RL loop or improving *experience efficiency*. SPA [14] first applies a self-experience SFT stage, in which the policy is trained on its own interaction trajectories to estimate the current state and predict state transitions, and then runs standard RL. The learned world model thus serves as a grounded initialization for RL. Dyna-Think [172] integrates planning with an internal

world model into the reasoning-acting loop, trains the policy with Dyna-Think Imitation Learning followed by Dyna-Think Dyna Training, and achieves best-of-n performance comparable to R1-style reasoning with $2\times$ fewer tokens. DynaWeb [23] applies the Dyna architecture to web agents, training a dedicated world model to generate multi-step imagined trajectories. WebWorld [155] trains the first open-web simulator at scale (up to 32B parameters on 1M+ real-world trajectories), supporting 30+ step long-horizon simulations. A 14B agent trained on WebWorld-synthesized trajectories improves by 9.2% on WebArena, matching GPT-4o. Agent World Model [137] proposes a fully synthetic environment generation pipeline scaling to 1,000 code-driven environments, demonstrating that training exclusively in synthetic environments yields strong out-of-distribution generalization. RWML [173] proposes self-supervised action-conditioned world model learning using sim-to-real gap rewards. Combined with task-success rewards, it outperforms direct task-success RL on ALFWorld and τ^2-Bench. On the experience efficiency side, DreamGym [16] introduces the first unified framework for synthesizing diverse agent experiences at scale. DreamGym distills environment dynamics into a reasoning-based experience model and outperforms all baselines by over 30% on non-RL-ready tasks. ERL [110] augments trial-and-error RL with an explicit experience-reflection-consolidation loop, achieving gains of up to 81% in complex multi-step environments. SkillRL [154] bridges raw experience and policy improvement through automatic skill discovery and a hierarchical skill library that co-evolves with the policy during RL.

Agent RL for Web, Code, and Reasoning. RL and related training methods for LLM agents have been applied across diverse domains. For *web agents*, WebRL [97] introduces a self-evolving online curriculum RL framework with an outcome-supervised reward model, improving Llama-3.1-8B from 4.8% to 42.4% success rate on WebArena-Lite. WebAgent-R1 [144] demonstrates that simple end-to-end multi-turn RL with binary success rewards can raise the success rate of Llama-3.1-8B on WebArena-Lite from 8.5% to 44.8%. For *device-control agents*, DigiRL [6] proposes a two-stage offline-to-online RL approach. For *software engineering*, SWE-RL [142] scales RL for LLM reasoning on real-world software evolution data, and Self-Play SWE-RL [143] trains a single agent that injects and repairs bugs through self-play without human-labeled issues or tests. SWE-Gym [91] provides the first training environment, with 2,438 executable task instances, for real-world software engineering agents. Golubev et al. [37] combine rejection fine-tuning with DAPO to train a 72B SWE agent, increasing Pass@1 on SWE-bench Verified from 11% to 39%. For *reasoning and tool use*, Search-R1 [49] extends R1-style RL to search-augmented settings, improving performance over RAG baselines by 24% with Qwen2.5-7B and 20% with Qwen2.5-3B. ARTIST [115] unifies agentic reasoning and tool integration via RL, enabling autonomous tool invocation within multi-turn reasoning chains. ML-Agent [82] applies RL to autonomous machine learning engineering, where a 7B agent trained on only 9 ML tasks achieves performance comparable to agents built on much larger proprietary LLMs (*e.g.*, GPT-5) at significantly lower computational cost. AgentRefine [32] uses instruction tuning to teach agents to self-refine erroneous actions based on environment feedback.

Scaling and Training Infrastructure. Scaling agent RL to multi-turn, multi-task, and long-horizon settings introduces significant engineering and algorithmic challenges. AgentRL [176] presents a scalable multi-turn, multi-task framework with a fully asynchronous generation-training pipeline, cross-policy sampling for exploration, and task advantage normalization for stable multi-task training. LOOP [12] proposes a data- and memory-efficient version of PPO for interactive digital agents, with a 32B agent outperforming OpenAI o1 on AppWorld. AgentGym-RL [153] proposes ScalingInter-RL, which gradually shifts from exploitation to exploration across training, matching or surpassing commercial models on 27 tasks. ReMA [123] decouples reasoning into a meta-thinking agent and a low-level reasoning agent via multi-agent RL. MAGRPO [79] models LLM collaboration as cooperative multi-agent RL with centralized group-relative advantages. MARTI [179] provides an open-source framework for training multi-agent LLM systems with dynamic multi-agent workflows and both rule-based and generative rewards. VAGEN [127] introduces world modeling RL for multi-turn VLM agents.

Benchmarks and Evaluation. Several benchmarks have been established to evaluate agent capabilities, including WebShop [165] for web-based shopping tasks, ALFWorld [114] for embodied household tasks, AndroidInTheWild [100] for mobile device control, SWE-bench [48] for real-world software engineering, WebArena [188] for realistic web navigation, AgentBench [80] for multi-dimensional agent evaluation, and GAIA [88] for general AI assistants. More recently, TheAgentCompany [158] simulates a small software company environment with integrated development and

communication tools. On this benchmark, the best agents solve only 30% of tasks autonomously. AgentGym [152] provides a unified framework featuring 7 scenarios, 14 environments, and 89 tasks for concurrent agent interaction and self-evolution. MultiAgentBench [191] evaluates multi-agent LLM systems across diverse collaboration and competition scenarios with novel milestone-based KPIs.

Positioning of Our Work. The rapid progress in LLM and LLM agent training (spanning policy optimization [107, 171], reward design [81, 140], world-model augmentation [14, 155], and scalable infrastructure [153, 176]) has been made with centralized training, in which training data (task queries and trajectories) are aggregated in one place. This centralized-data assumption is increasingly at odds with privacy regulation (GDPR, CCPA) and the sensitivity of user interaction data. We relax it by enabling collaborative agent RL across distributed clients without sharing local data, and by providing a *theoretical* account of when this decentralized paradigm preserves agent training quality (Pattern A under task-level heterogeneity, and Pattern B under environment-level heterogeneity when one of (C1)–(C3) holds and the optimizer reaches the optimum) and when it does not (Patterns C/D). Each case is tied to a precise structural condition on the LLM agent's MDP.

I FEDAGENT Algorithm

This appendix expands on §2 in three parts: (i) a line-by-line description of the FEDAGENT training protocol of Algorithm 1 (Appendix I.1), (ii) an account of FEDAGENT's intrinsic privacy properties together with a discussion of why standard gradient-inversion and membership-inference attacks are expected to transfer poorly from supervised classification to LLM agent RL (Appendix I.2), and (iii) communication cost and deployment considerations (Appendix I.3). The protocol follows the classical FEDAVG [86] base protocol (sample → broadcast → local update → average), but replaces the supervised SGD of the local update with *policy-gradient RL on LLM-induced policies*. This single change produces both the new forms of heterogeneity analyzed in §3 and the privacy properties discussed in Appendix I.2.

I.1 Line-by-Line Description

Algorithm 1 in §2 states the FEDAGENT training protocol, with client-side and server-side computation color-coded for readability. This subsection explains the algorithm line by line. Unlike the fixed loss over labeled examples in supervised FL, the local objective is an expected return over rollouts that depends on the per-client task distribution $\mathcal{D}_{\tau_i}$ and the per-client MDP components $(\mathcal{S}_i, \mathcal{A}_i, P_i, R_i)$, so the Input-Dynamics Asymmetry of §3 is encoded directly into the inner loop.

Notation. N is the total client population, T is the number of communication rounds, M is the per-round sampled cohort size ($M \leq N$), and E is the number of local optimization epochs each sampled client performs before returning its model (in each epoch the client samples a fixed batch of tasks from the local pool, with exact batch sizes listed in Appendices J.3 and J.4). Subscripts on θ encode round and local-epoch indices: θ_t is the global model at the start of round t, $\theta_{i,t,e}$ is client i's local iterate after e local epochs in round t, and $\theta_{i,t,E}$ is the model the client returns to the server. The learning rate η_{lr} is shared across clients in our experiments, but the protocol does not require a shared learning rate.

Server side (Lines 3–4 and 14). Each round begins with the server sampling M clients from $[N]$ uniformly without replacement and broadcasting the current global parameters θ_t to that cohort (Lines 3–4). At the end of the round, the server collects the returned local models $\{\theta_{i,t,E}\}_{i \in S_t}$ and aggregates them by uniform averaging $\theta_{t+1} \leftarrow \frac{1}{M} \sum_{i \in S_t} \theta_{i,t,E}$ (Line 14). Uniform averaging matches the federated objective $\mathcal{J}_{\text{fed}}(\theta) = \frac{1}{N} \sum_i \mathcal{J}_i(\theta)$ defined in §2 under the partial-participation sampling we use (M clients sampled uniformly without replacement). A sample-size-weighted average with weights $w_i = |X_i| / \sum_j |X_j|$ is the natural alternative when client pool sizes differ widely. In our setting, the COVERAGEPARTITION experiments in §5.2 indicate that uniform averaging tolerates such size dispersion under GRPO. Under PPO the extreme coverage curve ends about 7–10 points lower on average over the last ten checkpoints (Appendix G.3).

Client side (Lines 5–13). Each sampled client first re-initializes its local iterate to the broadcast global model ($\theta_{i,t,0} \leftarrow \theta_t$), then performs E local epochs of policy optimization on its own task-augmented MDP $\mathcal{M}_i$. Each local epoch consists of (i) collecting the epoch's rollout batch $B_{i,t,e}$ (the "minibatch of trajectories" of Algorithm 1) by rolling out the current local policy $\pi_{\theta_{i,t,e}}$, (ii) estimating a policy gradient $g_{i,t,e}$ from those trajectories, and (iii) updating the local parameters by $\theta_{i,t,e+1} \leftarrow \theta_{i,t,e} + \eta_{\text{lr}}\, g_{i,t,e}$. In the implementation, step (iii) runs several AdamW mini-batch updates on the epoch's samples (about 12 on WebShop and up to 50 on ALFWorld, Appendix J.5). The pseudocode is deliberately optimizer-agnostic: Line 9 represents the gradient estimator by the generic procedure call "$g_{i,t,e}$ from $B_{i,t,e}$", which can be instantiated by any actor-only or actor-critic policy-gradient method. In the main experiments we instantiate it with **GRPO** [107] (the group-relative method that drops the critic and normalizes each task's rewards by their group mean and standard deviation) and **PPO** [106] (with a learned value baseline and GAE, sharing GRPO's clipped actor loss in our runs), and we observe in §5.3 that the choice of local optimizer decides whether some variants reach Pattern B or stay in Pattern C, even though the FedAvg outer loop is identical. The exchanged parameters also depend on the optimizer. Under PPO, the critic is federated alongside the actor: both are uploaded, averaged, and re-broadcast each round. Under GRPO, only the actor is exchanged.

I.2 Intrinsic Privacy Properties

A central motivation for FEDAGENT is privacy: centralizing user task queries and agent trajectories at scale can conflict with legal constraints on data sharing [51] and with the sensitivity of natural-language interaction logs. We make this informal motivation precise by specifying the threat model and identifying what does and does not cross the client/server boundary. We then argue structurally that the dominant privacy-attack families developed against supervised federated classification transfer poorly to LLM agent RL. We do not claim formal privacy guarantees. We identify the structural reasons why standard attacks are weakened and point to orthogonal defenses (differential privacy, secure aggregation, parameter-efficient transmission) that remain compatible with the base FEDAGENT protocol.

Threat model. We adopt the standard *honest-but-curious* threat model for federated learning [51]: all parties (server and participating clients) follow the protocol exactly, but they may inspect the messages they observe and attempt to infer information about other clients' private data. The privacy-sensitive content in FEDAGENT is each client's task pool $X_i \subset \mathcal{T}$ together with the rollout trajectories produced during local optimization. Trajectories carry substantially more sensitive information than supervised labels: a single rollout in WebShop or ALFWorld interleaves the user's task descriptor (*e.g.*, a shopping request or a household goal), the agent's intermediate reasoning (up to 512 tokens per step and often thousands per trajectory), every tool call and environment observation, and the reward signal. Protecting these trajectories is the core privacy goal of FEDAGENT.

What crosses the boundary. By construction (Algorithm 1), the only quantities that traverse the client/server boundary are model parameters: θ_t on the downlink and $\theta_{i,t,E}$ (equivalently $\Delta\theta_{i,t} = \theta_{i,t,E} - \theta_t$) on the uplink, plus, under the PPO instantiation, the critic parameters ϕ_t / $\phi_{i,t,E}$ exchanged the same way. Raw task descriptors, rollout trajectories, intermediate token sequences, environment observations, action logits, advantage estimates, per-task reward signals, and per-client success rates all stay on the client in deployment, and our experiments collect per-client validation scores only for evaluation (Appendix J.7). The PPO critic is trained on local reward signals and carries the same caveats as the actor update discussed below. It is also the object that RL-specific inversion attacks such as RGIA [41] target. Under GRPO no critic exists. Restricting the uplink to model parameters applies the same data-minimization principle that FEDAVG [86] provides for supervised learning, but the threat surface that remains is structurally different because both the local objective and the sensitive content differ from the supervised case.

Why classical gradient-inversion attacks transfer poorly. The gradient-inversion attack (GIA) literature [35, 45, 192] has established that, for image classification, an honest-but-curious server can sometimes reconstruct private inputs and labels directly from gradient updates. Three structural features of FEDAGENT weaken this attack vector substantially.

- *Multi-step local updates break per-sample gradient correspondence.* Most GIAs are formulated against single-step gradients $g = \nabla L(\theta; (x, y))$ on small batches and search for a candidate $(\tilde{x}, \tilde{y})$ whose gradient matches g. FEDAGENT clients return $\theta_{i,t,E}$ after E local epochs of stochastic policy optimization across many mini-batches, so the linear correspondence between individual training samples and the returned delta $\Delta\theta_{i,t}$ is broken. Huang et al. [45] show that even in supervised classification, the attack weakens substantially once its strong assumptions (known BatchNorm statistics and private labels) are relaxed or the batch size grows. We expect inversion to be harder still under policy-gradient updates, which, unlike deterministic gradients of a loss on a labeled example, are themselves stochastic estimators of an expectation over trajectories.
- *Label-leakage attacks have no discrete label channel to exploit.* The label-leakage attack of Zhao et al. [182] recovers ground-truth class labels by inspecting the sign of the last-fully-connected-layer gradient under cross-entropy loss, exploiting the direct link between a single softmax cell's gradient and the class index. Agent RL has no fixed class set: the supervisory signal is a stochastic policy gradient against log-probabilities of long action sequences, with no one-hot target. The label signal that this attack exploits does not exist in the FEDAGENT protocol.
- *Long-sequence inversion is fundamentally hard.* Token-level GIAs against language-model gradients [8, 94] have made progress, but they remain limited in how much text they can recover: DAGER [94] guarantees exact recovery only when the total token count stays below the model's embedding dimension, and earlier approximate methods such as LAMP [8] were evaluated on

inputs of at most a few tens of words. Each training sample in our runs is a windowed per-turn prompt with a response of up to 512 tokens (Appendix J.5), already well beyond the inputs of a few tens of words used to evaluate LAMP. A mini-batch of 64 such samples also holds far more tokens than the embedding dimension that bounds DAGER's exact recovery.

RL-specific gradient inversion is weakened by environmental dynamics. An emerging line of work [41] adapts GIAs to federated RL by jointly reconstructing (s, a, r, s') tuples from shared value-function gradients. He [41] document a unique *pseudo-solution problem*: candidate reconstructions that match the gradient values may violate the underlying environment dynamics, so attacks without prior knowledge of the dynamics, such as Deep Leakage from Gradients (DLG) [192], cannot resolve this ambiguity. The proposed RGIA method mitigates this ambiguity by adding regularizers that constrain reconstructions to plausible state distributions, plausible reward ranges, and consistent transition dynamics. In FEDAGENT the MDP components $(\mathcal{S}_i, \mathcal{A}_i, P_i, R_i)$ are local to each client. When environments differ across clients, as in our environment-level heterogeneity construction (§3.3), which varies P_i across clients, the attacker lacks the most informative side information needed to make the inversion well-posed for any specific client. Even if the attacker has a perfect simulator of the average environment, per-client divergences along the environment-level axis (Catalog Split, Field-Subset Index, BM25 Reweighting, Lookalike Injection, and Rank Wrapper in our experiments) leave the inversion underdetermined.

Membership inference is similarly weakened. Membership-inference attacks [89, 113] determine whether a specific example was in the training set, exploiting what the model or its gradients reveal about individual training samples. Agent RL departs from this setup in two ways. First, the natural unit of training data is a task descriptor τ, and, unlike a labeled example, a single τ produces a distribution of trajectories under the stochastic policy. The membership signal is weakened by being spread across many stochastic rollouts of the same task. Second, reward computation is local to the client, and the server never observes per-example reward signals or per-task success indicators. The standard signal channels for membership inference, namely per-example loss and prediction confidence on the canonical input, therefore have no direct analogue in the FEDAGENT uplink. An attacker would have to reconstruct these signals from $\Delta\theta_{i,t}$. This reconstruction faces the same multi-step-update and long-sequence obstacles documented above.

Caveats and orthogonal defenses. The argument above is structural: it identifies where existing classification-oriented attack surfaces fail to transfer to LLM agent RL. The argument does not establish a worst-case formal bound. We do not claim that FEDAGENT is provably private. Three established orthogonal mechanisms are compatible with the base FEDAGENT protocol, and the first two add quantitative guarantees to this argument: (i) differential privacy applied during local training [1] or to the aggregate of clipped client updates [36], (ii) secure aggregation [10] so the server only observes the aggregated parameters θ_{t+1} rather than individual client updates, and (iii) communication-efficient transmission via parameter-efficient methods such as FLoRA [135] or zeroth-order seed-based protocols [98], which both reduce bandwidth (cf. Appendix I.3). We expect such compact updates to also make inversion harder because they carry less per-sample information. A formal privacy analysis tailored to LLM agent RL, in particular extending the pseudo-solution analysis of He [41] to long-horizon LLM rollouts, is a worthwhile direction for future work.

I.3 Communication Cost and Deployment Considerations

Per-round bandwidth. Each round of FEDAGENT transmits the full LLM parameters θ on both the downlink and uplink. For the `Qwen2.5-1.5B-Instruct` backbone used in the main experiments, this transmission requires $\approx$ 3 GB per direction in bf16 per participating client per round under GRPO (actor only), and $\approx$ 6 GB under PPO (whose critic is exchanged alongside the actor, cf. Appendix I). We deliberately use full-parameter transmission throughout the experiments to keep the communication channel a controlled variable so that any observed effect on training quality is attributable purely to the heterogeneity axes of §5.2–5.3. Compression-induced loss is excluded by design. In a practical deployment, the communication-efficient federated fine-tuning literature provides several alternatives that drastically reduce per-round bandwidth. LoRA-based methods [7, 121, 135] transmit only adapter matrices, segment-sharing methods [78] amortize the cost over several rounds, and zeroth-order methods [98] reduce the per-round message size to kilobytes. None of

these change the algorithmic structure analyzed in §3–4, and they can be combined directly with the privacy mechanisms of Appendix I.2.

Partial participation and client drift. Two practical considerations from supervised FL [51, 52] carry over to FEDAGENT. *Partial participation* ($M < N$) introduces extra variance into the aggregated update: with M clients per round, each client is selected with probability M/N per round (on average about 1.4 times over our $T = 70$ rounds with $N = 100$ and $M = 2$), so when client distributions differ along the heterogeneity axes of §3, this sampling variance interacts with the dispersion measures $\Delta^2_{\text{pref}}, \Delta^2_{\text{cov}}, \Delta^2_{\text{hard}}$. *Client drift* ($E > 1$ local epochs) is the well-known phenomenon that local iterates depart from the broadcast initialization within a round, and large E trades communication savings against drift-induced slowdown. We use moderate E in the main experiments, and the task-level heterogeneity experiments in §5.2 confirm that drift does not destabilize FEDAGENT even under extreme task-level dispersion (Pattern A). The environment-level results in §5.3 show that drift alone is not the source of Pattern D collapse either: the single-environment baseline uses the same uniform task partition and the same E, and it reaches $\sim$60% without collapsing.

Algorithmic degrees of freedom. Three hyperparameters govern the cost-quality tradeoff of the FEDAGENT protocol and are reported alongside the experiments. (i) *Cohort size* M controls how many clients update per round and thereby the per-round bandwidth and aggregator-side variance. (ii) *Local epochs* E controls how much local computation is amortized per communication round. Large E saves communication but increases client drift. (iii) *Total rounds* T controls the global compute budget. Throughout the main experiments we fix M, E, and T to the values reported in §5.1 so that the only varying axes are the heterogeneity hyperparameters of §5.2 and §5.3. A sensitivity analysis on $|X_i|$, M, and E is reported in Appendix G.1.

Deployment scenarios. Two natural deployment scenarios motivate FEDAGENT. (i) *Cross-silo:* a small number of organizations (*e.g.*, enterprises with their own user query logs and tool catalogs) train an agent jointly while keeping their data and tools internal. Each silo is a client, N is small ($\sim$ 10–100), and per-round bandwidth is not a primary bottleneck. The privacy properties of Appendix I.2 are the primary benefit. (ii) *Cross-device:* large numbers ($N \gg 10^3$) of end-user devices contribute training signal. Here partial participation, full-parameter bandwidth, and on-device compute become limiting constraints, and the parameter-efficient and zeroth-order drop-in replacements above are the natural way to obtain a deployable instantiation. The experiments in §5.2–5.3 use N=100 with M=2 per round. This setup matches the cross-silo client count, uses the strong partial participation typical of cross-device FL, and remains tractable on a research-scale cluster.

J Experimental Setup Details

This appendix consolidates the full experimental configuration used throughout §5 and the appendices, expanding on the condensed setup paragraphs in §5.1, §5.2, and §5.3. All numerical values below describe the configuration the reported experiments were run with: the verl-agent-based implementation, preserved unchanged on the repository's `paper-reproduce-verl-agent` branch (Appendix C). The repository's `main` branch is an equivalence-verified re-implementation of the same protocol as a small extension of unmodified `verl` 0.8 (Python 3.12). The re-implementation preserves the training semantics and changes engine placement and memory settings for the new engine (*e.g.*, `tensor_model_parallel_size` $4 \rightarrow 1$, GRPO `gpu_memory_utilization` $0.5 \rightarrow 0.6$), as documented in the repository.

J.1 Hardware and Software

All experiments are run on $4\times$ NVIDIA H100 (80 GB) GPUs per node, using `tensor_model_parallel_size=4` for vLLM rollout. The implementation is built on top of HuggingFace `transformers` (4.51.1) and the `verl-agent` framework [31] (itself built on `verl` 0.3.1.dev). Our federated extension is released alongside the paper at the public repository in Appendix C. The software stack uses Python 3.10 (3.10.0 for the WebShop conda environment and 3.10.18 for ALFWorld), PyTorch 2.6.0+cu124, and CUDA toolkit 12.4. Rollout inference uses vLLM 0.8.2 for WebShop and 0.8.5 for ALFWorld. We use bfloat16 mixed precision throughout (vLLM `dtype=bfloat16`, FSDP shards in bf16). FSDP is configured with gradient checkpointing enabled, `param_offload=false`, and `optimizer_offload=false` for both GRPO and PPO. The four WebShop variants other than Catalog Split (Field-Subset Index, BM25 Reweighting, Lookalike Injection, Rank Wrapper) run against our in-memory BM25 backend (Appendix M), and validation uses the default Lucene searcher. ALFWorld uses the standard textual interface from Shridhar et al. [114], with the data cache located at `$ALFWORLD_DATA/json_2.1.1/train` (3553 game files and 550 spec files).

J.2 Models and Tokenization

We test four agent backbones across the experiments: `Qwen2.5-1.5B-Instruct`, `Qwen2.5-3B-Instruct`, `Qwen2.5-7B-Instruct`, and `Llama-3.2-3B-Instruct`. The Qwen2.5-1.5B-Instruct backbone is used for all heterogeneity experiments (§5.2, §5.3) and the hyperparameter sensitivity ablation (Appendix G.1). Table 1 and Table 3 report effectiveness results for all four backbones. We update all model parameters (full fine-tuning, without LoRA), using each model's released chat template and tokenizer. No system-prompt modification is applied. The maximum prompt length is 4096 tokens for WebShop and 2048 tokens for ALFWorld. The maximum response length (max-new-tokens per agent action) is 512 tokens.

J.3 Federation Protocol

The federated training loop instantiates Algorithm 1 with the following hyperparameters, used in all experiments unless otherwise stated:

- Number of clients: $N = 100$.
- Clients selected per round: $M = 2$ (sampled deterministically by round seed for reproducibility, equivalent in distribution to uniform sampling without replacement at the per-round level).
- Local epochs per round: $E = 3$.
- Communication rounds: $T = 70$, giving $T \cdot E = 210$ total local epochs.
- Per-client task pool size: $|X_i| = 100$. On WebShop, the first 500 goals `goals[0:500]` are reserved as the validation pool and never seen during training (ALFWorld validates on its separate `valid_seen` split, Appendix J.7). Each client's pool is a contiguous slice of the remaining `goals[500:]` produced by `uniform_partition()` with `min_samples_per_client`= 100 and `data_sharding.seed`= 42. On WebShop, adjacent clients overlap on $\sim$36 goals due to contiguous slicing with boundary expansion.
- Per-epoch sampling: each local epoch collects a rollout batch of 64 trajectories (GRPO: a train batch of 8 tasks with $G = 8$ rollouts each; PPO: 64 tasks with one rollout each). Optimization

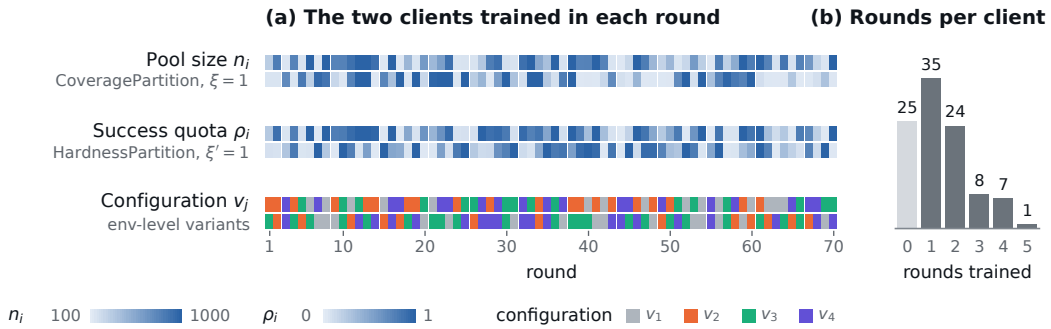


Figure 10: **The server selects the same client pairs in every run.** Panel (a) shows the two clients selected in each of the $T = 70$ rounds, colored by what they hold under two task-level partitions and four environment variants. The first two strips show the pool size under CoveragePartition($\xi = 1$) and the thresholded success rate ρ_i under HardnessPartition($\xi' = 1$), which are identical on WebShop and ALFWorld. The third shows the configuration that Field-Subset Index, BM25 Reweighting, Lookalike Injection, and Rank Wrapper assign to the client (Figure 17). Gray v_1 is the reference configuration of every pool except Lookalike Injection's, and Catalog Split gives every client its own catalog. Panel (b) shows the number of clients that train in 0 to 5 rounds, and its light bar counts the 25 clients that never train.

then proceeds in mini-batches of `ppo_mini_batch_size` $= 64$ *per-turn samples* under the windowed rollout formulation of Appendix J.5. At matched sequential local epochs, the two clients of a round together collect 128 trajectories per epoch, twice the 64 of the centralized baseline.

- Server aggregation: uniform parameter averaging $\theta_{t+1} \leftarrow \frac{1}{|S_t|} \sum_{i \in S_t} \theta_{i,t,E}$ over the round's selected clients S_t (FedAvg). FedProx is implemented but unused in our main experiments ($\mu_{\text{prox}} = 0.01$ is its unused default).
- Initial parameters: each backbone's released instruction-tuned checkpoint, used as θ_0 for the first round.
- Checkpoint format: `actor_rollout_ref.actor.checkpoint.contents = ['model']` (non-sharded), enabling cross-worker averaging. Per-round client checkpoints are pruned with `max_rounds_to_keep_client_checkpoints`$= 2$.

The sensitivity analysis in Appendix G.1 motivates the choice of $(M = 2,\ E = 3,\ |X_i| = 100)$.

Because the round seeds are fixed, every federated run with $N = 100$ and $M = 2$ uses the same sequence of client pairs (Figure 10). Over $T = 70$ rounds, 75 of the 100 clients train at least once and 25 never do. The trained clients still cover the full range of each heterogeneity axis. Under CoveragePartition($\xi = 1$), their pool sizes span 123–1,000 tasks. Under HardnessPartition($\xi' = 1$), 39 of the 140 client-rounds have a thresholded success rate $\rho_i \leq 0.1$ and 38 have $\rho_i \geq 0.9$. The four configurations in each environment pool receive 30–39 client-rounds each.

J.4 Policy Optimization Hyperparameters

We instantiate the local optimizer with either GRPO [107] (main text default) or PPO [106] (Appendix G.2). Both optimizers use the following settings:

- Optimizer: AdamW for both actor and critic.
- Adam betas: $(\beta_1, \beta_2) = (0.9, 0.999)$ (PyTorch default, not overridden).
- Weight decay: 0.01 (verl default).
- Learning rate: $1e{-6}$ for the actor, $1e{-5}$ for the critic (PPO only).
- LR schedule: constant (no warmup, no decay; `warmup_style=constant`, `lr_warmup_steps=-1`, `lr_warmup_steps_ratio=0.0`).
- Gradient clipping: max-norm 1.0 (verl default).

- Clipped actor loss: symmetric clip ratio $\epsilon_{\text{clip}} = 0.2$ (`actor.clip_ratio_low=actor.clip_ratio_high=` 0.2) and dual-clip parameter `clip_ratio_c` $= 3.0$ (verl defaults, which the GRPO scripts do not override).
- Update passes over each local epoch's batch (verl `ppo_epochs`): 1 (verl default).
- Discount factor: $\gamma = 1.0$ (verl default, undiscounted episodic).
- Invalid-action shaping: $\Delta r = -0.1 \times$ `invalid_count` per step on both benchmarks. This shaping is a small additive engineering term applied at the optimizer side to discourage malformed actions. The formal reward R_i of the underlying MDP remains the non-negative success-based component $r_u \in [0, R_{\max}]$ of §2, and the shaping magnitude ($\leq 0.1 \cdot \#$invalid per step, dominated by the success-based component once the agent learns valid syntax) does not affect the asymptotic claims of §4.

GRPO-specific. GRPO uses a group size of $G = 8$ (`env.rollout.n`) and a group-relative baseline advantage estimator (`algorithm.adv_estimator=` `grpo`) with within-group advantage normalization (`norm_adv_by_std_in_grpo=` True). The KL coefficient against the `verl` reference policy (distinct from the hardness checkpoint π_{ref}) is $\beta_{\text{KL}} = 0.01$ (`actor.kl_loss_coef`, with `kl_loss_type=` `low_var_kl`, the low-variance k_3 estimator of the KL divergence), and the entropy coefficient is 0.001. The mini-batch size is 64 turn samples, the micro-batch size per GPU is 8, and the train batch size is 8 tasks on both benchmarks.

PPO-specific. PPO uses the GAE advantage estimator (`algorithm.adv_estimator=` `gae`) with $\lambda = 1.0$ (verl default, equivalent to Monte-Carlo advantage). PPO also uses a value-loss clipping range of `cliprange_value` $= 0.5$ and an entropy coefficient of 0.001. The mini-batch size is 64 turn samples, the actor micro-batch size per GPU is 8, the critic micro-batch size per GPU is 4, and the train batch size is 64 tasks on both benchmarks. We do *not* add the KL term to the reward (`use_kl_in_reward` = False). The in-loss actor KL coefficient is $\beta_{\text{KL}} = 0.01$ (same as GRPO). In the first round, the critic is initialized from the same instruction-tuned checkpoint as the actor, with a value head added by verl's default value-head wrapper. From the second round onward, each client's critic is initialized from the previous round's federated critic average, since the critic is uploaded and averaged alongside the actor (Appendix I).

J.5 Rollout Protocol

Rollouts are generated with vLLM (`rollout.name=` `vllm`, `tensor_model_parallel_size=` 4, `gpu_memory_utilization=` 0.5, `enforce_eager=` false, `free_cache_engine=` false, `enable_chunked_prefill=` true, `kv_cache_dtype=` `auto`, resolving to bf16). We evaluated FP8 (e5m2) KV-cache quantization but reverted to the bf16 default, as FP8 gave no rollout speedup and triggered NaN crashes under 7B PPO. We use the following per-task rollout settings:

- Number of rollouts per task: $G = 8$ for GRPO (training, `env.rollout.n`). PPO collects one rollout per task (`env.rollout.n` disabled): GAE's advantage baseline comes from the learned critic rather than a rollout group, and PPO's larger train batch (64 tasks vs. GRPO's 8) yields the same 64-trajectory per-epoch budget (cf. §5.1).
- Training sampling: `temperature` $= 1.0$, `top_p` $= 1.0$, `top_k` $= -1$, `do_sample` = True.
- Validation sampling (`rollout.val_kwargs`): `temperature` $= 0.4$, `top_p` $= 1.0$, `top_k` $= -1$, `do_sample` = True.
- Maximum episode length: `env.max_steps` $= 15$ on WebShop, 50 on ALFWorld.
- Per-step generation budget: `max_response_length` $= 512$ tokens.

Windowed per-turn formulation. Rollouts follow the verl-agent windowed design (`env.history_length=` 2). At every turn, the policy receives a freshly rendered single-message prompt containing the task, the most recent two (observation, action) pairs, and the current observation. The policy then generates one action. Every turn is an independent training sample carrying the episode's terminal return (plus the per-step invalid-action shaping of Appendix J.4). Thus, a 64-trajectory local epoch yields roughly 750–800 turn samples on WebShop and more on ALFWorld, whose horizon is longer. The mini-batch size of 64 above is counted in turn samples, so

Prompt at step 2 identical for both clients except the two orange placeholders

```
You are an expert autonomous agent operating in the WebShop e-commerce environment.
Your task is to: Find me long lasting, moisture wicking men's boots for daily wear with
size: 9, and price lower than 130.00 dollars.
Prior to this step, you have already taken 1 step(s). Below are the most recent 1
observations and the corresponding actions you took: [Observation 1: ''Search'', Action 1:
'search[find me long lasting, moisture wicking men's boots for daily wear with size: 9]']
You are now at step 2 and your current observation is: ⟨current observation⟩.
Your admissible actions of the current situation are:
[
⟨admissible actions⟩
].

Now it's your turn to take one action for the current step.
You should first reason step-by-step about the current situation, then think carefully which
admissible action best advances the shopping goal. This reasoning process MUST be enclosed
within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for current step
and present it within <action> </action> tags.
```

Client 1 v_1, default ranking

⟨*current observation*⟩

```
'Back to Search' [SEP] 'Page 1 (Total
results: 200)' [SEP] 'Next >' [SEP]
'B01EX1BWE4' [SEP] 'Milwaukee Leather
MBM9075 Men's Black 6-inch Plain Toe Dual
Zipper Lock Leather Boots' [SEP] '$99.82 to
$134.99' [SEP] 'B074Z315KK' [SEP] 'Novia's
Choice Men Shiny Metallic Shirt Blouse
T-Shirt Tee for Nightclub' [SEP] '$18.99 to
$25.43' [SEP] [8 more results]
```

⟨*admissible actions*⟩

```
'click[back to search]',
'click[next >]',
'click[b01ex1bwe4]',
'click[b074z315kk]',
[8 more clicks]
```

Client 4 v_2, shuffled ranking

⟨*current observation*⟩

```
'Back to Search' [SEP] 'Page 1 (Total
results: 200)' [SEP] 'Next >' [SEP]
'B07BB1FJV8' [SEP] 'Edna Wig Color FS4/30 -
Foxy Silver Wigs Short Pixie Wispy Fringe
Synthetic Straight Layers African American
Women's Machine Wefted Lightweight Average
Cap Bundle with MaxWigs Hairloss Booklet'
[SEP] '$36.69' [SEP] 'B07PVZPS58' [SEP]
'BRIEF INSANITY Hot Rod Boxer Briefs for Men
and Women | Vintage Car Print Boxer Shorts -
Funny, Humorous, Novelty Underwear' [SEP]
'$19.95' [SEP] [8 more results]
```

⟨*admissible actions*⟩

```
'click[back to search]',
'click[next >]',
'click[b07bb1fjv8]',
'click[b07pvzps58]',
[8 more clicks]
```

Figure 11: **The task enters the policy through the prompt, and the environment enters only through the observations.** The figure shows the WebShop prompt at step 2 for two clients of the Rank Wrapper federation, which use its default (v_1) and shuffled (v_2) rankings. The prompt keeps only the most recent steps of the episode. The released prompt code produced it for the held-out goal of Figure 19, after the search query shown there. The top box, identical for both clients, shows the template with this episode's task (blue) and history. The bottom boxes show the contents of the two placeholders that differ (orange), namely the top-10 search results and the admissible actions. Client 1's default ranking puts the goal product (bold) first, while client 4's shuffled ranking shows no boots among its top-10 search results, and no other part of the prompt indicates which ranking is in use. Results 3–10 and their click actions are elided (gray italics). The rest is verbatim. In the observations, `[SEP]` separates the elements of the page, and codes such as `B01EX1BWE4` are product IDs.

each local epoch performs many optimizer steps, about 12 on WebShop and up to 50 on ALFWorld. Figure 11 shows such a prompt for two clients of one federation: the task enters through the template, and a client's environment enters only through the observations (current and recent) and the admissible actions.

J.6 Heterogeneity Construction Protocol

Task-level. The three sub-types are operationalized by PREFERENCEPARTITION(ω), COVERAGEPARTITION(ξ), and HARDNESSPARTITION(ξ'). The dispersion settings used in Figure 5 are: $\omega \in \{0.01, 0.99\}$ (preference: near-uniform vs. extreme one-hot, with $|X_i| = L = 100$), $\xi \in \{1, 256\}$ (coverage: extreme spread vs. near-uniform pool sizes over $[L_{\min}, L_{\text{avg}}, L_{\max}] = [100, 500, 1000]$),

and $\xi' \in \{1, 256\}$ (hardness: extreme success-rate skew vs. near-uniform, with $|X_i| = L = 100$ and success-count bounds $(\kappa_{\min}, \kappa_{\text{avg}}, \kappa_{\max}) = (0, 50, 100)$). Pseudocode and the four design properties (D1)–(D4) are in Appendix L (specifically Appendices L.1 and L.2). The formal dispersion measures and intrinsic-property results that ground these properties are in Appendix K. Per-client distribution figures for both settings are in Appendix L.3. The hardness reference checkpoint π_{ref} used to pre-label tasks as "successful"/"unsuccessful" is a `Qwen2.5-1.5B-Instruct` policy fine-tuned on each benchmark and frozen before federated training. When rolled out on the training pool, the checkpoint labels 27.8% of WebShop and 59.4% of ALFWorld training tasks as successful. We use a trained reference because the zero-shot backbone succeeds on only $\approx 1.4\%$ of WebShop goals, and this low rate would collapse the easy/hard split into a near-degenerate all-hard partition. The preference and coverage splits keep the mean success rate $\bar{\rho}$ near the labeled rates of 27.8% and 59.4%. The hardness splits set $\bar{\rho} = \kappa_{\text{avg}}/L = 1/2$, so they over-represent successful tasks on WebShop and under-represent them on ALFWorld. Both hardness settings share $\bar{\rho} = 1/2$, so the ξ' comparison is unaffected.

Environment-level. The five WebShop environment variants (Catalog Split, Field-Subset Index, BM25 Reweighting, Lookalike Injection, Rank Wrapper) are described in §3.3 with full implementation details in Appendix M. The curves in Figure 6 use $|\mathcal{V}| = 4$ for the four discrete-pool variants (Field-Subset Index, BM25 Reweighting, Lookalike Injection, Rank Wrapper) and `env_div` $= 1.0$, `keep_ratio` $= 0.7$ for Catalog Split. The wider ranges listed below are the configurable settings the partition code supports, and the paper reports no additional experiments with them. The variant pools and exact hyperparameter values are:

- **Catalog Split**: `env_div` $\in \{0.0, 0.3, 0.7, 1.0\}$ at `keep_ratio` $= 0.7$ ($\approx$ 702 products per client, on the uniform 100-goal-per-client task split).
- **Field-Subset Index**: $|\mathcal{V}| \in \{4, 8\}$ field subsets chosen from {name, Title, description, features, BulletPoints} and pairs/triples thereof.
- **BM25 Reweighting**: $|\mathcal{V}| \in \{4, 8\}$ (k_1, b) configurations, pairing the default $(1.2, 0.75)$ with the three extreme combinations $(0.3, 0.75), (5.0, 0.75), (1.2, 0.0)$ at $|\mathcal{V}|$=4 (the $|\mathcal{V}|$=8 extension adds $(0.1, 0.75), (1.2, 1.0), (2.0, 0.5), (0.3, 0.0)$).
- **Lookalike Injection**: $|\mathcal{V}| \in \{2, 4\}$ attacked-subterm configurations (price: 1,000 lookalikes, color: 286, size: 268, and price+color: 286). The 2-configuration setting is a reduced setting supported by the partition code and is not reported in this paper.
- **Rank Wrapper**: $|\mathcal{V}| = 4$ wrappers (default, shuffle, invert, partial-random).

`SEARCH_RETURN_N` is set to 200 throughout the environment-level experiments to keep target products in the results after post-retrieval catalog filtering (the non-environment-level experiments use the default 50). Validation always runs on the unperturbed WebShop environment (the `catalog_filter_asins`, `bm25_in_memory_config`, `extra_products`, and `search_engine_variant` kwargs are forced to `None` on the validation `SimServer`). Two residual differences remain, one between validation services and one between training and validation. First, the validation service inherits the run's `SEARCH_RETURN_N`, so the environment-level experiments validate with top-200 retrieval while the §5.1 baseline validates with the default top-50. The top-10 search results are identical under both, so the effect on the success rate of 15-step episodes is marginal. However, the two validation environments are not byte-identical. Second, the four discrete-pool variants run on the in-memory BM25 backend of Appendix M, whereas validation always uses the default Lucene searcher (Appendix M.3). The task partition is held *uniform* across all environment-level experiments, so the task split does not contribute to any divergence.

J.7 Evaluation Protocol

Test sets. WebShop is evaluated on 64 goals sampled from the held-out validation pool `goals[0:500]` (disjoint from training). ALFWorld is evaluated on 64 trials from the held-out `valid_seen` split: the task instances (game files) are disjoint from the training pool, but their underlying scenes also appear in training, so the metric measures held-out-task performance in familiar scenes (the `valid_unseen` split, whose scenes are absent from training, is not evaluated in this paper). The 64 trials span the six task types (Pick, Look, Clean, Heat, Cool, Pick2) with per-type episode counts $(14, 8, 12, 9, 11, 10)$, which are the denominators of the per-type columns in Table 1 and Table 3. For environment-level experiments, evaluation is performed on the *stan-*

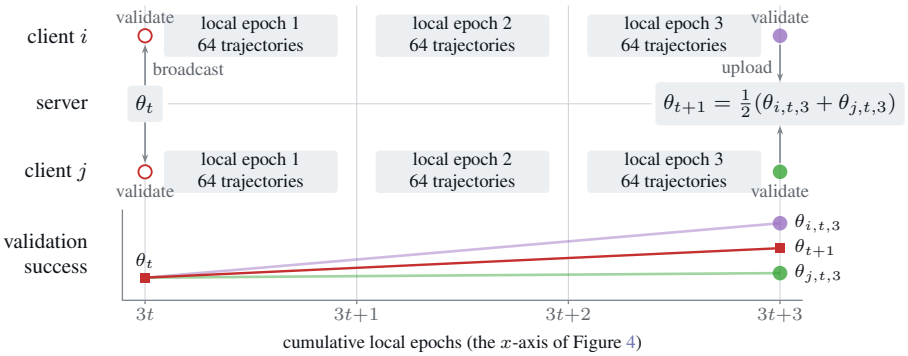


Figure 12: **One communication round and the points it adds to Figure 4.** The server broadcasts θ_t to the two clients of round t. Each client validates θ_t on 64 held-out tasks (open red markers) and then trains for $E = 3$ local epochs. Each epoch collects 64 trajectories and then runs optimizer steps on the per-turn training samples of these trajectories (about 12 on WebShop and up to 50 on ALFWorld, Appendix J.5). After the last epoch, each client validates its own model $\theta_{i,t,3}$ (filled markers) and uploads it, and the server averages the two uploads into θ_{t+1}. In the plot, the red square at epoch $3t$ is the mean of the two open-marker validations. Each colored circle at epoch $3t + 3$ is one filled-marker validation and is joined to the red square of its round. The red square at epoch $3t + 3$ comes from the next round, whose two clients validate θ_{t+1} before they train. The plotted heights are illustrative.

dard (unperturbed) WebShop environment so the metric isolates the post-aggregation generalization quality.

Metrics. Following Yao et al. [165], we report two complementary metrics. **Success Rate** measures binary task completion under the WebShop / ALFWorld convention. **Task Score** is reported only on WebShop and uses the environment's graded goal-match score $r \in [0, 1]$, which awards partial credit for the requested attributes, options, and price. The graded score is $r = 0$ when no purchase is made and $r = 1$ only for a perfect purchase. The score is averaged over evaluation episodes and scaled by 100, following the WebShop convention. Task Score is computed from this graded score signal, while training uses the sparse success reward plus the invalid-action penalty of Appendix J.4. Since every successful episode scores $r = 1$, Task Score on its 0–100 scale always upper-bounds Success Rate. The gap measures partial credit from partially matching purchases. On ALFWorld, we additionally break down success rate by the six task types and report the All-task aggregate (Tables 1 and 3). The breakdown partitions the same 64-trial evaluation, so the count-weighted average of the six per-type rates equals the All column.

Evaluation schedule. Two validation points in each sampled client's job generate the two visual elements of the training-dynamics figures: the aggregated server curve (shown in Figure 4, Figure 5, and Figure 6) and the per-client circle marks (shown in Figure 4 only). Each sampled client's local job runs validation before its first local epoch and after its last local epoch. Figure 12 shows one round and the points that it adds to Figure 4.

(i) The "server aggregated" curve (red in Figure 4). With `val_before_train` = true, each sampled client evaluates its broadcast model at local epoch 0, before any local training. Because all clients sampled in a round start from the same aggregated global model θ_t, these step-0 validations all measure θ_t and differ only by evaluation stochasticity. The reported *aggregated* value is their average over the round's sampled clients, giving one measurement of θ_t per round. These per-round measurements form the aggregated curve.

(ii) The colored circle marks (Figure 4 only). Each sampled client contributes one client-end point per round: its locally updated model is evaluated once at the terminal local epoch $E = 3$ (`verl`'s `is_last_step` evaluation), and the figure plots each of these per-client values as its own colored circle mark. The separate setting `test_freq` = 5 is `verl`'s step-level schedule *inside* a single client's job, which would launch validation whenever the internal global-step index is divisible by 5. However, each sampled client runs only $E = 3$ local epochs (= `verl` steps) per round ($3 < 5$), so

this schedule never triggers mid-training. Both evaluations use `val_batch_size` $= 64$, and all points are plotted against cumulative local epochs.

Random seeds and reporting. The seeds used by the experiment infrastructure are: `env.seed` $=$ 0 (training `SimServer` / ALFWorld step seed), `val SimServer.seed` $= 1000$ (validation, isolated from training RNG), `data_sharding.seed` $= 42$ (client-to-goal sharding, fixed across runs), $42 +$ `client_id` (configuration assignment for Field-Subset Index, BM25 Reweighting, Lookalike Injection, and Rank Wrapper), $42 + 1000 \cdot$ `client_id` (Catalog Split per-client distractor RNG with global seed 42 for the shared retention anchor u_c of Appendix M.4), and 99999 (offline lookalike synthesis). The federated and centralized rows in Table 1 and Table 3 report the mean and sample standard deviation over 3 runs, with the standard deviation shown as a subscript. The three FEDAGENT runs use training-goal shuffle seeds 42, 21, and 84 (`data_sharding.shuffle_seed`). This seed reshuffles the training goals before sharding, so it changes which tasks each client holds. The initial checkpoint (the released instruction-tuned model), the validation set, `env.seed` $=$ 0, and `data_sharding.seed` $= 42$ stay fixed across the three runs. Local-training rows use the indicated client indices (21, 42, 84), which share their values with the shuffle seeds but are unrelated to them.

J.8 Reproducibility Checklist

We summarize how the FEDAGENT experiments meet the standard NeurIPS reproducibility checklist:

- **Code, data, and configurations.** A public release containing our federated training code (the `paper-reproduce-verl-agent` branch holds the code of the reported runs, built on the `verl-agent` framework), dataset preprocessing scripts, all per-experiment configuration files, and scripts for regenerating the post-training outputs is linked in Appendix C.
- **Compute estimate.** The per-run wall-clock times are: WebShop GRPO ~ 20 min/round, ~ 24 h for $T = 70$ rounds (~ 93 GPU-hours per training run); WebShop PPO ~ 25 min/round, ~ 30 h (~ 117 GPU-hours); ALFWorld GRPO ~ 25 min/round, ~ 30 h (~ 117 GPU-hours); ALFWorld PPO ~ 30 min/round, ~ 35 h (~ 140 GPU-hours). Summing these costs over the 194 reported training runs gives an estimated total of about 17,000 H100 GPU-hours. The runs are the 144 runs of Tables 1 and 3, the 24 task-level runs of Figures 5 and 9, the 12 environment-level runs of Figure 6, and the 14 sensitivity runs of Appendix G.1. A Centralized or Local run counts as half a FEDAGENT run because it collects half as many rollouts (210 local epochs, against 2×210 for the $M = 2$ clients of FEDAGENT). The estimate applies the per-run costs above to all four backbones and excludes preliminary runs. Because 7B runs take longer than these per-run costs, the estimate is a lower bound.
- **Hyperparameters.** All federation, optimization, and rollout hyperparameters are reported above with values verified against the released configuration files.
- **Statistical reporting.** We report the mean and sample standard deviation over 3 runs (Appendix J.7). Training-dynamics figures show single-seed curves, and Figure 4 also overlays the per-client models as scatter points.
- **Datasets and licenses.** WebShop and ALFWorld are used under their respective open-source licenses. No human-subjects data is used. The synthetic Lookalike Injection products (§3.3, Appendix M) are generated programmatically from the WebShop catalog and inherit its license.

K Task-Level Heterogeneity Characterization

This appendix supports §3.2 with the formal measures used to quantify task-level heterogeneity across clients. We introduce the three inter-client dispersion measures $\Delta^2_{\text{pref}}, \Delta^2_{\text{cov}}, \Delta^2_{\text{hard}}$ (Definitions K.1–K.3), formalize the primary properties of the underlying first-order projections (Lemma 1), and establish the measures' definitional orthogonality and first-order sufficiency (Theorems 6 and 7). We also show that the converse of first-order sufficiency fails (Theorem 8). Appendix L covers the algorithms: how single-hyperparameter partitions realize each measure and how the four design properties (D1)–(D4) connect algorithms to measures.

K.1 Inter-Client Dispersion Measures

Throughout this section, we work with a federation of N clients, each holding a local task pool $X_i \subset \mathcal{T}$ of size $n_i = |X_i|$. As in §2, $\mathcal{D}_{\tau_i}$ is the uniform distribution on X_i, from which client i samples its training tasks. Each task $\tau \in \mathcal{T}$ has a categorical type $\ell(\tau) \in [C]$ and a reference success rate $\rho(\tau) \in [0,1]$, measured as the success rate of a fixed reference policy π_{ref} on τ (in our experiments, a benchmark-fine-tuned reference checkpoint frozen before federated training, see Appendix J.6). Higher $\rho(\tau)$ means an *easier* task. For a threshold $\eta \in (0,1)$, we write $S_\eta(\tau) := \mathbf{1}\{\rho(\tau) \geq \eta\}$ for the success indicator. The first-order task-distribution descriptor of client i is the triple

$$\mathbf{D}_i := (p_i,\, n_i,\, \rho_i) \in \Delta^{C-1} \times \mathbb{R}_+ \times [0,1],$$

where $p_i = (p_{i,c})_{c\in[C]}$ with $p_{i,c} = \Pr_{\tau\sim\mathcal{D}_{\tau_i}}[\ell(\tau) = c] = (1/n_i)\sum_{\tau\in X_i} \mathbf{1}\{\ell(\tau) = c\}$ is the type marginal, n_i is the pool size (the support size of $\mathcal{D}_{\tau_i}$, set directly by the partition protocol), and $\rho_i = \Pr_{\tau\sim\mathcal{D}_{\tau_i}}[\rho(\tau) \geq \eta] = (1/n_i)\sum_{\tau\in X_i} S_\eta(\tau)$ is the thresholded success rate. When a partition algorithm samples X_i from a client-level generating distribution (*e.g.*, the Dirichlet category mix q_i of Appendix L.2), p_i and ρ_i are unbiased estimates of that distribution's marginals with $O(1/\sqrt{n_i})$ deviation.

The three dispersion measures are inter-client variances of these projections, normalized so that each lies on a comparable scale.

Definition K.1 (Preference Heterogeneity)**.** *Given empirical type marginals* $\{p_i\}_{i=1}^N \subset \Delta^{C-1}$*,*

$$\Delta^2_{\textit{pref}} := \frac{1}{N}\sum_{i=1}^N \|p_i - \bar{p}\|_2^2 = \operatorname{tr}\big(\widehat{\operatorname{Cov}}_i(p_i)\big), \qquad \bar{p} := \frac{1}{N}\sum_i p_i.$$

The squared L^2 norm on the simplex arises naturally from a Cauchy–Schwarz bound on the gradient bias induced by the type marginals. Write $g_{i,c}(\theta)$ for client i's expected policy gradient on category-c tasks and $\bar{g}_c(\theta)$ for its cross-client average. Under the within-category-homogeneity assumption $g_{i,c}(\theta) \approx \bar{g}_c(\theta)$, the preference contribution to the gap between client i's gradient and the global mean gradient satisfies $\|\sum_c (p_{i,c} - \bar{p}_c)\bar{g}_c\|^2 \leq \big(\sum_c \|\bar{g}_c\|^2\big)\cdot\|p_i - \bar{p}\|_2^2$, so Δ^2_{pref} is the simplex-side factor in this bound. The range is $0 \leq \Delta^2_{\text{pref}} \leq 1 - 1/C$. The upper bound is attained by one-hot clients spread evenly across categories, which is possible when C divides N.

Definition K.2 (Coverage Heterogeneity)**.** *Given client pool sizes* $\{n_i\}_{i=1}^N$*,*

$$\Delta^2_{\textit{cov}} := \frac{1}{N}\sum_{i=1}^N \left(\frac{n_i - \bar{n}}{\bar{n}}\right)^2 = \widehat{\mathrm{CV}}^2(\{n_i\}), \qquad \bar{n} := \frac{1}{N}\sum_i n_i.$$

The squared coefficient of variation is a scale-invariant summary of pool-size dispersion: it vanishes precisely when all n_i are equal and is invariant to global rescaling of $\{n_i\}$. Under iterative-with-replacement RL sampling, a heuristic two-level variance decomposition suggests interpreting $\Delta^2_{\text{cov}}/\bar{n}$ as the leading-order Jensen gap in the variance term coming from pool composition. We do not formalize that decomposition here, and Definition K.2 does not rely on it.

Definition K.3 (Hardness Heterogeneity)**.** *Given client thresholded success rates* $\{\rho_i\}_{i=1}^N \subset [0,1]$*,*

$$\Delta^2_{\textit{hard}} := \frac{1}{N}\sum_{i=1}^N (\rho_i - \bar{\rho})^2 = \widehat{\operatorname{Var}}_i(\rho_i), \qquad \bar{\rho} := \frac{1}{N}\sum_i \rho_i.$$

Because the success indicator $S_\eta(\tau)$ is binary, ρ_i fully summarizes client i's thresholded success profile, and the sample variance $\widehat{\mathrm{Var}}_i(\rho_i)$ is the standard inter-client dispersion summary. The range is $0 \le \Delta^2_{\text{hard}} \le \bar{\rho}(1-\bar{\rho}) \le 1/4$. The upper bound is attained when each $\rho_i \in \{0, 1\}$.

K.2 Primary Properties of the Three Projections

The three projections (p_i, n_i, ρ_i) each summarize a distinct first-order aspect of $\mathcal{D}_{\tau_i}$. We collect their three key properties in a single lemma.

Lemma 1 (Primary Properties P1–P3)**.** *The three projections and their dispersion measures satisfy:*

- ***(P1) Well-defined under sampling:*** *each projection is a functional of $\mathcal{D}_{\tau_i}$ that is invariant to permutations of task identities preserving ℓ and ρ. When X_i is sampled from a generating distribution, p_i and ρ_i are unbiased for its marginals with $O(1/\sqrt{n_i})$ deviation, and n_i is set exactly by the protocol.*
- ***(P2) Bounded:*** *the dispersion measures are bounded: $\Delta^2_{pref} \le 1 - 1/C$ and $\Delta^2_{hard} \le 1/4$ intrinsically, and $\Delta^2_{cov} \le C_n$ within the* CoveragePartition *construction regime, where C_n is the construction constant of Proposition 2 (for general nonnegative pool sizes the intrinsic bound is $\mathrm{CV}^2 \le N - 1$).*
- ***(P3) Monotone-controllable:*** *each projection's inter-client dispersion is controlled by a single hyperparameter (Propositions 1, 2, 3) that varies the dispersion monotonically over the realizable range of Theorem 11 (an open interval inside the definitional bounds).*

Proof. For (P1), p_i is the marginal of $\mathcal{D}_{\tau_i}$ along ℓ, ρ_i is its mass on the success set of S_η, and $n_i = |X_i|$ is its support size, set by the partition protocol. All three are permutation-invariant. Relative to a generating distribution, $p_{i,c}$ and ρ_i are sample means of bounded indicators, so their standard deviation is $O(1/\sqrt{n_i})$ by standard concentration (*e.g.*, Hoeffding). The pool size n_i has zero estimation noise. For (P2), the three bounds follow from *distinct* constraints: $\Delta^2_{\text{pref}} \le 1 - 1/C$ from the simplex geometry (attained by one-hot clients spread evenly across the C categories); $\Delta^2_{\text{cov}} \le C_n$ is the upper limit of the realizable range under CoveragePartition, where pool sizes have bounded support $[L_{\min}, L_{\max}]$ at fixed mean and the squared coefficient of variation is bounded by the Bhatia–Davis/Popoviciu inequality (this limit comes from the construction regime, and the intrinsic bound for nonnegative sizes is $\mathrm{CV}^2 \le N - 1$); and $\Delta^2_{\text{hard}} \le \bar{\rho}(1-\bar{\rho}) \le 1/4$ from the Bernoulli variance. For (P3), PreferencePartition, CoveragePartition, and HardnessPartition provide closed-form monotone maps from ω, ξ, ξ' to the corresponding measure (Propositions 1–3, deferred to Appendix L). □

Remark 1 (Heuristic gradient-heterogeneity interpretation)**.** *Each measure is motivated by a distinct mechanism contributing to the inter-client gradient variance: preference via the Cauchy–Schwarz bound sketched after Definition K.1 (this bound relies on the within-category-homogeneity assumption $g_{i,c} \approx \bar{g}_c$), coverage via the heuristic two-level-variance/Jensen-gap interpretation noted after Definition K.2, and hardness via the degradation of group-relative advantage signals as $\rho_i \to \{0, 1\}$ (§3.2). We emphasize that these mechanisms are motivating heuristics: none is formalized in this paper, and none of our theorems depends on them. The formal content of this appendix is the boundedness, orthogonality, and sufficiency statements themselves.*

K.3 Definitional Orthogonality and First-Order Sufficiency

Theorem 6 (Definitional Orthogonality)**.** *The three measures depend on disjoint first-order projections of the partition. Specifically, there exist projection maps $\pi_p : \{X_i\} \mapsto \{p_i\}$, $\pi_n : \{X_i\} \mapsto \{n_i\}$, $\pi_\rho : \{X_i\} \mapsto \{\rho_i\}$ and functionals f_p, f_n, f_ρ such that*

$$\Delta^2_{pref} = f_p \circ \pi_p, \qquad \Delta^2_{cov} = f_n \circ \pi_n, \qquad \Delta^2_{hard} = f_\rho \circ \pi_\rho,$$

so that the three measures are functionally independent (each a functional of one projection, invariant to the other two). The definitional ranges (the intervals implied by the simplex, CV^2, and Bernoulli-variance constraints) are the outer feasibility bounds. The realizable region obtained by composing the three single-hyperparameter constructions is the open box $\mathcal{R} = (0,\, 1-\|p_0\|_2^2) \times (0,\, C_n) \times (0,\, C_h)$ of Theorem 11, with C_n and C_h the construction constants defined in Appendix L.2.1. This box lies

inside the definitional bounds (e.g., $1 - \|p_0\|_2^2 \le 1 - 1/C$*). Every target triple in* $\mathcal{R}$ *is achievable in expectation by varying* $\{p_i\}, \{n_i\}, \{\rho_i\}$ *independently, under the hypotheses of Theorem 11.*

Proof. The three functionals are taken directly from Definitions K.1–K.3: $f_p(\{p_i\}) = (1/N)\sum_i \|p_i - \bar{p}\|_2^2$ depends only on $\{p_i\}$, $f_n(\{n_i\}) = (1/N)\sum_i((n_i - \bar{n})/\bar{n})^2$ depends only on $\{n_i\}$, and $f_\rho(\{\rho_i\}) = (1/N)\sum_i(\rho_i - \bar{\rho})^2$ depends only on $\{\rho_i\}$. Each f_* is invariant to the other two projections, giving functional independence. Reachability in expectation over the realizable box $\mathcal{R}$ follows by independently varying the three projections, with concrete realizations given by Theorem 11 (under Assumption 1 and the theorem's balanced-pool condition) in Appendix L. The region $\mathcal{R}$ is contained in, and generally strictly smaller than, the outer definitional box. □

Theorem 7 (First-Order Sufficiency)**.** *The triple* (p_i, n_i, ρ_i) *is a sufficient first-order descriptor for the three dispersion measures: if two partitions* $\{X_i\}$ *and* $\{X_i'\}$ *satisfy* $\mathbf{D}_i = \mathbf{D}_i'$ *for all* $i \in [N]$*, then*

$$\Delta^2_{pref}(\{X_i\}) = \Delta^2_{pref}(\{X_i'\}), \quad \Delta^2_{cov}(\{X_i\}) = \Delta^2_{cov}(\{X_i'\}), \quad \Delta^2_{hard}(\{X_i\}) = \Delta^2_{hard}(\{X_i'\}).$$

Consequently, any other first-order moment of $\mathcal{D}_{\tau_i}$ *enters the dispersion measures only through* (p_i, n_i, ρ_i)*.*

Proof. The first statement follows directly from Theorem 6: each Δ^2_* is a function of one of the three projections, so equal projections imply equal measures. The second statement restates the first: if a moment m_i of $\mathcal{D}_{\tau_i}$ affected, *e.g.*, Δ^2_{pref}, but were not encoded in p_i, such a moment would contradict $\Delta^2_{\text{pref}} = f_p(\{p_i\})$. □

Theorem 8 (Converse Fails)**.** *The converse of Theorem 7 does not hold: two partitions can have identical* $(\Delta^2_{pref}, \Delta^2_{cov}, \Delta^2_{hard})$ *yet differ in the client-by-client descriptor sequence, i.e.,* $\mathbf{D}_i \neq \mathbf{D}_i'$ *for some* i*. Separately, the full distribution* $\mathcal{D}_{\tau_i}$ *cannot be recovered from* (p_i, n_i, ρ_i) *alone.*

Proof. A two-client, two-category counterexample suffices for Δ^2_{pref}. Take partition A with $p_1^A = (0.6, 0.4)$, $p_2^A = (0.4, 0.6)$, giving $\bar{p}^A = (0.5, 0.5)$ and $\Delta^2_{\text{pref}} = 0.02$. Also take partition B with $p_1^B = (0.5, 0.5)$, $p_2^B = (0.3, 0.7)$, giving $\bar{p}^B = (0.4, 0.6)$ and again $\Delta^2_{\text{pref}} = 0.02$. The two partitions share the dispersion value but differ at the client level. The same construction (with constant n_i, ρ_i) extends to any combination of measures. The non-recoverability of $\mathcal{D}_{\tau_i}$ is a separate (and immediate) observation: two distributions with the same (p_i, n_i, ρ_i) can differ in higher-order structure such as the within-client difficulty profile above the threshold η or the type–difficulty joint distribution, none of which is captured by the first-order descriptor. This limitation is the cost of the projection-based notion of heterogeneity that we adopt. □

L Task-Level Partition Algorithms

This appendix supports §3.2 with concrete details on how the three task-level sub-types of Agent Heterogeneity (preference, coverage, hardness) are operationalized in our experiments. Appendix L.1 presents the pseudocode of PREFERENCEPARTITION, COVERAGEPARTITION, and HARDNESSPARTITION. Appendix L.2 states and proves the four design properties (D1)–(D4) connecting these algorithms to the dispersion measures of Appendix K. Appendix L.3 visualizes the per-client distributions induced on WebShop and ALFWorld at the near-uniform and most-heterogeneous settings used in §5.2.

L.1 Pseudocode

We provide the pseudocode of the three partition procedures introduced in §3.2. Each procedure realizes one heterogeneity sub-type through a single hyperparameter (ω for preference, ξ for coverage, ξ' for hardness) that monotonically controls the corresponding dispersion measure (Δ^2_{pref}, Δ^2_{cov}, Δ^2_{hard}). At the same time, each keeps the first-order means $(\bar{p}, \bar{n}, \bar{\rho})$ invariant as the hyperparameters vary, so any observed effect is attributable to dispersion alone (cf. design properties (D1)–(D3) in Appendix F.2). The three procedures also admit a joint construction (D4), a composite sampler that reaches any feasible target triple under threshold independence (Theorem 11). Below we describe each algorithm in turn.

L.1.1 PREFERENCEPARTITION

PREFERENCEPARTITION (Algorithm 2) controls whether each client's category mix concentrates near the global category marginal p_0 (small ω) or spreads away from it (large ω). The Dirichlet concentration vector is reparameterized as $p_0 \cdot (1-\omega)/\omega$, which gives a one-parameter Dirichlet family with mean fixed at p_0 for all ω and variance monotonically increasing in ω: as $\omega \to 0$ all clients have the global marginal (i.i.d. limit), while as $\omega \to 1$ each client collapses onto a single category (one-hot vertex limit). The capacity-fix step (line 5) ensures that the per-client set size L is preserved exactly even when the multinomial count for a category exceeds that category's size, so $\bar{n}$ remains constant across all values of ω. The leftover-redistribution rule reassigns surplus samples proportionally to q over categories with spare capacity. This rule preserves the per-client category marginal in expectation up to the capacity constraint and avoids systematic bias toward larger pools. On WebShop, where fashion goals make up 78% of the training pool, the implementation first subsamples the training pool to a random 20% of its fashion goals and then smooths the marginal as $p_0 \leftarrow (p_0 + 0.01)/(1 + 0.01\,C)$. The WebShop p_0 is therefore $(0.088, 0.081, 0.407, 0.369, 0.055)$ over (beauty, electronics, fashion, garden, grocery), and the WebShop preference runs train on this mixture, while the other WebShop runs use the full pool.

Algorithm 2 PREFERENCEPARTITION

Require: Category pools $\{\mathcal{I}_c\}_{c=1}^{C}$ with sizes m_c; total clients N; per-client set size L; spread $\omega \in (0,1)$

Ensure: Client datasets $X_1, \ldots, X_N$ with $|X_i| = L$

1: $p_{0,c} \leftarrow m_c / \sum_{j=1}^{C} m_j$ for $c = 1 \ldots C$ ▷ global category marginal (reference)
2: **for** $i = 1$ to N **do**
3: $(q_1, \ldots, q_C) \sim \text{Dirichlet}\big(p_{0,1}\frac{1-\omega}{\omega}, \ldots, p_{0,C}\frac{1-\omega}{\omega}\big)$ ▷ larger $\omega \Rightarrow$ higher variance; $\mathbb{E}[q] = p_0$ for all ω
4: $(a_1, \ldots, a_C) \sim \text{Multinomial}(L;\, q_1, \ldots, q_C)$ ▷ category counts for client i
5: **if** any $a_c > m_c$ **then** set $a_c \leftarrow \min(a_c, m_c)$ and redistribute leftover by q to categories with spare capacity ▷ capacity fix within a set
6: $X_i \leftarrow \bigcup_{c=1}^{C} \text{SAMPLEWITHOUTREPLACEMENT}(\mathcal{I}_c, a_c)$
7: **end for**
8: **return** $\{X_i\}_{i=1}^{N}$

L.1.2 COVERAGEPARTITION

COVERAGEPARTITION (Algorithm 3) controls the spread of per-client pool sizes $\{n_i\}$ on a bounded interval $[L_{\min}, L_{\max}]$, while fixing two global quantities: the per-client mean L_{avg} and the average number of replicas per item r (and hence the total number of assignments $A_{\text{tot}} = \lfloor rN_{\text{pool}} \rceil$). The Beta family parameterized by $(\mu\xi, (1-\mu)\xi)$ has mean μ for all ξ, with $\xi \to \infty$ collapsing the sampled fractions $\{x_i\}$ to μ (so $\{n_i\}$ concentrates at L_{avg}) and smaller ξ widening the spread toward the boundaries $L_{\min}, L_{\max}$ ($\xi = 1$ is the most-dispersed setting used in our experiments). The post-renormalization to sum A_{tot} followed by the largest-remainder rounding (lines 5–6) realizes the prescribed sizes exactly while respecting the per-client bounds. The weighted no-replacement placement loop (lines 10–13) then distributes items so that each item j appears in exactly z_j clients, where z_j is either $\lfloor r \rfloor$ or $\lceil r \rceil$ (algorithm lines 7–8), keeping the global replica budget exact and decoupling cross-client dispersion from the total exposure budget. As a result, varying ξ changes only Δ^2_{cov} and leaves the global category and difficulty marginals invariant at leading order (Theorems 9 and 10).

Algorithm 3 COVERAGEPARTITION

Require: total items N_{pool} (indexed $1{:}N_{\text{pool}}$); total clients N; 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 $X_1, \ldots, X_N$

1: $A_{\text{tot}} \leftarrow \lfloor rN_{\text{pool}} \rceil$ ▷ total assignments (sum of all $|X_i|$); keeps global overlap fixed
2: **assert** $NL_{\min} \le A_{\text{tot}} \le NL_{\max}$ ▷ feasibility under per-client bounds
3: $\mu \leftarrow (L_{\text{avg}} - L_{\min})/(L_{\max} - L_{\min})$; $\alpha \leftarrow \mu\xi,\ \beta \leftarrow (1-\mu)\xi$ ▷ Beta mean μ maps to size mean L_{avg}
4: Sample $x_i \sim \text{Beta}(\alpha, \beta)$ for $i = 1 \ldots N$ ▷ larger $\xi \Rightarrow$ lower variance (sizes closer to L_{avg})
5: $u_i \leftarrow L_{\min} + x_i(L_{\max} - L_{\min})$; $u_i \leftarrow u_i \cdot \dfrac{A_{\text{tot}}}{\sum_j u_j}$ ▷ shape then renormalize to sum A_{tot}
6: $n_i \leftarrow \text{ROUNDTOSUM}(u,\ A_{\text{tot}},\ [L_{\min}, L_{\max}])$ ▷ largest remainder with clipping to $[L_{\min}, L_{\max}]$
7: $z_{\min} \leftarrow \lfloor r \rfloor$, $z_{\max} \leftarrow \lceil r \rceil$, $N_{\text{hi}} \leftarrow A_{\text{tot}} - z_{\min}N_{\text{pool}}$
8: Set $z_j \leftarrow z_{\max}$ for any N_{hi} items; $z_j \leftarrow z_{\min}$ otherwise
9: Initialize $X_i \leftarrow \emptyset$, $\text{rem}_i \leftarrow n_i$ for all i
10: **for** $j = 1$ to N_{pool} **do** ▷ weighted, no-replacement placement across clients
11: $\mathcal{C}_{\text{avail}} \leftarrow \{i : \text{rem}_i > 0\}$; choose z_j distinct $i \in \mathcal{C}_{\text{avail}}$ with $\Pr(i) \propto \text{rem}_i$
12: Add item j to each chosen X_i and decrement the corresponding rem_i
13: **end for**
14: **return** $\{X_i\}_{i=1}^N$

L.1.3 HARDNESSPARTITION

HARDNESSPARTITION (Algorithm 4) controls the spread of per-client thresholded success rates $\{\rho_i\}$, defined as the fraction of "successful" tasks (those with reference checkpoint success rate $\rho(\tau) \ge \eta$) in each client's pool. The procedure is a two-stage construction: it first invokes COVERAGEPARTITION on the successful index set $\mathcal{Y}$ with bounds $(\kappa_{\min}, \kappa_{\text{avg}}, \kappa_{\max})$ and dispersion ξ' to determine each client's count of successful items, then fills the remainder $L - |Y_i|$ from the unsuccessful pool $\mathcal{U}$ to enforce a constant per-client size $|X_i| = L$. As ξ' grows, the success counts $\{|Y_i|\}$ concentrate at κ_{avg} so all clients receive the same fraction of successful tasks (uniform ρ). As ξ' shrinks ($\xi' = 1$ in our experiments), the success counts spread toward $\kappa_{\min}$ and $\kappa_{\max}$ so ρ_i ranges from near-zero (few tasks that π_{ref} solves) to near-one (mostly tasks that π_{ref} solves). Because the unsuccessful filler uses simple without-replacement sampling and the per-client size is held at L, both $\bar{\rho}$ and $\bar{n}$ are preserved for every value of ξ'. This preservation isolates the effect of Δ^2_{hard} on federated training dynamics (Theorems 9 and 10). The isolation holds at leading order, under the approximate threshold-independence discussed in Assumption 1.

L.2 Design Properties (D1)–(D4)

This subsection states and proves the four design properties (D1)–(D4) connecting the three partition algorithms of Appendix L.1 (PREFERENCEPARTITION, COVERAGEPARTITION, and HARDNESSPAR-

Algorithm 4 HARDNESSPARTITION

Require: total items N_{pool} (indexed $1{:}N_{\text{pool}}$); disjoint index sets $\mathcal{Y}$ (successful) and $\mathcal{U}$ (unsuccessful) with $\mathcal{Y} \cup \mathcal{U} = \{1{:}N_{\text{pool}}\}$; total clients N; per-client set size L; hyperparameters for COVERAGEPARTITION: bounds $(\kappa_{\min}, \kappa_{\text{avg}}, \kappa_{\max})$ with $\kappa_{\max} \le L$, dispersion ξ', overlap r

Ensure: Client datasets $X_1, \ldots, X_N$ with $|X_i| = L$

$\{Y_i\}_{i=1}^N \leftarrow$ COVERAGEPARTITION$\big(|\mathcal{Y}|, N, (\kappa_{\min}, \kappa_{\text{avg}}, \kappa_{\max}), \xi', r\big)$ ▷ larger $\xi' \Rightarrow$ lower variance

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

  $\nu_i \leftarrow L - |Y_i|$; $F_i \leftarrow$ SAMPLEWITHOUTREPLACEMENT$(\mathcal{U}, \nu_i)$

  $X_i \leftarrow Y_i \cup F_i$

end for

return $\{X_i\}_{i=1}^N$

TITION) to the dispersion measures introduced in Appendix K. (D1) Each algorithm's hyperparameter monotonically controls one measure (Propositions 1–3). (D2) Varying that hyperparameter leaves the other two measures invariant in expectation, under a threshold-independence assumption that holds approximately in practice (Theorem 9). (D3) All first-order means are preserved as the hyperparameters vary (Theorem 10). (D4) The three partitions compose to reach any feasible target triple (Theorem 11).

L.2.1 Single-Hyperparameter Monotone Control (D1)

We now show that each partition algorithm has a single hyperparameter that monotonically controls the corresponding measure in closed form. Fix the global type marginal $p_0 \in \Delta^{C-1}$ with $p_{0,c} > 0$ for all c, the size bounds $(L_{\min}, L_{\text{avg}}, L_{\max})$ with $\mu = (L_{\text{avg}} - L_{\min})/(L_{\max} - L_{\min})$, and the success-count bounds $(\kappa_{\min}, \kappa_{\text{avg}}, \kappa_{\max})$ with $\mu' = (\kappa_{\text{avg}} - \kappa_{\min})/(\kappa_{\max} - \kappa_{\min})$. Set $C_n := (L_{\max} - L_{\min})^2 \mu(1-\mu)/L_{\text{avg}}^2$ and $C_h := (\kappa_{\max} - \kappa_{\min})^2 \mu'(1-\mu')/L^2$. These two constants will appear as the upper limits of the realizable dispersion ranges in the closed-form control laws of Propositions 2 and 3 below. We adopt two standing conventions so that the first-order references are shared across the three algorithms: (i) the replica budget is set to $r = NL_{\text{avg}}/N_{\text{pool}}$, so the total assignment budget satisfies $A_{\text{tot}} = \lfloor rN_{\text{pool}} \rceil = NL_{\text{avg}}$ and COVERAGEPARTITION's renormalization to A_{tot} fixes $\bar{n} = L_{\text{avg}}$; and (ii) the success-quota mean is matched to the global thresholded success rate, $\kappa_{\text{avg}}/L = \bar{s} = \Pr[S_\eta = 1]$, so the success-rate reference $\bar{\rho}^\star = \kappa_{\text{avg}}/L = \bar{s}$ is a single shared value across PREFERENCEPARTITION, COVERAGEPARTITION, and HARDNESSPARTITION. Only the joint construction of Theorem 11 uses convention (ii). Properties (D1)–(D3) do not use it, and Theorem 10 states the HARDNESSPARTITION reference as κ_{avg}/L. Our single-axis hardness experiments do not follow convention (ii). They use $(\kappa_{\min}, \kappa_{\text{avg}}, \kappa_{\max}) = (0, L/2, L)$, so $\mu' = 1/2$, $C_h = 1/4$, and $\bar{\rho} = 1/2$ exactly for every ξ'. PREFERENCEPARTITION and COVERAGEPARTITION keep $\bar{\rho}$ close to $\bar{s}$: the measured $\bar{\rho}$ is 0.28 on WebShop and 0.59–0.62 on ALFWorld. The control laws below are stated in the limit $N, L \to \infty$: at finite (N, L), expectations carry corrections of order $O(1/L) + O(1/N)$, and realized values deviate from them by $O(1/\sqrt{L}) + O(1/\sqrt{N})$.

Proposition 1 (Preference Control: ω for Δ^2_{pref})**.** *Under* PREFERENCEPARTITION *with* $q_i \sim \text{Dirichlet}(p_0 \cdot (1-\omega)/\omega)$ *followed by category counts of* X_i *sampled from* $\text{Multinomial}(L; q_i)$*,*

$$\mathbb{E}[\Delta^2_{\textit{pref}}(\omega)] = \omega \cdot (1 - \|p_0\|_2^2),$$

which is linear in ω*, strictly increasing on* $(0,1)$*, invertible by* $\omega^*(\delta_p) = \delta_p/(1-\|p_0\|_2^2)$*, and covers the open range* $(0, 1-\|p_0\|_2^2)$*.*

Proof. With $\alpha := (1-\omega)/\omega$, the per-client Dirichlet covariance has trace $\text{tr}(\text{Cov}(q_i)) = (1 - \|p_0\|_2^2)/(\alpha+1) = \omega(1-\|p_0\|_2^2)$. As $L \to \infty$, $p_i \to q_i$ almost surely (multinomial concentration), and as $N \to \infty$, $\bar{p} \to \mathbb{E}[q_i] = p_0$ almost surely (LLN). Combining these limits gives $\mathbb{E}[\Delta^2_{\text{pref}}] \to \mathbb{E}[\|q_i - p_0\|_2^2] = \omega(1-\|p_0\|_2^2)$. Ignoring the capacity-fix step, the computation is exact at finite (N, L): $\text{tr}\,\text{Cov}(p_i) = (1 - \|p_0\|_2^2)\big(\omega + (1-\omega)/L\big)$, hence $\mathbb{E}[\Delta^2_{\text{pref}}] = \frac{N-1}{N}(1-\|p_0\|_2^2)\big(\omega + (1-\omega)/L\big)$, which tends to $\omega(1-\|p_0\|_2^2)$. Strict monotonicity follows from $\partial_\omega[\omega(1-\|p_0\|_2^2)] = 1 - \|p_0\|_2^2 > 0$ when p_0 is non-degenerate. Linear inversion gives $\omega^*(\delta_p) = \delta_p/(1-\|p_0\|_2^2)$, and the image of the strictly monotone map on $(0,1)$ is the open interval $(0, 1-\|p_0\|_2^2)$. □

Proposition 2 (Coverage Control: ξ for Δ^2_{cov})**.** *Under* CoveragePartition *with* $x_i \sim \text{Beta}(\mu\xi, (1-\mu)\xi)$, $u_i = L_{\min} + x_i(L_{\max} - L_{\min})$, *followed by global renormalization* $u_i \leftarrow u_i \cdot NL_{avg}/\sum_j u_j$ *and integer rounding to* $\{n_i\}$,

$$\mathbb{E}[\Delta^2_{cov}(\xi)] = \frac{C_n}{\xi + 1},$$

which is strictly decreasing on $(0, \infty)$, *invertible by* $\xi^*(\delta_c) = C_n/\delta_c - 1$, *and covers the open range* $(0, C_n)$.

Proof. The Beta moments give $\mathbb{E}[x_i] = \mu$ and $\text{Var}(x_i) = \mu(1-\mu)/(\xi+1)$. Linear transformation yields $\mathbb{E}[u_i] = L_{\text{avg}}$ and $\text{Var}(u_i) = (L_{\max} - L_{\min})^2\mu(1-\mu)/(\xi+1)$. Renormalization enforces $\bar{n} = L_{\text{avg}}$ exactly (the rescaling factor concentrates to 1 as $N \to \infty$), so $\mathbb{E}[\Delta^2_{\text{cov}}] \to \text{Var}(u_i)/L^2_{\text{avg}} = C_n/(\xi+1)$. Monotonicity follows from $\partial_\xi[C_n/(\xi+1)] = -C_n/(\xi+1)^2 < 0$. Inversion gives $\xi^*(\delta_c) = C_n/\delta_c - 1$, and the image is the open interval $(0, C_n)$. Finite-N renormalization induces $O(1/N)$ negative correlation and integer rounding induces $O(1/L^2_{\text{avg}})$ residuals. Both are small at the experimental scale ($N = 100$ and $L_{\text{avg}} = 500$, cf. the coverage distributions of Figure 15). The closed form is an expectation in the large-N limit, and at $\xi = 1$ the realized $\Delta^2_{\text{cov}} = 0.46$ exceeds $C_n/2 = 0.40$ (Figure 13). □

Proposition 3 (Hardness Control: ξ' for Δ^2_{hard})**.** *Under* HardnessPartition *with* $\{|Y_i|\}$ *sampled via a Beta-shaped quota of dispersion* ξ' *from the success set* $\mathcal{Y}$ *(mean* κ_{avg} *on* $[\kappa_{\min}, \kappa_{\max}]$*), and* $X_i = Y_i \cup F_i$ *with* F_i *filled from* $\mathcal{U}$ *to size* L,

$$\mathbb{E}[\Delta^2_{hard}(\xi')] = \frac{C_h}{\xi' + 1},$$

which is strictly decreasing on $(0, \infty)$, *invertible by* $\xi'^*(\delta_h) = C_h/\delta_h - 1$, *and covers the open range* $(0, C_h)$.

Proof. Apply Proposition 2 to $\{|Y_i|\}$: $\text{Var}(|Y_i|) = (\kappa_{\max} - \kappa_{\min})^2\mu'(1-\mu')/(\xi'+1)$. Since $\rho_i = |Y_i|/L$, dividing by L^2 gives $\text{Var}(\rho_i) = C_h/(\xi'+1)$. The construction $|X_i| = L$ enforces $\bar{\rho} = \kappa_{\text{avg}}/L$ exactly. Monotonicity, inversion, and range coverage follow by the same arguments as in the proof of Proposition 2, with C_h in place of C_n and ξ' in place of ξ. □

L.2.2 Cross-Measure Isolation (D2)

To formalize the claim that varying one hyperparameter does not affect the other measures, we need a mild assumption on how type and difficulty relate in the global pool. In the thresholded interpretation of Definition K.3, the operational assumption is threshold-independence (TI):

Assumption 1 (Threshold-Independence at level η)**.** *The type label* ℓ *and the success indicator* S_η *are independent on the global task pool:* $\Pr[\ell = c, S_\eta = s] = \Pr[\ell = c] \cdot \Pr[S_\eta = s]$ *for all* $c \in [C], s \in \{0, 1\}$. *Equivalently,* $s_c := \Pr[S_\eta = 1 \mid \ell = c]$ *is constant across c, with common value* $\bar{s} = \Pr[S_\eta = 1]$.

TI is operational (it can be tested directly via a chi-squared test on (ℓ, S_η)) and is strictly weaker than full independence $\ell \perp \rho$. TI is also incomparable with the mean-independence condition $\mathbb{E}[\rho \mid \ell = c] \equiv \mathbb{E}[\rho]$ (here $\mathbb{E}[\rho]$ is the global mean of the continuous ρ and differs from the inter-client mean $\bar{\rho}$). Mean independence constrains the conditional mean but not the thresholded marginal. On our experimental task pools with the reference success labels, the test does not reject TI on WebShop ($\chi^2_4 = 6.7$, $p = 0.15$, $\epsilon_{\text{TI}} := \max_c |s_c - \bar{s}| = 0.07$), but rejects it on ALFWorld ($\chi^2_5 = 297$, $p < 10^{-60}$), where s_c ranges from 0.40 (Pick2) to 0.78 (Pick) across the six task types ($\epsilon_{\text{TI}} = 0.20$). TI therefore holds only approximately in practice. The results below weaken in a controlled way under approximate TI: if $\max_c |s_c - \bar{s}| \le \epsilon_{\text{TI}}$, then every cell of Theorems 9 and 10 that invokes TI holds up to an additional additive $O(\epsilon_{\text{TI}})$ term, and the TI-free cells are unaffected. The isolation guarantees should therefore be read as leading-order approximations, with an $O(\epsilon_{\text{TI}})$ cross-measure residual on the affected cells.

Theorem 9 (Cross-Measure Isolation, D2)**.** *Under Assumption 1, for each partition algorithm* $\text{Part}_\lambda \in \{\text{PreferencePartition}_\omega, \text{CoveragePartition}_\xi, \text{HardnessPartition}_{\xi'}\}$ *with*

hyperparameter λ and target measure Δ_a^2, every other measure Δ_b^2 ($b \neq a$, with $a, b \in \{pref, cov, hard\}$) satisfies

$$\mathbb{E}[\Delta_b^2(\lambda)] = \text{const} + O(1/L) + O(1/N) \qquad \forall\lambda,$$

with realized values within $O(1/\sqrt{L}) + O(1/\sqrt{N})$ of this expectation. In particular, at leading order, $\partial_\omega \mathbb{E}[\Delta_{cov}^2] = \partial_\omega \mathbb{E}[\Delta_{hard}^2] = \partial_\xi \mathbb{E}[\Delta_{pref}^2] = \partial_\xi \mathbb{E}[\Delta_{hard}^2] = \partial_{\xi'} \mathbb{E}[\Delta_{pref}^2] = \partial_{\xi'} \mathbb{E}[\Delta_{cov}^2] = 0$.

Proof. We sketch the six cross-cells. *(I.a)* PREFERENCEPARTITION $\times\Delta_{\text{cov}}^2$: the algorithm enforces $|X_i| = L$ exactly, so $\Delta_{\text{cov}}^2 \equiv 0$ for all ω. *(I.b)* PREFERENCEPARTITION $\times\Delta_{\text{hard}}^2$: the per-client expected thresholded success rate given q_i is $\sum_c q_{i,c} s_c$. Under TI, $s_c \equiv \bar{s}$, so $\mathbb{E}[\rho_i \mid q_i] = \bar{s} \sum_c q_{i,c} = \bar{s}$. This expectation is independent of q_i (hence of ω). The remaining inter-client variance is the sampling-noise floor $\bar{s}(1-\bar{s})/L + O(1/N)$. *(II.a, II.b)* COVERAGEPARTITION $\times(\Delta_{\text{pref}}^2, \Delta_{\text{hard}}^2)$: the algorithm performs weighted-without-replacement sampling with weights proportional to remaining capacity, which is independent of ℓ and S_η, so per-client p_i and ρ_i are unbiased estimates of the global marginals with $O(1/\sqrt{n_i})$ noise. No TI is needed here. *(III.a)* HARDNESSPARTITION $\times\Delta_{\text{pref}}^2$: under TI, both $\mathcal{Y}$ and $\mathcal{U}$ have type marginal equal to p_0, so successful and unsuccessful subsamples each yield per-client type marginal p_0 up to $O(1/\sqrt{L})$ noise. *(III.b)* HARDNESSPARTITION $\times\Delta_{\text{cov}}^2$: the algorithm enforces $|X_i| = L$ exactly via $X_i = Y_i \cup F_i$ with $|F_i| = L - |Y_i|$, so $\Delta_{\text{cov}}^2 \equiv 0$ for all ξ'. The leading-order partial derivatives in the statement follow from these constants. □

Measured on the experimental pools. In Figure 13, we run the three partitions of the released code on the task pools of our experiments (6,410 WebShop training goals and 3,553 ALFWorld training games) with the reference success labels of Appendix J.6. Property (D1) holds: each hyperparameter moves its own measure monotonically, and the measure follows its closed form except at $\xi = \xi' = 1$, where it exceeds the closed form by 15–20% (0.46 vs. 0.40 for coverage, 0.15 vs. 0.125 for hardness). The cells of Theorem 9 that do not need TI also hold. PREFERENCEPARTITION and HARDNESSPARTITION keep Δ_{cov}^2 at exactly 0, and COVERAGEPARTITION leaves both other measures at the sampling-noise level (≤ 0.003). The two TI-dependent cells agree with the chi-squared test above. On WebShop they stay at the noise floor. On ALFWorld, PREFERENCEPARTITION raises Δ_{hard}^2 from 0.002 to 0.020 and HARDNESSPARTITION raises Δ_{pref}^2 from 0.008 to 0.020. Both values match the prediction computed from the measured per-type success rates s_c, without assuming TI. These increases are the $O(\epsilon_{\text{TI}})$ residuals discussed above. Relative to the range that each measure's own hyperparameter spans, they amount to 13% and 2%. The extreme-preference ALFWorld run of Figure 5 therefore also carries a small hardness dispersion. Theorem 2 covers that run unchanged, because it holds for any task split under a shared environment.

L.2.3 Factor Invariance (D3)

Theorem 10 (Factor Invariance, D3)**.** *Under Assumption 1, for every partition algorithm* PART_λ *and every first-order mean $\bar{f} \in \{\bar{n}, \bar{p}, \bar{\rho}\}$, we have $\bar{f}(\lambda) = \bar{f}^*$ exactly (by construction), exactly in expectation (by the Dirichlet first moment), or at leading order with $O(1/\sqrt{L}) + O(1/\sqrt{N})$ residuals (for the TI-dependent and sampling cells, as marked in the table), for all λ, where the reference value $\bar{f}^*$ is fixed at L (or L_{avg}), p_0, and $\bar{s}$ (or κ_{avg}/L) respectively. The 9-cell preservation table is:*

Algorithm	$\bar{n}(\lambda)$	$\bar{p}(\lambda)$	$\bar{\rho}(\lambda)$
PREFERENCEPARTITION(ω)	$= L$ *exactly*	$= p_0$ *in expectation*	$= \bar{s}$ *leading-order (TI)*
COVERAGEPARTITION(ξ)	$= L_{avg}$ *exactly*	$= p_0$ *leading-order*	$= \bar{s}$ *leading-order*
HARDNESSPARTITION(ξ')	$= L$ *exactly*	$= p_0$ *leading-order (TI)*	$= \kappa_{avg}/L$ *exactly*

In particular, all 9 cells are preserved as any of the three hyperparameters varies, up to the residuals marked in the table.

Proof. We verify the cells in turn. *Exact-by-construction cells:* PREFERENCEPARTITION fixes $|X_i| = L$ via the $\text{Multinomial}(L; q_i)$ step. COVERAGEPARTITION enforces $\sum_i n_i = N L_{\text{avg}}$ via the renormalization step. HARDNESSPARTITION fixes $|X_i| = L$ via $X_i = Y_i \cup F_i$ and $\sum_i |Y_i| = N\kappa_{\text{avg}}$ via the construction. *Dirichlet first moment:* $q_i \sim \text{Dirichlet}(p_0\alpha)$ has $\mathbb{E}[q_i] = p_0$ for all $\alpha > 0$, so

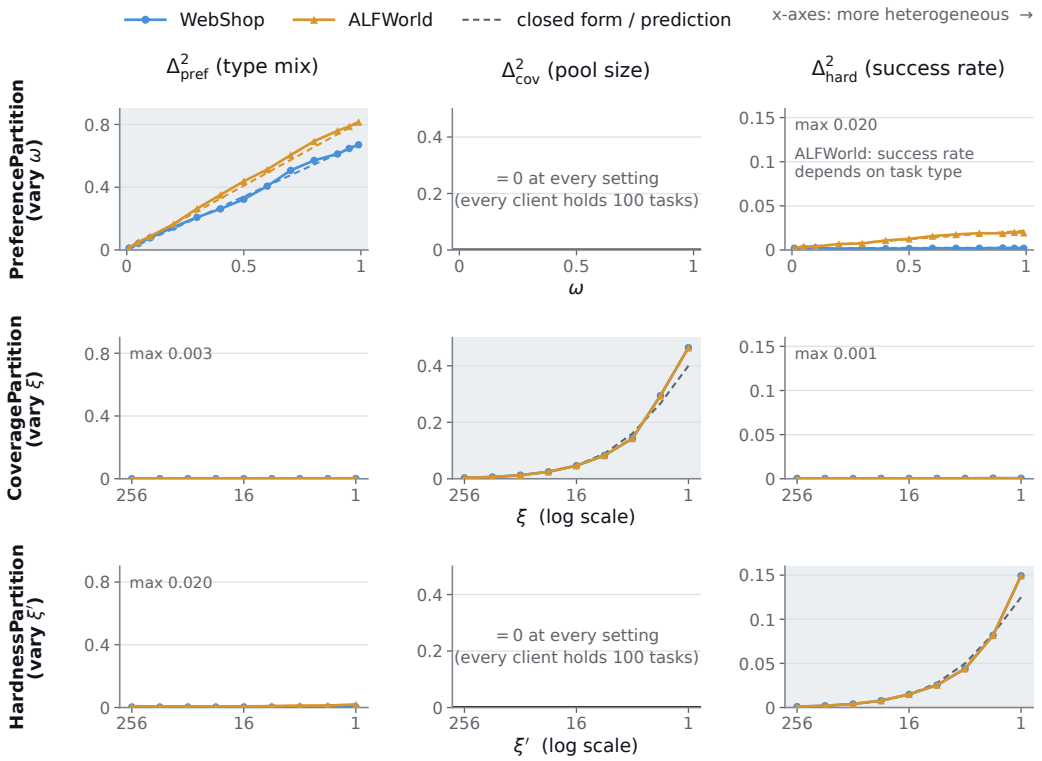


Figure 13: **Each partition moves its own measure and leaves the other two nearly unchanged.** In each row, we vary one hyperparameter of the released partition code on the real task pools, for all $N = 100$ clients with seed 42, so the partitions at the smallest and largest tested values are those run in Figure 5. Each column is one measure of Definitions K.1–K.3, on a single scale per column. Shaded panels hold the measure the row targets, with the closed forms of Propositions 1–3 (dashed). The two cells that rely on Assumption 1, which requires task type and success to be independent, are the top-right and bottom-left panels. They show the prediction computed from the measured per-type success rates s_c, without assuming independence (dashed), and they move only on ALFWorld, where the assumption fails. Larger ω and smaller ξ and ξ' give more heterogeneous splits. Each off-diagonal panel states its largest value over the tested settings. COVERAGEPARTITION and HARDNESSPARTITION set the same quotas on both pools, so their diagonal curves coincide.

$\mathbb{E}[\bar{p}] = p_0$ for all ω. *TI cells:* for PREFERENCEPARTITION $\times\bar{p}$, $\mathbb{E}[\rho_i \mid q_i] = \sum_c q_{i,c} s_c = \bar{s}$ under TI. For HARDNESSPARTITION $\times\bar{p}$, the type marginals of $\mathcal{Y}$ and $\mathcal{U}$ both equal p_0 under TI, so the merged per-client type marginal equals p_0 up to sampling noise. *No-TI cells:* for COVERAGEPARTITION, weighted-without-replacement sampling with capacity weights is independent of type and success indicator, so per-client p_i, ρ_i equal the global marginals at leading order. The sampling-noise residuals are $O(1/\sqrt{L}) + O(1/\sqrt{N})$. □

L.2.4 Joint Configurability (D4)

Theorem 11 (Constructional Orthogonality / Joint Configurability, D4)**.** *Suppose Assumption 1 holds and the global pool is balanced in the sense that every (type, success-stratum) cell contains at least $B_{\min} = \Theta(L)$ tasks. Define the realizable region*

$$\mathcal{R} := \big(0,\, 1 - \|p_0\|_2^2\big) \times \big(0,\, C_n\big) \times \big(0,\, C_h\big),$$

and the measure map $\Phi_{\text{het}}(\{X_i\}) := (\Delta^2_{pref}, \Delta^2_{cov}, \Delta^2_{hard})$. Then for any target triple $(\delta_p, \delta_c, \delta_h) \in \mathcal{R}$, there exists a composite stratified partition with hyperparameters

$$\xi^* = \frac{C_n}{\delta_c} - 1, \qquad \omega^* = \frac{\delta_p}{1 - \|p_0\|_2^2}, \qquad \xi'^* = \frac{C_h}{\delta_h} - 1,$$

all obtained in closed form from Propositions 1–3, such that $\mathbb{E}[\Phi_{\text{het}}(\{X_i\})] = (\delta_p, \delta_c, \delta_h)$ *under the standing conventions and in the limit* $N, L \to \infty$*, with sample-level deviation* $O(1/\sqrt{L}) + O(1/\sqrt{N})$.

Proof. The construction proceeds in three layers and a joint-sampling step. *Layer 1 (size):* sample $\{n_i\}$ from CoveragePartition at ξ^*, fixing Δ^2_{cov} to δ_c by Proposition 2. *Layer 2 (type):* sample per-client type vectors $q_i \sim \text{Dirichlet}(p_0(1-\omega^*)/\omega^*)$, fixing Δ^2_{pref} to δ_p by Proposition 1. *Layer 3 (success):* sample a per-client target success *fraction* ρ^{q}_i from a Beta-shaped quota of dispersion ξ'^* (mean $\bar{\rho}^\star = \kappa_{\text{avg}}/L$ on $[\kappa_{\min}/L, \kappa_{\max}/L]$) and set the success count *size-proportionally*, $|Y_i| = \text{round}(\rho^{\text{q}}_i\, n_i)$. Sampling the fraction rather than the raw count is essential once $\{n_i\}$ varies (Layer 1 with $\delta_c > 0$): it makes $\rho_i = |Y_i|/n_i \approx \rho^{\text{q}}_i$, so $\Delta^2_{\text{hard}} = \text{Var}(\rho_i) = C_h/(\xi'^*+1) = \delta_h$ by Proposition 3 regardless of the Layer-1 size dispersion, and it guarantees $0 \le |Y_i| \le n_i$ so the remaining count $\nu_i = n_i - |Y_i| \ge 0$ is always well-defined. *Joint sampling:* given $(n_i, q_i, |Y_i|)$, sample two stratified multinomials, $n_{i,\cdot,1} \sim \text{Multinomial}(|Y_i|; q_i)$ and $n_{i,\cdot,0} \sim \text{Multinomial}(n_i - |Y_i|; q_i)$, and fill each cell (c, s) by uniform sampling-without-replacement from $\mathcal{T}_{c,s} := \ell^{-1}(c) \cap S_\eta^{-1}(s)$. The balanced-pool condition ensures every cell has enough tasks: writing $N_{\text{pool}} := |\mathcal{T}|$ for the global pool size, under TI the cell $\mathcal{T}_{c,s}$ has size $\approx N_{\text{pool}}\, p_{0,c}\, p^{(s)}$ with $p^{(1)} = \bar{s}$ and $p^{(0)} = 1 - \bar{s}$ (the global success-stratum fractions). This size is at least $B_{\min} = \Theta(L)$ by the balanced-pool condition, and the cell is sampled with r-fold replication to meet the total per-cell demand $\sum_i n_{i,c,s} \approx N L_{\text{avg}}\, p_{0,c}\, p^{(s)} = r N_{\text{pool}}\, p_{0,c}\, p^{(s)}$. Cross-isolation (Theorem 9) ensures that the three layers' targets do not interfere: Layer 1 is exact by construction, Layer 2's type marginal equals q_i on each client (independent of the success split), and Layer 3's success rate equals ρ^{q}_i (independent of the type split by uniform within-cell sampling, and independent of $\{n_i\}$ by the size-proportional sampling). Sample-level concentration $O(1/\sqrt{L}) + O(1/\sqrt{N})$ follows from finite-sample multinomial residuals. □

With Theorem 11, all four properties (D1)–(D4) are proved. Monotone control (D1, Propositions 1–3), cross-measure isolation (D2, Theorem 9), and factor invariance (D3, Theorem 10) compose into joint configurability (D4) over the open realizable region $\mathcal{R}$. Together, the four properties provide the formal foundation for the partition-driven heterogeneity experiments used throughout §5.2 and Appendix L.3.

L.3 Per-Client Distribution Figures

We visualize the per-client task distributions induced by the three partition algorithms of Appendix L.1 at the two extreme dispersion settings used in §5.2: a near-uniform setting (small dispersion, gray curves in the main paper) and an extremely heterogeneous setting (large dispersion, blue curves). The near-uniform setting serves as the baseline, while the heterogeneous setting uses the most dispersed value of each hyperparameter that we run. Two qualitative observations are consistent across all three sub-types and both benchmarks. First, the global marginals are preserved between the two settings up to finite-sample noise (aggregating the per-client bars recovers closely matching global marginals across the ω, ξ, ξ' settings), confirming the factor-invariance property (D3) in Appendix F.2. Second, the per-client variability is much larger in the heterogeneous setting, with individual clients departing far from the global average. Together these observations confirm that each hyperparameter changes Δ^2 along its target axis without shifting the first-order means, up to the small ALFWorld leakage of Appendix L.2.

L.3.1 Preference Heterogeneity

Figure 14 shows per-client category compositions (stacked bars over client IDs) for WebShop (top) and ALFWorld (bottom), at $\omega = 0.01$ (near-uniform) versus $\omega = 0.99$ (extremely heterogeneous). At $\omega = 0.01$, each client's histogram closely matches the global category marginal p_0, with all product categories on WebShop (or all six household task types on ALFWorld) represented in roughly the proportions of p_0 (on WebShop, the fashion-subsampled marginal of Appendix L.1.1), so the cross-client spread of p_i is small. At $\omega = 0.99$, by contrast, almost every client holds a single category (a few hold two), with the dominant categories varying across clients. The pool-level marginal p_0 is unchanged: the client-averaged mix matches p_0 *in expectation* ($\mathbb{E}[q] = p_0$ for all ω by construction), and the realized aggregate deviates only by finite-N sampling noise (visible, *e.g.*, in the smaller categories at $\omega = 0.99$). Meanwhile Δ^2_{pref} rises about fiftyfold. The two benchmarks have

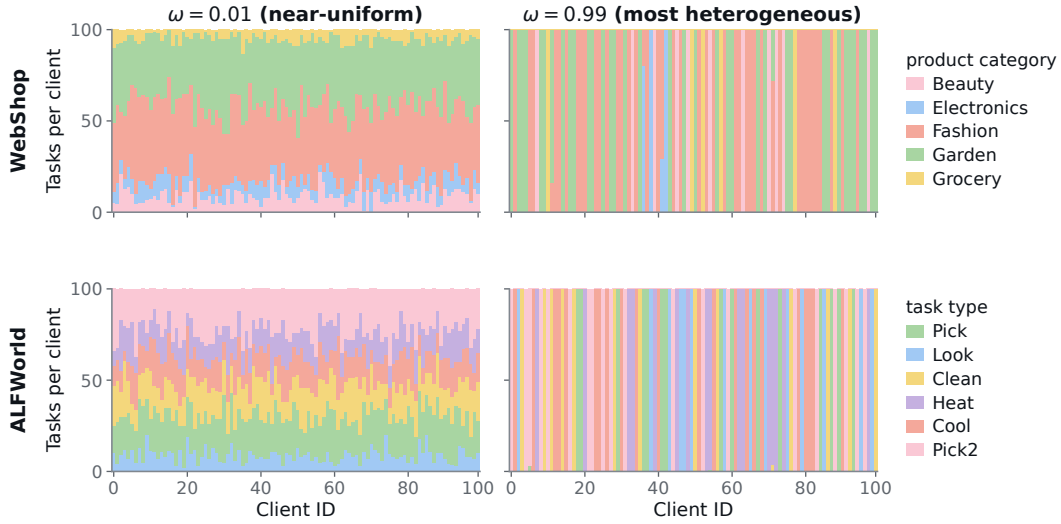


Figure 14: **Client Distribution under Preference Heterogeneity** on WebShop (top, five product categories) and ALFWorld (bottom, six household task types). Each bar is one client's pool of 100 tasks, stacked by category. The left panels show $\omega = 0.01$ (near-uniform), where every client's composition is close to the global marginal p_0. The right panels show $\omega = 0.99$ (most heterogeneous), where almost every client holds a single category. The aggregate across clients matches p_0 in expectation in both settings (up to finite-N sampling noise), so ω moves only the dispersion Δ^2_{pref}, up to the small ALFWorld leakage of Appendix L.2. Coverage and hardness counterparts are in Figures 15 and 16.

different category structures (WebShop has five top-level product categories, two of which are much more frequent than the others, while ALFWorld has six fairly balanced household task types), but the resulting concentration profiles are qualitatively similar on both benchmarks, confirming that ω controls dispersion comparably on both (and, by construction, on any benchmark with a categorical task structure).

L.3.2 Coverage Heterogeneity

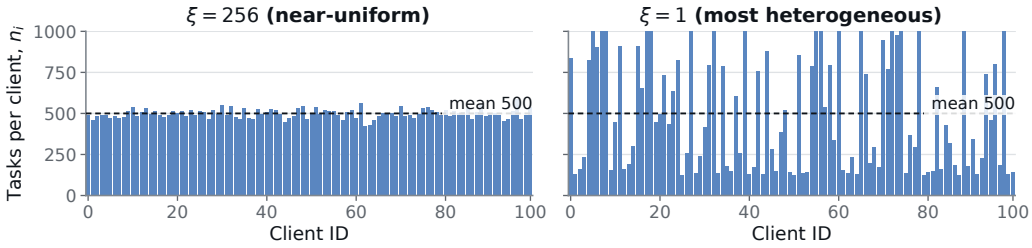


Figure 15: **Client Distribution under Coverage Heterogeneity.** Each bar is one client's pool size n_i, and the dashed line is the mean. The left panel shows $\xi = 256$ (near-uniform), and the right panel shows $\xi = 1$ (most heterogeneous). Smaller ξ thus gives *more* dispersed sizes, the opposite direction to ω in Figure 14. The mean pool size (500 tasks) and the total number of task copies across clients are the same in both settings, so only the spread changes. The construction acts only on task indices, so WebShop and ALFWorld receive the same pool sizes.

Figure 15 shows the empirical distribution of per-client pool sizes $\{n_i\}$ over $[L_{\min}, L_{\text{avg}}, L_{\max}] = [100, 500, 1000]$ at $\xi = 1$ (extremely heterogeneous) and $\xi = 256$ (near-uniform). At $\xi = 256$, the per-client bars are tightly concentrated around $L_{\text{avg}} = 500$, so all clients face essentially the

same overall exploration breadth, and the corresponding Δ^2_{cov} is close to zero. At $\xi = 1$, the sizes spread out to cover the full $[L_{\min}, L_{\max}] = [100, 1000]$ interval, with a non-trivial fraction of clients at or near each boundary. In this setting, some clients repeatedly traverse a narrow region while others explore broadly, and iterative-with-replacement RL sampling is most likely to reveal the per-client breadth limitation discussed in §3.2. The total number of task copies across clients and the per-client mean $L_{\text{avg}} = 500$ are held identical across the two settings, so any cross-setting comparison isolates the dispersion effect from total exposure. Because the partition operates only on task indices, the same construction applies identically to WebShop and ALFWorld and the resulting plots are visually indistinguishable across the two benchmarks. We therefore present a single figure for both benchmarks.

L.3.3 Hardness Heterogeneity

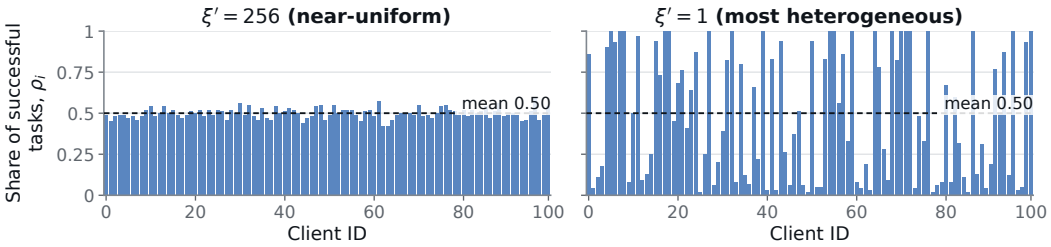


Figure 16: **Client Distribution under Hardness Heterogeneity.** Each bar is one client's thresholded success rate ρ_i, the share of its tasks that the reference checkpoint π_{ref} labels as successful, and the dashed line is the mean. The left panel shows $\xi' = 256$ (near-uniform), and the right panel shows $\xi' = 1$ (most heterogeneous). Every client holds 100 tasks and the mean rate is $\bar{\rho} = 1/2$ in both settings, so only the spread changes. WebShop and ALFWorld use the same quotas, each filled with that benchmark's own successful and unsuccessful tasks.

Figure 16 shows the empirical distribution of per-client thresholded success rates $\{\rho_i\}$ at $\xi' = 1$ (extremely heterogeneous) and $\xi' = 256$ (near-uniform). At $\xi' = 256$, every client receives essentially the same proportion of "successful" tasks (those with $\rho(\tau) \geq \eta$) and "unsuccessful" tasks, so the policy-gradient advantage signal is comparable across clients. At $\xi' = 1$, the thresholded success rate ρ_i spreads toward both extremes: some clients are dominated by easy tasks (those the reference checkpoint solves), while others are dominated by hard tasks where the agent rarely succeeds and produces little positive advantage, especially under group-relative methods such as GRPO whose advantage normalization uses within-group success statistics. Although the two settings differ sharply in inter-client signal availability, the per-client size $|X_i| = L$ and the mean quota $\bar{\rho} = 1/2$ are identical for every value of ξ' by construction (the unsuccessful pool $\mathcal{U}$ supplies the filler items), so any divergence in the federated training comparison of §5.2 can be attributed to Δ^2_{hard}, up to the small ALFWorld leakage of Appendix L.2.

M Environment-Level Heterogeneity Variants

This appendix expands on the construction outlined in §3.3 and supports the empirical results reported in §5.3. Our central methodological claim is that environment-level heterogeneity is intrinsically multi-axis: the transition kernel $P(s' \mid s, a)$ is the composition of several distinct mechanisms, so no single scalar describes it. Perturbing different mechanisms produces structurally different forms of $\pi^\star$ divergence. The resulting forms can preserve different subsets of (C1)–(C3) and hence place the variants at different positions on the Pattern B/C/D spectrum of §4.4. Appendix M.1 formalizes this view as a four-stage transition pipeline and locates several common agent benchmarks within it. Appendix M.2 makes the four stages concrete on WebShop. Appendix M.3 specifies the deterministic per-client environment assignment procedure shared across all five variants (Algorithms 5–6). The remaining subsections (M.4–M.8) instantiate one variant per stage (plus a joint perturbation). For each variant, we give the motivation, an MDP-level formalization of what is changed, an implementation summary (which, for Variants 2–5, specifies the variant pool $\mathcal{V}$ and the SimServer override key), and the predicted pattern under (C1)–(C3) of §4.3. The variants are designed so that, taken together, they span the Pattern B/C/D spectrum used in Figure 6 and isolate which sufficient condition fails along each axis. Finally, Appendix M.9 measures, on a fixed probe set, how each variant changes the search transition and how far apart it places the clients.

M.1 A Four-Stage Transition Pipeline

For any agent that interacts with an external resource (a product catalog, a simulated household, a code repository, an API surface), the transition kernel can be decomposed into a four-stage information-processing pipeline:

$$\underbrace{a}_{\text{action}} \longrightarrow \underbrace{\text{Stage 1}}_{\text{content/catalog}} \longrightarrow \underbrace{\text{Stage 2}}_{\text{encoding/index}} \longrightarrow \underbrace{\text{Stage 3}}_{\text{matching/score}} \longrightarrow \underbrace{\text{Stage 4}}_{\text{rendering}} \longrightarrow s'.$$

Stage 1 (content/catalog) fixes *what* information the environment can in principle return (product set, scene inventory, codebase contents, available API endpoints). **Stage 2 (encoding/index)** determines *how* that content is represented for retrieval (which fields enter an inverted index, which features describe an object, which symbols enter an AST). **Stage 3 (matching/score)** specifies *how* an action queries the encoded representation (BM25 score, similarity metric, action precondition checker, classifier). **Stage 4 (rendering)** maps the matched output back to the agent's observation (top-K ranking, page rendering, room transition, tool-output formatting).

Two clients can disagree at any nonempty subset of these stages, and disagreements at different stages induce structurally different forms of $\pi^\star$ divergence. Stage 1 differences induce a *state-coverage shift*: clients see different reachable (s, a), but the optimal action ordering on the shared support is unchanged. Stage 2 differences induce an *encoding mismatch*: the same catalog is indexed under different bag-of-tokens document representations, so the same query returns different results and the optimal query formulation differs across clients. Stage 3 differences induce a *score-function shift*: identical (encoded) queries return different rankings, so the optimal click order differs across clients. Stage 4 differences induce *observation distortion*: identical rankings are shown under different display rules, so the optimal way to navigate the results differs across clients. Stage 1 differences preserve a shared $\pi^\star$ in Π_Θ and are consistent with (C2) of §4.3 (and with (C1) after abstracting states to their reward-relevant content, cf. the Variant 1 formalization below). Stage 2–4 differences typically break (C2) and force the policy to specialize on quantities (encoding, score, rendering) that are not directly observable, so robustness can then be recovered only through (C1), an off-support shared optimum, or (C3), in-context environment identification. Lookalike Injection is the exception, because its visible cue supports a shared rule (Appendix M.7).

Table 4 illustrates that common retrieval-style and simulator-backed agent benchmarks admit an analogous factorization, and federated training across clients with different content, indexing, scoring, or rendering yields four structurally distinct forms of environment-level heterogeneity. We therefore treat the decomposition as a *systematic taxonomy* of environment-level heterogeneity constructions for agent benchmarks in general. We instantiate at least one variant per stage to test whether the Pattern B/C/D classification of §4.4 depends on the structural form of the perturbation rather than any single variant's idiosyncrasies. All five variants are instantiated on WebShop, and cross-benchmark instantiation is a natural follow-up.

Benchmark	Stage 1 (content)	Stage 2 (encoding)	Stage 3 (matching)	Stage 4 (rendering)
WebShop	product catalog	doc-text fields	BM25 score (k_1, b)	top-K ranking, paging
ALFWorld	scene + inventory	object visibility, room graph	precondition checker	next-room observation
Booking	flight/hotel database	filter index	sort criteria	UI rendering, availability
Code	repository files	AST or symbol index	grep, LSP matching	diff display, test runner
Tool-use	API endpoints	tool-schema parser	router, classifier	tool-output formatter

Table 4: We instantiate the four-stage transition pipeline on representative agent benchmarks. Each row lists the component that can be perturbed at each stage of one benchmark, illustrating that the decomposition extends naturally beyond WebShop.

M.2 The WebShop Pipeline as a Concrete Instantiation

On WebShop, the agent's primary external action $a = \texttt{search}\,\langle q \rangle$ is processed by four explicit components that align one-to-one with the abstract stages above. The agent emits a query q from the current LLM context. **Component 1 (catalog)** determines which products are visible at all: WebShop provides an ASIN-indexed catalog of 1,000 products together with a per-instance candidate filter. **Component 2 (doc-text)** concatenates a configured subset of the per-product fields `name`, `Title`, `description`, `features`, `BulletPoints` into the document string fed to the BM25 indexer. **Component 3 (BM25 score)** computes

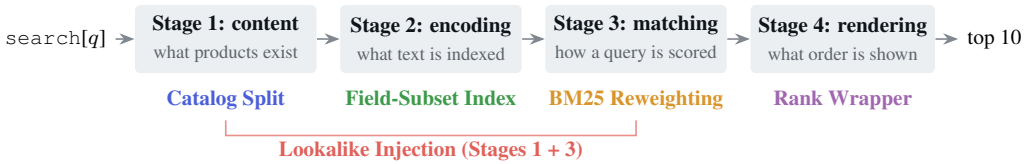
$$\text{score}(q, d) \;=\; \sum_{t \in q} \text{IDF}(t) \cdot \frac{\text{TF}(t, d) \cdot (k_1 + 1)}{\text{TF}(t, d) + k_1 \big(1 - b + b\,|d|/\text{avgdl}\big)},$$

with hyperparameters (k_1, b) controlling term-frequency saturation and length normalization, respectively. **Component 4 (ranking)** sorts by score, takes the top-K candidates, and renders them ten per page, so the agent first sees the top-10 results.

Each component is independently perturbable: the catalog can be sliced (Component 1), the doc-text can be assembled from a different field subset (Component 2), the score formula can be adjusted through (k_1, b) (Component 3), or the post-ranking step can be wrapped to alter the displayed order (Component 4). The five variants below are these four natural single-component perturbations together with one cross-stage adversarial construction, Lookalike Injection, which perturbs Components 1 and 3 jointly: lookalike products injected into the catalog (Component 1) propagate into the score (Component 3) because they are scored by the same BM25 against the same query, and they can be designed so that they are simultaneously promoted by BM25 *and* mismatched against one designated subterm set of the WebShop reward (one or two subterms). All variants share the same task partition (`partition_strategy=uniform`, with 100 goals per client), the same WebShop reward function, and the same `Qwen2.5-1.5B-Instruct` backbone, so that the variants differ from one another only in their transition perturbation (Appendix J.6 lists the backend and retrieval-depth differences from the baseline). Figure 17 places the five variants on this pipeline and lists the configurations of each variant and the number of distinct client environments they produce.

M.3 Deterministic Per-Client Environment Assignment

The five variants below share a common assignment procedure: each of the $N = 100$ clients is mapped *deterministically* to one environment perturbation, and the same client is mapped to the same perturbation in every communication round. Deterministic, round-stable assignment is required by the environment-level FedAvg theory of §4: Theorems 3/4 and 5 all assume each client owns a fixed MDP $\mathcal{M}_i$, so any re-shuffle of $i \mapsto \mathcal{M}_i$ across rounds would change the per-client gradient targets that FedAvg averages. The implementation falls into two templates. Variant 1 (Catalog Split, Appendix M.4) needs per-client catalog construction with two continuous hyperparameters (`env_div`, `keep_ratio`). We present this construction as Algorithm 5. Variants 2–5 share a discrete *variant-pool* template: each defines a finite pool of $|\mathcal{V}|$ configurations, and each client is mapped to one configuration by a deterministic seed. We present this template as Algorithm 6. Both algorithms take the base seed $\sigma_0 = 42$ as input, so all five variants are reproducible from σ_0 together with the variant-specific arguments listed below. Client IDs are 0-based throughout this appendix



Variant	Stage	v_1 (21 clients)	v_2 (19 clients)	v_3 (33 clients)	v_4 (27 clients)	Distinct envs
Catalog Split	1	*no configurations:* own catalog per client (Algorithm 5)				100
Field-Subset Index	2	all fields	name + Title	description‡	BulletPoints	4
BM25 Reweighting	3	$(1.2, 0.75)$	$(1.2, 0.0)$	$(0.3, 0.75)$	$(5.0, 0.75)$	4
Lookalike Injection	1 + 3	price	color	size	price + color	4
Rank Wrapper	4	default	shuffle†	invert	partial-random†	48

reference environment † seeded per client ‡ empty in the released data, so this index holds only product names

Figure 17: **The five environment-level variants on WebShop's search transition, their configurations, and the number of distinct client environments.** Top: the diagram marks the stage each variant perturbs. Bottom: the table lists the four configurations v_1–v_4 of each variant except Catalog Split, with BM25 settings given as (k_1, b). Each client draws one configuration index from a fixed seed and keeps it in all four variants (Algorithm 6), so the client counts in the header hold for all four of these variants. The Lookalike Injection entries name the attribute in which the injected copies differ from their originals, and the partial-random wrapper returns random products on about half of the queries. Rank Wrapper has 48 distinct environments because its shuffle and partial-random clients are seeded individually.

$(i = 0, \ldots, N-1)$, matching the implementation, whose per-client seeds are $\sigma_0 + i$ in Algorithm 6 and $\sigma_0 + 1000\,i$ in Algorithm 5. In formulas, the main text's client $i \in [N]$ corresponds to ID $i - 1$, and the main-text figures use the IDs directly.

Algorithm 5 CATALOGSPLITPARTITION (Variant 1)

Require: Global ASIN pool $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ with $|\mathcal{C}| = 1000$; global instruction-bearing ASIN set $\mathcal{G} \subset \mathcal{C}$ ($|\mathcal{G}| = 415$); per-client goal budget $L = 100$; diversity hyperparameter `env_div` $\in [0, 1]$; retention hyperparameter `keep_ratio` $\in (0, 1]$; total clients N; base seed σ_0

Ensure: Per-client catalogs $\mathcal{C}_0, \ldots, \mathcal{C}_{N-1}$, each paired with an L-goal task pool X_i; $|\mathcal{C}_i| = |\mathcal{G}_i| + \mathrm{round}(\texttt{keep_ratio} \cdot |\mathcal{R}_i|) \approx 702$ (at `keep_ratio` $= 0.7$)

$\{X_i\}_{i=0}^{N-1} \leftarrow$ UNIFORMPARTITION(goal pool, N, L) ▷ same task split as the homogeneous baseline; X_i is client i's pool of L=100 goals

$u_c \sim \mathrm{Uniform}[0, 1]$ via $\mathrm{RandomState}(\sigma_0)$ for each $c \in \mathcal{C}$ ▷ global retention anchor (shared across clients, keyed by ASIN)

for $i = 0$ to $N - 1$ **do** ▷ client ids are 0-based, matching the implementation

  $\mathcal{G}_i \leftarrow \{\mathrm{asin}(g) : g \in X_i\}$ ▷ this client's target ASINs; $|\mathcal{G}_i| \approx 7$ (WebShop goals cluster on ASINs)

  $\mathcal{R}_i \leftarrow \mathcal{C} \setminus \mathcal{G}_i$; $n_{\mathrm{keep},i} \leftarrow \mathrm{round}(\texttt{keep_ratio} \cdot |\mathcal{R}_i|)$ ▷ per-client distractor pool $|\mathcal{R}_i| \approx 993$; $n_{\mathrm{keep},i} \approx 695$ at `keep_ratio` $= 0.7$

  $v_{c,i} \sim \mathrm{Uniform}[0, 1]$ via $\mathrm{RandomState}(\sigma_0 + 1000\,i)$ for each $c \in \mathcal{C}$ ▷ per-client perturbation

  $e_{c,i} \leftarrow (1 - \texttt{env_div})\,u_c + \texttt{env_div}\,v_{c,i}$ for $c \in \mathcal{R}_i$ ▷ retention score: `env_div` $= 0 \Rightarrow$ shared, $1 \Rightarrow$ independent

  $\mathcal{R}_i^{\mathrm{keep}} \leftarrow$ the smallest $n_{\mathrm{keep},i}$ elements of $\mathcal{R}_i$ ordered by $e_{\cdot,i}$

  $\mathcal{C}_i \leftarrow \mathcal{G}_i \cup \mathcal{R}_i^{\mathrm{keep}}$ ▷ client catalog: this client's targets retained, distractors filtered

end for

return $\{(\mathcal{C}_i,\ X_i)\}_{i=0}^{N-1}$

The dispatch in Algorithm 6 differs across Variants 2–5 only in the choice of pool $\mathcal{V}$ and in which SimServer field `key` receives cfg_i. Table 5 summarizes both for the four variants in turn, and the variant subsections below specify the entries of $\mathcal{V}$ in full. Pool size $|\mathcal{V}|$ is configurable: the main experiments

Algorithm 6 EnvVariantPartition (Variants 2–5)

Require: Variant pool $\mathcal{V} = (v_1, \ldots, v_{|\mathcal{V}|})$, an ordered list of $|\mathcal{V}|$ configuration objects; total clients N; base seed σ_0; environment-override key `key` (the SimServer field that receives cfg_i)
Ensure: Per-client environment overrides $\mathrm{cfg}_0, \ldots, \mathrm{cfg}_{N-1}$
1: **for** $i = 0$ to $N - 1$ **do** ▷ client ids are 0-based, matching the implementation
2: $j_i \leftarrow \mathrm{RandomState}(\sigma_0 + i).\mathrm{randint}(|\mathcal{V}|)$ ▷ deterministic pool index, identical across rounds
3: $\mathrm{cfg}_i \leftarrow v_{j_i+1}$ ▷ randint returns $j_i \in \{0, \ldots, |\mathcal{V}| - 1\}$, indexing the 1-based list $\mathcal{V}$; one configuration per client, frozen for all T rounds
4: **end for**
5: **return** $\{\mathrm{cfg}_i\}_{i=0}^{N-1}$ ▷ SimServer for client i reads `env_kwargs[key]` $\leftarrow \mathrm{cfg}_i$

Variant	Pool $\mathcal{V}$	$\lvert\mathcal{V}\rvert$ (used/max)	Override key `key`
2 (Field-Subset Index)	field subsets of `name`, `Title`, `description`, `features`, `BulletPoints`	4 / 8	`bm25_config.fields`
3 (BM25 Reweighting)	(k_1, b) combinations with all fields (F_{full})	4 / 8	`bm25_config.{k1,b}`
4 (Lookalike Injection)	attacked-subterm sets $\mathcal{L}_{\text{price}}$, $\mathcal{L}_{\text{color}}$, $\mathcal{L}_{\text{size}}$, $\mathcal{L}_{\text{price+color}}$	4 / 4	`extra_products`
5 (Rank Wrapper)	post-ranking wrappers W_{default}, W_{shuffle}, W_{invert}, W_{partial}	4 / 4	`search_engine_variant`

Note. `bm25_config` abbreviates the SimServer field `bm25_in_memory_config`.

Table 5: This table lists the per-variant instantiation of Algorithm 6. Each row specifies the pool $\mathcal{V}$, the supported pool sizes, and the SimServer override key `key`. The configurations are listed explicitly in Appendices M.5–M.8.

use $|\mathcal{V}| = 4$, which assigns the 100 clients (IDs 0–99) to the four configurations (v_1, v_2, v_3, v_4) with counts $(21, 19, 33, 27)$ under $\sigma_0 = 42$ (uniform in expectation, with the finite-sample imbalance expected from 100 independent $\mathrm{randint}(4)$ samples). Variants 2–3 can be extended up to $|\mathcal{V}| = 8$ without code changes. Validation always uses the unperturbed reference environment (Lucene-backed BM25 on the full 1,000-product catalog with no rank wrapper): the validation service is launched with every training-time override key (`bm25_in_memory_config`, `extra_products`, `search_engine_variant`, `catalog_filter_asins`) cleared, so the per-client perturbation does not affect the reported success-rate metric. This validation protocol keeps the variant comparisons consistent with one another and comparable to the homogeneous-environment baseline of §5.1.

M.4 Variant 1: Catalog Split (Stage 1)

Motivation. The Catalog Split variant perturbs Stage 1 (content) only: each client's local catalog is a strict subset of the global 1,000-product pool. The motivation is to isolate the weakest form of environment-level heterogeneity, in which clients see different reachable states but the rule for what to do at any commonly visited state is identical. Concretely, the optimal action sequence "search for the goal attributes, click the matching ASIN, buy with the requested options" applies unchanged across catalog subsets, so we expect this variant to leave $\pi^\star$ invariant on the shared support, and we use it to test whether the shared-optimum condition (C2) (and (C1) after state abstraction) of §4.3 alone is sufficient for environment-level robustness.

MDP formalization. Let $\mathcal{C} = \{c_1, \ldots, c_{1000}\}$ be the global ASIN set and let $\mathcal{C}_i \subset \mathcal{C}$ be client i's catalog. The catalog enters the transition kernel through the search action:

$$P_i\big(s' \mid s,\, a = \texttt{search}\, q\big) = \mathrm{render}\Big(\text{top-}K\big\{\mathrm{score}_{\mathrm{BM25}}\big(q,\, \mathrm{doc}(p)\big) \;:\; p \in \mathcal{C}_i\big\}\Big),$$
$$\mathcal{A}_i = \mathcal{A}, \quad R_i = R.$$

On the shared initial and search states the returned pages thus differ across clients through their distractor composition (non-catalog hits are deleted from the ranking, moving the remaining entries

to earlier positions), so the disagreement region $\{(s,a) : P_i(\cdot \mid s,a) \neq P_j(\cdot \mid s,a)\}$ *is* visited on-support: P_i is not simply a restriction of a global kernel. What the construction preserves is the optimal action set: every reward-bearing target for client i's own goal slice is retained in $\mathcal{C}_i$, so at every reachable (s,τ) the optimal action of the shared policy "search for the goal attributes, click the matching ASIN, buy" is unchanged even though the surrounding distractors differ. The variant therefore satisfies (C2) (action-preserving with a shared $\pi^\star$) but does not literally satisfy (C1). (C1) holds only after abstracting states to their reward-relevant content (goal-product presence and relative order), under which the residual kernel disagreement is confined to reward-irrelevant page content.

Implementation. The per-client catalog $\mathcal{C}_i$ is constructed by Algorithm 5 (Appendix M.3). The procedure has three steps. *Step 1: task-aligned target core.* The task axis reuses the homogeneous baseline's split (`partition_strategy=uniform`, $L = 100$ goals per client via UNIFORM-PARTITION), so Variant 1 differs from the §5.1 baseline in the catalog and in the retrieval depth of Appendix J.6, with `env_div` as the sole added heterogeneity axis (at `keep_ratio` $= 1$ the catalog is the full pool and Variant 1 matches that baseline except for retrieval depth). Of the 1,000-product pool, $|\mathcal{G}| = 415$ ASINs have a non-empty WebShop instruction (*i.e.*, can be the target of a synthetic goal). Client i retains the target ASINs $\mathcal{G}_i$ of *its own* L goals, so every reward-bearing product for that client's goals stays reachable. Goal counts per ASIN are highly skewed: the median ASIN carries 2 goals, but a few ASINs carry many, so a contiguous 100-goal slice induces only $|\mathcal{G}_i| \approx 7$ target ASINs, leaving a per-client distractor pool $\mathcal{R}_i = \mathcal{C} \setminus \mathcal{G}_i$ with $|\mathcal{R}_i| \approx 993$. *Step 2: retention scoring.* Each $c \in \mathcal{C}$ receives a global anchor score u_c sampled once from $\mathrm{RandomState}(\sigma_0)$ (shared across all clients) and a per-client perturbation $v_{c,i}$ sampled from $\mathrm{RandomState}(\sigma_0 + 1000\,i)$ (independent across i). Scores are keyed by ASIN, so a given product has the same u_c in every client's pool. The retention score $e_{c,i} = (1 - \texttt{env_div})u_c + \texttt{env_div}\, v_{c,i}$, evaluated on $\mathcal{R}_i$, interpolates between fully shared distractor selection (`env_div` $= 0$, all clients keep the same lowest-anchor distractors) and fully independent selection (`env_div` $= 1$, each client keeps a distinct lowest-score slice). *Step 3: catalog assembly.* For each client we take the $\mathrm{round}(\texttt{keep_ratio} \cdot |\mathcal{R}_i|) \approx 695$ distractors with the smallest $e_{c,i}$ and take their union with $\mathcal{G}_i$. We use `keep_ratio` $= 0.7$, giving $|\mathcal{C}_i| \approx 702$ products per client (near-constant, varying by a few products as $|\mathcal{G}_i|$ varies). The size is stable across `env_div`, since `env_div` changes only *which* distractors are kept and leaves their number fixed. The curve in Figure 6 uses the strongest dispersion `env_div` $= 1.0$ (`env_div` is configurable over $\{0.0, 0.3, 0.7, 1.0\}$). Because the shared target core is small, the per-client catalogs are strongly separated: at `env_div` $= 1.0$ the pairwise full-catalog Jaccard between any two clients is ≈ 0.54 (approaching the independent-selection floor $\approx \texttt{keep_ratio}/(2 - \texttt{keep_ratio}) = 0.538$ at `keep_ratio` $= 0.7$), versus ≈ 0.99 at `env_div` $= 0$ (the corresponding top-10 overlaps are 0.47 and 0.97, Appendix M.9). The catalog is enforced at the SimServer layer, *post-retrieval*. The BM25 index (the default Lucene searcher, built once offline over the full 1,000-product corpus and shared by all clients) is never rebuilt. Instead, the constructor receives $\mathcal{C}_i$ as a list of ASINs and every search filters the top-`SEARCH_RETURN_N` ($= 200$) hits by catalog membership before rendering, so a client's result list is the global ranking with non-catalog rows deleted (≈ 138 of 200 candidates remain on average at `keep_ratio` $= 0.7$). Corpus statistics (IDF, average document length) are therefore identical across clients, and the heterogeneity enters purely through which candidates remain. The rest of the WebShop pipeline (reward function, page rendering, action grammar) is untouched. Validation rollouts use the unfiltered 1,000-product catalog, so the training-time perturbation does not affect the reported success-rate metric.

Expected pattern. We expect Pattern B under PPO and high-end Pattern C under GRPO (optimization-level ϵ_{opt}, §4.4). By Theorem 5 (§4.3), (C2) plus (R_{env}) is sufficient for $\Delta_{\text{pol}} = 0$. Empirically the federated curve coincides with the single-environment baseline under PPO (panel (b) of Figure 6) and stays in a high Pattern C band under GRPO (panel (a)), even at the most aggressive catalog split (`env_div` $= 1.0$, pairwise catalog Jaccard ≈ 0.54). Catalog heterogeneity alone, even when extreme, therefore did not produce environment-level collapse in our experiments, and the remaining variants perturb later stages of the pipeline.

M.5 Variant 2: Field-Subset Index (Stage 2)

Motivation. The Field-Subset Index variant perturbs Stage 2 (encoding) by varying the doc-text field subset used to build the BM25 index across clients. Different clients see the same catalog and the same BM25 hyperparameters, but the document strings indexed by their search engines

are constructed from different field combinations (title text only, bullet-point feature text only, their concatenation, etc., with the effective channels on the released product schema detailed under Implementation). Because the bag of tokens that BM25 indexes differs across clients, a query that retrieves the goal product on one client may not retrieve it on another, so the optimal query style differs across clients: short title-keyword queries for name-indexed clients and feature phrases for `BulletPoints`-indexed clients.

MDP formalization. Let $\mathrm{doc}_F(p)$ denote the doc-text built for product p using field subset $F \subseteq \{\texttt{name}, \texttt{Title}, \texttt{description}, \texttt{features}, \texttt{BulletPoints}\}$. Client i uses field subset F_i, so its transition kernel under the search action satisfies

$$P_i\big(s' \mid s,\, a = \texttt{search}\, q\big) \;=\; \mathrm{render}\Big(\text{top-}K\big\{\mathrm{score}_{\mathrm{BM25}}\big(q,\, \mathrm{doc}_{F_i}(p)\big) \;:\; p \in \mathcal{C}\big\}\Big).$$

Identical (s, q) pairs produce different rankings across clients whenever $F_i \neq F_j$. The optimal $\pi_i^\star$ depends on F_i through the query distribution it places on $a = \texttt{search}\, q$. Because F_i is not observed, a shared policy needs (C2), that is, some action that is optimal for every i on every reachable (s, τ), and this requirement fails whenever no single query ranks the goal product highly under every F_i.

Implementation. The default Lucene searcher is replaced by an in-memory BM25 backend `InMemoryBM25Searcher` (built on `rank_bm25`) exposing a per-client `fields` list, a `k1` hyperparameter, and a `b` hyperparameter. At index-build time, for product p the searcher concatenates the configured fields with a single-space separator into one document string $\mathrm{doc}_F(p)$, tokenizes by whitespace, and feeds the resulting bag of tokens to the standard BM25 score formula. Fields absent from the product dictionary are skipped without a warning, and a document for which every requested field is empty falls back to the product `name`. In the released product schema, three of the five SimServer field names resolve to non-empty text: `name`, `Title` (attached by the product loader as a copy of `name`), and `BulletPoints` (the bullet-point feature list). The lowercase fields `description` and `features` are not attached by the loader and resolve to empty strings. The per-client configuration is dispatched through Algorithm 6 with override key `bm25_in_memory_config.fields`. At $|\mathcal{V}| = 4$, the variant pool is the ordered list $(F_{\text{full}}, F_{\text{name}}, F_{\text{desc}}, F_{\text{bullets}})$, where $F_{\text{full}} = \{\texttt{name}, \texttt{Title}, \texttt{description}, \texttt{features}, \texttt{BulletPoints}\}$, $F_{\text{name}} = \{\texttt{name}, \texttt{Title}\}$, $F_{\text{desc}} = \{\texttt{description}\}$, and $F_{\text{bullets}} = \{\texttt{BulletPoints}\}$. Because `description` and `features` resolve to empty strings, the *effective* per-client indexes are $F_{\text{full}} \to \{\texttt{name}, \texttt{Title}, \texttt{BulletPoints}\}$, $F_{\text{name}} \to \{\texttt{name}, \texttt{Title}\}$, $F_{\text{desc}} \to \{\texttt{name}\}$ (via the all-empty fallback), and $F_{\text{bullets}} \to \{\texttt{BulletPoints}\}$. The realized heterogeneity axis is therefore title text versus bullet-point feature text (plus name-token doubling where `Title` co-occurs with `name`), which is narrower than the five-field design but still produces a strong encoding shift, as measured below. For $|\mathcal{V}| > 4$, the pool is extended with $\{\texttt{features}\}$, $\{\texttt{name}, \texttt{Title}, \texttt{BulletPoints}\}$, $\{\texttt{description}, \texttt{features}\}$, and $\{\texttt{description}, \texttt{features}, \texttt{BulletPoints}\}$ up to a code-level cap of $|\mathcal{V}| = 8$. Under the same field resolution, the eight configurations reduce to four distinct effective indexes, so $|\mathcal{V}| = 8$ adds only redundant configurations (one reason the reported experiments use $|\mathcal{V}| = 4$). The per-client BM25 hyperparameters are fixed at $(k_1, b) = (1.2, 0.75)$ across configurations, so the only varying axis is the field subset. The catalog is the full 1,000-product pool with no per-client filtering. Pool selection is itself a YAML configuration setting (`variant_pool: fields_only`) that the federated dispatcher propagates to the partition function via the `BM25_VARIANT_POOL` environment variable, so the same partition function serves both Variant 2 and Variant 3 with different defaults. To quantify the effective heterogeneity, we replay the 300 unique agent search queries recovered from baseline training logs against the effective indexes: at $|\mathcal{V}| = 4$ the mean between-client top-10 Jaccard overlap (statistic (iii) of Appendix M.9) is 0.39 with a mean pairwise top-1 disagreement of 70%. This encoding shift is *stronger* than the shift along Variant 3's score-parameter axis (0.62 / 64%, Appendix M.6). The pairs span the full range: F_{bullets} against the title-only indexes F_{name} and F_{desc} is nearly disjoint (Jaccard ≈ 0.14), while F_{desc} against F_{name} is nearly identical (Jaccard ≈ 0.90, differing only through name-token doubling). Validation always launches the reference environment with `bm25_in_memory_config` cleared, falling back to the Lucene searcher over the standard doc text, so the metric is measured on the unperturbed environment.

Expected pattern. We expect Pattern C. (C1) fails because the search results differ across clients on states the optimal trajectory visits. (C2) holds only partially because the optimal query under one

field subset can be suboptimal under another, so the clients share an optimal action on some states but not on all. (C3) holds only weakly because the field subset is not directly inferable from the observation. We therefore expect a finite, observable degradation without full collapse, consistent with the large but non-collapsing gap in Figure 6.

M.6 Variant 3: BM25 Reweighting (Stage 3)

Motivation. The BM25 Reweighting variant perturbs Stage 3 (matching) by setting per-client (k_1, b) to extreme combinations while keeping the catalog and doc-text identical. The hyperparameter k_1 controls term-frequency saturation: $k_1 \to 0$ reduces term frequency to a binary indicator (a term either occurs or it does not), while $k_1 \to \infty$ makes term frequency enter the score linearly so that long documents with repeated query terms dominate. The hyperparameter b controls length normalization: $b = 0$ disables it (long documents are favored), while $b = 1$ enforces strict normalization by the average document length. Placing different clients at $(1.2, 0.75)$ (default), $(0.3, 0.75)$, $(5.0, 0.75)$, and $(1.2, 0.0)$ induces a *score-function shift* that assigns different scores to identical (q, d) pairs and hence different rankings to the same query.

MDP formalization. Let $(k_1^{(i)}, b^{(i)})$ be client i's BM25 hyperparameters and let $\mathrm{score}_{(k_1,b)}(q, d)$ denote the BM25 score with those parameters. Then

$$P_i\big(s' \,\big|\, s,\, a = \texttt{search}\, q\big) \;=\; \mathrm{render}\Big(\text{top-}K\big\{\mathrm{score}_{(k_1^{(i)},\, b^{(i)})}\big(q,\, \mathrm{doc}(p)\big) \;:\; p \in \mathcal{C}\big\}\Big).$$

For the four combinations we use (the default and three extreme ones), the mean between-client top-10 Jaccard overlap is 0.62 and the mean pairwise top-1 disagreement is 64% across 300 replayed agent queries (recovered from baseline training logs). The most extreme pair, $(1.2, 0.0)$ versus $(5.0, 0.75)$, reaches Jaccard 0.48 with top-1 disagreement 91%. (C2) holds only partially because the same agent-emitted query returns differently ordered result pages across clients, so the query that brings the goal product onto the first page, and hence the optimal action at the same pre-search state, often differs across clients.

Implementation. We reuse the in-memory BM25 backend `InMemoryBM25Searcher` introduced for Field-Subset Index but fix the field set to the full concatenation $F_{\text{full}} = \{\texttt{name}, \texttt{Title}, \texttt{description}, \texttt{features}, \texttt{BulletPoints}\}$ across all clients, so only the score-formula hyperparameters (k_1, b) vary. The per-client configuration is dispatched through Algorithm 6 with the same partition function as Variant 2 but with the default pool (YAML setting `variant_pool` unset). At $|\mathcal{V}| = 4$ the pool is the ordered list of (k_1, b) combinations, in the released assignment order: $(1.2, 0.75)$ (default), $(1.2, 0.0)$ (no length normalization, long documents favored), $(0.3, 0.75)$ (saturated TF, term occurrence acts like a binary indicator), and $(5.0, 0.75)$ (near-linear TF, long documents dominate). For $|\mathcal{V}| > 4$ we extend $\mathcal{V}$ with $(0.1, 0.75)$, $(1.2, 1.0)$, $(2.0, 0.5)$, and $(0.3, 0.0)$ up to a code-level cap of $|\mathcal{V}| = 8$. The SimServer override $\mathrm{cfg}_i = (F_{\text{full}}, k_1^{(i)}, b^{(i)})$ is written to `bm25_in_memory_config`, and the catalog is the full 1,000-product pool with no per-client filtering or lookalike injection. Because Variants 2 and 3 share the same partition function, the only operational difference is the `BM25_VARIANT_POOL` environment variable that the federated dispatcher reads when calling `partition_strategy.py` (`fields_only` routes to Variant 2's pool, and the default routes to Variant 3's). The replay statistics quoted in the MDP-formalization paragraph above are obtained by replaying the same recovered baseline-rollout queries used for Variant 2. The two axes disrupt search differently: the (k_1, b) combinations reshuffle rankings computed on *identical* documents (top-1 disagreement up to 91% on the most extreme pair), while Variant 2's field subsets change which documents match the query at all and yield the lower mean between-client top-10 Jaccard overlap of the two (0.39 vs 0.62). Validation again launches the reference environment with `bm25_in_memory_config` cleared, falling back to the Lucene searcher at default (k_1, b).

Expected pattern. We expect Pattern C. The argument is the same as for Field-Subset Index. (C1) fails because the search results differ across clients on states the optimal trajectory visits. (C2) holds only partially because the score function is per-client, and (C3) again holds only weakly because the BM25 hyperparameter is not part of the observation. The empirical curve in Figure 6 stays below the single-environment baseline but does not collapse, unlike Rank Wrapper. The GRPO curve ends at $\sim$33%. Under PPO the curve fluctuates more (last-ten-checkpoint range $\sim$14–46% vs. $\sim$29–37%

under GRPO), and the PPO last-ten-checkpoint mean ($\sim$30%) is no higher than the GRPO mean ($\sim$33%).

M.7 Variant 4: Lookalike Injection (Stages 1+3)

Motivation. The Lookalike Injection variant has the most elaborate construction of the five. It perturbs Stage 1 and Stage 3 jointly: each client's catalog is augmented with a set of *lookalike* products designed to (i) tie with the goal product under BM25 (each lookalike is an exact clone of its origin's document text, so the two score identically on every query), and (ii) be systematically wrong on one designated subterm set of the WebShop reward (price, customization color, customization size, or the joint price+color pair). The lookalike is identifiable on inspection: its ASIN carries a visible `LK_` prefix that is rendered on the results page, and for the price pools the $9999.0 attack price is itself shown next to the item. The reward subterm that a lookalike fails differs across clients. The variant therefore tests whether training produces inspect-before-buying behavior at all.

MDP formalization. Let $\mathcal{L}_i$ be client i's lookalike pool, of cardinality 268–1,000 depending on which subterm is attacked. WebShop's reward multiplies a type-match indicator r_{type} by the normalized sum of the attribute-match count n_{attr}, the option-match count n_{opt}, and a price indicator r_{price}. The augmented catalog and the reward decomposition are

$$\begin{aligned} \mathcal{C}_i &= \mathcal{C} \cup \mathcal{L}_i, \\ R(s,a) &= r_{\text{type}} \cdot \frac{n_{\text{attr}} + n_{\text{opt}} + r_{\text{price}}}{|\text{attrs}| + |\text{opts}| + 1}. \end{aligned}$$

Each lookalike $p \in \mathcal{L}_i$ keeps its document text byte-identical to its origin product's, while its non-text fields (`pricing`, `customization_options`) are perturbed so that the targeted component of the numerator is driven to 0 for that client's pool: the price term r_{price}, the color or size contribution to the option-match count n_{opt}, or jointly r_{price} and the color contribution for the price+color pool. Because the document text is unchanged, BM25 scores tie exactly and p is placed adjacent to its origin product, with rank-order between the pair decided by sort tie-breaking. In replayed queries, the goal product's clone, when one exists, appears in the top-10 results essentially always and frequently at position 1. The type indicator r_{type} and the attribute-match count n_{attr} remain unchanged because the lookalike copies every field that enters them verbatim: the type match uses the product's query, category, and name, and the attribute matches use fuzzy string matching against the document text. The injection is therefore a clean attack on the designated reward subterms. As an empirical check, end-to-end verification tests on 100 replayed queries give a reward change ΔR (the reward for buying the clone minus the reward for buying its origin) of -0.42, -0.38, -0.35, and -0.46 for the price, color, size, and price+color pools, respectively, confirming that the attack reduces the reward itself.

Implementation. The pipeline has two phases. *Phase 1: offline lookalike synthesis.* A standalone script `synthesize_lookalike.py` (included in the release under `tools/env_heterogeneity/`) produces one lookalike for each catalog product that has the attacked field, by (a) cloning the product verbatim (the document text is left untouched, so the lookalike ties with its origin under BM25 on every query) and (b) perturbing the designated non-text attribute set: `pricing` is set to the constant $9999.0 (above every goal's price cap), and `customization_options.color/.size` values are swapped to alternatives from a fixed thesaurus that fail the WebShop fuzzy option match. The swap choices are sampled using a PRNG seeded with `SEED`$=99999$, so the four output JSON files $\mathcal{L}_{\text{price}}$ (1,000 entries, one per catalog product, including products that never serve as goal targets), $\mathcal{L}_{\text{color}}$ (286, the products with a non-empty color option), $\mathcal{L}_{\text{size}}$ (268), and $\mathcal{L}_{\text{price+color}}$ (286) are reproducible across runs and are included in the repository. Each lookalike ASIN is named `LK_<DIM>_<origin_asin>`, so the prefix unambiguously identifies it as a synthetic product. Because the prefix is rendered on the results page, it also provides the inspectable cue discussed under Expected pattern. *Phase 2: per-client dispatch.* The federated dispatcher invokes Algorithm 6 with the ordered list $\mathcal{V} = (\mathcal{L}_{\text{price}}, \mathcal{L}_{\text{color}}, \mathcal{L}_{\text{size}}, \mathcal{L}_{\text{price+color}})$ at $|\mathcal{V}| = 4$. The JSON files are lazily loaded into a per-process cache the first time each pool is requested. The SimServer override key is `extra_products`: when a non-empty list is detected, the WebShop environment constructor automatically redirects search from the default Lucene backend to `InMemoryBM25Searcher` (the original Lucene index is read-only and does not contain `LK_`$\star$ ASINs, so adding products to it would require a re-index at every constructor call). The augmented catalog $\mathcal{C} \cup \mathcal{L}_i$ (of size 1,268–2,000 depending on the pool) is re-indexed once at startup using

F_{full} and $(k_1, b) = (1.2, 0.75)$. The reward function and action grammar are untouched. Only the candidate set that search ranks is enlarged. Validation always launches the reference environment with `extra_products` cleared, falling back to the unmodified 1,000-product Lucene index for a direct comparison with the homogeneous-environment baseline (up to the retrieval depth of Appendix J.6).

Expected pattern. We expect Pattern C under both optimizers. Empirically the GRPO run ends at ∼34% (last-ten-checkpoint mean ∼24%) in the interior of Pattern C, because the variant leaves an inspectable cue that keeps the run from degrading to Pattern D. (C1) fails, because the injected lookalikes change the result pages on the optimal trajectory, but (C2) holds: because the environment renders each exact clone's `LK_` ASIN prefix on the results page and in the clickable list (and, for the price pools, the $9999.0 attack price directly in the top-10 results), a single client-independent rule (never buy an `LK_`-prefixed item) is optimal across all four configurations, and clients differ only in which reward subterm drops when the rule is violated. The cue is a property of the rendered state and is independent of the environment identity, so it acts through (C2) and does not require (C3). Theorem 5 then gives $\Delta_{\text{pol}} = 0$, and the remaining degradation is optimization slack in acquiring the inspect-before-buying behavior, which the policy can learn even under GRPO. Under GRPO, client-wise gradient conflict during training leaves a finite slack but does not collapse performance. PPO ends higher (∼46%), although the runs do not isolate which PPO component is responsible, and both settings fall into Pattern C as observed in §5.3. The rendered `LK_` cue is the structural feature that separates Lookalike Injection from Rank Wrapper (Appendix M.8).

M.8 Variant 5: Rank Wrapper (Stage 4)

Motivation. The Rank Wrapper variant perturbs Stage 4 (rendering) by post-processing the BM25 top-K output before it is rendered to the agent. The catalog, doc-text, and BM25 score are all identical across clients. Only the wrapper around the search-engine output differs. The purpose of this construction is to isolate *rendering-level* transition heterogeneity from *content-level* or *score-level* heterogeneity. Under our wrappers the underlying scored candidate pool is unchanged, but the presented order (and, for W_{partial}, possibly the presented top-K membership itself) is altered. This change breaks any "trust the top position" heuristic the policy might have learned.

MDP formalization. For client i with wrapper W_i,

$$P_i\big(s' \,\big|\, s,\, a = \texttt{search}\, q\big) \;=\; \text{render}\big(W_i\big(\text{top-}K_{\text{BM25}}(q,\, \mathcal{C})\big)\big).$$

We use four wrappers, each post-processing the full retrieved candidate list ($K = \texttt{SEARCH_RETURN_N} = 200$ in the environment-level experiments, paginated 10 per page): W_{default} is the identity, W_{shuffle} randomly permutes the candidate list, W_{invert} reverses it so that BM25's lowest-scoring candidate appears first and its best candidate appears on the last page, and W_{partial} replaces the BM25 ranking with a uniformly random sample of K products with probability 0.5 (and otherwise returns the BM25 ranking unchanged). We deliberately excluded an always-random fifth wrapper W_{random} at design time to avoid clients whose goal product is effectively unreachable (near-zero reward, so the client contributes only gradient noise to the FedAvg update). As discussed under Expected pattern, the executed W_{invert} configuration in fact produces a comparable problem, because reversing all 200 candidates moves the goal product ≈ 19 pages down the list, beyond the pages reachable within the 15-step budget.

Implementation. Each non-trivial wrapper is implemented as a small Python class that holds a reference to the base `InMemoryBM25Searcher` and post-processes its output. The base searcher uses the F_{full} field list (which resolves to the effective {`name`,`Title`,`BulletPoints`} channels of Appendix M.5), default parameters $(k_1, b) = (1.2, 0.75)$, and the unmodified 1,000-product catalog. Each `search` call requests $k = \texttt{SEARCH_RETURN_N} = 200$ candidates, so the wrappers act on the full 200-candidate list. `ShuffledTopKSearcher` fetches $\max(\texttt{shuffle_k}, k)$ hits and shuffles them all before returning the first k. At $k = 200$, the $\texttt{shuffle_k} = 50$ window therefore never takes effect, and the permutation spans all 200 candidates, although the design intended a top-50 permutation. The permutation is sampled using a per-client `random.Random` seeded with $\sigma_0 + i$. `InvertedTopKSearcher` reverses the full 200-candidate list deterministically, with no random state. `PartialRandomSearcher` makes an independent random choice per query (the same per-client PRNG produces an independent Bernoulli sample at every `search` call): with probability $\texttt{random_prob} = 0.5$ it returns 200 ASINs sampled uniformly without replacement from the catalog

and assigns them score 0. Otherwise, it forwards the base searcher's ranking. The federated dispatcher invokes Algorithm 6 with $\mathcal{V} = (W_{\text{default}}, W_{\text{shuffle}}, W_{\text{invert}}, W_{\text{partial}})$ at $|\mathcal{V}| = 4$, where W_{default} is the identity (the unwrapped base searcher). Each cfg_i records the wrapper type and any wrapper-specific arguments (`shuffle_k`=50, `random_prob`=0.5) together with the per-client seed $\sigma_0 + i$, so the shuffle and partial-random streams are deterministic across rounds yet independent across clients. The SimServer override key is `search_engine_variant`: the WebShop constructor reads this field, builds the base BM25 searcher first, and wraps it before exposing it to the agent loop. Validation always launches the reference environment with `search_engine_variant` cleared, falling back to the default Lucene backend with no wrapper, so the reported success rate is measured on the standard evaluation environment.

Expected pattern. We expect Pattern D under both optimizers. (C1) fails because the wrapper changes the result pages on states the optimal trajectory visits. (C2) fails, because under the shuffle and invert wrappers the position-1 product is no longer the highest-relevance one but the policy cannot detect this reordering without a per-wrapper specialization. (C3) effectively fails: the agent could in principle infer, from observed mismatches on previous queries, that the page no longer follows BM25 order, but doing so requires a non-trivial in-context inference that the small backbone we use does not consistently realize, and our two-step history window (Appendix J.5) keeps most earlier queries out of the prompt. Two effects of the executed setup ($K = 200$) make this variant more severe than pure presentation reordering. First, because the wrappers act on all 200 retrieved candidates, in replayed queries the shuffle wrapper places the goal product in the top-10 results only $\approx 5\%$ of the time (a top-50 permutation would leave it there $\approx$ 13–20% of the time). Second, the invert wrapper places the goal product at a median list position of ≈ 194 (≈ 19 pages down the list), beyond the pages reachable within the 15-step budget, so its 33 clients (under the $|\mathcal{V}| = 4$ assignment of Appendix M.3), whenever selected, train on episodes in which success is effectively impossible and contribute the near-zero-gradient noise that the W_{random} exclusion was designed to avoid. The GRPO floor for this variant therefore reflects both rank-reordering heterogeneity and the unreachable goal products of the invert clients. Lookalike Injection exposes a rendered cue that supports one shared rule (C2), whereas Rank Wrapper admits no shared rule and its environment-identifying signal (C3) is far harder to exploit. Under GRPO, Rank Wrapper therefore behaves like the worst-case construction (Pattern D) while Lookalike Injection stabilizes in Pattern C. Under PPO, Rank Wrapper rises to $\sim$20%, still the lowest of the six PPO curves, consistent with Pattern D.

M.9 Measuring the Induced Environment-Level Heterogeneity

The subsections above specify each variant through its effect on P_i. Definition 2 summarizes the divergence between clients by $\delta = \sup_{i \neq j, (s,a)} D_{\text{TV}}(P_i, P_j)$. Because the search engines are deterministic apart from the randomized Rank Wrapper configurations, $\delta = 1$ for every variant as soon as two clients return different pages for a single query, so δ cannot distinguish the five variants. We therefore measure the induced heterogeneity directly on the search transition, independently of any policy.

Probe protocol. Every environment receives the same probe set: WebShop's 500 held-out validation goals, each sent as one `search` query whose text is the goal instruction with the price clause removed. For each environment we record the 200-candidate result list and its top 10 results, which form the first results page. The *reference engine* is the unperturbed search engine: full 1,000-product catalog, all fields, $(k_1, b) = (1.2, 0.75)$, and no wrapper. This engine runs on the same in-memory BM25 backend as the variants, so all probed environments share one implementation. Catalog Split trains on the Lucene searcher, whereas the probe applies its catalog filter to the in-memory engine. We report three statistics, each averaged over the 500 queries:

- **(i) Top-10 overlap with the reference.** The Jaccard overlap of an environment's top-10 results with the reference top 10 measures *which* products are shown.
- **(ii) Order agreement.** The Spearman correlation r_{s} measures *in what order* products are shown, comparing the environment's ordering with the reference ordering of the products that both 200-candidate lists contain.
- **(iii) Between-client overlap.** The Jaccard overlap of two clients' top-10 results measures *how far apart* two clients are. Its mean over pairs of clients whose environments differ is the *mean between-client top-10 Jaccard overlap*.

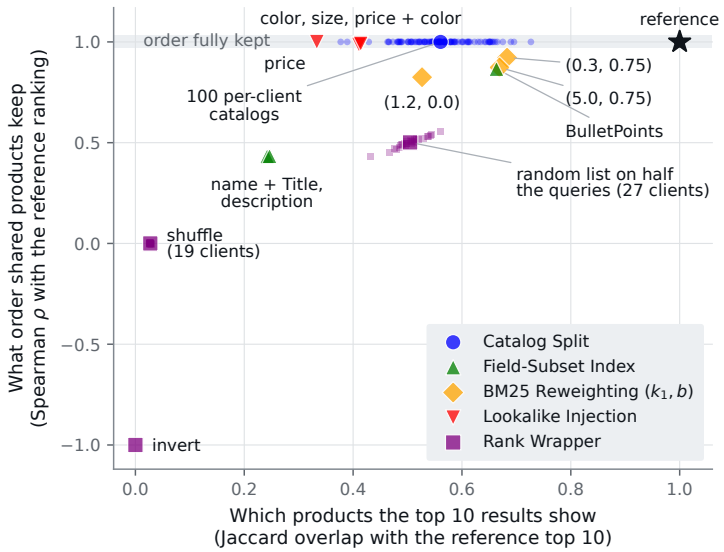


Figure 18: **Each variant changes the search transition in a stage-specific way.** The horizontal axis is the overlap of an environment's top-10 search results with those of the reference engine (Jaccard index, statistic (i)). The vertical axis is the rank correlation of the products that both result lists contain (statistic (ii)). Both are averaged over the same 500 held-out queries, and colors and markers are as in Figure 6. Large markers are means. Small markers are individual clients for Catalog Split and for the seeded Rank Wrapper configurations, in which every client has its own environment. Labels in parentheses are BM25 settings (k_1, b). The unperturbed configurations (all fields, $(k_1, b) = (1.2, 0.75)$, and the default wrapper) coincide with the reference engine ($\star$).

The probe queries are a fixed proxy for the agent's own queries, which drift during training. On the two variants for which we also replayed logged agent queries, (iii) closely matches those replays. The mean between-client top-10 Jaccard overlap is 0.39 (replay: 0.39, Appendix M.5) for Field-Subset Index and 0.57 (replay: 0.62, Appendix M.6) for BM25 Reweighting.

Each stage leaves a distinct signature. Figure 18 plots every client environment by its values of (i) and (ii), and the four stages of Appendix M.1 occupy distinct regions (Lookalike Injection, which perturbs Stages 1 and 3, falls in the content-stage region).

- **Content-stage perturbations (Catalog Split, and the injected clones of Lookalike Injection) keep the order.** All 100 Catalog Split catalogs lie on $r_s = 1$, with top-10 overlap between 0.38 and 0.73 (mean 0.56). Deleting products from the catalog only moves the remaining ones forward. The four lookalike pools lie at $r_s \geq 0.99$, because each clone ties with its origin and is inserted next to it. The matching-stage effect of Lookalike Injection, the exact tie between each clone and its origin, therefore leaves the order of the real products nearly intact. The clones still displace real products, so the top-10 overlap falls to 0.33–0.41.
- **Matching-stage perturbations keep most products but reorder them.** The three non-default (k_1, b) settings keep 53–68% of the reference top 10, at $r_s = 0.82$–0.92.
- **Encoding-stage perturbations change both.** The two title-only indexes (name+Title, and `description`, which falls back to the name) keep only 24–25% of the reference top 10, at $r_s = 0.43$. The BulletPoints index stays closer to the all-fields reference (0.66, $r_s = 0.87$), with which it shares the bullet-point text.
- **Rendering-stage perturbations destroy the order.** Invert lies at $r_s = -1$ with no top-10 overlap. Every shuffle client lies at $r_s \approx 0$ with 3% overlap. The partial-random wrapper lies near 0.5 on both axes, because it returns the reference list on about half of the queries.

The four mechanisms named in Appendix M.1 (state-coverage shift, encoding mismatch, score-function shift, and observation distortion) therefore correspond to measurably different changes to $P(s' \mid s, \texttt{search}[q])$, in addition to different code paths.

Query (same for every client)

```
search[find me long lasting, moisture wicking men's boots
       for daily wear with size: 9]
```

Tile colors

T target product · LK lookalike clone · in the reference top 10 · not in the reference top 10

Top-10 search results (number = rank under Uniform) · **Top-10 result similarity across clients** (higher = more similar, 1 = identical)

Uniform — every client uses the reference engine · mean similarity 1.00

		position 1	2	3	4	5	6	7	8	9	10	clients	c0	c1	c2
client 0	reference	T	2	3	4	5	6	7	8	9	10	100	1	1	1
client 1	reference	T	2	3	4	5	6	7	8	9	10	100	1	1	1
client 2	reference	T	2	3	4	5	6	7	8	9	10	100	1	1	1

Catalog Split — each client has its own catalog (4 of 100 shown) · mean similarity 0.49

													c1	c2	c3	c5
client 1	own catalog	T	3	6	7	8	9	11	14	15	16	1	1	.54	.49	.54
client 2	own catalog	T	2	3	5	6	7	8	11	12	15	1	.54	1	.46	.48
client 3	own catalog	T	2	5	7	8	10	11	12	14	15	1	.49	.46	1	.43
client 5	own catalog	T	2	4	6	7	9	10	11	13	14	1	.54	.48	.43	1

Field-Subset Index — each client indexes one of 4 field sets · mean similarity 0.39

													c1	c4	c0	c3
client 1	v_1 all fields	T	2	3	4	5	6	7	8	9	10	21	1	.24	.25	.66
client 4	v_2 name + Title	10	T	29	9	63	–	75	31	94	61	19	.24	1	.87	.16
client 0	v_3 description	10	T	–	29	9	61	75	94	63	55	33	.25	.87	1	.16
client 3	v_4 BulletPoints	T	2	5	4	8	6	3	11	13	7	27	.66	.16	.16	1

BM25 Reweighting — each client uses one of 4 settings of (k_1, b) · mean similarity 0.57

													c1	c4	c0	c3
client 1	v_1 (1.2, 0.75)	T	2	3	4	5	6	7	8	9	10	21	1	.53	.68	.67
client 4	v_2 (1.2, 0.0)	T	5	11	4	7	22	14	8	13	21	19	.53	1	.64	.43
client 0	v_3 (0.3, 0.75)	T	2	4	3	7	11	5	6	8	22	33	.68	.64	1	.48
client 3	v_4 (5.0, 0.75)	T	5	2	6	8	3	10	9	13	4	27	.67	.43	.48	1

Lookalike Injection — each client's clones change one of 4 attributes · mean similarity 0.36

													c1	c4	c0	c3
client 1	v_1 price	LK	T	2	LK	3	LK	5	LK	4	LK	21	1	.32	.32	.32
client 4	v_2 color	T	LK	2	LK	3	5	LK	4	LK	6	19	.32	1	.38	.42
client 0	v_3 size	LK	T	LK	2	3	LK	LK	6	4	LK	33	.32	.38	1	.38
client 3	v_4 price + color	T	LK	2	LK	3	5	LK	4	LK	6	27	.32	.42	.38	1

Rank Wrapper — each client uses one of 4 wrappers · mean similarity 0.10

													c1	c4	c10	c0	c3	c16
client 1	v_1 default	T	2	3	4	5	6	7	8	9	10	21	1	.03	.03	.00	.50	.54
client 4	v_2 shuffle	133	85	87	38	151	44	123	147	149	39	1	.03	1	.03	.03	.01	.02
client 10	v_2 shuffle	95	72	148	12	104	54	47	124	T	154	1	.03	.03	1	.02	.02	.02
client 0	v_3 invert	200	199	198	197	196	195	194	193	192	191	33	.00	.03	.02	1	.00	.00
client 3	v_4 → random	–	184	–	–	–	16	–	–	–	–	1	.50	.01	.02	.00	1	.28
client 16	v_4 → unchanged	T	2	3	4	5	6	7	8	9	10	1	.54	.02	.02	.00	.28	1

Figure 19: **How the client environments differ within each variant.** Each row is one client, and its tiles show the client's top-10 search results for the same held-out query. A tile's number is the product's rank under the reference engine (the Uniform rows), and "–" marks a rank beyond 200. The clients column counts the clients whose environment is the same as the row's, so 1 marks an environment that no other client shares. v_1–v_4 are the configurations of Figure 17. Each matrix gives the Jaccard similarity of two clients' top-10 results (statistic (iii)), averaged over all 500 held-out queries, and column ck refers to client k. The group title reports the mean between-client top-10 Jaccard overlap. On this query, Rank Wrapper's v_4 returned a random list for client 3 and the unchanged ranking for client 16. The Catalog Split rows are four clients whose catalogs contain this query's target.

How far apart the clients are. Figure 19 compares the clients with one another. Under the Uniform baseline every client returns the same top 10.

- **Catalog Split.** Each client's top 10 keeps the reference order but is filled with different distractors, so two clients share about half of their top 10 (0.47 averaged over 20 clients, 0.49 among the four shown). At `env_div` $= 0$, where all clients select distractors by the same ranking, the same statistic is 0.97. This comparison confirms that `env_div` controls how far apart the client catalogs are.
- **Field-Subset Index, BM25 Reweighting, and Lookalike Injection** split the clients into four groups that share an environment within a group and differ across groups. The groups do not always differ to the same degree:
  - *Field-Subset Index* effectively has two families. Clients on name+Title and on `description` see nearly the same top 10 (0.87), and clients on all fields and on BulletPoints share two thirds of theirs (0.66). Across the two families clients share only 16–25% of their top 10.
  - *BM25 Reweighting* clients overlap more (0.43–0.68).
  - *Lookalike Injection* clients keep the reference order of the real products but differ in which clones occupy the top 10 (0.32–0.42). Whether a clone appears above the target also depends on the configuration. In the example it does for price and size but not for color and price+color, with the clone-versus-origin order following the sort's tie-breaking.
- **Rank Wrapper** splits the clients into the most environments. Every shuffle client and every partial-random client has its own seed, so the 100 clients occupy 48 distinct environments, and two shuffle clients share only 3% of their top 10.

Discussion. These measurements bear on how the patterns of Figure 6 arise.

- **The extreme cases agree, but magnitude alone does not order the patterns.** Rank Wrapper, the only Pattern D variant (under both optimizers), also gives the lowest mean between-client top-10 Jaccard overlap of any variant (0.10). Beyond that extreme, however, the size of the kernel disagreement does not order the outcomes. BM25 Reweighting clients share the most top-10 products (0.57) yet stay in Pattern C under both optimizers. Catalog Split clients share fewer (0.47) yet reach the baseline under PPO (Pattern B).
- **The order axis separates these two cases.** Catalog Split preserves the reference order for every client, so the shared rule "search, click the best-matching product, buy" remains optimal everywhere, consistent with (C2). BM25 Reweighting changes which product appears first, and with it the optimal click order.
- **Order agreement predicts the attained plateau only partially.** Field-Subset Index has a lower order agreement than BM25 Reweighting ($r_s = 0.43$ for its title-only indexes), yet PPO raises Field-Subset Index to ∼42%, whereas the PPO last-ten-checkpoint mean of BM25 Reweighting (∼30%) is no higher than the GRPO mean (∼33%). Lookalike Injection preserves the order ($r_s \geq 0.99$), yet the federated runs degrade under both optimizers. The rendered `LK_` cue, that is, (C2), keeps Lookalike Injection in Pattern C under both optimizers, and no transition-level statistic captures that cue.

Graded statistics of P_i thus order the variants only partially, while a scalar such as δ does not order them at all. This partial ordering supports the analysis of §4.3, which predicts the pattern from which of (C1)–(C3) a variant preserves and does not use the distance between the client kernels.

Caveats. Four limitations apply to these measurements.

- The probe queries are goal instructions, which differ from the queries the agent submits during training.
- The reference engine uses the in-memory BM25 backend, whereas validation uses the Lucene backend (Appendix M.3).
- The tiles in Figure 19 show a single query and are illustrative. The matrices and Figure 18 average over all 500 queries.
- Catalog Split statistics use `keep_ratio` $= 0.7$ as in the reported runs. The between-client values are averaged over the first 20 clients.

N Asymmetric Robustness Mechanism: Theory and Proofs

This appendix provides the full theoretical foundation underlying §4: the LLM-tailored realizability assumptions on the policy class (Appendix N.1), the proofs of Theorems 1–5 (Appendices N.2–N.4), and the implications for the gradient-heterogeneity quantity ζ^2 at the federated solution (Appendix N.5). Throughout, rewards and initial states are task-conditional as in §2, rewards are defined on transitions, $r_u = R(s_u, a_u, s_{u+1}; \tau)$, and we suppress the step index of finite-horizon value functions.

N.1 Realizability Assumptions for LLM Agents

We work under five realizability assumptions on the LLM-induced policy class $\Pi_\Theta = \{\pi_\theta : \theta \in \Theta\}$. All five are motivated by LLM agents (they build on the engineering effort already invested in pre-training and instruction tuning), and none constrains the federated optimizer beyond an abstract sub-optimality ϵ_{opt}.

- **(R)** *Task-conditional pointwise optimality*: under Definition 1, there exists $\theta^\star \in \Theta$ such that $\pi_{\theta^\star}$ is optimal pointwise on every $\tau \in \text{supp}(\bar{\mathcal{D}}_\tau)$.
- **(R′)** *ϵ-realizability in expectation*: there exists $\theta^\star \in \Theta$ whose expected sub-optimality $\mathbb{E}_{\bar{\mathcal{D}}_\tau}[\sup_\pi \mathcal{J}(\pi; \tau, \mathcal{M}_{\text{env}}) - \mathcal{J}(\pi_{\theta^\star}; \tau, \mathcal{M}_{\text{env}})]$ is bounded by a capacity slack $\epsilon_{\text{approx}} \geq 0$, relaxing the pointwise condition of (R) to a population-level one.
- **($\mathbf{R}_{\text{env}}$)** *Environment-shared optimum*: under Definition 2, if a single policy is simultaneously optimal across all $\mathcal{M}_i$ (cf. (C1) or (C2) in §4.3), it lies in Π_Θ.
- **($\mathbf{R}_{\text{aug}}$)** *History-augmented realizability*: the augmented class $\Pi_\Theta^{\text{aug}} = \{\pi_\theta(a \mid s, \tau, h_u)\}$, in which the LLM receives the trajectory prefix h_u through its context window, contains the environment-conditional policies used in (C3). Each such policy follows a default prefix policy until a reveal step, infers the environment j from the prefix, and then acts optimally with respect to a kernel model $\hat{P}_j$ of that environment (cf. (C3) in §4.3). Exact environment-conditional optima correspond to the case $\hat{P}_j = P_j$.
- **(LR)** *Local realizability for lower bounds*: in the worst-case constructions, Π_Θ contains, for every client i, a policy that is ϵ_{LR}-optimal in $\mathcal{M}_i$. "Local" refers to the client and is unrelated to local optima of the objective. The lower bounds below hold up to an additive ϵ_{LR}, which we suppress.

For LLMs, (R)/(R′) align with instruction tuning, (R_{aug}) uses in-context posterior inference, and (LR) is satisfied by the simple decision rules used in our worst-case bandit construction (Appendix N.3).

Remark 2 (What these assumptions do not constrain)**.** *None of (R)/(R′)/(R_{env})/(R_{aug})/(LR) is an optimization-side assumption: there is no convexity, smoothness, gradient-Lipschitz, or PL condition on $\mathcal{J}_{fed}$, no bound on the stochastic-gradient variance, and no assumption of a specific federated rate. (Appendix N.5 invokes smoothness only to translate value-level slacks into a bound on ζ^2, and none of the theorems uses it.) The five assumptions are purely representational statements about whether a target policy lies inside Π_Θ (or Π_Θ^{aug}). The federated optimizer is treated as a black box that returns some $\hat{\theta}$ near the federated optimum with abstract slack ϵ_{opt}. This separation is the reason the Asymmetric Robustness Mechanism is structural and independent of the optimization rate: the task-level slack vanishes as model capacity grows (R′) and as the optimizer slack shrinks (Theorem 2), while the environment-level floor of Theorem 4 holds for any optimizer choice.*

Remark 3 ((R) and (R′) as a soft form of instruction-tuning realizability)**.** *(R) asserts that Π_Θ contains a policy that is optimal pointwise on every task in the federated mixture support. For instruction-tuned LLMs, such a policy is the standard target of supervised fine-tuning on multi-task instruction data, and this fine-tuning already trains the model to produce approximately task-conditional optimal behavior given the prompt as input. (R′) relaxes the pointwise condition to a population-level ϵ_{approx}-slack so that imperfect realizability (the most common practical regime) is admissible: as the LLM capacity grows or supervised-fine-tuning coverage broadens, ϵ_{approx} shrinks toward zero. The key structural ingredient that makes (R)/(R′) more realistic than a generic "the optimum lies in the class" statement is I-D Asymmetry: the task descriptor τ enters $\pi_\theta(a \mid s, \tau)$ as input, so a single parameter $\theta^\star$ can encode task-conditional optimal behavior without needing a per-task copy of the policy.*

Remark 4 ((R_{aug}) and in-context inference)**.** *(R_{aug}) augments the policy input with the trajectory prefix $h_u = (s_0, a_0, r_0, \ldots, s_u)$, so that the policy can condition on the current state and on all earlier observations of the current episode. Operationally, this augmentation is what LLM-induced policies do by design: the trajectory prefix is appended to the prompt, and the LLM performs in-context posterior inference over the latent environment identity (e.g., reading a homepage banner to infer which website it is on, then conditioning subsequent tool calls accordingly). No new parameter is added relative to Π_Θ. (R_{aug}) is the assumption that the LLM's existing context-window capability suffices to represent such environment-conditional behavior once the environment identity becomes inferable from history (the (C3) regime), possibly through an imperfect internal model $\hat{P}_j$ of the identified environment.*

Remark 5 (Why (LR) is mild)**.** *(LR) requires Π_Θ to contain an ϵ_{LR}-optimal policy for each client in the worst-case constructions of Theorems 3 and 4, whose exact per-environment optima are simple decision rules such as the always-a_1 and always-a_2 policies in the transition-swap bandit. Modern instruction-tuned LLMs trivially realize such constant-action policies (the constant policy $\pi(a_1 \mid s) = 1$ corresponds to a deterministic prompt-response template), so (LR) imposes essentially no restriction on Π_Θ in the regimes we consider. The sole role of (LR) is to ensure that the lower bounds do not fail merely because the policy class cannot represent its own per-environment optima.*

N.2 Proofs of Theorems 1 and 2 (Task-Level Robustness)

This subsection proves the task-level robustness theorems stated in §4.1 of the main text, namely Theorem 1 (idealized) and Theorem 2 (approximate, black-box optimizer). Both proofs operate under the realizability assumptions detailed in Appendix N.1: Theorem 1 invokes (R), while Theorem 2 relaxes (R) to (R′) and additionally accounts for a federated-optimization slack ϵ_{opt}. Throughout, $\mathcal{J}_\tau(\theta) := \mathcal{J}(\pi_\theta; \tau, \mathcal{M}_{\text{env}})$ denotes the expected return of π_θ on task τ in the shared environment $\mathcal{M}_{\text{env}}$, and $\bar{\mathcal{D}}_\tau := \frac{1}{N}\sum_{i=1}^{N} \mathcal{D}_{\tau_i}$ is the federated task mixture.

Theorem (Task-Level Robustness, idealized (Theorem 1))**.** *Under Definition 1 and Assumption (R), the following hold.*

(i) The federated objective reduces to the mixture expectation:

$$\mathcal{J}_{fed}(\theta) \;=\; \mathbb{E}_{\tau\sim\bar{\mathcal{D}}_\tau}\big[\mathcal{J}(\pi_\theta; \tau, \mathcal{M}_{env})\big].$$

(ii) Any $\theta^\star$ that realizes (R) maximizes $\mathcal{J}_{fed}$ and is simultaneously optimal for every client: $\mathcal{J}_i(\theta^\star) = \sup_{\theta\in\Theta} \mathcal{J}_i(\theta)$ for all $i \in [N]$. In particular, no client–federation tradeoff exists at the global optimum.

Proof. Part (i). Under Definition 1, all clients share the same environment $\mathcal{M}_{\text{env}} = (\mathcal{S}, \mathcal{A}, P, R, \gamma, \mu_0)$, so the per-task expected return is client-agnostic, $\mathcal{J}(\pi_\theta; \tau, \mathcal{M}_i) = \mathcal{J}_\tau(\theta)$. The per-client objective $\mathcal{J}_i$ depends on i only through the task distribution $\mathcal{D}_{\tau_i}$ it integrates against. By the per-client objective definition $\mathcal{J}_i(\theta) = \mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\mathcal{J}_\tau(\theta)]$ and linearity of expectation,

$$\mathcal{J}_{\text{fed}}(\theta) \;=\; \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\mathcal{J}_\tau(\theta)] \;=\; \mathbb{E}_{\tau\sim\bar{\mathcal{D}}_\tau}[\mathcal{J}_\tau(\theta)],$$

where the second equality uses the definition of the mixture $\bar{\mathcal{D}}_\tau = \frac{1}{N}\sum_i \mathcal{D}_{\tau_i}$.

Part (ii). Assumption (R) provides $\theta^\star \in \Theta$ such that $\mathcal{J}_\tau(\theta^\star) = \sup_\pi \mathcal{J}(\pi; \tau, \mathcal{M}_{\text{env}}) =: \mathcal{J}_\tau^\star$ for every $\tau \in \operatorname{supp}(\bar{\mathcal{D}}_\tau)$. Each $\mathcal{D}_{\tau_i}$ is absolutely continuous with respect to $\bar{\mathcal{D}}_\tau$, because $\bar{\mathcal{D}}_\tau \geq N^{-1}\mathcal{D}_{\tau_i}$ as measures. This inequality also gives $w_i := d\mathcal{D}_{\tau_i}/d\bar{\mathcal{D}}_\tau \leq N$ and hence the bound $\chi^2(\mathcal{D}_{\tau_i} \,\|\, \bar{\mathcal{D}}_\tau) = \int w_i^2 \, d\bar{\mathcal{D}}_\tau - 1 \leq N - 1$ stated in §4.1. By (R), the pointwise inequality $\mathcal{J}_\tau(\theta^\star) \geq \mathcal{J}_\tau(\theta)$ holds for all $\theta \in \Theta$ and all $\tau \in \operatorname{supp}(\bar{\mathcal{D}}_\tau)$. Absolute continuity makes this inequality hold almost surely under any distribution $\mathcal{D} \ll \bar{\mathcal{D}}_\tau$, and taking expectation under such a $\mathcal{D}$ gives

$$\mathbb{E}_{\tau\sim\mathcal{D}}[\mathcal{J}_\tau(\theta^\star)] \;\geq\; \mathbb{E}_{\tau\sim\mathcal{D}}[\mathcal{J}_\tau(\theta)] \qquad \forall\, \theta \in \Theta.$$

Setting $\mathcal{D} = \bar{\mathcal{D}}_\tau$ yields federation-optimality of $\theta^\star$. Setting $\mathcal{D} = \mathcal{D}_{\tau_i}$ yields per-client optimality. □

Remark 6. *The argument uses I-D Asymmetry only at the step that invokes (R): a single $\theta^\star$ realizes the task-conditional pointwise-optimal policy precisely because τ enters $\pi_\theta(a \mid s, \tau)$ as an input,*

allowing one parameter to encode different task-conditional behaviors. Without that input, a single θ could be pointwise optimal only if all tasks in the support shared the same optimal behavior, and this condition fails as soon as two tasks require different actions at a common reachable state.

Theorem (Task-Level Robustness, approximate, black-box optimizer (Theorem 2))**.** *Under Definition 1 and Assumption (R′), let $\hat{\theta}_{fed} \in \Theta$ be any parameter satisfying*

$$\mathcal{J}_{fed}(\hat{\theta}_{fed}) \;\geq\; \sup_{\theta\in\Theta} \mathcal{J}_{fed}(\theta) - \epsilon_{opt}$$

for some $\epsilon_{opt} \geq 0$. Then for every client $i \in [N]$,

$$\sup_{\theta\in\Theta} \mathcal{J}_i(\theta) - \mathcal{J}_i(\hat{\theta}_{fed}) \;\leq\; \sqrt{\big(1 + \chi^2(\mathcal{D}_{\tau_i} \,\|\, \bar{\mathcal{D}}_\tau)\big) \cdot R_{\max} H \cdot \big(\epsilon_{approx} + \epsilon_{opt}\big)}.$$

Proof. The proof bounds the per-task gap $\hat{f}(\tau) := \mathcal{J}^\star_\tau - \mathcal{J}_\tau(\hat{\theta}_{\text{fed}})$ on the federated mixture, translates it to client i via importance-weighted Cauchy–Schwarz, and then connects it to the per-client objective gap. Here $\mathcal{J}^\star_\tau = \sup_\pi \mathcal{J}(\pi; \tau, \mathcal{M}_{\text{env}})$ as in the proof of Theorem 1. Note that $\hat{f}$ is well-defined and client-agnostic because $\mathcal{M}_{\text{env}}$ is shared (Definition 1), and $\hat{f}(\tau) \in [0, R_{\max} H]$ since per-task return is bounded by $R_{\max} H$.

Step 1 (mixture-level bound on $\mathbb{E}[\hat{f}]$). By Theorem 1 (i), $\mathcal{J}_{\text{fed}}(\theta) = \mathbb{E}_{\bar{\mathcal{D}}_\tau}[\mathcal{J}_\tau(\theta)]$, so

$$\mathbb{E}_{\bar{\mathcal{D}}_\tau}[\hat{f}(\tau)] \;=\; \mathbb{E}_{\bar{\mathcal{D}}_\tau}[\mathcal{J}^\star_\tau] - \mathcal{J}_{\text{fed}}(\hat{\theta}_{\text{fed}}).$$

Decomposing the gap using any $\theta^\star$ realizing (R′) gives

$$\mathbb{E}_{\bar{\mathcal{D}}_\tau}[\hat{f}(\tau)] \;=\; \underbrace{\mathbb{E}_{\bar{\mathcal{D}}_\tau}[\mathcal{J}^\star_\tau] - \mathcal{J}_{\text{fed}}(\theta^\star)}_{\leq\, \epsilon_{\text{approx}} \text{ by (R}')} \;+\; \underbrace{\mathcal{J}_{\text{fed}}(\theta^\star) - \mathcal{J}_{\text{fed}}(\hat{\theta}_{\text{fed}})}_{\leq\, \epsilon_{\text{opt}} \text{ since } \theta^\star \in \Theta} \;\leq\; \epsilon_{\text{approx}} + \epsilon_{\text{opt}}.$$

The first inequality is exactly the (R′) statement (capacity slack on the mixture). The second uses $\theta^\star \in \Theta$, hence $\mathcal{J}_{\text{fed}}(\theta^\star) \leq \sup_{\theta\in\Theta} \mathcal{J}_{\text{fed}}(\theta)$, combined with the assumed sub-optimality of $\hat{\theta}_{\text{fed}}$.

Step 2 (importance-weighted Cauchy–Schwarz). Let $w_i(\tau) := \mathrm{d}\mathcal{D}_{\tau_i}/\mathrm{d}\bar{\mathcal{D}}_\tau$ denote the Radon–Nikodym derivative, well-defined because $\mathcal{D}_{\tau_i} \ll \bar{\mathcal{D}}_\tau$. Then

$$\mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\hat{f}(\tau)] \;=\; \mathbb{E}_{\tau\sim\bar{\mathcal{D}}_\tau}\big[w_i(\tau)\, \hat{f}(\tau)\big].$$

Applying Cauchy–Schwarz in $L^2(\bar{\mathcal{D}}_\tau)$,

$$\mathbb{E}_{\bar{\mathcal{D}}_\tau}[w_i \cdot \hat{f}] \;\leq\; \|w_i\|_{L^2(\bar{\mathcal{D}}_\tau)} \cdot \|\hat{f}\|_{L^2(\bar{\mathcal{D}}_\tau)}.$$

The squared first factor equals one plus a χ^2 divergence:

$$\|w_i\|^2_{L^2(\bar{\mathcal{D}}_\tau)} \;=\; \int w_i(\tau)^2 \,\mathrm{d}\bar{\mathcal{D}}_\tau \;=\; \int \frac{\mathrm{d}\mathcal{D}^2_{\tau_i}}{\mathrm{d}\bar{\mathcal{D}}_\tau} \;=\; 1 + \chi^2(\mathcal{D}_{\tau_i} \,\|\, \bar{\mathcal{D}}_\tau).$$

For the second factor, the boundedness $\hat{f}(\tau) \in [0, R_{\max} H]$ gives the pointwise inequality $\hat{f}(\tau)^2 \leq R_{\max} H \cdot \hat{f}(\tau)$. Hence

$$\|\hat{f}\|^2_{L^2(\bar{\mathcal{D}}_\tau)} \;=\; \mathbb{E}_{\bar{\mathcal{D}}_\tau}[\hat{f}^2] \;\leq\; R_{\max} H \cdot \mathbb{E}_{\bar{\mathcal{D}}_\tau}[\hat{f}] \;\leq\; R_{\max} H \cdot (\epsilon_{\text{approx}} + \epsilon_{\text{opt}}),$$

where the last step invokes Step 1. Combining the two factor bounds gives

$$\mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\hat{f}(\tau)] \;\leq\; \sqrt{\big(1 + \chi^2(\mathcal{D}_{\tau_i} \,\|\, \bar{\mathcal{D}}_\tau)\big) \cdot R_{\max} H \cdot (\epsilon_{\text{approx}} + \epsilon_{\text{opt}})}.$$

Step 3 (reduction to per-client gap). We claim

$$\sup_{\theta\in\Theta} \mathcal{J}_i(\theta) - \mathcal{J}_i(\hat{\theta}_{\text{fed}}) \;\leq\; \mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\hat{f}(\tau)].$$

Indeed, $\sup_{\theta\in\Theta} \mathcal{J}_i(\theta) = \sup_{\theta\in\Theta} \mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\mathcal{J}_\tau(\theta)] \leq \mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\mathcal{J}^\star_\tau]$ (the unrestricted pointwise optimum upper-bounds the parameterized average). Subtracting $\mathcal{J}_i(\hat{\theta}_{\text{fed}}) = \mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\mathcal{J}_\tau(\hat{\theta}_{\text{fed}})]$ yields exactly $\mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\hat{f}(\tau)]$. Combining this inequality with Step 2 completes the proof. □

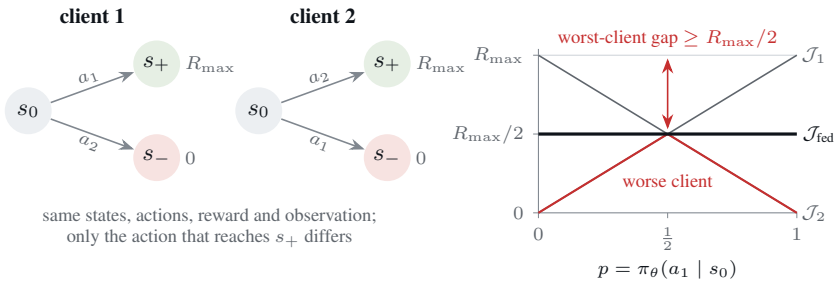


Figure 20: **The transition-swap bandit used in Theorem 3.** Left: the two clients share states, actions, rewards, and observations, and differ only in which action reaches the rewarding state s_+. Right: the curves show client values under a shared policy that picks a_1 with probability p. The federated objective $\mathcal{J}_{\text{fed}}$ is flat at $R_{\max}/2$, so every p is federation-optimal, while the worse-off client (red) stays at least $R_{\max}/2$ below its own optimum. Smoothing the transitions to $(1 \pm \delta)/2$, where δ is the total-variation distance between the two clients' transitions, scales both slopes by δ and leaves a worst-client gap of at least $R_{\max}\delta/2$. Repeating this step H times with observations that never reveal the client gives the $\Omega(R_{\max}H\delta)$ bound of Theorem 4.

Remark 7. *The proof never uses any optimization-side hypothesis on the federated optimizer beyond the definitional sub-optimality bound on ϵ_{opt}. As a consequence, the inequality is optimizer-agnostic: it applies to any federated procedure for which ϵ_{opt} can be bounded (either empirically, by tracking validation-loss plateaus, or theoretically, by applying a convergence rate from the federated optimization literature). A complementary linear bound also holds: since $w_i \leq N$ (cf. Part (ii) of the proof of Theorem 1), one has $\sup_\theta \mathcal{J}_i(\theta) - \mathcal{J}_i(\hat{\theta}_{fed}) \leq \mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\hat{f}(\tau)] \leq \|w_i\|_\infty(\epsilon_{approx} + \epsilon_{opt}) \leq N\,(\epsilon_{approx} + \epsilon_{opt})$, so the per-client gap is at most the minimum of the square-root and linear forms. The linear form is the sharper one exactly when $\epsilon_{approx} + \epsilon_{opt} < \big(1 + \chi^2(\mathcal{D}_{\tau_i}\|\bar{\mathcal{D}}_\tau)\big) R_{\max}H/\|w_i\|_\infty^2$, which includes the vanishing-slack limit. When $\mathcal{D}_{\tau_i} \approx \bar{\mathcal{D}}_\tau$ ($w_i \approx 1$), the linear form gives a per-client gap of order $\epsilon_{approx} + \epsilon_{opt}$. Under Definition 1, w_i depends on the split only through which clients hold each task and how large their pools are, and it does not depend on how different the clients' category mixes are: with disjoint pools, $w_i \equiv N$ on X_i and $\chi^2(\mathcal{D}_{\tau_i}\|\bar{\mathcal{D}}_\tau) = N - 1$ whether the category split is near-uniform or extreme, so $w_i \approx 1$ requires overlapping pools. The square-root form is informative at moderate slack and makes the dependence on the split explicit through χ^2. Its three inequalities (Step 1, Cauchy–Schwarz, and $\hat{f}^2 \leq R_{\max}H\hat{f}$) are tight together only when $\mathbb{E}_{\bar{\mathcal{D}}_\tau}[\hat{f}] = \epsilon_{approx} + \epsilon_{opt}$ and $\hat{f}$ is proportional to w_i and takes values in $\{0, R_{\max}H\}$, which forces $(1 + \chi^2)(\epsilon_{approx} + \epsilon_{opt}) = R_{\max}H$, in which case the bound equals the trivial value $R_{\max}H$.*

N.3 Proofs of Theorems 3 and 4 (Environment-Level Non-Robustness)

This subsection proves the environment-level non-robustness results stated in §4.2: Theorem 3 establishes the existence of environment constructions with a strictly positive worst-client gap, and Theorem 4 quantifies that gap as $\Omega(R_{\max}H\delta)$ for multi-step constructions. Both rely on Assumption (LR) of Appendix N.1. This assumption guarantees that Π_Θ contains an ϵ_{LR}-optimal policy for every client of our worst-case constructions, whose exact optima are simple deterministic decision rules. The results delimit the scope of Theorem 2: in contrast to the task-level slack, which vanishes as capacity and optimization improve, the environment-level gap persists as a structural property of the federated solution.

Theorem (Environment-Level Non-Robustness, idealized (Theorem 3))**.** *Under Definition 2 and Assumption (LR), there exist environment constructions $\{\mathcal{M}_i\}_{i=1}^N$ (with a single shared task in the construction) such that any federated optimum $\theta^\star_{fed} \in \arg\max_{\theta\in\Theta} \mathcal{J}_{fed}(\theta)$ satisfies*

$$\sup_{i\in[N]} \Big[\sup_{\theta_i\in\Theta} \mathcal{J}_i(\theta_i) - \mathcal{J}_i(\theta^\star_{fed}) \Big] \;>\; 0.$$

Proof. We construct the transition-swap bandit (Figure 20). Let $N = 2$, $\mathcal{S} = \{s_0, s_+, s_-\}$, $\mathcal{A} = \{a_1, a_2\}$, $\mu_0 = \mathbf{1}_{s_0}$ (point mass at s_0), with one decision step from s_0 followed by absorption. The

shared reward is given on arrival, $R(s,a,s') = R_{\max}\mathbf{1}\{s' = s_+\}$, so reaching s_+ earns $R_{\max}$ and reaching s_- earns 0. The task is trivial (single-task) so that all heterogeneity is confined to the transition kernels. The two clients differ only in dynamics:

$$\begin{aligned}\mathcal{M}_1: &\quad P_1(s_+ \mid s_0, a_1) = 1, \quad P_1(s_- \mid s_0, a_2) = 1,\\ \mathcal{M}_2: &\quad P_2(s_- \mid s_0, a_1) = 1, \quad P_2(s_+ \mid s_0, a_2) = 1.\end{aligned}$$

The construction satisfies Definition 2: $P_1 \neq P_2$, but the state space, action space, reward function, task distribution, discount, and initial-state distribution are all shared. Per-client optima are deterministic: $\pi_1^\star$ always selects a_1 (value $R_{\max}$), and $\pi_2^\star$ always selects a_2 (value $R_{\max}$). By (LR), Π_Θ contains an ϵ_{LR}-optimal policy for each client, so $\sup_{\theta_i \in \Theta} \mathcal{J}_i(\theta_i) = R_{\max}$ up to ϵ_{LR}.

Now let any candidate federated θ induce a stochastic mixture $\pi_\theta(a_1 \mid s_0) = p$ for some $p \in [0,1]$. By linearity,

$$\begin{aligned}\mathcal{J}_1(\theta) &= p\,R_{\max} + (1-p)\cdot 0 \;=\; p\,R_{\max},\\ \mathcal{J}_2(\theta) &= p\cdot 0 + (1-p)\,R_{\max} \;=\; (1-p)\,R_{\max},\\ \mathcal{J}_{\text{fed}}(\theta) &= \tfrac{1}{2}\big(\mathcal{J}_1(\theta) + \mathcal{J}_2(\theta)\big) \;=\; R_{\max}/2 \quad \text{(constant in } p\text{)}.\end{aligned}$$

Every $p \in [0,1]$ is therefore federation-optimal, but the worst-client gap is

$$\sup_{i\in\{1,2\}} \big[R_{\max} - \mathcal{J}_i(\theta)\big] \;=\; R_{\max}\cdot\max(p, 1-p) \;\geq\; R_{\max}/2.$$

Hence the worst-client gap is at least $R_{\max}/2 > 0$, completing the existence proof. □

Remark 8. *The construction is small: $|\mathcal{S}| = 3$, $|\mathcal{A}| = 2$, $N = 2$, horizon 1, and none of N, $|\mathcal{A}|$, or the horizon can be reduced without making the problem trivial. The fact that $\mathcal{J}_{fed}$ is identically $R_{\max}/2$ in p shows that no algorithm operating only on the federated objective (FedAvg, weighted FedAvg, Reptile-style aggregators) can resolve the indeterminacy. The failure is a property of the objective and is independent of the optimizer. Disambiguation requires either personalization (distinct θ_i per client) or making environment identity available to the policy (Corollary 1, or (C3) of Theorem 5).*

To extend the qualitative existence result to a quantitative lower bound, we build an explicit multi-step construction and compute its policy-disagreement gap directly. For reference we first record a standard cross-MDP simulation bound that makes the horizon scaling of value differences explicit. The lower bound below is obtained by direct computation on the explicit construction and does not rely on this simulation bound, which is used in the tightness remark after the proof and in the (C3) recovery proof (Appendix N.4).

Lemma 2 (Cross-MDP simulation)**.** *For two MDPs $\mathcal{M}_1, \mathcal{M}_2$ that share state space, action space, reward function, discount factor, and initial-state distribution but differ in transition kernels with $\sup_{s,a} D_{\text{TV}}(P_1(\cdot \mid s,a), P_2(\cdot \mid s,a)) \leq \delta$, the value of any policy π satisfies*

$$\big|\mathcal{J}(\pi;\mathcal{M}_1) - \mathcal{J}(\pi;\mathcal{M}_2)\big| \;\leq\; R_{\max}H^2\,\delta,$$

where H is the horizon when $\gamma = 1$ and $H = 1/(1-\gamma)$ when $\gamma < 1$, in which case the bound reads $R_{\max}\delta/(1-\gamma)^2$. The same bound holds for the values of π from any fixed start state s over any remaining horizon of at most H steps. The H^2 factor decomposes into one factor of H from the value range $V_{\max} \leq R_{\max}H$ and a second factor of H from telescoping per-step transition errors along the trajectory. The bound is tight up to constants when $\delta \lesssim 1/H$. With d^π the occupancy of π in $\mathcal{M}_1$, an alternative form $R_{\max}H^2 \cdot \mathbb{E}_{(s,a)\sim d^\pi}[D_{\text{TV}}(P_1, P_2)]$ holds, replacing the worst-case divergence δ with the occupancy-weighted divergence, which is no larger.

Bounds of this type are standard and originate in the Simulation Lemma of Kearns and Singh [53]. We restate it here only to make the horizon scaling explicit.

Theorem (Environment-Level Non-Robustness, quantitative (Theorem 4))**.** *Under Definition 2 with worst-case TV divergence $\delta := \sup_{i\neq j,(s,a)} D_{\text{TV}}(P_i, P_j)$ and Assumption (LR), there exist multi-step environment constructions (with a single shared task in the construction) whose observations reveal neither past rewards nor outcomes (so that (C3) is unavailable) and a corresponding policy class satisfying (LR), such that the policy-disagreement gap*

$$\Delta_{pol} \;:=\; \frac{1}{N}\sum_{i=1}^{N} \sup_{\theta_i\in\Theta} \mathcal{J}_i(\theta_i) - \sup_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} \mathcal{J}_i(\theta) \;\geq\; \Omega(R_{\max}H\delta).$$

The worst-client gap satisfies $\sup_i[\sup_{\theta_i\in\Theta}\mathcal{J}_i(\theta_i) - \mathcal{J}_i(\theta^\star_{fed})] \geq \Delta_{pol}$ *for any* $\theta^\star_{fed} \in \arg\max_{\theta\in\Theta}\mathcal{J}_{fed}(\theta)$.

Proof. The argument proceeds in three parts: a smoothed bandit yielding $\Omega(R_{\max}\delta)$, its amplification over H observation-silent steps yielding $\Omega(R_{\max}H\delta)$, and a $\sup \geq \text{avg}$ argument that converts Δ_{pol} into a worst-client gap.

Bandit lower bound. Consider the same single-step bandit as in Theorem 3 but with smoothed transitions parameterized by $\delta \in [0,1]$:

$$P_1(s_+ \mid s_0, a_1) = \tfrac{1+\delta}{2}, \quad P_1(s_+ \mid s_0, a_2) = \tfrac{1-\delta}{2},$$
$$P_2(s_+ \mid s_0, a_1) = \tfrac{1-\delta}{2}, \quad P_2(s_+ \mid s_0, a_2) = \tfrac{1+\delta}{2},$$

and the same arrival reward $R(s,a,s') = R_{\max}\mathbf{1}\{s' = s_+\}$. Then $\sup_{s,a} D_{\text{TV}}(P_1,P_2) = \delta$ and, by (LR), the per-client optimum is $\sup_{\theta_i\in\Theta}\mathcal{J}_i(\theta_i) = R_{\max}\frac{1+\delta}{2}$ up to ϵ_{LR}, which we suppress hereafter. For any federated $\pi_\theta(a_1 \mid s_0) = p$,

$$\mathcal{J}_1(\theta) = R_{\max}\left[\tfrac{1}{2} + \tfrac{(2p-1)\delta}{2}\right],$$
$$\mathcal{J}_2(\theta) = R_{\max}\left[\tfrac{1}{2} - \tfrac{(2p-1)\delta}{2}\right],$$
$$\mathcal{J}_{\text{fed}}(\theta) = R_{\max}/2 \quad \text{(constant in } p\text{)}.$$

Hence $\sup_{\theta\in\Theta}\mathcal{J}_{\text{fed}}(\theta) = R_{\max}/2$ and

$$\Delta_{\text{pol}} \;=\; R_{\max}\tfrac{1+\delta}{2} - R_{\max}/2 \;=\; R_{\max}\delta/2.$$

Horizon amplification via an observation-silent multi-step chain. We now repeat the bandit for H steps within one episode, one bandit play per agent step, while preserving $\sup_{s,a} D_{\text{TV}}(P_1,P_2) = \delta$. From every state $s \in \{s_0, s_+, s_-\}$, the client's good action leads to s_+ with probability $\frac{1+\delta}{2}$ and its bad action leads there with probability $\frac{1-\delta}{2}$ (otherwise the next state is s_-). The arrival reward is the same, and the roles of a_1 and a_2 are swapped between $\mathcal{M}_1$ and $\mathcal{M}_2$. Only the transition kernel differs across clients (the reward is shared, as required by Definition 2), and $D_{\text{TV}}(P_1(\cdot \mid s,a), P_2(\cdot \mid s,a)) = \delta$ at every (s,a). Critically, every state emits the same observation o, and we restrict the policy class to the *observation-silent* class of memoryless policies $\pi_\theta(a \mid o)$ of this single shared observation, so the policy uses neither past rewards nor outcomes. This restriction realizes the hypothesis of Theorem 4 that observations reveal neither past rewards nor outcomes and places the construction outside the (C3) regime, and the remark after the proof quantifies how much the restriction matters.

In this observation-silent class a policy is a single probability $p = \pi_\theta(a_1 \mid o)$. Per-client optima are always-a_1 for client 1 and always-a_2 for client 2, with episode return $HR_{\max}\frac{1+\delta}{2}$ each. For any federated $\pi_\theta(a_1 \mid o) = p$,

$$\mathcal{J}_1(\theta) = HR_{\max}\left[\tfrac{1}{2} + \tfrac{(2p-1)\delta}{2}\right],$$
$$\mathcal{J}_2(\theta) = HR_{\max}\left[\tfrac{1}{2} - \tfrac{(2p-1)\delta}{2}\right],$$
$$\mathcal{J}_{\text{fed}}(\theta) = HR_{\max}/2 \quad \text{(constant in } p\text{)}.$$

Therefore

$$\Delta_{\text{pol}} \;=\; HR_{\max}\tfrac{1+\delta}{2} - HR_{\max}/2 \;=\; \tfrac{R_{\max}H\delta}{2} \;=\; \Omega(R_{\max}H\delta),$$

the stated bound. The reward stays in $[0, R_{\max}]$, the policy class contains the per-client optima by (LR), and the divergence is exactly δ.

Discounted analogue. With $\gamma < 1$ and $H = 1/(1-\gamma)$, run the same chain indefinitely. The per-client optimum is $\sum_{u\geq 0}\gamma^u R_{\max}\frac{1+\delta}{2} = R_{\max}H\frac{1+\delta}{2}$, and the federated objective equals $R_{\max}H/2$ for every p, so again $\Delta_{\text{pol}} = R_{\max}H\delta/2$.

Worst-client gap. For any $\theta^\star_{\text{fed}} \in \arg\max_{\theta\in\Theta}\mathcal{J}_{\text{fed}}(\theta)$,

$$\sup_{i\in[N]}\big[\sup_{\theta_i\in\Theta}\mathcal{J}_i(\theta_i) - \mathcal{J}_i(\theta^\star_{\text{fed}})\big] \;\geq\; \frac{1}{N}\sum_{i=1}^{N}\big[\sup_{\theta_i\in\Theta}\mathcal{J}_i(\theta_i) - \mathcal{J}_i(\theta^\star_{\text{fed}})\big] \;=\; \Delta_{\text{pol}},$$

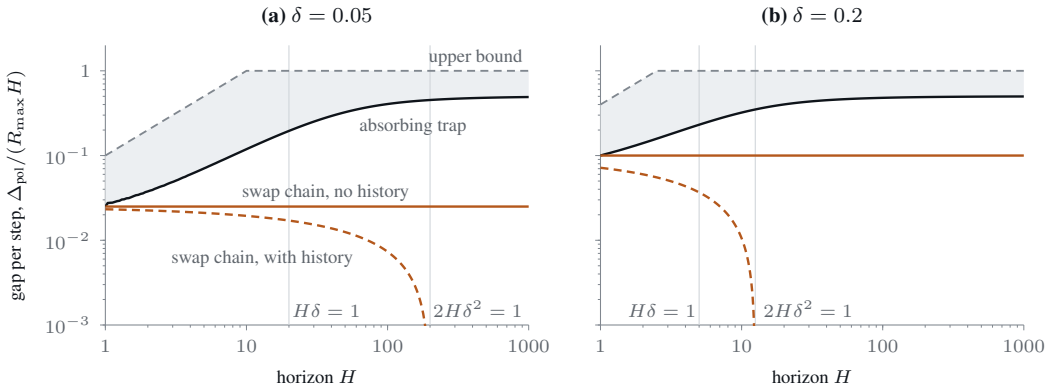


Figure 21: **How the environment-level gap grows with the horizon.** Each curve is a closed form from Appendix N.3, divided by $R_{\max}H$ so that it gives the gap per step, for $\delta = 0.05$ in (a) and $\delta = 0.2$ in (b). When the policy cannot use past outcomes (silent observations), the swap chain of Theorem 4 has a gap of $\delta/2$ per step at every H (solid orange). When the policy may condition on past outcomes, the gap of the chain is bounded below by the dashed orange curve, which reaches zero at $2H\delta^2 = 1$. The gap of the absorbing trap of Remark 10 is about $(H+1)\delta/4$ per step while $H\delta$ is small and approaches $1/2$ per step as $H\delta$ grows (black). The trap gives this gap for every policy class satisfying (LR), with or without history. If the policy class contains one client's optimal policy, as (LR) posits for our constructions, no construction with divergence δ has a gap above $\min\{1, 2H\delta\}$ per step (dashed gray). The worst-case gap therefore lies in the shaded band, so it is $\Theta(R_{\max}H\min\{1, H\delta\})$. The vertical lines mark $H\delta = 1$ and $2H\delta^2 = 1$.

where the first inequality is $\sup \geq \text{avg}$ and the equality uses the optimality of $\theta^\star_{\text{fed}}$ for the federated objective. Combining this bound with the chain construction completes the proof. □

Remark 9. *The asymmetry with Theorem 2 is now quantitative: the task-level slack $\sqrt{(1+\chi^2)R_{\max}H(\epsilon_{approx}+\epsilon_{opt})}$ vanishes as capacity and optimization improve, whereas the environment-level gap persists for any optimizer. The single-step gap $\Omega(R_{\max}\delta)$ holds for any Π_Θ satisfying (LR), since no history exists before the sole decision. For the horizon-amplified gap, the observation-silent restriction matters only when the two environments can be distinguished within an episode. Suppose instead that the shared policy may condition on past outcomes, as LLM policies do through their context window (cf. (R_{aug})), and let m_2 count the steps in which it plays a_2. Client 1 loses $R_{\max}\delta$ in expectation in each such step and client 2 in each remaining step, so the two regrets sum to $R_{\max}\delta\,(H + \mathbb{E}_1[m_2] - \mathbb{E}_2[m_2]) \geq R_{\max}\delta H\,(1 - \text{TV})$, where TV is the total variation between the laws of the H-step history under $\mathcal{M}_1$ and $\mathcal{M}_2$. Each step contributes a KL divergence of $\delta \ln \frac{1+\delta}{1-\delta} \leq \frac{8}{3}\delta^2$ for $\delta \leq \frac{1}{2}$, so Pinsker's inequality gives $\text{TV} \leq \delta\sqrt{2H}$ and, for every history-dependent policy class satisfying (LR),*

$$\Delta_{pol} \;\geq\; \frac{R_{\max}H\delta}{2}\big(1 - \delta\sqrt{2H}\big) \qquad \big(\delta \leq \tfrac{1}{2}\big).$$

The $\Omega(R_{\max}H\delta)$ floor therefore persists without the observation-silent restriction whenever $\delta \lesssim H^{-1/2}$, the regime in which environment identity cannot be inferred within the episode. Outcome-based inference removes the floor only when $H \gtrsim \delta^{-2}$, which is the (C3) regime: a reveal step $u^\star = O(\delta^{-2}\log(1/\eta_{cls}))$ of outcome observations identifies the environment with error η_{cls}. The observation-silent restriction keeps the construction in the (C3)-unavailable case for every δ, matching the statement of Theorem 4. Within the observation-silent class, (C3) is unavailable, so among the three conditions of Theorem 5 only (C1) or (C2) can apply.

Remark 10 (Tightness in H)**.** *The order $\Omega(R_{\max}H\delta)$ is not the largest possible. If Π_Θ contains one client's optimal policy $\pi_1^\star$, as (LR) posits for our constructions, Lemma 2 gives $\mathcal{J}_i(\pi_i^\star) \leq \mathcal{J}_1(\pi_i^\star) + R_{\max}H^2\delta \leq \mathcal{J}_1(\pi_1^\star) + R_{\max}H^2\delta \leq \mathcal{J}_i(\pi_1^\star) + 2R_{\max}H^2\delta$ for every i, so $\Delta_{pol} \leq 2R_{\max}H^2\delta$, alongside the trivial $\Delta_{pol} \leq R_{\max}H$. An absorbing-trap construction attains the smaller of these two orders. Give each client a live state and a zero-reward absorbing trap, let client 1's action a_2 and client 2's action a_1 fall into the trap with probability δ, keep every other transition live, and*

give reward $R_{\max}$ for every step that ends in the live state. While the agent is in the live state, every history-dependent policy receives the same observations, so it reduces to an open-loop action sequence. By convexity of $x \mapsto (1-\delta)^x$, the best shared sequence is constant, and

$$\Delta_{pol} \;=\; \frac{R_{\max}}{2}\Big[H - \sum_{u=1}^{H}(1-\delta)^u\Big] \;\geq\; \frac{R_{\max}H}{12}\min\{1, H\delta\} \qquad (H \geq 2)$$

for every policy class satisfying (LR), with or without history. The worst-case gap is therefore $\Theta(R_{\max}H\min\{1, H\delta\})$. We state Theorem 4 with the transition-swap chain because it isolates the observation-silent mechanism that the WebShop variants probe. Figure 21 compares the swap chain and the trap across horizons.

N.4 Proof of Theorem 5 (Recovery via (C1)–(C3))

This subsection proves Theorem 5, the recovery result that identifies when the worst-case lower bound of Theorem 4 no longer applies. The recovery theorem distinguishes three structurally distinct regimes ((C1) common optimal off-support, (C2) action-preserving with shared $\pi^\star$, (C3) self-revealing environment) under which the policy-disagreement gap reduces to zero or to a controllable horizon-scaled slack. Each case relies on a realizability assumption from Appendix N.1: (C1) and (C2) use ($\mathrm{R}_{\mathrm{env}}$), and (C3) uses ($\mathrm{R}_{\mathrm{aug}}$). Both suprema in the gap Δ_{pol} depend on the policy class: the (C3) case evaluates both over the history-augmented class $\Pi_\Theta^{\mathrm{aug}}$ with the same Θ, whereas the lower bound of Theorem 4 uses an observation-silent class, so the two statements concern different classes and do not conflict. Throughout, $\mathcal{J}_i(\pi)$ is the expected return of a policy π in $\mathcal{M}_i$ (so $\mathcal{J}_i(\theta) = \mathcal{J}_i(\pi_\theta)$), $V_i^\star := \sup_\pi \mathcal{J}_i(\pi)$ is the optimal value of $\mathcal{M}_i$ over all policies, and $V_P^\pi(s)$ is the value of π from state s under kernel P over the remaining horizon. Also, $d^\pi_{\mathcal{M}_i}$ denotes the normalized state-action occupancy of π in $\mathcal{M}_i$ (step-averaged for $\gamma = 1$), and $\delta_{\mathrm{eff}}(\pi; i, j) := \mathbb{E}_{(s,a)\sim d^\pi_{\mathcal{M}_i}}[D_{\mathrm{TV}}(P_i(\cdot \mid s,a), P_j(\cdot \mid s,a))]$ is the policy-induced (effective) divergence between $\mathcal{M}_i$ and $\mathcal{M}_j$. Among modifications of the bandit of Figure 20, the bandit in Figure 7(c) satisfies (C1) and (C2), and the bandit in Figure 7(d) satisfies (C3) with $u^\star = 0$ (with a client-dependent initial observation) and attains the (C3) bound at $H = 1$ (see the end of Case (C3)).

Theorem (Recovery via (C1)–(C3) (Theorem 5))**.** *Under Definition 2:*

(C1) *Under (R_{env}), if some policy $\pi^\star$ is optimal in some $\mathcal{M}_k$ with $V_k^\star \geq \max_j V_j^\star$, and its occupancy in $\mathcal{M}_k$ avoids the disagreement region, $\max_j \delta_{eff}(\pi^\star; k, j) = 0$, then $\pi^\star$ is optimal in every $\mathcal{M}_i$ and $\Delta_{pol} = 0$.*

(C2) *Under (R_{env}), if a shared policy $\pi^\star$ puts all mass on $\arg\max_a Q_i^\star(s, \tau, a)$ for every i and every (s, τ) in the support of the state marginal of $d^{\pi^\star}_{\mathcal{M}_i}$ (with $\tau \sim \mathcal{D}_\tau$), then $\pi^\star$ is optimal in every $\mathcal{M}_i$ and $\Delta_{pol} = 0$. (The main-text (C2), which asks for an action that is optimal for every client at every reachable state, implies this condition: a policy that selects such actions serves as this $\pi^\star$.)*

(C3) *Under (R_{aug}), if environment identity is $(u^\star, \eta_{cls}, \eta_{ker})$-self-revealing with a recoverable prefix ((C3-cls), (C3-ker), and (C3-rec) of Appendix F.5), then*

$$\Delta_{pol} \;\leq\; c_0 R_{\max}\, u^\star + R_{\max} H \eta_{cls} + 2R_{\max} H^2 \eta_{ker},$$

with c_0 the constant of (C3-rec). In particular, $\Delta_{pol}/(R_{\max}H) \to 0$ as $(u^\star/H, \eta_{cls}, H\eta_{ker}) \to (0,0,0)$, and $\Delta_{pol} = 0$ when $u^\star = 0$ and $\eta_{cls} = \eta_{ker} = 0$ (this case requires the first observation to reveal the environment and thus relaxes the shared μ_0 of Definition 2, whereas the (C3) bound itself does not use μ_0).

Proof. The three implications are proved separately. Each proof uses the corresponding realizability assumption in a different optimality argument. Recall that $\Delta_{\mathrm{pol}} \geq 0$ always holds, because the supremum of an average never exceeds the average of the suprema, so in each case it suffices to bound Δ_{pol} from above.

Case (C1). Because $\delta_{\mathrm{eff}}(\pi^\star; k, j) = 0$, the kernels $P_j(\cdot \mid s,a)$ and $P_k(\cdot \mid s,a)$ coincide on every pair (s,a) that $\pi^\star$ visits in $\mathcal{M}_k$ with positive probability. Since the task distribution, initial state, and

reward are shared (Definition 2), induction over steps shows that the trajectory law of $\pi^\star$ is the same in $\mathcal{M}_j$ as in $\mathcal{M}_k$, so

$$V_j^\star \;\geq\; \mathcal{J}(\pi^\star;\mathcal{M}_j) \;=\; \mathcal{J}(\pi^\star;\mathcal{M}_k) \;=\; V_k^\star \;\geq\; V_j^\star.$$

All inequalities are therefore equalities, and $\pi^\star$ is optimal in every $\mathcal{M}_j$. By ($\mathrm{R_{env}}$), some $\theta^\star \in \Theta$ realizes $\pi^\star$, hence $\mathcal{J}_i(\theta^\star) = \sup_{\theta\in\Theta} \mathcal{J}_i(\theta)$ for every i. The federated objective satisfies

$$\mathcal{J}_{\text{fed}}(\theta^\star) \;=\; \frac{1}{N}\sum_i \mathcal{J}_i(\theta^\star) \;=\; \frac{1}{N}\sum_i \sup_{\theta\in\Theta} \mathcal{J}_i(\theta),$$

so any $\theta^\star_{\text{fed}} \in \arg\max_\theta \mathcal{J}_{\text{fed}}(\theta)$ also achieves this value, yielding $\Delta_{\text{pol}} = 0$.

Both hypotheses are needed (the swap bandit shows that the value condition alone fails). The occupancy condition alone does not suffice: a client may have a strictly better policy that passes through states outside $\pi^\star$'s occupancy, and the value condition $V_k^\star \geq \max_j V_j^\star$ excludes exactly this case. The occupancy condition is also what makes (C1) checkable, since δ_{eff} can be estimated from rollouts of a candidate $\pi^\star$ (test (T1) of Appendix O.8.1), whereas verifying optimality in every $\mathcal{M}_i$ directly would require solving each of them.

Case (C2). For each i, $\pi^\star$ selects an action in $\arg\max_a Q_i^\star(s,\tau,a)$ at every (s,τ) that it reaches in $\mathcal{M}_i$. By the standard MDP optimality theorem (selecting Bellman-optimal actions on every reachable state yields a globally optimal policy), $\pi^\star$ is globally optimal in $\mathcal{M}_i$ for every i, even though the values $V_i^\star$ may differ across i. By ($\mathrm{R_{env}}$), some $\theta^\star \in \Theta$ realizes $\pi^\star$, hence $\mathcal{J}_i(\theta^\star) = \sup_{\theta\in\Theta} \mathcal{J}_i(\theta)$ for every i and $\Delta_{\text{pol}} = 0$ by the same averaging argument as in (C1).

Case (C3). By ($\mathrm{R_{aug}}$), history-conditioned policies are realized by the *same* parameter class Θ (the LLM's context window already carries h_u, and no new parameter is added, cf. the remark after ($\mathrm{R_{aug}}$) in Appendix N.1). In the (C3) case we write $\mathcal{J}_i(\theta)$ for the return of π_θ^{aug}, so both suprema in Δ_{pol} range over Π_Θ^{aug} with the same Θ. Consider the parameter $\theta^{\text{aug}} \in \Theta$ realizing the following environment-conditional policy, which ($\mathrm{R_{aug}}$) places in the augmented class: it follows the default prefix policy π_0 of (C3) for $u < u^\star$, adopts at the reveal step the environment label $\hat{\imath}(h_{u^\star})$ inferred from the prefix, and thereafter executes the policy $\hat{\pi}_{\hat{\imath}}$ that is optimal for the kernel model:

$$\pi^{\text{aug}}_{\theta^{\text{aug}}}(a \mid s,\tau,h_u) \;=\; \hat{\pi}_{\hat{\imath}(h_{u^\star})}(a \mid s,\tau) \quad (u \geq u^\star),$$

where $\hat{\imath} : \mathcal{H} \to [N]$ is the classifier of (C3) and $\hat{\pi}_j$ is a (step-dependent) policy that is optimal for the kernel model $\hat{P}_j$ from every state and every remaining horizon, which exists by backward induction. The step index u, which h_u determines, is suppressed in $\hat{\pi}_j$. Likewise, $\pi_i^\star$ denotes a policy that is optimal for P_i from every state and every remaining horizon, so that $V_{P_i}^{\pi_i^\star}(s) = V_i^\star(s)$. Since no policy, augmented or not, can exceed the optimal value of $\mathcal{M}_i$, we have $\sup_{\theta_i\in\Theta} \mathcal{J}_i(\theta_i) \leq V_i^\star$, while $\sup_{\theta\in\Theta} \mathcal{J}_{\text{fed}}(\theta) \geq \mathcal{J}_{\text{fed}}(\theta^{\text{aug}})$. Combining,

$$\Delta_{\text{pol}} \;\leq\; \frac{1}{N}\sum_{i=1}^{N} \big[V_i^\star - \mathcal{J}_i(\theta^{\text{aug}})\big],$$

so it suffices to bound the regret of θ^{aug} on each $\mathcal{M}_i$. We decompose this regret into three sources: prefix regret before $u^\star$, misclassification regret after $u^\star$, and kernel-modeling regret from planning with $\hat{P}_{\hat{\imath}}$ instead of $P_{\hat{\imath}}$.

Step (a): pre-reveal prefix regret. For $u < u^\star$, the policy follows π_0 and has not yet identified the environment. Since rewards are non-negative, dropping the prefix rewards of θ^{aug} gives the decomposition

$$V_i^\star - \mathcal{J}_i(\theta^{\text{aug}}) \;\leq\; \underbrace{V_i^\star - \mathbb{E}_{h_{u^\star}\sim(\mathcal{M}_i,\pi_0)}\big[V_i^\star(s_{u^\star})\big]}_{\text{prefix reward + handoff drift}} \;+\; \underbrace{\mathbb{E}_{h_{u^\star}\sim(\mathcal{M}_i,\pi_0)}\big[V_i^\star(s_{u^\star}) - V_i^{\text{post}}(h_{u^\star})\big]}_{\text{post-reveal regret: Steps (b), (c)}},$$

where $V_i^\star(s_{u^\star})$ and $V_i^{\text{post}}(h_{u^\star}) := V_{P_i}^{\hat{\pi}_{\hat{\imath}(h_{u^\star})}}(s_{u^\star})$ are values in $\mathcal{M}_i$ from $s_{u^\star}$ over the remaining $H - u^\star$ steps, the latter for the post-reveal policy adopted after the prefix $h_{u^\star}$. The first bracket accounts both for the reward lost during the prefix and for the *state drift* induced by π_0. This term is *not* controlled by $u^\star$ alone in a general MDP: a single prefix step can irreversibly drive the agent into a low-value (*e.g.*, zero-reward absorbing) region, costing $\Omega(R_{\max}(H - u^\star))$ even when $u^\star = 1$.

This irreversible loss is exactly what the recoverability condition (C3-rec) rules out: under (C3-rec), the first bracket is at most $c_0 R_{\max}\, u^\star$.

Step (b): post-reveal misclassification regret. For $u \geq u^\star$, the policy adopts the label $\hat{\imath}(h_{u^\star})$ for the rest of the episode, and this label is wrong with probability at most η_{cls} by (C3-cls). Here (C3-cls) applies because it is stated under the prefix distribution generated by the same π_0 that θ^{aug} follows. On this misclassified event, the policy follows $\hat{\pi}_{\hat{j}}$ in $\mathcal{M}_i$ for some $\hat{j} \neq i$ throughout the post-reveal trajectory. Since rewards lie in $[0, R_{\max}]$, the loss against the optimal value satisfies $V_i^\star(s) - V_i^\pi(s) \leq V_{\max} = R_{\max} H$ for *any* policy π and state s. The misclassified event therefore contributes at most $\eta_{\text{cls}} R_{\max} H$ to the second bracket. The horizon scaling here is only linear in H because the label is fixed once at the reveal step $u^\star$: misclassification is then a single trajectory-level event (the whole post-reveal trajectory follows one wrong label), so the value-range bound applies once per trajectory and does not accumulate over steps.

Step (c): kernel-modeling regret. On the correctly classified event, the post-reveal policy is $\hat{\pi}_i$, which is optimal for the kernel model $\hat{P}_i$ and not necessarily for the true kernel. With $\sup_{s,a} D_{\text{TV}}(\hat{P}_i, P_i) \leq \eta_{\text{ker}}$ by (C3-ker), applying Lemma 2 (cross-MDP simulation) twice from the fixed start state $s_{u^\star}$ yields the standard model-error bound, in which every value is evaluated at $s_{u^\star}$ over the remaining $H - u^\star$ steps:

$$V_{P_i}^{\pi_{\hat{i}}^\star} - V_{P_i}^{\hat{\pi}_i} \;=\; \big(V_{P_i}^{\pi_{\hat{i}}^\star} - V_{\hat{P}_i}^{\pi_{\hat{i}}^\star}\big) + \big(V_{\hat{P}_i}^{\pi_{\hat{i}}^\star} - V_{\hat{P}_i}^{\hat{\pi}_i}\big) + \big(V_{\hat{P}_i}^{\hat{\pi}_i} - V_{P_i}^{\hat{\pi}_i}\big) \;\leq\; 2R_{\max} H^2\, \eta_{\text{ker}},$$

since the middle term is ≤ 0 by optimality of $\hat{\pi}_i$ under $\hat{P}_i$ from $s_{u^\star}$ and each outer term is at most $R_{\max} H^2 \eta_{\text{ker}}$ by Lemma 2. The correctly classified event therefore contributes at most $2R_{\max} H^2 \eta_{\text{ker}}$ to the second bracket. Here the H^2 scaling occurs because the kernel error compounds at every step of the rollout (one H from the value range, one H from telescoping per-step transition errors).

Summing the three contributions,

$$\Delta_{\text{pol}} \;\leq\; c_0 R_{\max}\, u^\star + R_{\max} H \eta_{\text{cls}} + 2R_{\max} H^2 \eta_{\text{ker}},$$

as claimed. In particular, when $u^\star = 0$, $\eta_{\text{cls}} = 0$, and $\eta_{\text{ker}} = 0$, the bound reduces to $\Delta_{\text{pol}} = 0$, agreeing with the (C1) and (C2) results. Under the shared μ_0 of Definition 2, $h_0 = (s_0)$ has the same law in every environment, so (C3-cls) with $u^\star = 0$ implies $\eta_{\text{cls}} \geq 1 - 1/N$. This case of the proof never uses the shared μ_0, however, so the bound also holds when the first observation already reveals the environment. Figure 7(d), whose banner names the client with error η_{cls}, is such a case, and there the bound holds with equality at $u^\star = 0$ and $H = 1$. □

Remark 11. *The horizon scaling difference between η_{cls} (H) and η_{ker} (H^2) is structural and matters for analysis. Because the classifier fixes a single environment label at the reveal step $u^\star$, post-reveal misclassification is a trajectory-level event (one wrong label per trajectory, costing one value range $V_{\max} = R_{\max} H$), which gives the H scaling. Were the policy instead to re-evaluate the classifier at every step under a per-step slack, a wrong label could recur up to H times and the corresponding bound would increase to H^2. The kernel slack η_{ker}, by contrast, is a per-step divergence whose effect compounds along the rollout via the simulation lemma, so it acquires the second factor of H that the classifier slack does not. For LLM agents we expect the classifier slack to dominate. When the post-identification behavior is itself trained on the identified environment, it targets the true kernel, so (R_{aug}) holds with $\hat{P}_j = P_j$ and $\eta_{ker} = 0$. A positive η_{ker} captures behavior driven by an imperfect internal model of the identified environment, such as API conventions from pre-training that the environment no longer follows. When $\eta_{ker} = 0$, only the $u^\star$ and $H\eta_{cls}$ terms remain and the bound is mild.*

Remark 12. *The (C3) bound does not depend on δ. The bound ranges from the vanishing gap of (C1) and (C2), reached at $u^\star = 0$ and $\eta_{cls} = \eta_{ker} = 0$ (possible only when the first observation reveals the environment, which relaxes the shared μ_0 of Definition 2), to the trivial $R_{\max} H$ as the slacks grow, while Theorem 4 shows that gaps of order $R_{\max} H \delta$ do occur when (C3) is unavailable. The (C3) bound predicts when* FEDAGENT *under environment-level heterogeneity can work without modification on real LLM-agent deployments: wherever branding, tool messages, and DOM structure make $u^\star = O(1)$ and η_{cls} small from the very first step. In such deployments, environment identity is effectively recast as part of the input, so the situation behaves like task-level heterogeneity. Corollary 1 makes this reduction formal for explicit descriptors, and the (C3) bound of Theorem 5 covers the implicit case. In terms of the four patterns of §4.4, this reduction is the structural reason to expect*

Pattern B to dominate Pattern D in deployments with verbose, identity-revealing observations, with Pattern D appearing mainly on adversarial constructions like the observation-silent chain of Theorem 4.

N.5 Gradient Heterogeneity ζ^2 at the Federated Solution

The previous subsections established *quality-of-solution* statements: Theorem 2 controls the per-client sub-optimality of any near-optimal federated parameter by vanishing slacks, while Theorem 4 exhibits constructions with a structural gap of order $R_{\max} H \delta$ in the environment-level regime. The present subsection relates these statements to the gradient-heterogeneity quantity ζ^2 that FedAvg, SCAFFOLD, and most other supervised-FL and FRL convergence analyses use. Our point is that the mechanism controls ζ^2 *at the federated solution*. By Equation (2) below, ζ^2 at any near-optimal federated parameter is at most $2L_{\mathrm{sm}}(\Delta_{\mathrm{pol}} + \epsilon_{\mathrm{opt}})$. It therefore vanishes under task-level heterogeneity as $\epsilon_{\mathrm{approx}} + \epsilon_{\mathrm{opt}} \to 0$ and under (C1) or (C2) as $\epsilon_{\mathrm{opt}} \to 0$. Under (C3), it is bounded by a slack-controlled residual whose ratio to $L_{\mathrm{sm}} R_{\max} H$ vanishes as $(u^\star/H, \eta_{\mathrm{cls}}, H\eta_{\mathrm{ker}}) \to 0$ and $\epsilon_{\mathrm{opt}}/(R_{\max} H) \to 0$. On the worst-case construction of Theorem 4, by contrast, some federated stationary points keep ζ^2 of order $R_{\max}^2 H^2 \delta^2$, whatever the optimizer, batch size, or local-step count. These statements concern solutions only, so they can be combined with any convergence rate the reader may wish to apply.

Definition. For any $\theta \in \Theta$, define the cross-client gradient heterogeneity at θ by

$$\zeta^2(\theta) \;:=\; \frac{1}{N} \sum_{i=1}^{N} \big\| \nabla \mathcal{J}_i(\theta) - \nabla \mathcal{J}_{\mathrm{fed}}(\theta) \big\|^2,$$

where $\nabla \mathcal{J}_i$ is the per-client policy gradient and $\nabla \mathcal{J}_{\mathrm{fed}} = \frac{1}{N} \sum_{i=1}^{N} \nabla \mathcal{J}_i$ is the federated gradient. The uniform bound $\sup_{\theta \in \Theta} \zeta^2(\theta) \le \bar{\zeta}^2$ is the inter-client heterogeneity assumption used throughout the federated optimization literature. Standard FedAvg / local-SGD convergence analyses [52, 67] yield bounds that, up to additional local-drift terms, take the prototypical form

$$\Phi_{\mathrm{alg}}(N, K, T) \;\le\; \widetilde{O}\!\left(\left(\frac{\sigma^2}{NKT} \right)^{\alpha_\sigma} \;+\; \frac{\bar{\zeta}^2}{T^{\alpha_\zeta}} \;+\; \frac{L_{\mathrm{sm}}\, D_0}{T^{\alpha_0}} \right), \tag{1}$$

where Φ_{alg} is an algorithm-specific convergence measure (*e.g.*, $\frac{1}{T}\sum_t \mathbb{E}\|\nabla \mathcal{J}_{\mathrm{fed}}(\theta_t)\|^2$ for stationary-point analyses, or the federated objective gap for objective-gap analyses), K is the number of local gradient steps per round ($K = E$ in the pseudocode of Algorithm 1, which takes one step per local epoch, and larger in our implementation, which runs several mini-batch updates per epoch), σ^2 is the intra-client stochastic-gradient variance, L_{sm} is a smoothness constant (unrelated to the pool-size symbol L of §5.2), D_0 is the initial sub-optimality, and $(\alpha_\sigma, \alpha_\zeta, \alpha_0) \in (0, 1]^3$ are exponents specific to the analysis. The exponent α_σ is set by the convexity regime and does not depend on the algorithm (*e.g.*, $\alpha_\sigma = 1/2$ in non-convex analyses of both FedAvg and SCAFFOLD). Control variates remove the dependence on $\bar{\zeta}^2$ and leave α_σ unchanged, and in specific analyses $\bar{\zeta}^2$ may also enter through a fractional power. We do not assume any particular choice of these exponents: Equation (1) is used only to identify the role of the uniform heterogeneity bound $\bar{\zeta}^2$. We deliberately write the heterogeneity term as decaying in T alone: in several analyses it does not improve with the local-step count K or the client count N, and can even degrade with K through client drift [52]. We also note that Li et al. [67] is a strongly convex analysis whose heterogeneity measure is an objective-value gap, which differs from $\bar{\zeta}^2$. We cite it only for the overall shape of the rate.

A solution-level bound on ζ^2. If each $\mathcal{J}_i$ is L_{sm}-smooth on $\Theta = \mathbb{R}^d$, then for every θ the ascent step $\theta + \nabla \mathcal{J}_i(\theta)/L_{\mathrm{sm}}$ gains at least $\|\nabla \mathcal{J}_i(\theta)\|^2/(2L_{\mathrm{sm}})$, so $\|\nabla \mathcal{J}_i(\theta)\|^2 \le 2L_{\mathrm{sm}}\big[\sup_{\theta'} \mathcal{J}_i(\theta') - \mathcal{J}_i(\theta)\big]$. Since $\zeta^2(\theta) \le \frac{1}{N}\sum_i \|\nabla \mathcal{J}_i(\theta)\|^2$, averaging over clients gives, for any $\hat{\theta}$ with $\mathcal{J}_{\mathrm{fed}}(\hat{\theta}) \ge \sup_\theta \mathcal{J}_{\mathrm{fed}}(\theta) - \epsilon_{\mathrm{opt}}$ and under any form of heterogeneity,

$$\zeta^2(\hat{\theta}) \;\le\; \frac{2L_{\mathrm{sm}}}{N} \sum_{i=1}^{N} \Big[\sup_{\theta} \mathcal{J}_i(\theta) - \mathcal{J}_i(\hat{\theta}) \Big] \;\le\; 2L_{\mathrm{sm}} \big(\Delta_{\mathrm{pol}} + \epsilon_{\mathrm{opt}} \big), \tag{2}$$

where the second inequality uses $\frac{1}{N}\sum_i \sup_\theta \mathcal{J}_i(\theta) = \Delta_{\mathrm{pol}} + \sup_\theta \mathcal{J}_{\mathrm{fed}}(\theta)$. This derivation is the only place where smoothness enters, and none of Theorems 1–5 depends on it.

Task-level: ζ^2 vanishes at near-optimal solutions. Under task-level heterogeneity (Definition 1) and (R′), $\frac{1}{N}\sum_i \sup_\theta \mathcal{J}_i(\theta) \le \mathbb{E}_{\bar{\mathcal{D}}_\tau}[\mathcal{J}_\tau^\star] \le \mathcal{J}_{\text{fed}}(\theta^\star) + \epsilon_{\text{approx}}$, so $\Delta_{\text{pol}} \le \epsilon_{\text{approx}}$, and Equation (2) gives $\zeta^2(\hat{\theta}_{\text{fed}}) \le 2L_{\text{sm}}(\epsilon_{\text{approx}} + \epsilon_{\text{opt}})$. The heterogeneity measured at near-optimal federated solutions therefore vanishes in the same limit as the slack of Theorem 2. Under the exact assumption (R), the realizing $\theta^\star$ maximizes every per-task return, so if it lies in the interior of Θ and each $\mathcal{J}_\tau$ is differentiable, then $\nabla\mathcal{J}_i(\theta^\star) = \mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\nabla\mathcal{J}_\tau(\theta^\star)] = 0$ for every i and $\zeta^2(\theta^\star) = 0$. (For softmax policies, exact optimality is typically reached only in a saturating limit, along which the gradients also vanish.) These results describe solutions and make no claim about rates. Rate bounds of the form Equation (1) use the uniform bound $\sup_\theta \zeta^2(\theta)$, which task-level heterogeneity does not make small, because, away from the optimum, one-hot splits produce strongly disagreeing client gradients. We therefore do not claim that FedAvg attains the centralized rate. Under task-level heterogeneity, ζ^2 is *transient*: we expect ζ^2 to be large early in training, and it is provably small near the solution. Its effect on the trajectory is the client drift discussed in Appendix O.2. On the quality side, the federated solution recovers the centralized one as the slacks vanish (Theorem 2).

Environment-level under (C1)–(C3): ζ^2 is controlled by the recovery slacks. Under (C1) or (C2) (with (R_{env})), $\Delta_{\text{pol}} = 0$, so Equation (2) gives $\zeta^2(\hat{\theta}_{\text{fed}}) \le 2L_{\text{sm}}\epsilon_{\text{opt}}$, and at an interior common optimum $\theta^\star$ every per-client gradient vanishes, so $\zeta^2(\theta^\star) = 0$. Under (C3) (with ($\text{R}_{\text{aug}}$)), Equation (2) and Theorem 5 give $\zeta^2(\hat{\theta}_{\text{fed}}) \le 2L_{\text{sm}}\big(c_0 R_{\max} u^\star + R_{\max} H \eta_{\text{cls}} + 2R_{\max} H^2 \eta_{\text{ker}} + \epsilon_{\text{opt}}\big)$, a slack-controlled residual. Along the training trajectory we expect, but do not prove, a different transient shape of ζ^2 under each condition. Under (C1) the disagreement region lies off the optimal trajectory, so $\zeta^2(\theta_t)$ should shrink as the iterates' policies concentrate on that trajectory, although the policies of earlier iterates can still visit the region. Under (C2) the value differences across environments can make $\zeta^2(\theta_t)$ large early in training. Under (C3), $\zeta^2(\theta_t)$ should stay large until the LLM learns in-context environment identification and then shrink.

Worst-case environment-level: ζ^2 stays large at some federated stationary points. Consider the transition-swap chain underlying Theorem 4 (Appendix N.3), in which two clients have swapped optimal actions and the observation-silent policy class is a softmax over the two actions with logits θ, so that $p = \pi_\theta(a_1 \mid o)$. Then $\nabla\mathcal{J}_1(\theta) = -\nabla\mathcal{J}_2(\theta) = R_{\max} H\delta\, \nabla p(\theta)$ with $\|\nabla p(\theta)\| = \sqrt{2}\, p(1-p)$. Because $\mathcal{J}_{\text{fed}}$ is identically constant on this construction, *every* θ is a federated stationary point (for $H = 1$ and $\delta = 1$, the returns of these points form the black segment in Figure 7(b)). At the symmetric point θ_{sym} ($p = \frac{1}{2}$) the two clients' gradients are equal and opposite, each of magnitude $\sqrt{2}R_{\max}H\delta/4$, so they cancel in the average while leaving a large cross-client disagreement:

$$\sup_{\theta:\, \nabla\mathcal{J}_{\text{fed}}(\theta)=0} \zeta^2(\theta) \;\ge\; \zeta^2(\theta_{\text{sym}}) \;=\; \frac{R_{\max}^2 H^2 \delta^2}{8} \;>\; 0.$$

This lower bound holds at θ_{sym} and fails at stationary points near saturation: as the softmax saturates, $\zeta^2(\theta)$ decays toward zero. At such saturated points the policy concentrates on one client's optimal action, so the *other* client's regret is maximal, and the objective-side gap of Theorem 4 still applies even though ζ^2 vanishes. We therefore do not claim a universal gradient-level floor. The construction shows that ζ^2 can stay bounded away from zero on part of the federated stationary set. This result is consistent with Equation (2) because Δ_{pol} is large here, and it complements the objective-side gap of Theorem 4.

Two convergence dimensions must be kept separate. First, on the *rate-to-stationary* side, the heterogeneity term $\bar{\zeta}^2/T^{\alpha_\zeta}$ in Equation (1) still tends to zero as $T \to \infty$ even when $\bar{\zeta}^2$ is bounded below, so any standard federated optimizer (FedAvg, SCAFFOLD, FedSVRPG-M) still converges to a federated stationary point at the usual rate, possibly with a larger constant. We make no claim of a rate floor. Second, on the *quality-of-stationary* side, the worst-client gap at any $\hat{\theta}_T$ on the construction of Theorem 4 is bounded below by

$$\sup_i \Big[\, \sup_{\theta_i\in\Theta} \mathcal{J}_i(\theta_i) - \mathcal{J}_i(\hat{\theta}_T)\Big] \;\ge\; \Delta_{\text{pol}} \;=\; \Omega(R_{\max} H\delta),$$

because the $\sup \ge \text{avg}$ argument of Appendix N.3 applies at every θ. The bound holds regardless of T, the local-step count K, the number of clients N, the model capacity, or the optimizer. The floor is inherent to the federated solution *structure* and is independent of the optimization *rate*. No fast-rate

optimizer (variance reduction, control variates, momentum, FedSVRPG-M-style heterogeneity-independent rate) can eliminate it, since the Theorem 4 bound is on the worst-client objective gap, which is invariant to the optimizer. This rate-independence is the convergence-side counterpart to the structural distinction emphasized in §4 of the main text and in Appendix H. In particular, our bound is consistent with, and complementary to, fast-rate-to-stationary results such as FedSVRPG-M [125]: they bound the rate at which $\nabla \mathcal{J}_{\text{fed}} \to 0$ independently of $\bar{\zeta}^2$. We bound the worst-client quality of any such stationary point from below.

Three-regime comparison. Table 6 summarizes the three regimes along both dimensions.

Regime	ζ^2 at the federated solution	Quality of $\hat{\theta}_{\text{fed}}$ as per-client solution
Task-level (Definition 1, under (R′))	$\zeta^2(\hat{\theta}_{\text{fed}}) \le 2L_{\text{sm}}(\epsilon_{\text{approx}} + \epsilon_{\text{opt}}) \to 0$	$\sqrt{(1+\chi^2)\, R_{\max} H (\epsilon_{\text{approx}} + \epsilon_{\text{opt}})} \to 0$
Environment-level under (C1)–(C3)	$\zeta^2(\hat{\theta}_{\text{fed}}) \le 2L_{\text{sm}}(\Delta_{\text{pol}} + \epsilon_{\text{opt}})$; $\Delta_{\text{pol}} = 0$ under (C1)/(C2), slack-bounded under (C3)	$\le N\epsilon_{\text{opt}}$ under (C1)/(C2) ($= 0$ at θ^*_{fed}); $\le N(\Delta_{\text{pol}} + \epsilon_{\text{opt}})$ under (C3)
Worst-case environment-level (under (LR))	$R^2_{\max} H^2 \delta^2/8$ at θ_{sym}; $\to 0$ at saturated stationary points	$\ge \Omega(R_{\max} H\, \delta)$, irreducible

Table 6: **Three-regime comparison at the federated solution.** Under task-level heterogeneity and under environment-level heterogeneity satisfying (C1) or (C2), ζ^2 at near-optimal federated solutions is bounded through Equation (2) by slacks that vanish, and the per-client gap vanishes with them. Under (C3), the bound on ζ^2 and the per-client gap also contain the (C3) residual of Theorem 5, whose ratio to $R_{\max} H$ vanishes as $(u^\star/H, \eta_{\text{cls}}, H\eta_{\text{ker}}) \to 0$. The per-client gap in the middle row follows because the client-average gap is at most $\Delta_{\text{pol}} + \epsilon_{\text{opt}}$ and each client's gap is non-negative. On the worst-case construction of Theorem 4, ζ^2 reaches $R^2_{\max} H^2 \delta^2/8$ at the symmetric federated stationary point, and, as a complementary objective-level statement, the $\sup \ge \text{avg}$ argument of Theorem 4 gives a worst-client quality floor of order $R_{\max} H \delta$ that holds at *every* parameter. No fast-rate optimizer can eliminate this floor.

Closing remark. Existing FRL convergence analyses bound the rate at which $\hat{\theta}_T$ approaches a federated stationary point: FedAvg- and SCAFFOLD-style analyses do so under a uniform assumption $\sup_\theta \zeta^2(\theta) \le \bar{\zeta}^2$ treated as a black-box constant, while FedSVRPG-M-style analyses achieve rates independent of $\bar{\zeta}^2$ altogether. On the worst-case construction, ζ^2 reaches order $R^2_{\max} H^2 \delta^2$ at some federated stationary points, and the rate-side bounds remain valid. However, the stationary point they certify can be structurally biased against at least one client, so convergence no longer implies per-client quality. Our solution-level analysis complements those rate-side bounds: it characterizes the *quality* of the converged $\hat{\theta}_T$ as a per-client solution, whichever rate-side bound one prefers.

O Empirical Patterns: Extended Discussion

This appendix describes how the four empirical training-curve patterns of §4.4 arise from a single architectural fact about LLM agents. We first restate that fact and follow the unified causal chain from it to each pattern (Appendix O.1). We then examine each pattern's mechanism (Appendices O.2 through O.5) and connect the four patterns to optimization through the gradient-heterogeneity quantity ζ^2 (Appendix O.6). Next, we present two complementary tools based on the mechanism: a post-hoc diagnostic decision tree (Appendix O.7) and a pre-deployment mitigation protocol (Appendix O.8). We conclude by summarizing the mechanism's predictions and falsifiable claims (Appendix O.9). An operator who only needs the deployment procedure can skip directly to Appendix O.8.

O.1 The Unified Causal Chain

The mechanism in §4 can be restated as a single causal chain based on one architectural fact about LLM-induced policies. This subsection states that fact, explains its two qualitatively different consequences, and describes the full chain. The remaining subsections examine each step in detail.

O.1.1 The Single Architectural Fact: Input-Dynamics Asymmetry

The chain starts from how LLM agents are built in practice:

> **I-D Asymmetry.** The task descriptor τ enters the policy through its input channel $\pi_\theta(a \mid s, \tau)$. The transition kernel P does not. The policy has access to P only indirectly, through the observations that follow its actions.

I-D Asymmetry is descriptive: the prompt to a deployed LLM agent contains τ (the user's task) but does not contain P (the environment it is acting in). The environment is the HTTP endpoint, terminal, or DOM tree that the agent's tool calls access. The policy can obtain information about the environment only through observation streams. Every theorem and pattern in this appendix is a consequence of I-D Asymmetry applied to the federated objective $\mathcal{J}_{\text{fed}}(\theta) := \frac{1}{N}\sum_i \mathcal{J}_i(\theta)$, with no further structural assumptions about Π_Θ beyond the realizability assumptions of Appendix N.1.

O.1.2 The Two Sides of Asymmetry

I-D Asymmetry has qualitatively different consequences depending on whether the heterogeneity occurs in the input or the dynamics.

Task-level side: τ is in the input. A single θ can encode the function $\tau \mapsto b(\tau)$ from task descriptors to behaviors, because τ is part of the policy's input. The same parameters that handle summarization can also handle code synthesis, because the LLM conditions its behavior on the prompt. Concretely, under Definition 1 the federated objective reduces to

$$\mathcal{J}_{\text{fed}}(\theta) \;=\; \mathbb{E}_{\tau\sim\bar{\mathcal{D}}_\tau}\big[\mathcal{J}(\pi_\theta;\tau,\mathcal{M}_{\text{env}})\big] \quad \text{(Theorem 1(i))},$$

where $\bar{\mathcal{D}}_\tau$ is the federated task mixture. The population objective depends only on the mixture and is invariant to how task mass is split across clients. This split-invariance is the "benign" side of I-D Asymmetry: no matter how extreme the per-client task split, federated training optimizes the same objective as i.i.d. mixture training. The per-client objectives followed during local steps still differ, which produces the transient client drift of Appendix O.2.

Environment-level side: P_i is in the dynamics. A single θ *cannot* encode the function $i \mapsto b_i$ from environment identity to behavior, because i is not part of the policy's input: the same prompt at state s_0 must produce the same action distribution regardless of which $\mathcal{M}_i$ the agent is currently interacting with. Federated robustness is therefore conditional on a single structural question:

> *Does there exist a shared policy $\pi^\star \in \Pi_\Theta$ (or in Π_Θ^{aug} once history is admitted) that is simultaneously optimal in every $\mathcal{M}_i$?*

The three sufficient conditions of §4.3 provide three structurally distinct ways to obtain such a shared policy: (C1) common optimal off-support, (C2) action-preserving with shared $\pi^\star$, and (C3) a self-revealing environment, where an augmented class Π_Θ^{aug} realizes per-environment optima from history. When all three fail *and* the heterogeneity is of the adversarial type realized by Theorem 4's construction (observation-silent and action-flipping on visited states), no single θ encodes the per-environment optima and the $\Omega(R_{\max}H\delta)$ floor applies. The conditions are sufficient but not necessary, so robustness without any of (C1)/(C2)/(C3) remains possible outside that construction class (Appendix O.8.4).

O.1.3 The Full Causal Chain

As Appendices O.1.1 and O.1.2 explain, the causal chain from I-D Asymmetry to each empirical pattern has two branches.

Task-level branch ($\rightarrow$ Pattern A).

(1) I-D Asymmetry: τ is in the input, so a single θ can encode $\tau \mapsto b(\tau)$.

(2) Theorem 1(i) gives the mixture-equivalence identity: $\mathcal{J}_{\text{fed}}(\theta) = \mathbb{E}_{\bar{\mathcal{D}}_\tau}[\mathcal{J}(\pi_\theta;\tau,\mathcal{M}_{\text{env}})]$.

(3) The population objective depends only on $\bar{\mathcal{D}}_\tau$, so it is identical across all task splits with the same mixture.

(4) At near-optimal federated solutions, the gradient heterogeneity satisfies $\zeta^2 \le 2L_{\text{sm}}(\epsilon_{\text{approx}} + \epsilon_{\text{opt}})$ (Equation (2)), which vanishes with the slacks of (R′) and of the optimizer.

(5) If the split does not raise the optimizer slack, the federated training plateau coincides with the near-uniform baseline (the i.i.d.-style split of §5.2) up to estimation noise. Step (4) makes this coincidence plausible near convergence, where client gradients nearly agree, while the transient curve can still differ through local-SGD drift. This outcome is **Pattern A**.

Environment-level branch ($\rightarrow$ Patterns B, C, D).

(1) I-D Asymmetry: P_i is in the dynamics, so a single θ must choose one fixed behavior independent of environment identity.

(2) The structural question: does Π_Θ (or Π_Θ^{aug}) contain a shared optimum across all $\mathcal{M}_i$?

(3) This question has three structurally distinct positive answers via (C1)/(C2)/(C3). These conditions are sufficient but not necessary: when all three fail, the answer *may* be negative, and on Theorem 4's construction class it provably is (Appendix O.8.4 discusses the empirically robust exceptions). The slack quantities $(\eta_{\text{cls}}, \eta_{\text{ker}}, u^\star)$ of Appendix O.4 parameterize how closely (C3) holds, and $\widehat{\delta}_{\text{eff}}$ is the empirical signature of residual (C1) failure.

(4) One of three branches applies:

- *Full-(C) branch:* some condition holds exactly ((C1) or (C2), or (C3) with zero slacks), so a shared optimum exists and $\zeta^2 \leq 2L_{\text{sm}}\epsilon_{\text{opt}}$ at near-optimal solutions. Provided the optimizer attains this shared optimum (ϵ_{opt} small), the federated curve coincides with the single-environment baseline (**Pattern B**). A nonzero ϵ_{opt} leaves the run in Pattern C despite full structural (C) satisfaction (the Catalog Split/GRPO and Lookalike Injection cases of §5.3).
- *Partial-(C) branch:* a shared $\pi^\star$ exists only approximately, and ζ^2 at near-optimal solutions is small but positive, bounded through Equation (2) by the slacks (cf. Appendix O.6). The plateau is stable but biased below the single-environment baseline, by an amount that depends on these slacks (Theorem 5 bounds the (C3) terms). This outcome is **Pattern C**.
- *No-(C) branch:* on adversarial constructions such as that of Theorem 4, no shared $\pi^\star$ exists, and the plateau is structurally biased by $\Omega(R_{\max}H\delta)$ at the objective level. At the gradient level, the federated stationary set of the worst-case construction contains points with $\zeta^2 = \Omega(R_{\max}^2 H^2 \delta^2)$, attained at the symmetric point θ_{sym}. We deliberately claim no universal trajectory-level floor (Appendix N.5). The local-SGD trajectory can oscillate between conflicting client gradients, and the repeated conflicting updates can overwrite the LLM's pre-trained world model (prior degradation). This outcome is **Pattern D**.

The chain is therefore: *single architectural fact $\rightarrow$ two structural sides $\rightarrow$ four empirical patterns*. Patterns A and B share the same plateau-overlap signature but for different reasons (mixture-equivalence vs. shared-optimum existence). Patterns B/C/D form a continuous spectrum parameterized by the slacks. Pattern D's possible behavior decomposes further into quality bias, oscillating updates, and prior degradation, as detailed in Appendix O.5. The remainder of this appendix elaborates each link of the chain in turn.

O.2 Pattern A: Plateau Identity vs. Trajectory Drift

The mixture-equivalence identity (Theorem 1(i), underlying Theorem 2)

$$\mathcal{J}_{\text{fed}}(\theta) \;=\; \mathbb{E}_{\tau\sim\bar{\mathcal{D}}_\tau}\left[\mathcal{J}(\pi_\theta; \tau, \mathcal{M}_{\text{env}})\right]$$

makes the federated population objective identical across any task split sharing the same mixture $\bar{\mathcal{D}}_\tau$. As a direct consequence, every split poses the same population optimization problem, so the achievable plateau is identical. The empirical interpretation of Pattern A is therefore a two-part statement:

(A-i) **Plateau identity (solution-level).** The long-horizon plateau matches the near-uniform baseline up to estimation noise. Population-objective identity (Theorem 1(i)) makes the achievable plateau the same for every split. FedAvg attains this plateau equally well only if the split also does not raise the optimizer slack ϵ_{opt}. The common optimum of Theorem 1(ii) supports equal attainment near convergence, where client gradients nearly agree (Equation (2)), but no theorem guarantees equal attainment. Theorem 2 adds a per-client guarantee, which the global validation curves of Figure 5 do not test.

(A-ii) **Trajectory drift (algorithm-level).** The transient training curve under extreme task heterogeneity need not track the near-uniform baseline curve. The transient drift depends on the number E of local epochs, the learning rate, and the aggregation frequency, and is bounded by standard FedAvg drift terms in E and the inter-round step size. In our single-run curves

the drift exceeds 10 points mid-training in several comparisons, in both directions, before the curves rejoin (Figure 5 and Appendix G.3).

This decomposition matters because Pattern A can be misread as predicting exactly identical training curves. Under that interpretation, any visible transient gap would be treated as a falsification. The actual prediction is plateau identity, and the evidence in Figures 5 and 9 is consistent with it: across all three task-level sub-types (preference, coverage, hardness) and both optimizers, the twelve plateau gaps lie between -10.8 and $+9.5$ points, and ten of their checkpoint ranges contain zero (Figure 23).

Falsifiability. A single observation would contradict the Pattern A prediction: a task split $\{\mathcal{D}_{\tau_i}\}$ whose mixture matches the near-uniform baseline but whose federated training plateau differs from the near-uniform baseline's plateau by more than algorithmic noise. Since the two splits share one population objective (Theorem 1(i)), such an observation would show that the split changes what the optimizer attains, or that the two mixtures differ in practice. A failure of (R′) on the actual Π_Θ is one possible cause of the former: without a common near-optimum, client gradients still disagree at the solution, so the fixed point of local SGD depends on the split. A split-dependent optimizer bias with (R′) intact is the other. No task-level plateau gap in our single runs exceeds 11 points (Figure 23), and this absence of a large divergence is the empirical content of Pattern A.

O.3 Pattern B vs. Pattern A: Same Plateau, Different Mechanism

Patterns A and B both produce a federated plateau that matches a "best baseline" (the near-uniform baseline for A and the single-environment baseline for B). The mechanisms differ, and this difference has observable experimental signatures:

- Pattern A: the mixture-equivalence identity (Theorem 1(i)) forces *population-objective identity* across splits. Different splits yield the same $\mathcal{J}_{\text{fed}}$ surface.
- Pattern B: Theorem 5 guarantees the existence of a *shared optimum*, in Π_Θ under (C1) or (C2), or, up to its slack bound, in Π_Θ^{aug} under (C3). Per-client population objectives differ (because $P_i \neq P_j$ in general), but at every reachable state their $\arg\max$ sets share at least one action.

The difference between the mechanisms is visible in the transient phase and does not appear at the plateau. Under Pattern A, transient curves across different splits differ only through client drift, because the population objective is the same. Under Pattern B, transient curves may visibly differ in shape across environment-level settings (the per-client objectives differ), yet they converge to the same shared optimum and therefore reach the same plateau.

For LLM agents, we expect (C3) to be the condition that most commonly applies in practice: branding, tool messages, and DOM structure can let the LLM identify the environment in-context. Environment identity then becomes part of the input, and a task-level-style mixture-equivalence argument applies within Π_Θ^{aug}. In this regime, Pattern B therefore arises because (C3) turns environment identity into an input dimension, so Pattern A's mixture-equivalence argument applies.

O.4 The B/C/D Spectrum and Its Slack Parameters

The Recovery Theorem (Appendix N.4) uses quantitative slack parameters to describe how closely (C3) holds. Patterns B, C, and D therefore form a continuous spectrum. The position of any environment-level deployment on the spectrum is determined by the four structural quantities listed below, together with a fifth, optimization-level factor: the optimizer's slack ϵ_{opt} (§4.4), which determines whether a structurally achievable plateau is actually attained.

- $\eta_{\text{cls}} \in [0, 1]$: classifier slack on the environment identity inferable from observation history (governs (C3)).
- $\eta_{\text{ker}} \in [0, 1]$: kernel-modeling slack $\sup_{s,a} D_{\text{TV}}(\hat{P}_i, P_i)$ of the (C3) kernel model (governs (C3), jointly with η_{cls} and $u^\star$).
- $u^\star \in \mathbb{N}$: reveal step after which the environment identity becomes inferable from history (governs the strength of (C3) as a function of episode length H).
- $\widehat{\delta}_{\text{eff}} \geq 0$: occupancy-weighted effective divergence between P_i and P_j along a candidate shared optimum $\hat{\pi}^\star$ (test (T1) of Appendix O.8.1 produces $\hat{\pi}^\star$ from a short per-client policy-improvement

probe). This quantity is the empirical detection signature for (C1) and does not appear as a term in the Theorem 5 bound (cf. Appendix O.8.2).

Pattern B is the limit $(\eta_{\text{cls}}, H\eta_{\text{ker}}, u^\star/H, \widehat{\delta}_{\text{eff}}) \to (0,0,0,0)$ along the relevant axes. Pattern D is the regime where the slacks are large enough that the (C3) bound becomes vacuous and, on adversarial constructions, the gap reaches the order $R_{\max}H\delta$ of Theorem 4. Pattern C corresponds to intermediate values. Varying a single experimental quantity, such as reducing how much the observations reveal about the environment while holding task and reward fixed, moves the outcome continuously from B through C to D. On the bandit of Figure 7(d), this quantity is the banner's error rate η_{cls}, and the worst-client gap $\eta_{\text{cls}}R_{\max}$ grows continuously from 0 at $\eta_{\text{cls}} = 0$ to $R_{\max}/2$ at $\eta_{\text{cls}} = \frac{1}{2}$. This spectrum is visible in Figure 6. Catalog Split and Lookalike Injection lie near the B end structurally, because (C2) holds for both: PPO attains the B-end plateau on Catalog Split, while the nonzero ϵ_{opt} leaves Lookalike Injection, and Catalog Split under GRPO, in Pattern C. Field-Subset Index and BM25 Reweighting fall in the C interior, because (C1) fails and (C2) holds only partially. Only Rank Wrapper lies at the D end, under both optimizers, because it is the only variant that breaks all of (C1)–(C3).

O.5 Pattern D Mechanism: Decomposing Oscillation and Forgetting

Pattern D can show more than a stable biased plateau: oscillation across rounds, divergence across random seeds, and catastrophic forgetting of pre-trained world knowledge. Theorem 4's $\Omega(R_{\max}H\delta)$ floor establishes structural per-client suboptimality at the federated stationary point but does not by itself predict oscillation or forgetting. The full set of possible Pattern D behaviors decomposes into three mechanisms that can act together:

(D-i) **Quality bias.** The federated optimum $\theta^\star_{\text{fed}}$ is structurally suboptimal for at least one client because no single θ can simultaneously encode mutually contradictory transition kernels in Π_Θ. On the construction of Theorem 4, this bias is the $\Omega(R_{\max}H\delta)$ floor, and it holds for any optimizer.

(D-ii) **Oscillating local updates.** At interior federated stationary points such as θ_{sym}, individual client gradients can be large (each client locally pushes θ toward its own environment's optimum), and their directions conflict. After averaging, the net gradient is approximately zero (stationary in expectation), but along the local-SGD trajectory client gradients update the parameter in conflicting directions. Multi-step local SGD ($E > 1$) amplifies these conflicting updates and can produce visible round-to-round oscillation in the global curve. Such oscillation can also make different seeds diverge. Neither of our two Pattern D runs shows such oscillation (see below).

(D-iii) **Prior degradation.** We hypothesize that when the individual gradient magnitude at $\theta^\star_{\text{fed}}$ exceeds the effective regularization that maintains the pre-trained world model in θ, repeated averaging of conflicting gradients can overwrite the pre-training prior. Capabilities the base LLM originally possessed (factual recall, multi-turn coherence, instruction following on out-of-distribution prompts) would then degrade. Such degradation would appear as catastrophic forgetting in addition to the structural plateau bias.

Pattern D always includes (D-i), structural quality bias at the federated optimum, and may add (D-ii), oscillating local updates around it, and (D-iii), prior degradation when those updates are large enough to overwrite the LLM's pre-trained representation. In our experiments, Pattern D (Rank Wrapper, Figure 6) appears as a low plateau under both optimizers. Under GRPO, the curve never exceeds 13%, and its checkpoint-to-checkpoint changes are the smallest of the twelve curves in that figure. A flat plateau of this kind is consistent with the (D-i) bias or with a large optimizer slack. PPO raises Rank Wrapper to $\sim$20%, still the lowest of the six PPO curves, so the variant stays in Pattern D. Under PPO, its checkpoint-to-checkpoint changes (3.1 points on average) are no larger than those of any other PPO curve. Neither curve shows (D-ii) oscillation, and our runs do not measure (D-iii). Because the GRPO curve does not oscillate, PPO's gain cannot come from suppressing (D-ii). The gain may reflect a smaller optimizer slack, and the runs do not isolate which PPO-specific component (the learned critic with GAE, value-loss clipping, or the 64-task-by-1-rollout batch) is responsible. No optimizer change can remove (D-i).

O.6 ζ^2 Connects the Mechanism to Optimization: Rate vs. Quality

The gradient-heterogeneity quantity ζ^2 formalized in Appendix N.5 unifies all four patterns at the optimization level by distinguishing the *rate of convergence to a stationary point* from *stationary-point quality*:

- Pattern A: $\zeta^2 \leq 2L_{\text{sm}}(\epsilon_{\text{approx}} + \epsilon_{\text{opt}})$ at near-optimal federated solutions (Equation (2)), and the per-client gap vanishes with the same slacks. The plateau matches the near-uniform (i.i.d.-style) baseline.
- Pattern B: $\zeta^2 \leq 2L_{\text{sm}}\epsilon_{\text{opt}}$ at near-optimal solutions, with $\zeta^2(\theta^\star) = 0$ at an interior common optimum, in Π_Θ under (C1)/(C2) or in Π_Θ^{aug} under (C3) with zero slacks. The mechanism differs from that of Pattern A (shared-optimum existence vs. mixture-equivalence), but the optimization-level signature is the same.
- Pattern C: ζ^2 at near-optimal solutions is small but positive and is bounded by $2L_{\text{sm}}(\Delta_{\text{pol}} + \epsilon_{\text{opt}})$, where $(\eta_{\text{cls}}, \eta_{\text{ker}}, u^\star)$ control Δ_{pol} under partial (C3). Under partial (C2), Δ_{pol} is positive because the clients' optimal action sets share no action on some reachable states, and our theorems give no bound for this case. The quantity $\widehat{\delta}_{\text{eff}}$ serves as the empirical signature of residual (C1) failure and is not a term in the bound. Per-client plateau quality is biased by an amount bounded by these slacks, well below the worst-case floor.
- Pattern D: on adversarial constructions, worst-client quality at stationarity is structurally suboptimal by $\Omega(R_{\max}H\delta)$, an *objective-level* bound (Theorem 4). At the gradient level, the federated stationary set of the worst-case construction contains points with $\zeta^2 = R_{\max}^2 H^2 \delta^2/8$, attained at the symmetric point θ_{sym}. However, $\zeta^2(\theta_t)$ itself may decay along the trajectory as the softmax saturates. We deliberately claim no universal trajectory-level ζ^2 floor (Appendix N.5).

Under the smoothness and step-size assumptions of the rate analyses quoted in Appendix N.5, all four patterns exhibit standard convergence *rates* to the federated stationary point. The differences across patterns that our theorems capture are at the level of *solution quality*. In Pattern D, the optimizer reaches a stationary point, but that point has poor per-client quality (the federated stationary point $\neq$ per-client optima). Table 6 of Appendix N.5 consolidates this rate-vs-quality separation across the three regimes. The present subsection augments that table with the explicit Pattern A/B/C/D mapping.

O.7 Diagnostic Decision Tree

The patterns yield an actionable diagnostic that maps an observed training-curve shape to the underlying mechanism and the appropriate mitigation. Given a federated training curve and access to baselines, the procedure is:

Step 1. Identify the heterogeneity level. If clients differ only in $\mathcal{D}_{\tau_i}$ (task distribution) with shared $\mathcal{M}_{\text{env}}$, the regime is task-level: proceed to Step 2 with Pattern A as the prediction. If clients differ in P_i, the regime is environment-level: proceed to Step 3.

Step 2. Verify Pattern A. Compare the federated plateau to the near-uniform baseline.

- If the plateaus match within run-to-run noise, Pattern A is confirmed. No mitigation is required. Given (R′) and a shared environment, robustness holds for any task split (Theorem 2).
- If the plateaus diverge, either one of Theorem 2's premises fails in practice or the split raises the optimizer slack ϵ_{opt}. To diagnose the divergence, check that the task-conditional optimal policy is realizable on the actual Π_Θ, as (R′) requires (a frequent cause is insufficient instruction-tuning coverage), and then check the local optimizer for systematic bias (*e.g.*, a per-client KL constraint that saturates differently across heterogeneity levels).

Step 3. Identify the position on the B/C/D spectrum. Compare the federated plateau to the best single-environment baseline.

- If the plateaus match within run-to-run noise, the run exhibits **Pattern B**. At least one of (C1)/(C2)/(C3) likely holds, unless a recovery mechanism outside them is at work (Appendix O.8.4). Identify which condition holds:

  - Ablate the environment identity from the observation history (*e.g.*, remove brand strings, normalize tool message templates). If the plateau collapses, (C3) was responsible.
  - Hold the context window fixed and vary how much the clients' optimal-action sets overlap. If the plateau collapses when the overlap is removed, (C2) was responsible. A small $\widehat{\delta}_{\text{eff}}$ in (T1) instead points to (C1).
- If the plateau is lower than the baseline by a finite stable margin and at least one (C) test of Appendix O.8.1 passes fully or approximately (*e.g.*, (T2) finds a common optimal action on most but not all reachable states), the run exhibits **Pattern C**. Either the relevant (C) holds partially, or a condition holds fully but the optimizer leaves a nonzero ϵ_{opt}. Identify the dominant slack from the curve shape:
  - If (T3) identifies the environment only after a long history prefix, $u^\star$ is the dominant slack. Mitigation: inject an environment descriptor early in the prompt so $u^\star \to 0$, recovering Pattern B.
  - If (T2) finds a common optimal action on most but not all reachable states, or if plateau bias grows roughly linearly with the on-support divergence $\widehat{\delta}_{\text{eff}}$ of (T1), partial (C2) or the residual (C1) failure signaled by $\widehat{\delta}_{\text{eff}}$ is dominant (a shared optimum exists only approximately). Mitigation: apply per-client fine-tuning or a LoRA adapter to the worst clients.
  - If plateau bias is non-monotone or seed-dependent, η_{cls} is dominant (history is partially informative but classifier slack is large). Mitigation: enlarge the observation context window or add an explicit environment tag.
  - If the plateau recovers under a different local optimizer (*e.g.*, a short PPO probe) while at least one of the tests (T1)–(T3) of Appendix O.8.1 passes, the gap comes from the optimization-level slack ϵ_{opt}. Mitigation: switch to the optimizer that recovered the plateau, or tune the number of local steps and the learning rate. Both of our optimizers already clip their local updates.
- If all (C) tests fail and the plateau stays far below the baseline, possibly with oscillation or capability degradation, the run exhibits **Pattern D**, the outcome expected when all of (C1)/(C2)/(C3) fail on an adversarial construction like that of Theorem 4. Optimizer instability can produce a similar curve, so confirm with the federated probe (F) of Appendix O.8.4 before concluding. Plain FedAvg with a single global θ is structurally inadequate. Mitigation: switch to per-client personalization (*e.g.*, per-client LoRA on a frozen global base) or hierarchical aggregation (cluster clients by environment, aggregate within clusters). A clipped local update is expected to mitigate (D-ii) and (D-iii) but does not change the (D-i) structural floor. Both of our optimizers already clip their local updates, and our runs include PPO but do not isolate its effect on (D-ii) or (D-iii).

The decision tree converts the deployment question "is FEDAGENT working for my deployment?" into the observable question "which of Patterns A–D does my training curve show?", and points to the structural change predicted to recover robustness when the answer is not Pattern A. This tree is the operational form of the Asymmetric Robustness Mechanism's claim that the patterns serve as a diagnostic tool.

O.8 Operational Mitigation Protocol

The diagnostic decision tree of Appendix O.7 maps an *observed* training curve to the underlying mechanism and its mitigation. It is post-hoc (run an experiment first, then determine the regime). A practitioner planning a federated deployment also needs the converse: a way to decide, *before* spending compute on a full run, which of the three sufficient conditions of §4.3 holds (or holds approximately), what bound the per-client gap satisfies under the measured slacks, and which mitigation to apply. This subsection turns the Recovery Theorem (Theorem 5) and the worst-case bound (Theorem 4) into such a protocol: a sequence of pre-deployment tests for (C1)/(C2)/(C3), a closed-form bound parameterized by the measured slacks, a set of mitigations indexed by the dominant slack, and an empirical fallback when none of the tests succeed. Figure 22 summarizes the protocol as a decision tree.

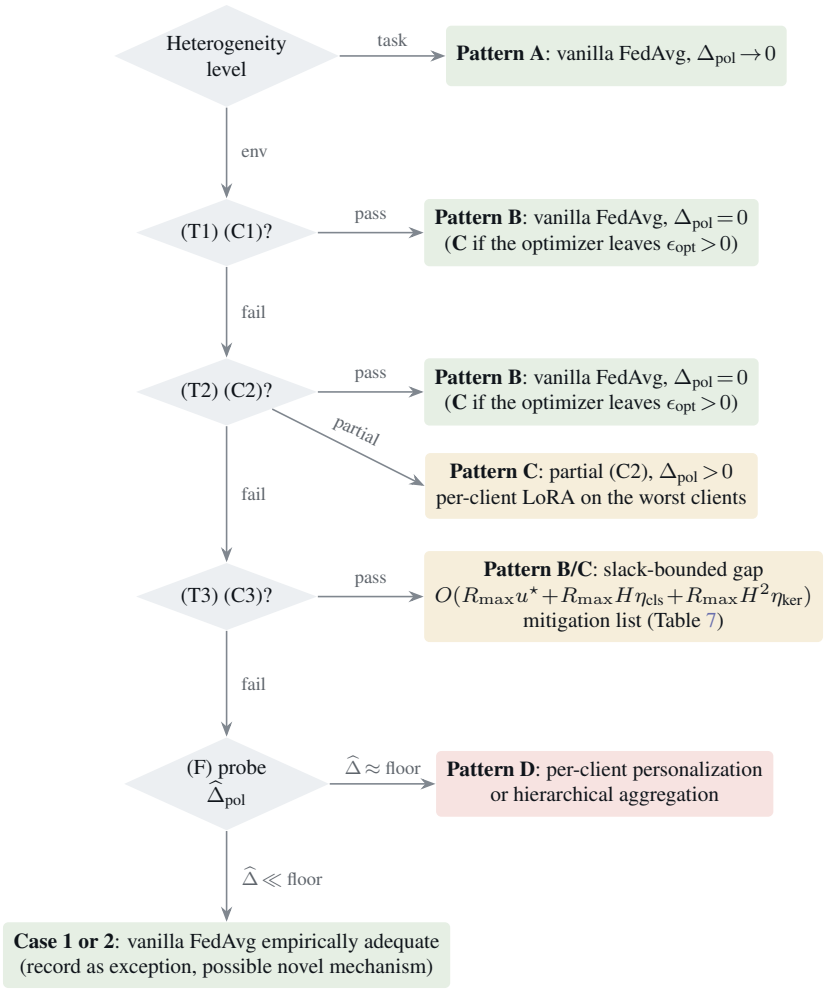


Figure 22: **Mitigation decision tree for federated agent deployments.** Diamonds are tests defined in Appendix O.8.1 (T1, T2, T3) and Appendix O.8.4 (F). Rectangles are outcomes paired with the structural mitigation. Δ_{pol} is the policy-disagreement gap of Theorem 4, and $\widehat{\Delta}$ is its estimate from the probe. In the yellow (T3) box, $u^\star$ is the step at which the environment becomes identifiable, η_{cls} is the error of identifying it, and η_{ker} is the error of the per-client transition models (Theorem 5). Tests are evaluated sequentially top-to-bottom on a small probe rollout. At the first passing test, the procedure exits to the right with a vanilla-FedAvg or slack-bounded prescription. A passing (T1) or (T2) yields Pattern B only if the optimizer attains the shared optimum, and a nonzero ϵ_{opt} leaves the run in Pattern C. A partial pass of (T2), with a common optimal action on most but not all reachable states, exits to Pattern C. The empirical probe (F) at the bottom resolves the case in which (C1)–(C3) all fail into either Pattern D (personalization required, red) or Case 1 or 2 of Appendix O.8.4 (vanilla FedAvg empirically adequate, green). "Floor" here refers to $R_{\max} H \widehat{\delta}_{\text{eff}}$, the on-support proxy for the $R_{\max} H \delta$ scale of Theorem 4. Green indicates $\Delta_{\text{pol}} = 0$ or $\to 0$. Yellow indicates a bounded but nonzero gap. Red indicates that the structural floor applies.

O.8.1 Pre-Deployment Tests for (C1), (C2), (C3)

Each of the three sufficient conditions admits a measurable proxy that can be evaluated from a small per-client probe rollout, well before launching a full federated training run:

(T1) **Test for (C1) (common-optimal-off-support).** Run a short per-client policy-improvement loop (a few PPO epochs on each $\mathcal{M}_i$ from the same shared base θ_0) to obtain $\hat{\pi}_i^\star$ for every client. Check (a) whether $\hat{\pi}_i^\star \approx \hat{\pi}_j^\star$ on rollout-visited states (an approximate-shared-optimal

check), and (b) the occupancy-weighted divergence $\widehat{\delta}_{\text{eff}}(\hat{\pi}^\star; i, j) := \widehat{\mathbb{E}}_{(s,a)\sim d^{\hat{\pi}^\star}_{\mathcal{M}_i}}\left[D_{\text{TV}}(P_i(\cdot \mid s, a), P_j(\cdot \mid s, a))\right]$ along the candidate $\hat{\pi}^\star := \hat{\pi}^\star_k$ of the client k with the highest probe value. If (a) holds and (b) is below the run-to-run noise floor for every (i, j) pair, (C1) is empirically supported. Because the probe covers only rollout-visited states, this evidence is a finite-sample proxy rather than a certificate (cf. Appendix O.8.4).

(T2) **Test for (C2) (action-preserving with shared $\pi^\star$).** Run the same per-client probe as in (T1) and compute $\arg\max_a \hat{Q}^\star_i(s, \tau, a)$ at every reachable state. Condition (C2) is satisfied when these argmax sets share at least one action at every state of the shared support, even if $\hat{V}^\star_i \neq \hat{V}^\star_j$. The signature distinguishing (C2) from (C1) is that $\widehat{\delta}_{\text{eff}}$ is allowed to be *nonzero*: kernels can disagree as long as the optimal action ranking is preserved.

(T3) **Test for (C3) (self-revealing-environment).** Train a lightweight environment-identity classifier $\hat{\imath} : \mathcal{H} \to [N]$ on observation prefixes h_u (we expect a logistic regression on bag-of-tokens or a small transformer head to suffice). Vary $u = 1, 2, \ldots, H$ and record the smallest $u^\star$ at which the classifier accuracy on a held-out probe set saturates. The residual error rate at $u \geq u^\star$ is the empirical $\hat{\eta}_{\text{cls}}$. For the kernel slack $\hat{\eta}_{\text{ker}}$, fit a one-step transition predictor $\hat{P}_{\hat{\imath}(h_u)}(\cdot \mid s, a)$ on probe rollouts and report its TV distance to per-client kernels (or set $\hat{\eta}_{\text{ker}} = 0$ when the post-identification behavior is trained on the identified environment itself, so that the behavior targets the true kernel). In environments without a within-episode way back to the initial state, additionally probe the recoverability condition (C3-rec) of Appendix F.5: roll out the default prefix policy for $u^\star$ steps and check that post-prefix success rates are not systematically depressed relative to fresh starts.

The tests are independent: a deployment may satisfy (C1) and (C3) simultaneously, (C3) alone, or another combination of conditions. The measured degree of satisfaction enters directly into the bound of Appendix O.8.2.

O.8.2 The Quantitative Policy-Disagreement Gap Bound

The Recovery Theorem (Theorem 5, restated and proved in Appendix N.4) gives an explicit bound on the policy-disagreement gap, parameterized by the measured slacks:

- **Under (C1) or (C2):** $\Delta_{\text{pol}} = 0$. The federated optimum coincides with the per-client optimum, so no structural mitigation is needed.
- **Under (C3) with $(u^\star, \eta_{\text{cls}}, \eta_{\text{ker}})$ slacks:**

$$\Delta_{\text{pol}} \;\leq\; \underbrace{c_0 R_{\max}\, u^\star}_{\text{prefix regret}} + \underbrace{R_{\max}\, H\, \eta_{\text{cls}}}_{\text{misclassification regret}} + \underbrace{2 R_{\max}\, H^2\, \eta_{\text{ker}}}_{\text{kernel-modeling regret}}.$$

  The three terms have distinct origins: the trajectory prefix, a trajectory-level event, and per-step compounding. Their relative magnitudes determine which slack dominates the residual gap. The prefix term assumes the recoverability condition (C3-rec) of Appendix F.5, which holds when a back or home action restores the initial state within the episode. Where the default prefix policy can drive the agent into irrecoverable states, the prefix term grows toward $O(R_{\max} H)$ and the (C3-rec) probe of (T3) should be run alongside the classifier test.
- **When no condition holds (worst case):** Theorem 4 constructs instances on which the gap equals $\Delta_{\text{pol}} = R_{\max} H \delta / 2$, where $\delta = \sup_{i,j,s,a} D_{\text{TV}}(P_i(\cdot \mid s, a), P_j(\cdot \mid s, a))$ is the unrestricted (occupancy-unweighted) per-client divergence, and the tightness remark after the proof of Theorem 4 gives constructions with gaps of order $R_{\max} H \min\{1, H\delta\}$. Both results are existence statements: the failure of (C1)–(C3) does not by itself imply such a gap. If every policy reaches the disagreement region with probability at most ε, for instance, the gap is at most $2\varepsilon R_{\max} H$. On the constructions where the gap does occur, it is irreducible and independent of optimizer, capacity, or training budget.

The interpolation between the (C3) upper bound and the worst-case lower bound forms the B/C/D spectrum of Appendix O.4. As $(u^\star, \eta_{\text{cls}}, \eta_{\text{ker}})$ shrink toward zero, the upper bound reduces to the zero gap of the (C1)/(C2) regime (Pattern B). As these slacks grow, the upper bound becomes vacuous, and on adversarial constructions the gap can reach the order of Theorem 4 (Pattern D). Intermediate values correspond to Pattern C.

A note on $\widehat{\delta}_{\text{eff}}$. The occupancy-weighted divergence $\widehat{\delta}_{\text{eff}}(\hat{\pi}^\star; i, j)$ measured in (T1) does not appear as an additive term in the (C3) bound, because the bound is stated under the (R_{aug}) assumption that Π_Θ^{aug} realizes the environment-conditional policies of (C3). Instead, $\widehat{\delta}_{\text{eff}}$ functions as the practical detection signature for (C1): when it vanishes along a candidate $\hat{\pi}^\star$ that also passes (T1)'s shared-optimum check (a), (C1) is supported and the gap is zero. Check (b) alone is *not* sufficient (cf. the discussion after the (C1) case of the proof in Appendix N.4). When $\widehat{\delta}_{\text{eff}}$ is moderate but the (C3) classifier $\hat{\imath}$ is accurate, the deployment is in the (C3) approximate regime and the displayed bound applies.

O.8.3 Mitigations Indexed by the Dominant Slack

The four operational slacks $(u^\star, \eta_{\text{cls}}, \eta_{\text{ker}}, \widehat{\delta}_{\text{eff}})$ each admit a structurally distinct mitigation. Table 7 lists these mitigations. Each entry is justified by the bound term or test it targets.

Dominant slack	Empirical signature	Mitigation
$u^\star$ (reveal step)	(T3) identifies the environment only after a long history prefix	Inject an environment descriptor early in the prompt (*e.g.*, a one-line environment tag prepended to the system prompt). The classifier $\hat{\imath}$ becomes trivial, $u^\star \to 0$, and the prefix regret term vanishes, recovering Pattern B.
η_{cls} (classifier slack)	Plateau bias non-monotone in the heterogeneity level, with a seed-dependent run-to-run gap	Enlarge the observation context window so $\hat{\imath}$ has access to a longer history. Alternatively, add an explicit environment tag in the prompt. The misclassification term shrinks linearly in η_{cls}.
η_{ker} (kernel slack)	Plateau bias scales with H^2 (long-horizon tasks degrade faster than short ones)	Normalize the action set across environments (*e.g.*, wrap each environment's API in a shared interface). Alternatively, use environment-specific tokenization so the policy does not need to model environment-conditional kernels in θ. The H^2 term is the most costly on long-horizon agent tasks.
$\widehat{\delta}_{\text{eff}}$ (residual divergence)	Plateau bias grows roughly linearly with $\widehat{\delta}_{\text{eff}}$, and the (C1) test fails while the (C3) test succeeds	Per-client LoRA adapter on the worst clients (keep the global θ shared, attach a small per-client correction). Alternatively, use hierarchical aggregation that clusters clients by similarity in $\widehat{\delta}_{\text{eff}}$ and aggregates within clusters.
All slacks large *(Pattern D)*	Plateau collapse, oscillation across rounds, divergence across seeds, capability degradation	Replace plain FedAvg with full per-client personalization (*e.g.*, per-client LoRA on a frozen global base, pFedMe, Ditto). Alternatively, use hierarchical FedAvg with environment-clustered groups. Clipped local updates are expected to mitigate (D-ii) and (D-iii) but do not change the (D-i) structural floor. Our runs include PPO but do not isolate its effect on (D-ii) or (D-iii).

Table 7: Mitigations indexed by the dominant slack. The signature column is what the practitioner observes on a probe run. The mitigation column is the structural change that targets the dominant slack (Appendix O.8.2). Multiple slacks can be dominant simultaneously, in which case the mitigations compose.

These mitigations are structural: they identify which axis of the deployment to change ($u^\star$ via prompt, η_{cls} via context window, η_{ker} via API wrapping, $\widehat{\delta}_{\text{eff}}$ via per-client correction) and leave the choice of optimizer and learning-rate schedule open. The optimizer choice (GRPO vs. PPO, FedAvg vs. SCAFFOLD) sets ϵ_{opt}, so it is the mitigation when the (C) tests pass, and it can affect (D-ii) and (D-iii).

O.8.4 The Uncertain Regime: When All (C) Tests Fail but Collapse Does Not Occur

The Recovery Theorem provides sufficient but not necessary conditions: a deployment in which (T1)/(T2)/(T3) all fail might *still* be empirically robust (no Pattern D collapse on the actual training curve). Two cases account for this possibility, and the practitioner protocol depends on which one applies:

- **Case 1: a recovery condition holds that the tests do not detect.** Conditions (C1)/(C2)/(C3) capture the most common LLM-agent recovery mechanisms but do not exhaust the space of conditions under which the worst-case lower bound no longer applies. Case 1 can arise in two ways: a different structural mechanism may apply, or a condition may hold while its finite-sample test fails. For example, heterogeneity confined to transitions that the optimal policy never visits satisfies (C1), yet (T1) can still fail if the probe policies visit that disagreement region. In both cases, Theorem 4's worst-case construction does not apply and FedAvg can be robust despite all three tests failing.
- **Case 2: the worst-case construction has not been encountered yet.** Theorem 4's lower bound is realized by a specific transition-swap construction. Deployments may satisfy the abstract assumptions of Theorem 4 without containing a perturbation of the construction's form. In this case Pattern D has not yet been triggered: it does not appear on the current training curve but can be elicited by adversarial environment constructions of the kind in Appendix M.

Empirical fallback protocol. When all three tests of Appendix O.8.1 fail, run a short federated probe (a few rounds at full federation but with a small token budget) and measure the policy-disagreement gap $\widehat{\Delta}_{\text{pol}}$ (or the worst-client gap, which upper-bounds it). Compare the measurement with the scale $R_{\max} H \widehat{\delta}_{\text{eff}}$, where $\widehat{\delta}_{\text{eff}}$ is the on-support divergence measured in (T1). The unrestricted δ of Definition 2 cannot be estimated from rollouts, and Theorem 4 shows only that gaps of this order are possible on some constructions:

(F1) If $\widehat{\Delta}_{\text{pol}}$ is within a small constant factor of $R_{\max} H \widehat{\delta}_{\text{eff}}$, the deployment behaves like the worst-case construction (Pattern D). The deployment is therefore not in Case 2. Switch to per-client personalization or hierarchical aggregation as recommended in Table 7.

(F2) If $\widehat{\Delta}_{\text{pol}}$ is substantially below this scale, a recovery mechanism holds that (T1)–(T3) did not detect (Case 1), or the deployment lacks a perturbation of the worst-case form (Case 2). Vanilla FedAvg is empirically adequate, but the situation is worth recording (and ideally investigating) because it may extend the framework with a new sufficient condition. Before concluding, conservatively re-run (T3) with a longer prefix or a larger classifier to rule out an overestimate of $\hat{\eta}_{\text{cls}}$ (*i.e.*, that (C3) actually holds but the probe classifier was too weak to certify it).

The fallback completes the protocol by assigning each deployment one of three outcomes: a (C1)/(C2)/(C3) certification with a quantitative bound, a Pattern D personalization recommendation, or a documented Case 1 or Case 2 exception. These three outcomes make the mechanism usable as a deployment diagnostic.

O.9 Predictive and Falsifiable Use of the Mechanism

This subsection synthesizes Appendices O.1 through O.8 into a single statement of what the Asymmetric Robustness Mechanism predicts, what observation would falsify it, and how the mechanism is meant to be used in practice. The central claim of this appendix is that the empirical patterns of federated agent training follow from I-D Asymmetry under varying degrees of (C1)/(C2)/(C3) satisfaction:

- **Task-level heterogeneity is the benign side of I-D Asymmetry.** Because τ enters the input, a single θ can encode $\tau \mapsto b(\tau)$ (under (R′)), and Pattern A (federated curve $\approx$ near-uniform baseline) is the expected consequence of mixture-equivalence. Pattern A is unconditional at the solution level, in the sense that no choice of per-client task split can create a structural floor. Attaining the plateau still requires an optimizer slack ϵ_{opt} that the split does not raise.
- **Environment-level heterogeneity is the pathological side of I-D Asymmetry.** Because P_i does not enter the input, federated robustness is conditional on whether a single θ can simultaneously be

optimal across all $\mathcal{M}_i$. The three sufficient conditions (C1)/(C2)/(C3) provide three structurally distinct ways for such a shared optimum to exist. The slacks $(\eta_{\text{cls}}, \eta_{\text{ker}}, u^\star)$ measure how closely (C3) is satisfied, and $\widehat{\delta}_{\text{eff}}$ signals residual (C1) failure. Partial (C2), as in Field-Subset Index and BM25 Reweighting, adds a gap that these slacks do not measure. The position on the Pattern B/C/D spectrum is determined by these quantities together with the optimizer's optimization-level slack ϵ_{opt} (§4.4).

O.9.1 Connecting Theory and Experiment

The mechanism links theory and experiment in both directions. A reader who observes a particular training-curve shape can follow it through Appendix O.7 back to the specific theorem and condition responsible for the shape:

- A curve that matches the near-uniform baseline under task-level heterogeneity indicates Pattern A, which follows from the mixture-equivalence identity (Theorem 1(i)) behind Theorem 2.
- A curve that matches the single-environment baseline under environment-level heterogeneity indicates Pattern B, which follows from the shared optimum of Theorem 5. Which (C) holds can be further identified via the ablation prescribed in Step 3 of Appendix O.7.
- A stable plateau biased below the single-environment baseline indicates Pattern C: either a (C) condition holds partially (nonzero $(\eta_{\text{cls}}, \eta_{\text{ker}}, u^\star)$ for (C3), partial (C2) with a common optimal action on only some reachable states, or residual (C1) failure signaled empirically by $\widehat{\delta}_{\text{eff}}$), or (C) holds fully with a nonzero optimizer slack ϵ_{opt}. The dominant slack determines which structural mitigation applies (Table 7).
- A plateau far below the baseline, with all (C) tests failing and the (F) probe near the floor, indicates Pattern D, which matches the $\Omega(R_{\max} H \delta)$ floor of Theorem 4 on adversarial constructions. Mitigation requires replacing vanilla FedAvg with per-client personalization or hierarchical aggregation.

Conversely, a practitioner who has measured (T1)/(T2)/(T3) on a probe deployment can predict which pattern the full training curve will exhibit, before spending the compute, via the mitigation protocol of Appendix O.8. The diagnostic and operational protocols make this correspondence explicit at the level of curve shapes rather than theorem statements.

Figure 23 places all 22 comparisons of §5 and Appendix G on one axis, as the mean gap to the baseline over the last ten checkpoints. The twelve task-level gaps fall on both sides of zero, between -10.8 and $+9.5$ points, and for ten of them the checkpoint-level range contains zero. Nine of the ten environment-level gaps lie below every task-level gap, between -14.9 and -48.9 points, and their ranges never reach zero. These nine runs are the ones assigned Patterns C and D. The remaining run, Catalog Split under PPO (-4.3), falls among the task-level gaps, and it is the one run assigned Pattern B. Each curve is a single run, so the figure shows how the two regimes separate and does not test each gap for significance.

O.9.2 Falsifiability per Pattern

Patterns A–C are each paired with an observation that would contradict the corresponding prediction, and Pattern D is paired with an observation that would show that (C1)–(C3) are not exhaustive. Because the theorems are proved, such an observation would show that one of their premises fails in the deployment (for instance a realizability assumption or a measured slack), while the theorems themselves remain true. We list these observations explicitly so that future work can test the framework's premises against the following observations:

- **Pattern A (premises of Theorem 2).** The contradicting observation is a task split $\{\mathcal{D}_{\tau_i}\}$ whose mixture matches the near-uniform baseline and whose federated training plateau differs from that baseline's plateau by more than algorithmic noise, while (R′) holds on the actual Π_Θ and the split does not raise ϵ_{opt}.
- **Pattern B (premises of Theorem 5).** The contradicting observation is a deployment in which (T1) or (T2) or (T3) is verifiably satisfied (*e.g.*, a small environment-classifier reaches near-zero error on early observation prefixes) but the federated plateau nevertheless falls substantially below the single-environment baseline, with no algorithmic explanation.

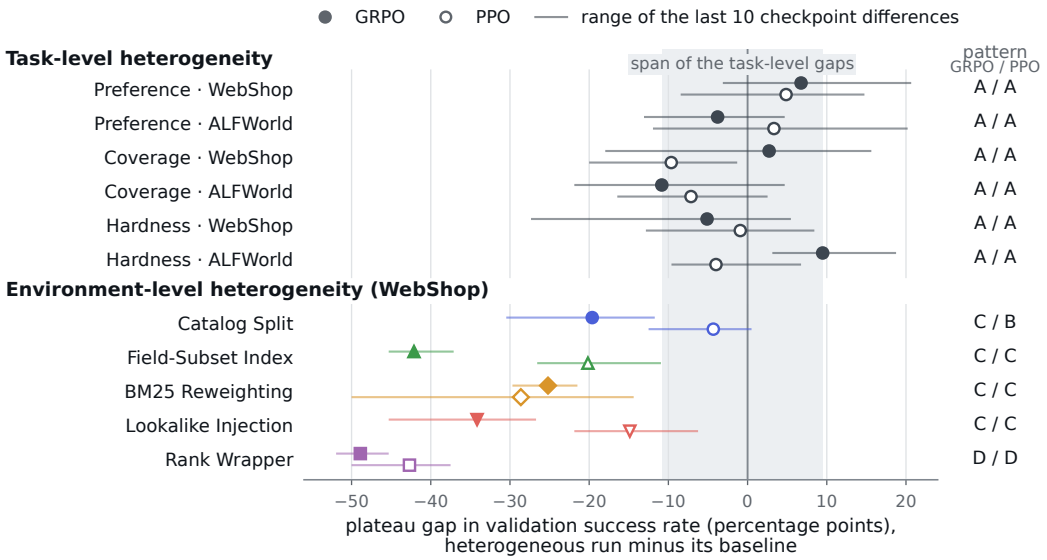


Figure 23: **Task-level gaps scatter around zero, while environment-level gaps do not.** Each marker is the mean, over the last 10 checkpoints (epochs 183–210), of the paired difference between a heterogeneous run and its baseline. The baseline is the near-uniform split for the task-level sub-types (Figures 5 and 9) and the single-environment Uniform run for the WebShop variants (Figure 6). Filled markers are GRPO and open markers PPO, with the colors and markers of Figure 6. Lines span the 10 checkpoint differences, and the shaded band spans the twelve task-level means. The right column gives the pattern assigned to each run in §5 and Appendix G.3.

- **Pattern C (slack bound of Theorem 5).** The contradicting observation is a deployment in which the measured slacks $(u^\star, \eta_{\text{cls}}, \eta_{\text{ker}})$ are small but the gap $\widehat{\Delta}_{\text{pol}}$ exceeds $c_0 R_{\max} u^\star + R_{\max} H \eta_{\text{cls}} + 2R_{\max} H^2 \eta_{\text{ker}}$. Since the bound cannot be exceeded when its premises hold, such an observation would show that the slacks were mismeasured or that (R_{aug}) fails.
- **Pattern D (non-exhaustiveness of (C1)–(C3)).** The informative observation is a deployment in which all of (T1)/(T2)/(T3) fail and $\delta = \sup_{i,j,s,a} D_{\text{TV}}(P_i, P_j)$ is large, yet no plateau collapse occurs and the gap remains far below $R_{\max} H \widehat{\delta}_{\text{eff}}$. Such an observation would reveal a recovery mechanism outside (C1)/(C2)/(C3) (Appendix O.8.4, Case 1), which the framework allows, since the conditions are sufficient but not necessary. Such an observation would *not* contradict Theorem 4, an existence statement about its own construction class: reproducing that construction with a silent policy class must yield $\Delta_{\text{pol}} = R_{\max} H \delta / 2$.

The difference in falsifiability across patterns is itself informative: Patterns A and B are falsified by a single counterexample (one split or deployment with a divergent plateau), Pattern C's falsification requires a calibrated comparison between measured slacks and the predicted bound, and the Pattern D observation tests only whether (C1)–(C3) are exhaustive. This contrast reflects the structural difference between the strict equalities behind Patterns A and B (mixture-equivalence and $\Delta_{\text{pol}} = 0$) and the slack bound of the Recovery Theorem (a quantitative inequality with explicit constants).

O.9.3 Predictive Use Given Deployment Characteristics

Given the structural characteristics of a planned deployment, the mechanism predicts which pattern the federated training curve will exhibit before any training is done. The relevant deployment characteristics are:

(1) **Heterogeneity type**: do clients differ in $\mathcal{D}_{\tau_i}$, in P_i, or in both? If clients differ only in $\mathcal{D}_{\tau_i}$, the prediction is Pattern A. If clients differ in P_i, proceed to (2).

(2) **How much the observations reveal**: are environment-identity-revealing tokens (URLs, branding, error templates, tool message formats) present in the early observation prefix h_u for

$u = O(1)$? If yes, (T3) is likely to be satisfied with $u^\star = O(1)$ and small η_{cls}, predicting Pattern B if the optimizer attains the shared optimum and Pattern C otherwise.

(3) **Optimal-action overlap**: do the optimal action sets $\arg\max_a Q_i^\star(s, \tau, a)$ share an action across clients on shared support? Affirmative answers (action-preserving across environments) predict that (C2) holds, independent of (C3). The run then reaches Pattern B if the optimizer attains the shared optimum and stays in Pattern C otherwise. We expect this overlap in deployments where environments share an underlying API but differ in latency or error rates.

(4) **Adversarial environment construction**: is the environment perturbation engineered to be observation-silent (no environment identity in observations) and action-flipping (optimal actions differ across environments on states the optimal trajectory visits)? If yes, all of (C1)/(C2)/(C3) fail and the prediction is Pattern D. Among the five WebShop variants of Appendix M, Rank Wrapper is the only variant in this regime (§5.3).

(5) **Episode horizon** H: long horizons amplify the η_{ker} term (H^2 scaling) more than the η_{cls} term (H scaling). Long-horizon deployments with non-trivial kernel slack are predicted to exhibit Pattern C even when (C3) holds, because the H^2 term dominates.

Each prediction comes with a structural reason (which (C) holds or fails, and which slack dominates), so a wrong prediction is itself diagnostic and can be attributed to a misspecified deployment characteristic.

Closing remark. Together, the unified causal chain (Appendix O.1), the per-pattern mechanism analyses (Appendices O.2 through O.5), the optimization-level ζ^2 analysis (Appendix O.6), the post-hoc diagnostic (Appendix O.7), and the pre-deployment mitigation protocol (Appendix O.8) turn a single architectural fact about LLM agents into a deployment-time toolkit: a prediction before training, a post-hoc diagnostic, and a pre-deployment mitigation protocol. The toolkit has not yet been run end to end, and its parts are motivated by our WebShop/ALFWorld results. Every observation along the chain is paired with a falsifiable theoretical claim and a structural mitigation. The theorems apply to any agent with the same relationship between inputs and dynamics (Appendix P). LLM agents satisfy I-D Asymmetry without engineering, because tasks arrive as text in the prompt.

P Why the Mechanism Is Sharper on LLM Agents

The Asymmetric Robustness Mechanism (§4, with full proofs in Appendix N) is stated for federated RL in general: it requires only I-D Asymmetry (§3.1) and the realizability assumptions (R)/(R′)/(R_{env})/(R_{aug})/(LR) of Appendix N.1. Theorems 2 and 4 would apply unchanged to a non-LLM agent with the same relationship between inputs and dynamics. This appendix argues that LLM agents make the mechanism *sharper* in three orthogonal aspects. First, the realizability assumptions on the task-level side are plausible for instruction-tuned LLMs because pre-training can align Π_Θ with task-conditional optima (Appendix P.1). Second, the environment-level side may incur an LLM-specific cost from pre-training prior degradation that is not captured by the formal $\Omega(R_{\max} H\delta)$ floor (Appendix P.2). Third, the (C3) self-revealing-environment condition admits two LLM-specific instantiations: an explicit form captured by environment-aware prompting (with a precise theorem-level reduction in Appendix P.3) and an implicit form realized by in-context posterior inference (Appendix P.4). These three properties explain why we expect the mechanism to be sharpest for LLM agents, even though it is stated at the general MDP level.

P.1 Task-as-Input: Pre-Training as Implicit Realizability

The task-level robustness bound in Theorem 2 includes a capacity slack ϵ_{approx} that controls how closely Π_Θ realizes the task-conditional Bayes-optimal policy. For LLM agents, we expect the pre-training and instruction-tuning pipeline to make this slack small.

Pre-training already trains task-conditional policies. The defining objective of supervised instruction-tuning is to produce a model that, given a prompt τ, generates a completion satisfying τ. Across diverse instructions seen during pre-training, this objective trains an approximation to the task-conditional map $\tau \mapsto b(\tau)$, evaluated under the mixture $\bar{\mathcal{D}}_\tau^{\text{pretrain}}$ that pre-training samples from. Assumption (R′) requires the federated mixture $\bar{\mathcal{D}}_\tau$ to admit a $\theta^\star \in \Theta$ with bounded expected suboptimality on that mixture. Whenever the support of the federated $\bar{\mathcal{D}}_\tau$ lies within that of $\bar{\mathcal{D}}_\tau^{\text{pretrain}}$, the initialization θ_0 can already be near-optimal on the mixture. The initialization's suboptimality $\epsilon_0 := \mathbb{E}_{\bar{\mathcal{D}}_\tau}[\mathcal{J}_\tau^\star - \mathcal{J}_\tau(\theta_0)]$ upper-bounds ϵ_{approx} (take $\theta^\star = \theta_0$ in (R′)), so (R′) then holds with a small slack before any federated update. We expect this containment for downstream applications whose task formats resemble those of instruction-tuning datasets. In that regime the federated optimizer refines an existing capability of Π_Θ rather than creating a new one, and the residual ϵ_{approx} is bounded by how much $\bar{\mathcal{D}}_\tau$ extends beyond the pre-training distribution. Multi-turn agent tasks such as WebShop need not satisfy this containment, and our runs are not in this regime. Under GRPO, θ_0 succeeds on at most 6.2% of validation episodes (Figure 4, epoch 0), and the PPO curves start at 7.8% on WebShop.

Federated task-heterogeneous fine-tuning is expected to extend pre-training. Each client's $\mathcal{D}_{\tau_i}$ is, in this view, simply a sample from the same "follow instructions" target that pre-training already approximates. A federated client's gradient updates (approximately) refine the parameters within the instruction-following manifold of Θ. Pre-training already places θ_0 on this manifold. As a result, we expect the cross-client gradient cosines to be biased toward positive values in the instruction-following subspace. Two clients whose tasks differ in semantic content but share the instruction-following format produce gradients that agree on the dominant direction (improving instruction adherence) while differing on second-order content-specific corrections. The bounded gradient heterogeneity $\zeta^2(\theta)$ that limits generic FedRL convergence (Appendix N.5) should thereby be small near the federated solution on LLM agents (Equation (2)). We do not measure gradient alignment directly, but our task-level results are consistent with this picture. This gradient-alignment effect is an empirically testable prediction complementary to, but not implied by, Theorem 2: the theorem's capacity slack $\epsilon_{\text{approx}} \leq \epsilon_0$ is small when the coverage condition above holds, and the aligned gradients additionally keep the optimization-side cost small.

When $\bar{\mathcal{D}}_\tau$ exceeds the pre-training coverage. This plausibility argument no longer applies when the federated task distribution lies substantially outside $\bar{\mathcal{D}}_\tau^{\text{pretrain}}$. In niche-domain federations (*e.g.*, specialized clinical workflows, low-resource languages, idiosyncratic enterprise tools), $\bar{\mathcal{D}}_\tau$ differs more from the distribution covered by pre-training, and the bound ϵ_0 on ϵ_{approx} in (R′) grows accordingly. The bound of Theorem 2 is still valid, but its effective magnitude on the per-client

gap is no longer dominated by optimization slack. The per-client gap then includes a term that scales with how far the federated mixture deviates from pre-training coverage. We do not vary pre-training coverage in our experiments, but the theoretical bound continues to apply. The practical recommendation is to verify that the initial suboptimality ϵ_0, an upper bound on ϵ_{approx}, is small before launching a federated run, *e.g.*, by evaluating the released checkpoint zero-shot on a probe task pool.

P.2 Environment-as-Implicit-Knowledge: Pre-Training Prior Degradation (Informal Claim)

The environment-level side of I-D Asymmetry parallels the task-level side at the MDP level, but *may be* more costly for LLM agents than for from-scratch RL agents. The argument below is informal: we identify the structural mechanism but do not give a formal lower bound separating the LLM cost from the from-scratch cost. Establishing such a lower bound is left for future work.

LLMs encode an implicit default world model. Pre-training equips the LLM with an implicit "default world model" composed of factual, procedural, and behavioral priors: how shells respond to commands, how websites are structured, how APIs are conventionally typed, what an OS error message looks like. This prior is encoded across the parameters θ_0 and explains why an LLM agent initialized at θ_0 can already act sensibly in many environments without any RL fine-tuning. Concretely, θ_0 differs from the from-scratch initialization assumed in classical FedRL theory. The pre-trained θ_0 encodes substantial environment-level knowledge acquired as a side effect of pre-training.

Conflicting per-client kernels force θ to encode incompatible world models. When clients hold mutually inconsistent transition kernels $\{P_i\}$, federated training requires a single parameter tensor θ to encode environment-conditional dynamics for every $\mathcal{M}_i$ simultaneously. By I-D Asymmetry, the policy has *no* input channel indicating which environment it is currently in. The same prompt at s_0 must produce one action distribution. When the environments require different actions, two natural options are (i) to devote θ to a single world model and incur error in every other environment, and (ii) to average across world models and incur error in all environments. Either choice can degrade the pre-training prior in the regions of θ that encoded it.

Representation-level interference. The degradation above is structurally distinct from the objective-level gap already captured by Theorem 4. In addition to the federated stationary point being suboptimal for at least one client (the formal claim of Theorem 4), the parameters themselves can drift away from the pre-training prior into a region of parameter space that is optimal for no client. This degradation would appear as catastrophic forgetting of capabilities the base model originally possessed (factual recall, instruction following on out-of-distribution prompts, multi-turn coherence). Our runs do not measure such forgetting. This forgetting is the (D-iii) "prior degradation" component of Pattern D in Appendix O.5.

Why we do *not* claim a formal lower bound here. Formally separating the LLM cost from the from-scratch cost would require a quantitative argument comparing the effective ϵ_{approx} of an LLM under conflicting $\{P_i\}$ to the same quantity for a from-scratch agent on the same federation. We do not provide that comparison. Such a comparison would require a careful experimental design that quantifies catastrophic forgetting as ϵ_{approx} degradation at the parameter level. In our view, the existing forgetting literature does not frame the question in those terms. We therefore label the claim as informal: LLM agents incur an *additional* prior-degradation risk under environment-level heterogeneity, beyond the formal $\Omega(R_{\max} H \delta)$ floor. This informal claim motivates the (D-iii) component of Pattern D (Appendix O.5). Our two Pattern D runs, Rank Wrapper under GRPO and PPO, do not test this component. Both curves stay low, and we do not measure capability degradation. A formal LLM-specific lower bound that quantifies prior degradation is a worthwhile target for future work.

Practical implication. The (D-i) quality bias of Appendix O.5 is the formal floor that no optimizer removes. The (D-ii) oscillation and (D-iii) prior degradation components are additional possible behaviors, and (D-iii) is specific to LLM agents. Our runs show no (D-ii) oscillation and do not measure (D-iii) (Appendix O.5). Mitigations targeting (D-i) (per-client personalization, hierarchical aggregation) address the formal floor. Mitigations targeting (D-iii) (limiting per-step parameter movement via clipped local updates, KL regularization to θ_0, parameter-efficient fine-tuning that

freezes the base model) address the LLM-specific prior-degradation component. The "all slacks large" row of Table 7 implicitly combines both classes of mitigation. The present subsection clarifies why both are needed for LLM agents specifically.

P.3 (C3) Explicit Form: Environment-Aware Prompting and Corollary 1

A key implication of I-D Asymmetry for LLM agents is that environment identity can be *represented* in the input channel as well as in the dynamics. The explicit way to represent environment identity in the input is a deliberate prompt-engineering operation: prepending an environment descriptor e_i to the agent's prompt. The (C3) condition of §4.3 formally states that this representation is feasible. Condition (C3) admits an explicit instantiation (environment-aware prompting, this subsection) and an implicit one (in-context posterior inference, Appendix P.4).

Environment descriptors represent environment identity in the input. Suppose the federation is augmented with an environment descriptor map $e : [N] \to \mathcal{E}$ such that each client i prepends e_i to its prompt before invoking the policy. Examples of e_i from real LLM-agent deployments include:

- "*You are operating in macOS 14.5 with zsh. The user's home directory is /Users/foo.*"
- "*You are connecting to API version 2.3.1, which uses OAuth 1.0 and returns XML.*"
- "*You are interacting with United Airlines' booking flow, last updated 2024-Q3.*"
- "*You are running on a Linux container with restricted file system access, where only /tmp is writable.*"

With e_i in the input, the policy class becomes the *augmented* class Π_Θ^{aug} of Appendix N.1, $\{\pi_\theta(a \mid s, \tau, e) : \theta \in \Theta\}$, with the descriptor e serving as the history input. The federation then transitions from a Definition 2 (environment-level) instance to a Definition 1 (task-level) instance over augmented tasks (τ, e) with input distribution $\bar{\mathcal{D}}_{(\tau,e)}$, because the descriptor conditions the initial state and selects the kernel (proof of Corollary 1). Theorem 2 then applies on the augmented mixture, with two residual slacks corresponding to the two ways the input representation can be imperfect: a *classifier slack* η_{cls} if the descriptor is sometimes assigned incorrectly (*e.g.*, the classifier $\hat{e}$ reports the wrong OS version on some prefixes) and a *kernel-modeling slack* η_{ker} if the shared kernel model $\hat{P}_e$ matches the true per-client kernels only approximately (*e.g.*, a partial OS version string that several hosts share, condition (i) of Corollary 1).

Corollary 1 (Environment-Aware Prompting Reduces Environment-Level to Task-Level Heterogeneity)**.** *Under Definition 2 with assumption (R_{aug}), suppose there exist an environment descriptor map $e : [N] \to \mathcal{E}$ and a classifier $\hat{e} : \mathcal{H} \to \mathcal{E}$ (the descriptor-valued counterpart of the classifier $\hat{\imath}$ of (T3)) with kernel model $\hat{P}$ such that:*

(i) *For every client i, every policy π, and every step $u \le H$, the trajectory-marginal misclassification rate is bounded (a stronger form of (C3-cls) in Appendix F.5):*

$$\Pr_{h_u \sim \mathcal{M}_i, \pi}\big[\hat{e}(h_u) \neq e_i\big] \le \eta_{cls},$$

and the kernel model satisfies $\sup_{s,a} D_{\text{TV}}(\hat{P}_{e_i}(\cdot \mid s,a), P_i(\cdot \mid s,a)) \le \eta_{ker}$.

(ii) *The augmented policy class Π_Θ^{aug} satisfies the relaxed realizability (R′) of Appendix N.1, with capacity slack ϵ_{approx}^{aug}, on the surrogate federation in which client i's kernel is replaced by $\hat{P}_{e_i}$ and its tasks are the augmented pairs (τ, e_i).*

Then any $\hat{\theta}_{fed}$ whose suboptimality on the surrogate federated objective is at most ϵ_{opt} satisfies, for every client i,

$$\mathcal{J}_i^{aug}(\hat{\theta}_{fed}) \;\ge\; \sup_{\theta\in\Theta} \mathcal{J}_i(\theta) \;-\; \sqrt{(1+\chi_i^{2,aug})\, R_{\max}\, H\, (\epsilon_{approx}^{aug} + \epsilon_{opt})} \\ -\; R_{\max}\, H\, \eta_{cls} \;-\; 2R_{\max}\, H^2\, \eta_{ker},$$

where $\mathcal{J}_i^{aug}$ is client i's deployed return with the descriptor pipeline active, $\mathcal{J}_i$ is the descriptor-free return of Theorem 4, and $\chi_i^{2,aug} := \chi^2(\mathcal{D}_{(\tau_i,e_i)} \,\|\, \bar{\mathcal{D}}_{(\tau,e)})$ is the chi-squared divergence on the

augmented input distribution. If ϵ_{opt} is instead measured on the deployed federated objective, it must be enlarged by $2R_{\max}H^2\eta_{ker} + 2R_{\max}H\eta_{cls}$ before entering the square root.

Proof sketch. The argument is a two-step reduction. First, consider the surrogate federation in which client i's kernel is replaced by $\hat{P}_{e_i}$. On the augmented state space $\mathcal{S} \times \mathcal{E}$, the surrogate kernel $\hat{P}((s', e) \mid (s, e), a) := \hat{P}_e(s' \mid s, a)$ reads the descriptor from the state, and the augmented task (τ, e_i) conditions the initial state (s_0, e_i) with $s_0 \sim \mu_0(\cdot \mid \tau)$. All clients therefore share one environment tuple and differ only in their distributions over augmented tasks, so the surrogate is a Definition 1 (task-level) instance, and Theorem 2 gives

$$\hat{\mathcal{J}}_i(\hat{\theta}_{\text{fed}}) \;\geq\; \sup_{\theta} \hat{\mathcal{J}}_i(\theta) \;-\; \sqrt{(1 + \chi_i^{2,\text{aug}})\, R_{\max}\, H\, (\epsilon_{\text{approx}}^{\text{aug}} + \epsilon_{\text{opt}})},$$

where $\hat{\mathcal{J}}_i$ is client i's surrogate return with the true descriptor in the input. Second, transfer the bound to the deployed return. Let $\mathcal{J}_i^{\text{aug},\star}$ be the return with the true descriptor in the input and the true kernel P_i. By Lemma 2 applied to $\hat{P}_{e_i}$ and P_i, $|\hat{\mathcal{J}}_i(\theta) - \mathcal{J}_i^{\text{aug},\star}(\theta)| \leq R_{\max}H^2\eta_{\text{ker}}$ for every θ, and applying this bound costs $R_{\max}H^2\eta_{\text{ker}}$ on each side of the inequality. Augmenting the input only enlarges the class, $\Pi_\Theta^{\text{aug}} \supseteq \Pi_\Theta$, so $\sup_\theta \mathcal{J}_i^{\text{aug},\star}(\theta) \geq \sup_\theta \mathcal{J}_i(\theta)$, the descriptor-free per-client optimum of Theorem 4. Finally, the deployed policy reads the descriptor through $\hat{e}$, fixed once per episode, so it differs from the true-descriptor policy only on the misclassification event. That event has probability at most η_{cls} by condition (i) and costs at most $R_{\max}H$, as in Step (b) of the proof of Theorem 5 in Appendix N.4. Combining these terms yields the displayed inequality, and the same perturbation bounds give the stated enlargement of ϵ_{opt} when it is measured on the deployed objective. In the explicit environment-aware-prompting form the descriptor is prepended at $u = 0$, so the reveal is immediate ($u^\star = 0$) and the prefix-recoverability condition (C3-rec) of Appendix F.5 is vacuous. The descriptor makes the first input client-specific, which Definition 2 excludes, but the (C3) case of the proof of Theorem 5 does not use the shared μ_0 (Appendix N.4). If the descriptor is instead inferred from a nonempty prefix ($u^\star > 0$, the implicit form of Appendix P.4), the corollary additionally inherits (C3-rec) and the $c_0 R_{\max}\, u^\star$ prefix term of Theorem 5. □

Interpretation: environment-aware prompting as a structural mitigation. Corollary 1 is the precise mathematical content of "environment-aware prompting reduces environment-level heterogeneity to task-level heterogeneity". Under perfect environment description ($\eta_{\text{cls}} = \eta_{\text{ker}} = 0$), the bound reduces to Theorem 2's task-level form on the augmented input, and the formal $\Omega(R_{\max}H\delta)$ floor of Theorem 4 is bypassed entirely. One caveat qualifies this picture. With deterministic, mutually distinct descriptors ($\mathcal{D}_{(\tau_i,e_i)} = \mathcal{D}_{\tau_i} \otimes \mathbf{1}_{e_i}$, with $\mathbf{1}_{e_i}$ the point mass at e_i), the augmented divergence is maximal, $\chi_i^{2,\text{aug}} = N - 1$, regardless of how similar the raw task distributions are. Environment-aware prompting therefore does not literally reduce the problem to the benign small-χ^2 regime. Instead it trades the *irreducible* $\Omega(R_{\max}H\delta)$ floor for a *reducible* slack: with $\chi_i^{2,\text{aug}} = N - 1$ the square-root term becomes $\sqrt{NR_{\max}H\,(\epsilon_{\text{approx}}^{\text{aug}} + \epsilon_{\text{opt}})}$, and the linear form of Appendix N.2 bounds the same gap by $N(\epsilon_{\text{approx}}^{\text{aug}} + \epsilon_{\text{opt}})$. Both bounds vanish with the capacity and optimizer slacks. Coarsening the descriptors so that several clients share an e lowers $\chi_i^{2,\text{aug}}$ at the cost of raising η_{ker} (a single shared kernel model $\hat{P}_e$ must then approximate several distinct per-client kernels), shifting weight from the vanishing $\sqrt{(1 + \chi_i^{2,\text{aug}})(\cdot)}$ slack onto the $O(R_{\max}H^2\eta_{\text{ker}})$ term. With imperfect description, the bound thus interpolates between the task-level form and the worst-case environment-level floor in proportion to the slacks, mirroring the B/C/D spectrum of Appendix O.4. This trade-off makes environment-aware prompting the natural candidate structural mitigation for environment-level heterogeneity in LLM-agent deployments where environment identity can be reliably described in natural language. Environment-aware prompting is the mitigation in the $u^\star$ row of Table 7. The empirical effectiveness of environment-aware prompting is not tested here.

Composability. Environment-aware prompting (Corollary 1) can be combined with the mitigations of Appendix O.8. Environment-aware prompting reduces $u^\star \to 0$ and $\eta_{\text{cls}} \to 0$ and, when η_{ker} is also small, recovers Pattern B in the bound. Per-client LoRA adapters compensate for the residual (C1) failure signaled by $\widehat{\delta}_{\text{eff}}$. Clipped local updates are meant to attenuate the (D-ii) oscillation and (D-iii) prior-degradation components. Both of our optimizers already clip their local updates. Our

runs include PPO, which raises Rank Wrapper from below 13% to $\sim$20% while the variant stays in Pattern D. The runs do not isolate which PPO-specific component (the learned critic with GAE, value-loss clipping, or the 64-task-by-1-rollout batch) causes this gain, and they do not isolate PPO's effect on (D-ii) or (D-iii). The first two mitigations are theory-motivated and untested here, and the third has no unclipped control run. The three classes of mitigation address structurally distinct components and can be applied independently or jointly.

P.4 (C3) Implicit Form: In-Context Posterior Inference

Environment-aware prompting (Appendix P.3) is the explicit form of (C3): a deliberate prompt-engineering operation that adds environment identity to the input. The implicit form is the one most specific to LLM agents: *the LLM identifies the environment from observation history without an explicit environment descriptor*.

Frontier LLMs may already perform in-context posterior inference. We expect frontier LLM agents in environments that reveal identity through observation to take actions consistent with maintaining a posterior over environment identity, a behavior we do not measure here. Such an agent would:

- probe early ("*let me check your OS version*", running `uname -a` or equivalent),
- infer from error messages ("*ah, this is a Windows path issue*", switching to backslash path separators),
- adjust strategy mid-rollout based on observed responses (*e.g.*, shifting from REST to GraphQL after an unexpected schema response),
- recognize benchmark-specific cues (URL branding, page layout, message templates) and condition subsequent actions accordingly.

These behaviors would realize $\hat{\imath} : \mathcal{H} \to [N]$ in (C3) implicitly: the LLM itself serves as the classifier by updating its posterior over environment identity from the trajectory prefix h_u observed so far, without a separate classifier module. Assumption ($\mathrm{R}_{\mathrm{aug}}$) of Appendix N.1 is the formal statement that $\Pi_{\Theta}^{\mathrm{aug}}$ realizes the resulting environment-conditional policies from such history. The assumption is plausible for LLMs with a sufficient context window, because pre-training already equips them with the natural-language understanding required to interpret tool error messages, DOM templates, and protocol responses as environment-identifying signals.

How LLM agents differ from classical RL agents. Classical RL agents must learn such posterior inference through trial and error on the federated task itself, even when augmented with recurrent policies that can in principle condition on history. Learning posterior inference on the task is what meta-RL methods do. The policy class of classical agents lacks the natural-language understanding to interpret observations as environment-identifying signals zero-shot. We expect LLM agents instead to inherit the inference capability from pre-training as a side effect of next-token prediction on heterogeneous text. In our view, pre-trained posterior inference is the most LLM-specific component of the Asymmetric Robustness Mechanism: (C3-implicit) gives LLM agents a way to recover robustness that from-scratch agents can access only at additional on-task learning cost. Condition (C3-implicit) is the structural basis for expecting environment-heterogeneous federated fine-tuning of LLM agents to work without modification on cue-rich real web/tool deployments, while still allowing failure on adversarial constructions like the observation-silent transition-swap chain of Theorem 4.

Whether (C3-implicit) holds yields a testable deployment-level prediction. We expect federated LLM-agent deployments to be robust to environment-level heterogeneity when environment identity is revealed through observation, which we consider likely in most real web and tool settings. Such deployments can fail when environment identity is suppressed and neither (C1) nor (C2) holds (observation-silent benchmarks engineered to remove identifying cues). Our WebShop variants represent the regime in which identifying cues are suppressed. Environments with informative cues remain untested here. The two regimes correspond to "(C3-implicit) holds" versus "(C3-implicit) fails". Test (T3) for (C3) in Appendix O.8.1 makes this distinction a measurable diagnostic. A single benchmark cannot distinguish "FedAvg is robust" from "FedAvg is robust because (C3-implicit) holds", and that distinction matters because robustness that relies on (C3-implicit) does not generalize

to observation-silent environments. The five WebShop variants of Appendix M all lie on the side where (C3-implicit) fails or holds only weakly, and they vary (C1)/(C2) instead. Catalog Split preserves a shared optimum ((C2) holds, and (C1) holds only under reward-relevant state abstraction) and falls into Pattern B under PPO (high-end Pattern C under GRPO). Lookalike Injection exposes an inspectable cue that makes one client-independent rule optimal, so (C2) holds, and it degrades but stabilizes in Pattern C. Field-Subset Index and BM25 Reweighting fall in Pattern C. Their search results differ across clients on states the optimal trajectory visits, so (C1) fails. The optimal query for one client's search engine can be suboptimal for another's, so the clients share an optimal action on some states but not on all, and (C2) holds only partially. Only Rank Wrapper admits no shared rule and no usable cue, and it stays in Pattern D under both optimizers (§5.3).

Sharper than the generic bound. Together, (C3-explicit) and (C3-implicit) provide LLM agents with *two* distinct ways to recover from environment-level heterogeneity. Neither mechanism is readily available to generic FedRL agents: the explicit mechanism requires engineered environment featurization, and the implicit one requires on-task meta-learning. Either the federation's engineers can add environment descriptors to the prompt (and Corollary 1 certifies the resulting bound), or the LLM realizes (C3) implicitly via in-context inference (and (R_{aug}) posits a θ^{aug} achieving it). Both mechanisms allow environment identity to be represented in the input as well as in the dynamics, so environment identity is no longer confined to the dynamics side of I-D Asymmetry. These two ways to recover robustness are the main sense in which the mechanism is sharper for LLM agents than for agents without natural-language pre-training.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and Introduction state three contributions (the two-level Agent Heterogeneity formalization grounded in the Input-Dynamics Asymmetry, the Asymmetric Robustness Mechanism with sufficient conditions (C1)–(C3), and empirical validation across four models and two benchmarks). These contributions are delivered in Sections 2 and 3, Section 4, and Section 5, respectively.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The two-level heterogeneity abstraction (task vs. environment) and its scope are discussed in Section 3. The assumptions underlying the Asymmetric Robustness Mechanism (bounded reward, finite horizon, the LLM-tailored realizability assumptions, the (C1)–(C3) sufficient conditions) are stated explicitly with each theorem in Section 4 and Appendix N. The empirical scope (text-based agents, 1.5B–7B parameter range, GRPO/PPO) is delineated in Section 5. Appendix E collects these limitations in four parts: the scope of the heterogeneity abstraction, the theoretical assumptions, the gap between the theory and the experimental setup, and the empirical scope.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: Theorems in Section 4 state assumptions inline (bounded reward $R_{\max}$, horizon H, the realizability assumptions (R), (R′), ($\mathrm{R}_{\mathrm{env}}$), ($\mathrm{R}_{\mathrm{aug}}$), (LR), the (C1)–(C3) recovery conditions). Complete proofs of Theorems 1–5 and of the solution-level ζ^2 bound are deferred to Appendix N (subsections N.1, N.2, N.3, N.4, N.5), Lemma 2 is a standard result restated with references, and Corollary 1 is given with a proof sketch (Appendix P.3). Partition-side results are stated and proved in Appendices K and L.2. All theorems, lemmas, and propositions are numbered and cross-referenced.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Appendix J fully documents hardware, software stack (Python 3.10, PyTorch 2.6.0+cu124, transformers 4.51.1, vLLM 0.8.2 and 0.8.5, verl 0.3.1.dev), base models, federation hyperparameters (M=2, E=3, T=70, $|X_i|$=100), GRPO/PPO settings, rollout configuration, evaluation protocol, and seeds. Algorithm 1 gives the FEDAGENT pseudocode, and Appendix I.1 describes it line by line. Partition algorithms with hyperparameters (ω, ξ, ξ') are in Appendix L.1. The five WebShop environment variants and their per-client assignment are in Appendix M. A public code repository with all configuration files and the scripts that regenerate the reported curves and tables is linked from Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often

one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: All datasets used (ALFWorld, WebShop) and base models (Qwen2.5-1.5B/3B/7B-Instruct, Llama-3.2-3B-Instruct) are publicly available. A public repository with the full FEDAGENT implementation, dataset preprocessing scripts, all per-experiment configuration files, and the training and evaluation scripts that regenerate the reported curves and tables is available at `https://github.com/sunblaze-ucb/FedAgent` and described in Appendix C. The configuration files on the `paper-reproduce-verl-agent` branch contain the same hyperparameters reported in Appendix J. The repository does not include large benchmark data, trained checkpoints, or training logs.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 5 specifies the federation setup (M=2, E=3, T=70, $|X_i|$=100, 64 trajectories per local epoch). Appendix J.4 lists the optimizer settings (AdamW, betas $(0.9, 0.999)$, weight decay 0.01, gradient clipping 1.0, constant learning rate, actor LR $1e{-6}$, critic LR $1e{-5}$), GRPO/PPO settings (group size G=8, KL 0.01 low-var, clip 0.2, dual-clip 3.0, value-clip 0.5, GAE λ=1.0, γ=1.0). Appendix J.5 lists rollout details (vLLM tp= 4, train temperature 1.0, eval temperature 0.4, max steps 15/50, max-new-tokens 512). Appendix J.7 describes the evaluation protocol. Data partitions for the heterogeneity experiments are described in Appendix L.1, and environment variants in Appendix M.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Table 1 (GRPO) and Table 3 (PPO) report mean $\pm$ standard deviation across three random seeds for the Centralized and FEDAGENT rows. For FEDAGENT, the three runs use the training-goal shuffle seeds 42, 21, and 84, which change which tasks each client holds. The initial checkpoint, the validation set, and the environment seed stay fixed. Each row reports the sample standard deviation over its three runs. The training-curve figures (*e.g.* Figures 4 to 6) show single-seed runs without error bars, as stated in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

Answer: [Yes]

Justification: Appendix J.1 reports the compute environment (NVIDIA H100 80GB GPUs, four-GPU tensor-parallel rollout via vLLM). Appendix J.8 gives the GPU-hours of one training run for each benchmark and optimizer and an estimated total of about 17,000 H100 GPU-hours over the 194 reported runs. The estimate applies these per-run costs to all four backbones. Per-run wall time grows with model size and is dominated by rollout, so 7B runs take longer and the estimate is a lower bound.

Guidelines:

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

9. **Code of ethics**

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

Answer: [Yes]

Justification: The work uses publicly available datasets (ALFWorld, WebShop) and open-weight base models, involves no human subjects, no personally identifiable data, and no high-risk deployment.

Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: Appendix D discusses positive impacts (privacy-preserving agent training compatible with GDPR/HIPAA/CCPA and the EU AI Act, reduced data centralization for sensitive workflows like medical, financial, and on-device assistants) and negative impacts (heterogeneity-induced training instability, residual privacy leakage from gradients, and the need for orthogonal defenses such as differential privacy and secure aggregation, which Appendix I.2 elaborates).

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper releases training code and configurations only. No new pre-trained models, generative models, or scraped datasets that pose high misuse risk are released.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The datasets and algorithms we use are cited in the main paper and Appendix J: ALFWorld [114], WebShop [165], the GRPO algorithm [107], and PPO [106]. The base models Qwen2.5-1.5B/3B/7B-Instruct (Qwen Team) and Llama-3.2-3B-Instruct (Meta), and the rollout and training software (vLLM, verl, verl-agent [31], and HuggingFace transformers), are open-weight or open-source assets used under their licenses and within their terms of use. ALFWorld and WebShop use the MIT License. Qwen2.5-1.5B/7B-Instruct, vLLM, verl, verl-agent, and transformers use Apache 2.0, Qwen2.5-3B-Instruct uses the Qwen Research License, and Llama-3.2-3B-Instruct uses the Llama 3.2 Community License.

    Guidelines:

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

has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The new assets are the FEDAGENT training pipeline, the task-level partition scripts (Appendix L.1), and the five WebShop environment-variant generators (Appendix M). The public release at `https://github.com/sunblaze-ucb/FedAgent` contains a README, configuration files, exact commands, and seeds, and is also referenced from Appendix C.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects. All experiments use publicly available simulated environments (ALFWorld, WebShop).

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve human subjects. No IRB review is required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: LLMs are the central object of study: FEDAGENT trains pre-trained instruction-tuned LLMs (Qwen2.5-1.5B/3B/7B-Instruct, Llama-3.2-3B-Instruct) as decision-making agents via federated reinforcement learning. The specific models, prompting templates, rollout configurations, and policy-optimization details are documented in Appendices J.2, J.4, and J.5.

    Guidelines:

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