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 *Equal Contribution. +Equal Advising.

fed-agent.github.io github.com/sunblaze-ucb/FedAgent

Training AI agents powered by Large Language Models (LLMs) requires large volumes of user data, yet privacy concerns limit centralized collection and any single party alone holds insufficient data. We explore **FEDAGENT** (**Federated Agent Reinforcement Learning**), a decentralized reinforcement learning paradigm that collaboratively trains LLM agents across distributed clients without sharing local data. The central reliability question is: Is FEDAGENT effective under *uniform* client distribution, and more importantly, is it robust to client *heterogeneity*? For the former, we provide the first empirical evidence that FEDAGENT matches Centralized Agent Training and outperforms Local Agent Training. For the latter, we first formalize **Agent Heterogeneity** at two structurally distinct levels: task-level (what clients ask the agent to do) and environment-level (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 through its input channel, while environments do not. Then, we theoretically establish an **Asymmetric Robustness Mechanism**: FEDAGENT is robust to task-level heterogeneity but non-robust to environment-level heterogeneity. We further identify three sufficient conditions under which FEDAGENT recovers robustness despite environment-level heterogeneity, and illustrate four possible training-curve patterns. On real-world agent benchmarks WebShop and ALFWorld, we empirically verify that FEDAGENT remains robust under extreme task-level heterogeneities and forms a stable-degrade-collapse spectrum under environment-level heterogeneities.

1. Introduction

LLM-based AI agents have shown remarkable capabilities across web navigation, embodied tasks, and tool use (Gao et al., 2025; Liu et al., 2025a; Zhang et al., 2026a), yet training them requires large volumes of user task queries and trajectories: privacy concerns limit centralized collection of such data, and any single party alone holds insufficient data to train a capable agent. We study **FEDAGENT** (**Federated Agent Reinforcement Learning**), a decentralized paradigm for LLM agent RL in which each client runs local policy optimization on its own

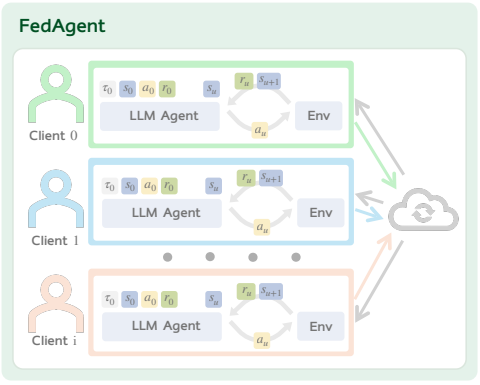

Figure 1 | **FEDAGENT framework.**

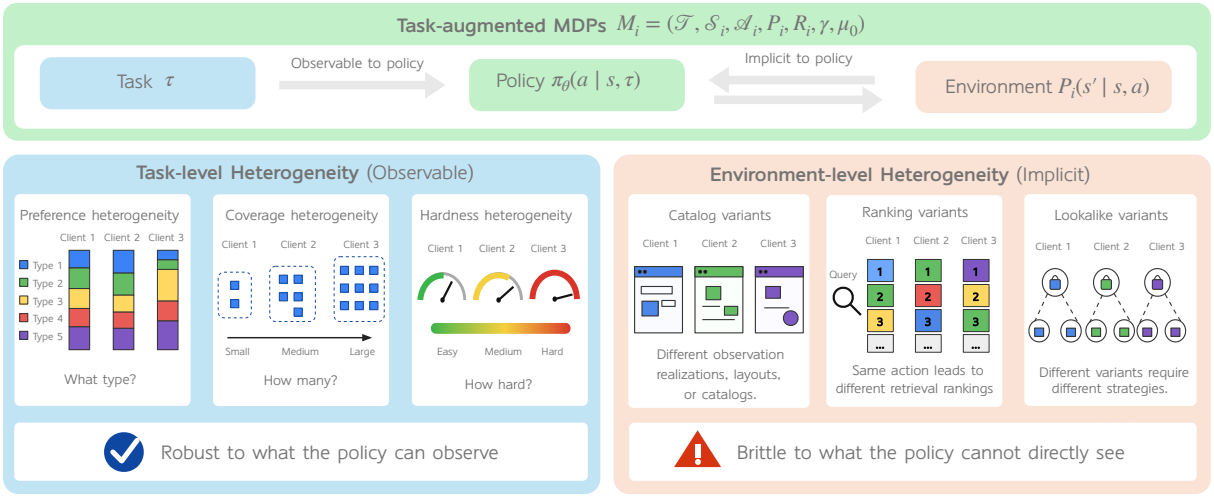

Figure 2 | **Asymmetric Robustness Mechanism of FEDAGENT under Agent Heterogeneity. Top:** in a task-augmented MDP the task τ is *observable* to the policy through its input channel, while the transition kernel P_i is *implicit*, visible only through successor states. This is the Input-Dynamics Asymmetry (§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 (illustrated by catalog, ranking, and lookalike variants, §3.3) perturbs the implicit dynamics and is worst-case non-robust (Theorems 3/4) unless one of the recovery conditions (C1)–(C3) holds (§4.3).

data, and the server averages the client models into a global model and broadcasts it back to the clients for the next round. Only model parameters are exchanged, so local data is never shared.

The reliability of FEDAGENT hinges on two questions, both *structurally new* relative to conventional federated learning (FL), which spans both supervised FL on static datasets and federated reinforcement learning (FRL) on simple MDPs with low-dimensional states, low-dimensional actions, and bounded rewards. **(Q1)** Is FEDAGENT *effective* under a uniform client distribution, i.e., does it match centralized agent training? Supervised FL has long established that FedAvg matches centralized SGD on input-label pairs, and FRL (Kairouz and McMahan, 2021; Liu et al., 2024a; Qi et al., 2021) extends this to the simple-MDP settings above. LLM agent RL, by contrast, involves *natural-language state and action spaces, diverse task formulations, and complex multi-step environment interactions*. In this regime the FedAvg-matches-centralized result does not directly transfer, and whether federated aggregation preserves training quality remains under-explored. **(Q2)** Is FEDAGENT *robust* when clients are *heterogeneous*? In conventional FL and FRL, client heterogeneity is treated as a single-axis *data-distribution skew*: supervised FL captures it with label, feature, or quantity skew on static datasets (Kairouz and McMahan, 2021; Li et al., 2020a; Ye et al., 2024a), while FRL (Jin et al., 2022; Wang et al., 2024a) captures it with transition differences across simple MDPs. In LLM agent RL, by contrast, heterogeneity is no longer a single-axis data property: it has an intrinsic *two-level MDP structure* with no counterpart in either paradigm, in which clients differ in *what task* they ask the agent to perform and *which environment* the agent acts in. The two levels behave structurally differently for an instruction-following LLM policy.

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. This asymmetry is a property of LLM-induced policies in agent MDPs and is not exploited by conventional FL or FRL analyses.

The Input-Dynamics Asymmetry has a direct theoretical consequence 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. It is *worst-case non-robust* to environment-level heterogeneity (Theorems 3/4): an irreducible $\Omega(R_{\max} H \delta)$ per-client gap remains that additional capacity, samples, or training cannot remove (here $R_{\max}$ is the reward scale, H the episode horizon, and δ the level of environment divergence, all formalized in §2–3.3). Three sufficient conditions **(C1)–(C3)** recover robustness in the env-level regime: common optimal off-support, action-preserving with shared optimum, and self-revealing environment via observation history. The mechanism further illustrates four observable training-curve Patterns A/B/C/D that bridge theory and experiment (Figure 2). The mechanism is stated at the MDP level but is sharpest for LLM agents (Appendix O), and it targets the quality of the policy that decentralized LLM agent RL produces, rather than the convergence rate of the federated optimizer that conventional FL/FRL heterogeneity theory analyzes.

With empirical studies on WebShop (Yao et al., 2022a) and ALFWorld (Shridhar et al., 2021) with Qwen/Llama agents (1.5B–7B) and GRPO/PPO algorithms, we demonstrate that FEDAGENT matches centralized training, remains robust under extreme task-level heterogeneities, and forms a stable-degrade-collapse spectrum across environment-level heterogeneities, in line with the theory.

Our contributions are summarized as follows:

- We make an early attempt to explore decentralized LLM agent RL and formalize **Agent Heterogeneity** as a two-level object (*task-level* with three sub-types: preference, coverage, hardness; vs. *environment-level*), grounded in the **Input-Dynamics Asymmetry** of task-augmented MDPs. We also design corresponding strategies (*e.g.*, PREFERENCEPARTITION, COVERAGEPARTITION, HARDNESSPARTITION) that isolate each form of heterogeneity by a single hyperparameter.
- 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). The mechanism illustrates four distinct training-curve patterns (A through D).
- We empirically verify the effectiveness and mechanism on WebShop and ALFWorld: FEDAGENT matches centralized training, remains robust under extreme task-level heterogeneities, and forms a stable-degrade-collapse spectrum under environment-level heterogeneities.

2. FEDAGENT: Federated Agent Reinforcement Learning

A population of N clients runs T communication rounds: at round $t \in \{0, \ldots, T-1\}$, the server samples $S_t \subset [N]$ with $|S_t| = M$ uniformly without replacement, broadcasts the global parameters θ_t, and aggregates the participating clients' locally updated parameters.

Each client i has its own **task-augmented MDP** $\mathcal{M}_i = (\mathcal{T}, \mathcal{S}_i, \mathcal{A}_i, P_i, R_i, \gamma, \mu_0)$ and per-client task distribution $\mathcal{D}_{\tau_i}$, and holds a finite local task pool $X_i \subset \mathcal{T}$ sampled from $\mathcal{D}_{\tau_i}$, from which its rollouts draw tasks. (Throughout our constructions the action space and reward function are shared, $\mathcal{A}_i = \mathcal{A}$ and $R_i = R$. We keep the subscripts for generality.) The client acts under an LLM-induced policy $\pi_\theta(a \mid s, \tau)$ that, given a task descriptor $\tau \in \mathcal{T}$ and state s, produces an action a which may span thousands of tokens (intermediate reasoning plus an environment-facing tool call). Per-step rewards are non-negative and bounded, $r_u \in [0, R_{\max}]$. H denotes the (finite)

Algorithm 1 FEDAGENT with Client and Server training.

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 $\theta_{t+1} \leftarrow \frac{1}{M} \sum_{i \in S_t} \theta_{i,t,E}$
end for
return $\theta_{\text{final}} \leftarrow \theta_T$

episode horizon, the maximum number of agent steps per episode in the episodic-undiscounted setting ($\gamma = 1$) we use. In the discounted setting, equivalently $H = 1/(1-\gamma)$ (cf. Appendix M.3's discounted analogue). Rolling out trajectories $\chi = (\tau, s_0, a_0, r_0, \ldots, s_H)$ in $\mathcal{M}_i$ yields the per-client expected return $\mathcal{J}_i(\theta) = \mathbb{E}_{\tau \sim \mathcal{D}_{\tau_i}} \mathbb{E}_{\chi \sim (\pi_\theta, \mathcal{M}_i, \tau)} [\sum_u \gamma^u r_u]$.

On each round, every $i \in S_t$ initializes $\theta_{i,t,0} \leftarrow \theta_t$, runs E local epochs of stochastic policy optimization on local rollouts (FEDAGENT is algorithm-agnostic and we instantiate **GRPO** and **PPO** as representative choices), and returns $\theta_{i,t,E}$ to the server. The server aggregates by uniform model averaging $\theta_{t+1} = \frac{1}{M} \sum_{i \in S_t} \theta_{i,t,E}$, optimizing the federated objective $\mathcal{J}_{\text{fed}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{J}_i(\theta)$. Algorithm 1 states the full protocol with client- and server-side computation color-coded. Appendix H gives a line-by-line description together with privacy and communication-cost discussions.

3. Two-Level Agent Heterogeneity

3.1. Input-Dynamics Asymmetry

Building on the per-client task-augmented MDP $\mathcal{M}_i = (\mathcal{T}, \mathcal{S}_i, \mathcal{A}_i, P_i, R_i, \gamma, \mu_0)$, task distribution $\mathcal{D}_{\tau_i}$, and policy $\pi_\theta(a \mid s, \tau)$, we additionally write the *federated mixture task distribution* as $\bar{\mathcal{D}}_\tau := \frac{1}{N} \sum_i \mathcal{D}_{\tau_i}$, the effective task distribution from which federated training draws. The robustness analysis below depends on which components of these MDPs are observable to the policy.

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 $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ that arise after acting.

We refer to this as **Input-Dynamics Asymmetry** (**I-D Asymmetry** hereafter; the appendices abbreviate it as **Asymmetry (A)**). Operationally, because τ is observable as input, a single θ can encode a task-conditional decision rule $(s, \tau) \mapsto a$ that maps different prompts to different actions. This is exactly what instruction-tuned LLMs already do. Because P is implicit in dynamics, a single θ cannot encode an environment-conditional function unless env identity is revealed through its input or observation history: when $P_1 \neq P_2$ on the same (s, τ), θ must choose one behavior, which is wrong in at least one of them (*e.g.*, two shops whose search engines rank differently: the same query returns the target at the top of the results for one client and near the bottom for the other).

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 the transition kernel P_i varies across clients over a shared $(\mathcal{S}, \mathcal{A})$ (client-specific reachable subsets $\mathcal{S}_i \subseteq \mathcal{S}$ are induced by P_i). The first is observable to the policy, the second is not. This asymmetry drives the robustness analysis in §4.

3.2. Task-Level Heterogeneity

Definition 1 (Task-Level Heterogeneity). Clients share a single environment tuple $\mathcal{M}_{\text{env}} := (\mathcal{S}, \mathcal{A}, P, R, \gamma, \mu_0)$ 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. Concrete instances 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. 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 electronics while another searches mostly 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 pretraining 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 pushing $\mathcal{D}_{\tau_i}$ 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 global mean, $\Delta^2_{\text{pref}} := \frac{1}{N} \sum_i \|p_i - \bar{p}\|_2^2$.
- **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 (Feng et al., 2025), the pool sets the *per-epoch exploration breadth*: a small pool causes the agent to repeatedly explore the same narrow region, which is specific to RL training and is not equivalent to supervised quantity-imbalance (where the entire pool is traversed each epoch). We measure this axis by the per-client local pool size $n_i = |X_i|$. Since pool sizes can range over orders of magnitude across clients, we use the scale-invariant squared coefficient of variation $\Delta^2_{\text{cov}} := \widehat{\text{CV}}^2(\{n_i\}) = \frac{1}{N} \sum_i \left((n_i - \bar{n})/\bar{n}\right)^2$

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 short Pick & Place tasks that the pretrained model already solves, while another's is dominated by the two-object Pick2 tasks that it mostly fails. Difficulty matters for training because the *policy-gradient advantage signal depends on it*: 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, and group-relative methods such as GRPO (Shao et al., 2024) normalize advantages by within-group success statistics. 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 a checkpoint's measured success rate, rather than any intrinsic task property, is the operationally relevant notion of difficulty. 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 the pretrained instruct model before any federated training, and higher $\rho(\tau)$ means an *easier* task. We then measure client i's difficulty profile 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 is one whose pool concentrates at low $\rho(\tau)$ and hence has 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 heterogeneity axis, which requires the three dispersion measures $(\Delta^2_{\text{pref}}, \Delta^2_{\text{cov}}, \Delta^2_{\text{hard}})$ to be *operationally separable*: each must be variable independently of the other two while the first-order means stay fixed (properties (D1)–(D4) below). The triple (p_i, n_i, ρ_i) gives exactly this granularity. It is a *first-order task-distribution descriptor* of client i: each component is a marginal projection of $\mathcal{D}_{\tau_i}$ (along the category map ℓ, the pool cardinality, and the thresholded success indicator, respectively), invariant under resampling of the pool. The descriptor is deliberately a projection rather than a complete characterization: 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 J formalizes the projections' properties, the sufficiency direction, and an explicit converse counterexample).

Partition algorithms and design properties. We design three sampling procedures, each instantiating one heterogeneity axis under a single dispersion hyperparameter (pseudocode in Appendix K.1):

- PREFERENCEPARTITION(ω) draws each client's category distribution from 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). $\omega \to 0$ *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 3 visualizes the two endpoints on both benchmarks.
- COVERAGEPARTITION(ξ) draws each client's pool size n_i from a Beta-shaped distribution on the admissible range $[L_{\min}, L_{\max}]$, holding the global overlap r (the average number of clients holding each task) fixed. This controls the spread of $\{n_i\}$ without changing the per-client mean: 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). Note that this direction is opposite to* ω*.*
- HARDNESSPARTITION(ξ') first measures hardness operationally rather than by human annota-

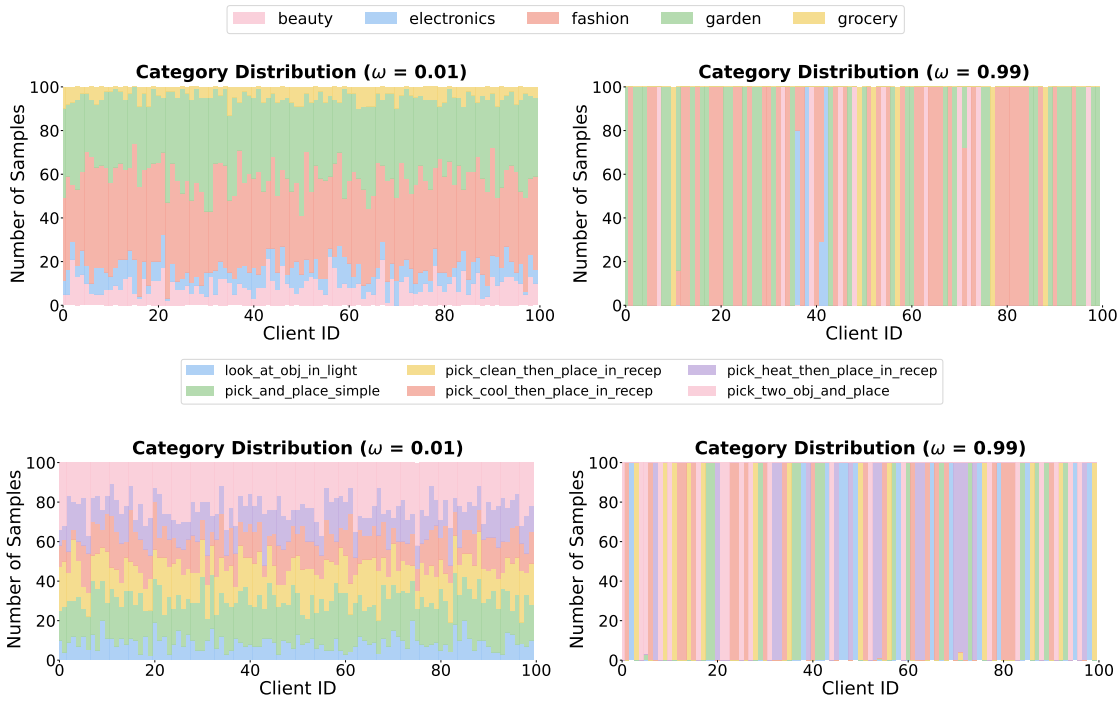

Figure 3 | **Client distributions under Preference Heterogeneity** at the smallest and largest tested values of ω (left: $\omega = 0.01$, near-uniform; right: $\omega = 0.99$, extremely Heterogeneous), on WebShop (top, five product categories) and ALFWorld (bottom, six household task types). Each panel shows per-client category compositions as stacked bars over client IDs: at $\omega = 0.01$ every client mirrors the global marginal p_0, while at $\omega = 0.99$ each client concentrates on one or two categories. The aggregate across clients matches p_0 in expectation at both endpoints (up to finite-N sampling noise), so ω moves only the dispersion Δ^2_{pref}. Coverage and hardness counterparts are in Appendix K.3.

tion: it rolls out the reference checkpoint π_{ref} on every task and pre-labels tasks with measured success rate $\rho(\tau) \geq \eta$ as "successful" and the rest as "unsuccessful" (the labels are measured once at the pretrained checkpoint and used only to construct the partition. Difficulty relative to the evolving policy naturally drifts during training). It then draws 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 K.2): **(D1) target control**, each hyperparameter monotonically controls its target measure (Propositions 1–3); **(D2) cross-measure invariance**, varying one hyperparameter leaves the other two measures invariant in expectation under a mild threshold-independence assumption between type and difficulty (its approximate validity in practice is discussed in Appendix K.2), *e.g.* $\partial_\omega \mathbb{E}[\Delta^2_{\text{cov}}] = \partial_\omega \mathbb{E}[\Delta^2_{\text{hard}}] = 0$ (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**, the three partitions compose to reach any feasible target triple (Theorem 11). Together, these properties let us empirically separate and vary each task-level sub-type independently in §5.2.

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 $\delta := \sup_{i \neq j,\,(s,a)} D_{\mathrm{TV}}(P_i(\cdot \mid s,a), P_j(\cdot \mid s,a)) \in [0,1]$. 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. 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 axes: *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. Each axis yields a different form of divergence in the optimal policy $\pi^\star$, and our construction covers all four.

We make this concrete on WebShop, whose transition pipeline naturally factors into four stages aligned with these axes: *content/catalog*, *encoding/index*, *matching/score*, and *rendering*. We then instantiate one or more variants per stage, yielding five WebShop perturbations spanning all four layers (Appendix L). The simplest, **Catalog Split**, perturbs the *content* layer 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 only vs. description+features), so identical queries yield different rankings and force per-client query crafting. **BM25 Reweighting** perturbs *matching* by setting per-client extreme (k_1, b) combinations, changing TF 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 mislead BM25 ranking *and* target one reward subterm. Finally, **Rank Wrapper** perturbs *rendering* by post-processing BM25 outputs (shuffle, invert), breaking any "trust the top position" heuristic the policy might otherwise learn. 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.

4. Asymmetric Robustness Mechanism

We now derive the central theoretical consequence of I-D Asymmetry: FEDAGENT's robustness is *asymmetric* across the two heterogeneity levels, robust to task-level heterogeneity but worst-case non-robust to environment-level heterogeneity, with three structural conditions that recover robustness in the latter case. The mechanism is grounded in five LLM-tailored realizability assumptions on the policy class $\Pi_\Theta = \{\pi_\theta : \theta \in \Theta\}$. Each theorem compares the federated solution against a task- or environment-conditional optimum, so each needs an assumption that Π_Θ can express the relevant optimum: (R) and its ϵ-approximate version (R′) posit that Π_Θ (approximately) realizes task-conditional optima; ($\mathrm{R}_{\mathrm{env}}$) and ($\mathrm{R}_{\mathrm{aug}}$) are their environment-level and history-augmented analogues; and (LR) posits that Π_Θ locally realizes per-environment optima (all formally stated in Appendix M.1). The mechanism yields four observable training-curve patterns that bridge theory and experiments (§4.4).

4.1. 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: for every client i, the federated per-client return $\mathcal{J}_i(\hat{\theta}_{\mathrm{fed}})$ 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 in two layers, an idealized statement (Theorem 1) and an approximate statement that handles imperfect realizability and optimizer slack (Theorem 2).

Theorem 1 (Task-Level Robustness, idealized)**.** *Under Definition 1 and (R), (i) the federated objective collapses 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.*

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 optimizer faces the same loss landscape regardless of how task mass is distributed across clients, so extreme one-hot splits and i.i.d. uniform splits are indistinguishable at the optimization level. Part (ii) adds that the federated optimum is simultaneously optimal for every client: no client-versus-federation tradeoff exists at the global optimum. In this idealized limit, federated and centralized training therefore coincide in both objective and optimum.

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,*

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

Theorem 2 is the practical form of Theorem 1: it relaxes (R) to its ϵ-version (R′), which allows the LLM to realize the task-conditional optimum only up to a capacity slack ϵ_{approx} and the optimizer to 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 factor χ^2 need not vanish, but it is bounded by $N-1$ in the standard uniform-weight federated setup, regardless of how aggressive the per-client split is (here χ^2 is the chi-squared divergence between task distributions, unrelated to the trajectory symbol χ of §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 provide θ with different pieces of the task-conditional optimal behavior $\tau \mapsto b(\tau)$ (the behavior an optimal policy implements on task τ), and they add *complementarily* rather than conflicting. The federation cannot face a worse loss landscape than centralized training because the two coincide exactly (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 ϵ_{approx} at initialization is typically small and federated fine-tuning extends an existing capability rather than learning $\tau \mapsto b(\tau)$ from scratch (Appendix O.1).

4.2. Mechanism 2: Environment-Level Non-Robustness

We next show that under environment-level heterogeneity (Definition 2), federated agent training is *fundamentally different*: there exist constructions where the converged θ is worst-case non-robust. Concretely, *non-robustness* here means: there is an irreducible per-client gap between

$\mathcal{J}_i(\hat{\theta}_{\text{fed}})$ and the per-client optimum $\sup_\theta \mathcal{J}_i(\theta)$, of structural origin, that no optimizer, capacity, or sample budget can close. We establish this in two layers, an existence claim (Theorem 3) and a quantitative lower bound (Theorem 4).

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

Theorem 3 says we can construct envs that force any federated optimum to be strictly sub-optimal on at least one client. A canonical construction (detailed in Appendix M.3) is a two-client transition-swap bandit: client 1's optimal action is client 2's worst, and vice versa. Any single θ must choose one client's optimal action, and the other client necessarily incurs strictly positive sub-optimality. This is qualitatively unlike Theorem 1, whose federated optimum was simultaneously optimal for *every* client. Under Definition 2, no single θ can satisfy both.

Theorem 4 (Environment-Level Non-Robustness, quantitative)**.** *Under Definition 2 with worst-case TV divergence $\delta := \sup_{i\neq j,(s,a)} D_{\text{TV}}(P_i, P_j)$ 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 $\theta^\star_{fed} \in \arg\max_\theta \mathcal{J}_{fed}(\theta)$.*

Theorem 4 is the quantitative form of Theorem 3: it characterizes the magnitude of the gap as a function of the inter-env 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, training, or optimizer choice can close it. (The single-step $\Omega(R_{\max}\delta)$ gap holds for any policy class satisfying (LR). The H-amplification additionally requires that (C3) be unavailable.) The gap is a property of the federated *solution* given Definition 2's transition heterogeneity, not of the optimization *trajectory* that reaches it. This is the central contrast with Theorem 2, whose bound consisted entirely of slacks that vanish under standard centralized-training conditions.

The mechanism is again I-D Asymmetry, operating in the opposite direction: env identity i is *not* part of the prompt, so a single θ cannot switch behavior across envs: at the same observed (s, τ), 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 envs, 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 structural floor $\Omega(R_{\max} H\delta)$ on the resulting bias. 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 is overwritten, manifesting empirically as catastrophic forgetting of capabilities the base model originally possessed (Appendix O.2).

4.3. Three Sufficient Conditions to Recover Robustness

Theorem 4's worst-case construction applies only when three requirements hold simultaneously: (a) the optimal trajectory *visits* the kernel-disagreement region, (b) optimal actions on the visited region *disagree* across envs, and (c) env identity is *not* recoverable as input to the policy. We expect that in most real-world deployments, env heterogeneity has additional structure that

breaks at least one of (a)/(b)/(c), which would let a single θ still encode optimal behavior across all clients and recover Theorem 2-style robustness despite $P_i \neq P_j$. We identify three structural conditions accordingly, each targeting one requirement:

- **(C1) Common optimal off-support** (breaks requirement (a)). There exists a shared policy $\pi^\star$ that is globally optimal in every $\mathcal{M}_i$, and its occupancy avoids the disagreement region $\{(s,a) : P_i(\cdot \mid s,a) \neq P_j(\cdot \mid s,a)\}$: kernels disagree, but the optimal trajectory never visits the disagreement region, *e.g.*, clients see different household room layouts but the optimal navigation corridor passes through identical rooms.
- **(C2) Action-preserving with shared $\pi^\star$** (breaks requirement (b)). The optimal action set $\arg\max_a Q_i^\star(s,\tau,a)$ on every reachable (s,τ) is the same across clients even if the underlying per-client optimal Q-functions $Q_i^\star$ differ: kernels disagree but action rankings are preserved, *e.g.*, robot dynamics differ in friction but the optimal torque ordering at each state is invariant.
- **(C3) Self-revealing environment via observation history** (breaks requirement (c)). Env identity is inferable from the trajectory prefix $h_t := (s_0, a_0, r_0, \ldots, s_t) \in \mathcal{H}$ that the policy has observed so far. Formally, there exist a classifier $\hat{\imath} : \mathcal{H} \to [N]$, kernel estimators $\{\hat{P}_i\}_{i\in[N]}$, a reveal step $t^\star \leq H$, and a fixed default prefix policy π_0 (followed for $t < t^\star$) such that:
  - **(C3-cls)** $\Pr_{h_{t^\star}\sim(\mathcal{M}_i,\pi_0)}[\hat{\imath}(h_{t^\star}) \neq i] \leq \eta_{\text{cls}}$, where $h_{t^\star}$ is the prefix generated by rolling out π_0 in $\mathcal{M}_i$ (classifier slack: the env label inferred at the reveal step $t^\star$ is kept fixed thereafter);
  - **(C3-ker)** $\sup_{s,a} D_{\text{TV}}(\hat{P}_i, P_i) \leq \eta_{\text{ker}}$ for every i (kernel-modeling slack);
  - **(C3-rec)** the prefix is *recoverable*: $\mathbb{E}_{(\mathcal{M}_i,\pi_0)}[V_i^\star(s_{t^\star})] \geq V_i^\star - c_0 R_{\max} t^\star$ for every i and an absolute constant c_0, where $V_i^\star(\cdot)$ is the optimal state-value function of $\mathcal{M}_i$ and, slightly abusing notation, $V_i^\star := \mathbb{E}_{s_0\sim\mu_0}[V_i^\star(s_0)]$ is its initial-state expectation. In words: following π_0 for $t^\star$ steps does not drive the state into a low-value region (automatic in per-round-reset environments, and in web/tool environments where a back or home action restores the initial state. Both our benchmarks satisfy it via per-episode resets).

  Under (C3) the history-augmented policy class Π_Θ^{aug} (the original class with the trajectory prefix h_t appended to the input) realizes per-environment optima, *e.g.*, a web agent reads "united.com" from a banner and conditions subsequent actions accordingly.

Theorem 5 (Recovery via (C1)–(C3))**.** *Under (C1) or (C2) (with (R_{env})), the policy-disagreement gap $\Delta_{pol} = 0$. Under (C3) (with (R_{aug}), by which the same parameter class Θ realizes the history-augmented policies, so both suprema in Δ_{pol} remain over Θ), $\Delta_{pol} \to 0$ as the slacks $(t^\star, \eta_{cls}, \eta_{ker}) \to 0$, for any TV divergence δ.*

Theorem 5 says each condition, when satisfied, brings the env-level case back to Mechanism 1's structurally-equivalent regime: (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 estimator become accurate. In each case, robustness is recovered not by closing Theorem 4's floor but by *circumventing* it: one of requirements (a)/(b)/(c) above is structurally precluded. These three conditions are sufficient but not necessary: a deployment can have all three apparently fail empirically and still train without collapse (Appendix N.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 which environment it is in, exploiting branding, error messages, prompt cues, and structural signals that synthetic benchmarks intentionally suppress. Assumption (R_{aug}) captures this formally: the LLM's context window can implement the classifier $\hat{\imath}$ as a side effect of next-token prediction. The in-context form of (C3) predicts that env-heterogeneous federated fine-tuning can work without modification on cue-rich real web/tool deployments, where branding alone can make (C3-cls) trivial. Our

synthetic variants deliberately suppress such cues. The prediction still admits failure on adversarial constructions like the observation-silent transition-swap bandit of Theorem 4, whose observations carry no information about the environment identity. Appendix O.3 formalizes the explicit form of (C3) via env-aware prompting (Corollary 1), and Appendix O.4 discusses the implicit (in-context inference) form.

4.4. Four Empirical Patterns Explained by the Mechanism

The Asymmetric Robustness Mechanism above (Theorems 2, 4, 5) predicts four possible training-curve patterns that organize federated agent runs. Pattern A captures the task-level regime (Mechanism 1), where mixture-equivalence makes robustness *unconditional*. Patterns B/C/D capture the env-level regime (Mechanism 2) as a graded spectrum from full robustness recovery to complete collapse. Which pattern a variant falls into depends on two factors: how closely (C1)/(C2)/(C3) hold, which sets the achievable plateau, and the optimization-level slack ϵ_{opt} of the optimizer in use, which decides whether that plateau is attained. Because the second factor is orthogonal to the first, the same variant can fall into Pattern B under one optimizer and Pattern C under another (§5.3). Each pattern is paired with a specific theorem and a specific empirical signature, turning the mechanism into a diagnostic tool: an observed curve shape indicates which regime is likely active and which mitigation applies.

- **Pattern A (Task-level robust, unconditional).** 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. 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 optimizer therefore faces the same loss landscape as centralized training on $\bar{\mathcal{D}}_\tau$ and converges to the same plateau, with per-client gap $\to 0$. The empirical signature is this split-invariance of the converged curve, up to the run-to-run variability of agent RL.
- **Pattern B (Env-level robust: one of (C1)–(C3) holds and the optimizer attains the shared optimum).** The federated curve nearly matches the homogeneous-environment (single-env) baseline despite $P_i \neq P_j$. Theorem 5 guarantees a single $\theta^\star$ that is simultaneously optimal for every $\mathcal{M}_i$, so client gradients all pull θ toward this shared $\theta^\star$ and FedAvg converges to it. 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, recasting env as input and 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 (Env-level degrade-but-stable: (C1)/(C2)/(C3) partially hold, or hold 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*: env identity is inferable from history with classifier slack $\eta_{\text{cls}} \in (0, \frac{1}{2})$, optimal action sets coincide on most but not all reachable states, or Π_Θ contains only an approximate common optimum. Alternatively, the structural conditions hold in full but the optimizer's ϵ_{opt} keeps the plateau below the baseline (the Catalog Split/GRPO case of §5.3). The resulting per-client gap is controlled by these slacks rather than by Theorem 4's worst-case $\Omega(R_{\max} H \delta)$ floor. The federation incurs a small structural cost for partial (C) satisfaction but does not collapse.
- **Pattern D (Env-level collapse: all of (C1)–(C3) fail and the heterogeneity is of the adversarial, observation-silent type realized by Theorem 4).** Performance degrades severely, often 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 applies: client gradients systematically disagree on $\arg\max_a Q_i^\star$, so no single θ can simultaneously satisfy all clients' env-conditional

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	71.9	37.5
Local (Client 42)	7.1	12.5	8.3	33.3	9.1	10.0	12.5	62.5	32.8
Local (Client 84)	42.9	50.0	33.3	11.1	45.5	30.0	35.9	60.9	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$_{\pm 5.6}$	76.0$_{\pm 2.4}$	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	64.1	43.8
Local (Client 42)	21.4	50.0	16.7	33.3	36.4	30.0	29.7	71.9	37.5
Local (Client 84)	35.7	25.0	50.0	44.4	45.5	50.0	42.2	60.9	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$_{\pm 7.9}$	81.8$_{\pm 7.0}$	**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**$_{\pm 3.6}$	**83.3**$_{\pm 5.5}$	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	75.0	45.3
Local (Client 42)	50.0	25.0	50.0	44.4	36.4	40.0	42.2	70.3	48.4
Local (Client 84)	7.1	12.5	16.7	22.2	9.1	20.0	14.1	62.5	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$_{\pm 3.3}$	88.0$_{\pm 3.3}$	**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**$_{\pm 5.0}$	**89.1**$_{\pm 4.1}$	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	65.6	26.6
Local (Client 42)	28.6	37.5	41.7	**33.3**	18.2	50.0	34.4	60.9	34.4
Local (Client 84)	14.3	25.0	16.7	22.2	**45.5**	20.0	23.4	71.9	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$_{\pm 6.5}$	77.1$_{\pm 6.3}$	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**$_{\pm 1.8}$	**77.6**$_{\pm 3.9}$	**57.8**$_{\pm 6.2}$

Table 1 | **Performance Comparison on ALFWorld and WebShop**. The policy optimization algorithm is GRPO; PPO counterpart is in Table 3 (Appendix F.2). We report the averaged performance and the corresponding standard deviation over three random seeds. For ALFWorld, the **Success Rate** (%) is reported. For WebShop, both the **Task Score** (0–100, the WebShop convention of 100× mean episode reward) and the **Success Rate** (%) are reported.

optima. For LLM agents specifically, this overwrites the pre-training prior, manifesting as visible degradation of capabilities the base model originally possessed (Appendix N.5).

5. Empirical Results

We empirically validate the Asymmetric Robustness Mechanism on two widely used agent benchmarks: WebShop (Yao et al., 2022a), a simulated e-commerce environment where the agent searches, browses, and purchases products against natural-language shopping instructions, and ALFWorld (Shridhar et al., 2021), a text-based embodied household benchmark spanning six task types (Pick, Look, Clean, Heat, Cool, Pick2). §5.1 addresses (Q1) by testing whether FEDAGENT matches centralized training under uniform client distributions. §5.2 and §5.3 jointly address (Q2): §5.2 tests whether Pattern A holds across all three task-level subtypes, and §5.3 tests 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 both GRPO and PPO as policy optimizers. Full details are in Appendix I.

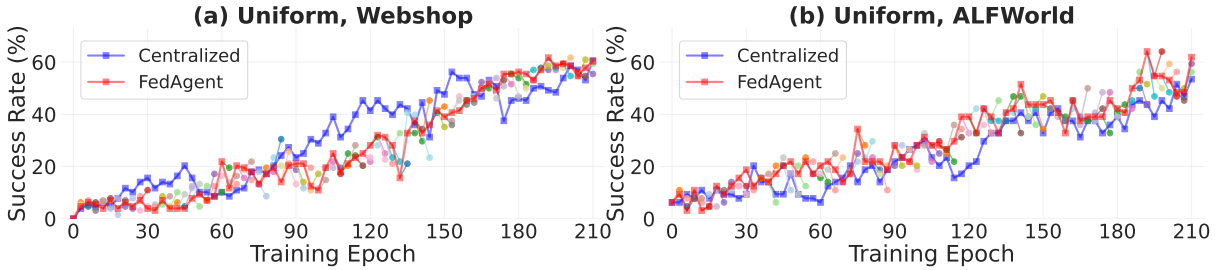

Figure 4 | **Training Dynamics of FEDAGENT (GRPO) and Centralized Training** (validation success rate, `Qwen2.5-1.5B-Instruct`). The red line is FEDAGENT's *aggregated server model* throughout training. The blue line is the centralized baseline. Circle marks with different colors show the model right after local training on the *specific clients selected in each round*. FEDAGENT reaches end-of-training success rates nearly identical to centralized training on both benchmarks.

5.1. FEDAGENT under Uniform Client Distribution

Setup. We address (Q1) under a uniform client distribution to isolate the effect of the federated paradigm itself from any heterogeneity factor. We partition the dataset (WebShop / ALFWorld) into 100 clients with 100 task instructions each (potential overlap), select $M = 2$ clients per communication round, train each selected client for $E = 3$ epochs per round, and run $T = 70$ rounds, for a sequential training depth of $T \cdot E = 210$ local epochs along each client's path (the x-axis of our training curves). Across the $M = 2$ clients sampled per round, the federation performs 420 client-epochs of rollouts in total. This $(M, E, |X_i| = 100)$ configuration is justified by the hyperparameter sensitivity ablation in Appendix F.1. Each local epoch draws a minibatch of 64 rollout trajectories, sampling tasks iteratively with replacement from local data (for GRPO, 8 tasks with $G = 8$ rollouts each; for PPO, 64 tasks with one rollout each). We instantiate FEDAGENT with GRPO (Shao et al., 2024). The PPO counterpart of the table is reported in Appendix F.2 with consistent conclusions. Following (Liu et al., 2024a), we compare against two baselines: **Centralized Agent Training** (full dataset, matched at the same 210 epochs of the same per-epoch sampling) and **Local Agent Training** (single client's dataset; we report three random indices 21, 42, 84).

Results. Table 1 and Figure 4 answer (Q1) affirmatively: **FEDAGENT matches centralized agent training and outperforms single-client local training, often substantially**, holding 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 ALFWorld with `Qwen2.5-1.5B-Instruct`, FEDAGENT reaches 63.0% total success rate versus 12.5–35.9% for local training and 56.2% for centralized. On WebShop with `Llama-3.2-3B-Instruct`, the corresponding numbers 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. 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 nearly identical end-of-training success rates of ~60% on both benchmarks. On WebShop, centralized leads by up to ~15 points mid-training before FEDAGENT closes

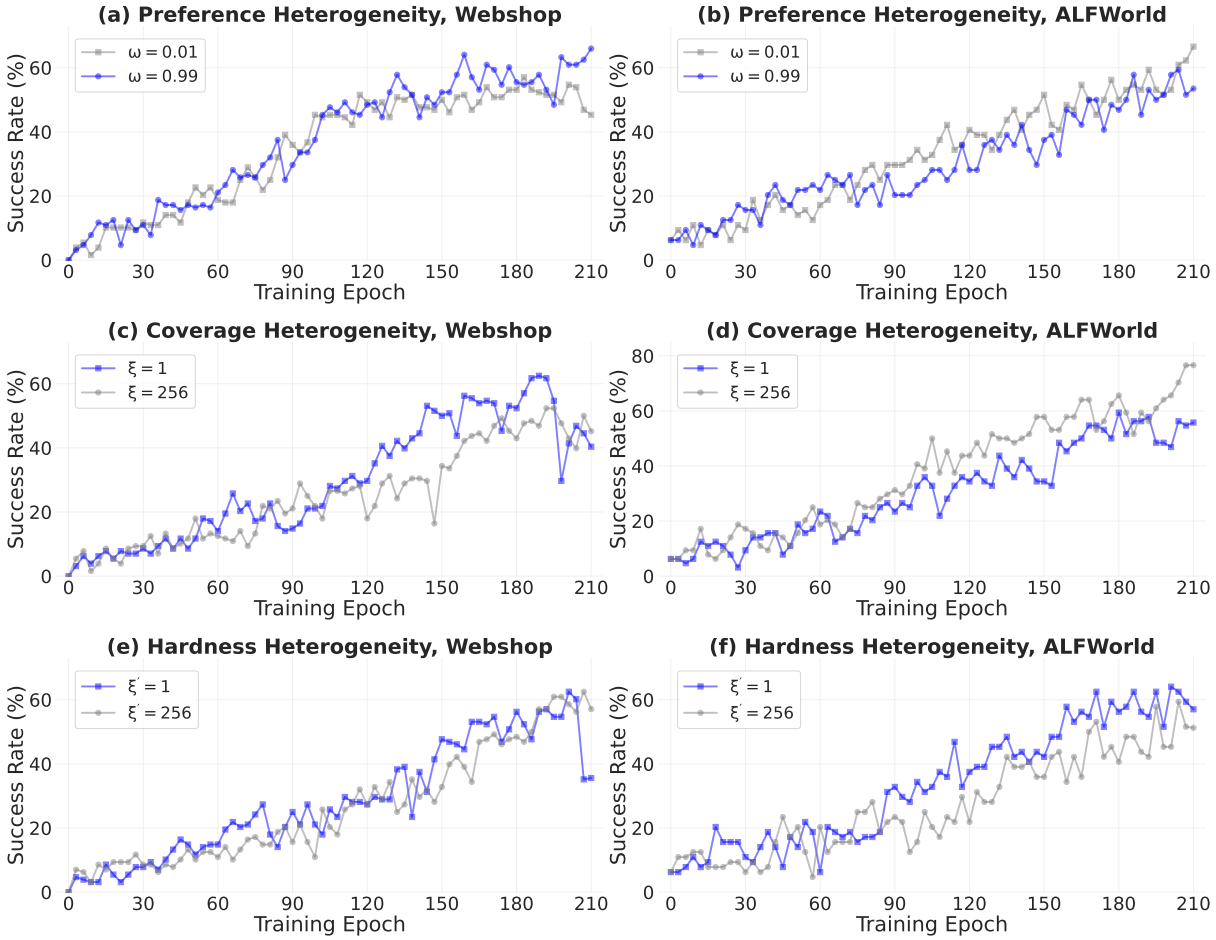

Figure 5 | **Training dynamics of FEDAGENT under task-level heterogeneity.** The policy optimization algorithm is GRPO. The PPO counterpart is in Appendix F.3. Panels (a,b) show preference, (c,d) coverage, and (e,f) hardness, on WebShop and ALFWorld. The federated training curve under extreme heterogeneity (blue: $\omega = 0.99$, $\xi = 1$, $\xi' = 1$) closely tracks the near-uniform baseline (grey: $\omega = 0.01$, $\xi = 256$, $\xi' = 256$), aligning with **Pattern A**.

the gap. On ALFWorld, which shows the higher run-to-run variability typical of agent RL, FEDAGENT matches or slightly exceeds centralized. This resolves (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-matches-centralized was previously established.

5.2. Impact of Task-Level Heterogeneity

Setup. We use §3.2 partition algorithms to vary each sub-type independently: PREFERENCEPARTITION(ω), COVERAGEPARTITION(ξ), and HARDNESSPARTITION(ξ'). We vary each hyperparameter from a near-uniform setting (grey: $\omega = 0.01$, $\xi = 256$, $\xi' = 256$) to an extreme heterogeneous setting (blue: $\omega = 0.99$, $\xi = 1$, $\xi' = 1$). The resulting client distributions are visualized in Figure 3 (preference) and Appendix K.3.2–K.3.3 (coverage, hardness). Other settings (100 clients, 210 total epochs, `Qwen2.5-1.5B-Instruct`) match §5.1.

Results. Figure 5 shows that FEDAGENT **remains robust even under extreme task-level heterogeneities**, across all three sub-types (panels a,b: preference; c,d: coverage; e,f: hardness).

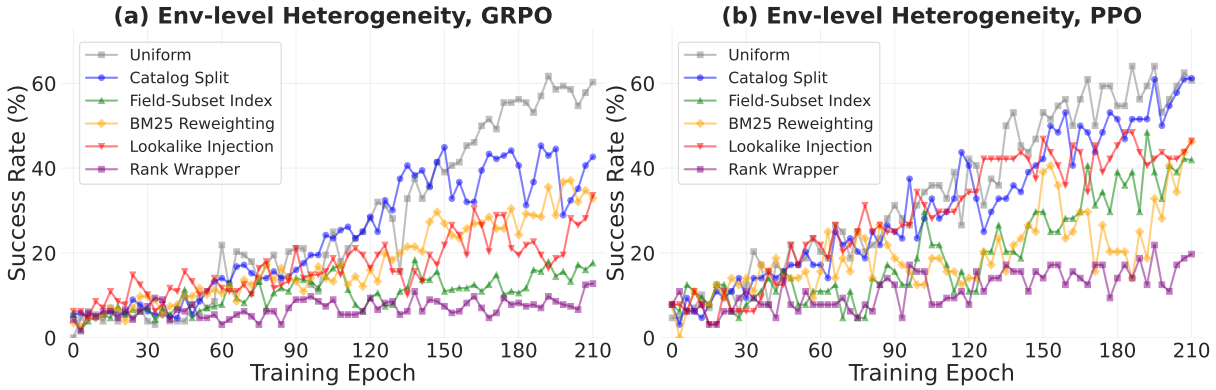

Figure 6 | **Training dynamics of FEDAGENT on WebShop under environment-level heterogeneity**, GRPO (left) and PPO (right). The grey *Uniform* curve is the homogeneous-environment (single-env) baseline of §5.1. The five env-variants (§3.3) form the **stable → degrade → collapse** spectrum: under GRPO, Catalog Split plateaus stably closest to the baseline (**Pattern C**, high end). Field-Subset Index, BM25 Reweighting, and Lookalike Injection plateau lower but remain stable (**Pattern C**). Rank Wrapper stays near ~10–13% (**Pattern D**). Under PPO, Catalog Split attains the baseline (**Pattern B**) and no variant exhibits Pattern D.

Moving each axis from near-uniform (grey) to its extreme (blue) shifts the converged plateau only marginally, and in several panels the extreme-heterogeneity curve even converges *above* the near-uniform one. The largest separation appears in Coverage/ALFWorld (panel d), yet even there it is comparable to the run-to-run variability we observe in LLM-agent RL training, where validation curves can vary by 10–20 points from one round to the next under sparse episode rewards and noisy on-policy rollouts. The extreme-vs-uniform spread therefore lies within this training-noise floor rather than reflecting a systematic penalty. This is Pattern A: under task-level heterogeneity, the population objective collapses to a single mixture expectation (Theorem 1), so the loss landscape faced by the optimizer is invariant to how tasks are split across clients. The empirical curves confirm this: the converged plateau is indistinguishable from the near-uniform baseline up to this training noise. This stands in sharp contrast to *environment*-level heterogeneity (§5.3): the same federation can there degrade far beyond the noise floor, up to collapse (Pattern D).

5.3. Impact of Environment-Level Heterogeneity

Setup. For each WebShop env-variant (§3.3), we run a federated experiment where clients differ along that variant's perturbation axis (per-client catalog subsets for Catalog Split, per-client (k_1, b) combinations for BM25 Reweighting, etc.). The task partition is held *uniform* as in §5.1 so that any divergence is attributable purely to environment-level heterogeneity. We match the §5.1 configuration in all other respects: 100 clients, 2 clients per round over 70 rounds (210 total local epochs), and `Qwen2.5-1.5B-Instruct` as the agent backbone. We instantiate FEDAGENT with both GRPO and PPO as policy optimizers to compare their robustness under env-level adversarial constructions. The four discrete-pool variants (Field-Subset Index, BM25 Reweighting, Lookalike Injection, Rank Wrapper) each use a pool of $|\mathcal{V}| = 4$ environment configurations, assigned deterministically and near-uniformly to the 100 clients (Appendix L.3). Catalog Split instead perturbs each client's catalog independently at dispersion `env_div` = 1.0, `keep_ratio` = 0.7. Full per-variant settings are in Appendix I.6.

Results. In sharp contrast to §5.2, Figure 6 shows that FEDAGENT **is non-robust under environment-level heterogeneity**: the five env-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 unchanged for everyone, so (C2) holds and the structural conditions place the achievable plateau at the homogeneous-environment baseline ((C1) holds only after abstracting states to their reward-relevant content, see Appendix L). 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 orthogonal optimizer axis of §4.4) rather than a structural floor. **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 optimal query and click order differ across clients on some states and (C1)/(C2) only partially hold. GRPO plateaus span ~15–35% (Pattern C). PPO raises Field-Subset Index to ~42%, while BM25 Reweighting remains in a similar, unstable ~25–40% band under both optimizers. **Lookalike Injection and Rank Wrapper** break the conditions most aggressively: the former jointly perturbs content and matching by injecting per-client lookalike products, while the latter post-processes search results (shuffle or invert the ranking) so the same query returns different orderings across clients. Both break (C1)/(C2). They differ on (C3). Lookalike retains a usable partial (C3): the attack is exposed once the agent inspects the specific reward component the lookalikes degrade (*e.g.* price). For Rank Wrapper, (C3) effectively fails: recovering the mismatch would require cross-query in-context inference that the 1.5B backbone does not consistently realize. This difference in (C3) usability decides which variant actually reaches Theorem 4's worst-case floor. Under GRPO, **Rank Wrapper** stays at ~10–13% with round-to-round oscillation and no learning progress (Pattern D). **Lookalike Injection**, despite being the more elaborate construction, exploits its inspectable cue: it degrades but recovers to ~34% by the end of training (Pattern C, comparable to BM25 Reweighting). Under PPO, the clipped per-step update raises both into Pattern C (Lookalike Injection: ~46%, Rank Wrapper: ~20%). **The B/C/D spectrum is thus algorithm-agnostic, but the pattern that actually appears for each variant depends on the optimizer's local robustness.** PPO's clipped update moves the GRPO Pattern D variant (Rank Wrapper) back to Pattern C and raises the Catalog Split, Field-Subset Index, and Lookalike Injection plateaus. BM25 Reweighting is the exception, plateauing similarly under both optimizers.

6. Conclusion

We explored **FEDAGENT**, a decentralized RL paradigm for training LLM agents without sharing local data. Grounded in the **Input-Dynamics Asymmetry** of task-augmented MDPs, we formalized **Agent Heterogeneity** at two levels (task and environment) and proved an **Asymmetric Robustness Mechanism**: FEDAGENT is robust to task-level heterogeneity (Pattern A) but worst-case non-robust to environment-level (Pattern D), with three sufficient conditions (C1)–(C3) recovering full or approximate robustness (Patterns B/C). Experiments on WebShop and ALFWorld under GRPO and PPO support this mechanism, positioning FEDAGENT as a promising paradigm for training LLM agents without centralizing user data (its privacy properties and their limits are discussed in Appendix H.2).

References

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.

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, Austria, May 3-7, 2021. OpenReview.net, 2021. URL https://openreview.net/forum?id=B7v4QMR6Z9w.

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, 2024.

D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. volume 30, 2017.

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.

H. Bai, Y. Zhou, J. Pan, M. Cemri, A. Suhr, S. Levine, and A. Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. In A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/1704ddd0bb89f159dfe609b32c889995-Abstract-Conference.html.

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. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024b. URL http://papers.nips.cc/paper_files/paper/2024/hash/1a134b50202088aa8c595cc99b310e5a-Abstract-Conference.html.

M. Balunovic, D. I. Dimitrov, N. Jovanovic, and M. T. 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/32375260090404f907ceae19f3564a7e-Abstract-Conference.html.

J. Bian, L. Wang, L. Zhang, and J. Xu. Fedalt: Federated fine-tuning through adaptive local training with rest-of-the-world lora. *CoRR*, abs/2503.11880, 2025. doi: 10.48550/ARXIV.2503.11880. URL https://doi.org/10.48550/arXiv.2503.11880.

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 B. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *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.

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.

K. Chen, M. F. 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, 2025a. doi: 10.48550/ARXIV.2502.01600. URL https://doi.org/10.48550/arXiv.2502.01600.

S. Chen, T. Zhu, Z. Wang, J. Zhang, K. Wang, S. Gao, T. Xiao, Y. W. Teh, J. He, and M. Li. Internalizing world models via self-play finetuning for agentic RL. *CoRR*, abs/2510.15047, 2025b. doi: 10.48550/ARXIV.2510.15047. URL https://doi.org/10.48550/arXiv.2510.15047.

X. Chen, Y. Shi, Q. Lan, Y. Qiu, and X. Gu. Fed-se: Federated self-evolution for privacy-constrained multi-environment LLM agents. *CoRR*, abs/2512.08870, 2025c. doi: 10.48550/ARXIV.2512.08870. URL https://doi.org/10.48550/arXiv.2512.08870.

Z. Chen, H. Cui, E. Wu, and Y. Xi. Efficient adaptive federated optimization of federated learning for iot. *arXiv preprint arXiv:2206.11448*, 2022.

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. *CoRR*, abs/2511.03773, 2025d. doi: 10.48550/ARXIV.2511.03773. URL https://doi.org/10.48550/arXiv.2511.03773.

M. Cheng, J. Ouyang, S. Yu, R. Yan, Y. Luo, Z. Liu, D. Wang, Q. Liu, and E. Chen. Agent-r1: Training powerful llm agents with end-to-end reinforcement learning. *arXiv preprint arXiv:2511.14460*, 2025.

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.

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. 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.

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.

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

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. *CoRR*, abs/2601.22149, 2026. doi: 10.48550/ARXIV.2601.22149. URL https://doi.org/10.48550/arXiv.2601.22149.

C. T. Dinh, N. H. Tran, and T. D. Nguyen. Personalized federated learning with moreau envelopes. In H. Larochelle, M. Ranzato, R. Hadsell, M. 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.

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*, volume ACL 2025 of *Findings of ACL*, pages 15267–15291. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.791/.

A. Fallah, A. Mokhtari, and A. E. Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In H. Larochelle, M. Ranzato, R. Hadsell, M. 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.

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. N. Dauphin, P. 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.

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. 2024.

L. Feng, Z. Xue, T. Liu, and B. An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025.

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.

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.

H. Gao, J. Geng, W. Hua, M. Hu, X. Juan, H. Liu, S. Liu, J. Qiu, X. Qi, Y. Wu, H. Wang, H. Xiao, Y. Zhou, S. Zhang, J. Zhang, J. Xiang, Y. Fang, Q. Zhao, D. Liu, Q. Ren, C. Qian, Z. Wang, M. Hu, H. Wang, Q. Wu, H. Ji, and M. Wang. A survey of self-evolving agents: On path to artificial super intelligence. *CoRR*, abs/2507.21046, 2025. doi: 10.48550/ARXIV.2507.21046. URL https://doi.org/10.48550/arXiv.2507.21046.

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. 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.

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.

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.

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*, 2025a.

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, 2025b. URL https://openreview.net/forum?id=iX3uESGdsO.

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*, 2025c.

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.

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.

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*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/huang25ae.html.

Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora. Evaluating gradient inversion attacks and defenses in federated learning. volume 34, pages 7232–7241, 2021.

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.

N. Hyeon-Woo, M. Ye-Bin, and T.-H. Oh. Fedpara: Low-rank hadamard product for communication-efficient federated learning. 2021.

C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. R. 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.

B. Jin, H. Zeng, Z. Yue, D. Wang, H. Zamani, and J. Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516, 2025. doi: 10.48550/ARXIV.2503.09516. URL https://doi.org/10.48550/arXiv.2503.09516.

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.

P. Kairouz and H. B. McMahan. Advances and open problems in federated learning. *Foundations and trends in machine learning*, 14(1-2):1–210, 2021.

S. M. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In C. Sammut and A. G. Hoffmann, editors, *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*, pages 267–274. Morgan Kaufmann, 2002.

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

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

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. Szepesvári, 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.

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.

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 R. Baeza-Yates and F. Bonchi, editors, *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.

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 M. 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.

G. Lan, D. Han, A. Hashemi, V. Aggarwal, and C. 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.

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

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, 2021a.

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, 2020a.

T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. volume 2, pages 429–450, 2020b.

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, 2021b. URL http://proceedings.mlr.press/v139/li21h.html.

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, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020c. URL https://openreview.net/forum?id=HJxNAnVtDS.

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, Austria, May 3-7, 2021*. OpenReview.net, 2021c. URL https://openreview.net/forum?id=6YEQUn0QICG.

Z. Li, G. Long, and T. Zhou. Federated recommendation with additive personalization. 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=xkXdE81mOK.

X. Liang, Y. Liu, T. Chen, M. Liu, and Q. Yang. Federated transfer reinforcement learning for autonomous driving. *CoRR*, abs/1910.06001, 2019. URL http://arxiv.org/abs/1910.06001.

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*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/liao25c.html.

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*, 2023.

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. 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.

Y. Lin, S. Han, H. Mao, Y. Wang, and B. 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.

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, 2024.

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.

B. Liu, N. Lv, Y. Guo, and Y. Li. Recent advances on federated learning: A systematic survey. *Neurocomputing*, 597:128019, 2024a.

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*, 2025a.

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 20743–20757, 2025b.

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. AAAI Press, 2026. doi: 10.1609/AAAI.V40I38.40487. URL https://doi.org/10.1609/aaai.v40i38.40487.

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, 2024b. URL https://openreview.net/forum?id=zAdUB0aCTQ.

X. Liu, K. Wang, Y. Wu, F. Huang, Y. Li, J. Zhang, and J. Jiao. Agentic reinforcement learning with implicit step rewards. *arXiv preprint arXiv:2509.19199*, 2025c.

Z. Liu, J. Chai, X. Zhu, S. Tang, R. Ye, B. Zhang, L. Bai, and S. Chen. Ml-agent: Reinforcing LLM agents for autonomous machine learning engineering. *CoRR*, abs/2505.23723, 2025d. doi: 10.48550/ARXIV.2505.23723. URL https://doi.org/10.48550/arXiv.2505.23723.

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 X. 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.

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.

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.

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.

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.

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*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

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/.

J. Peng, Y. Liu, R. Zhou, C. Fleming, Z. Wang, A. García, and M. Hong. Hiper: Hierarchical reinforcement learning with explicit credit assignment for large language model agents. *CoRR*, abs/2602.16165, 2026. doi: 10.48550/ARXIV.2602.16165. URL https://doi.org/10.48550/arXiv.2602.16165.

I. Petrov, D. I. Dimitrov, M. Baader, M. N. Müller, and M. T. Vechev. DAGER: exact gradient inversion for large language models. In A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 38: 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.

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.

J. Qi, Q. Zhou, L. Lei, and K. Zheng. Federated reinforcement learning: Techniques, applications, and open challenges. *CoRR*, abs/2108.11887, 2021. URL https://arxiv.org/abs/2108.11887.

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.

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. A. 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 / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/qin24a.html.

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.

C. Rawles, A. Li, D. Rodriguez, O. Riva, and T. P. 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.

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.

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

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.

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.

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

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, 2025a. URL https://openreview.net/forum?id=B8akWa62Da.

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, 2025b. doi: 10.18653/V1/2025.EMNLP-MAIN.239. URL https://doi.org/10.18653/v1/2025.emnlp-main.239.

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.

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.

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.

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.

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

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.

R. Singhal, K. Ponkshe, and P. Vepakomma. Fedex-lora: Exact aggregation for federated and efficient fine-tuning of foundation models. 2024.

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. 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.

F. Spadea and O. Seneviratne. Federated fine-tuning of large language models: Kahneman-tversky vs. direct preference optimization. In G. Long, M. Blumestein, Y. Chang, L. Lewin-Eytan, Z. H. Huang, and E. Yom-Tov, editors, *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.

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.

S. U. Stich. Local sgd converges fast and communicates little. 2018.

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=NLPzL6HWN1.

Z. Wan, Y. Li, X. Wen, Y. Song, H. Wang, L. Yang, M. Schmidt, J. Wang, W. Zhang, S. Hu, et al. Rema: Learning to meta-think for llms with multi-agent reinforcement learning. *arXiv preprint arXiv:2503.09501*, 2025.

H. Wang, M. Yurochkin, Y. Sun, D. S. Papailiopoulos, and Y. Khazaeni. Federated learning with matched averaging. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020a. URL https://openreview.net/forum?id=BkluqlSFDS.

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. A. 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 / OpenReview.net, 2024a. URL https://proceedings.mlr.press/v235/wang24v.html.

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. 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*, 2020b. URL https://proceedings.neurips.cc/paper/2020/hash/564127c03caab942e503ee6f810f54fd-Abstract.html.

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. *CoRR*, abs/2510.16907, 2025a. doi: 10.48550/ARXIV.2510.16907. URL https://doi.org/10.48550/arXiv.2510.16907.

L. Wang, J. Bian, L. Zhang, and J. Xu. Adaptive lora experts allocation and selection for federated fine-tuning. 2025b.

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. 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, 2024b. doi: 10.18653/V1/2024.ACL-LONG.510. URL https://doi.org/10.18653/v1/2024.acl-long.510.

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

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, 2020c. doi: 10.1109/JIOT.2020.2986803. URL https://doi.org/10.1109/JIOT.2020.2986803.

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. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, editors, *Advances in Neural*

Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024, 2024c. URL http://papers.nips.cc/paper_files/paper/2024/hash/28312c9491d60ed0c77f7fff4ad86dd1-Abstract-Conference.html.

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*, 2025c.

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. *CoRR*, abs/2602.10090, 2026. doi: 10.48550/ARXIV.2602.10090. URL https://doi.org/10.48550/arXiv.2602.10090.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. volume 35, pages 24824–24837, 2022.

Q. Wei, S. Zeng, C. Li, W. Brown, O. Frunza, W. Deng, A. Schneider, Y. Nevmyvaka, Y. K. Zhao, A. Garcia, et al. Reinforcing multi-turn reasoning in llm agents via turn-level reward design. *arXiv preprint arXiv:2505.11821*, 2025a.

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

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. *CoRR*, abs/2502.18449, 2025c. doi: 10.48550/ARXIV.2502.18449. URL https://doi.org/10.48550/arXiv.2502.18449.

Y. Wei, Z. Sun, E. McMilin, J. Gehring, D. Zhang, G. Synnaeve, D. Fried, L. Zhang, and S. I. Wang. Toward training superintelligent software agents through self-play SWE-RL. *CoRR*, abs/2512.18552, 2025d. doi: 10.48550/ARXIV.2512.18552. URL https://doi.org/10.48550/arXiv.2512.18552.

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, 2025e. doi: 10.18653/V1/2025.EMNLP-MAIN.401. URL https://doi.org/10.18653/v1/2025.emnlp-main.401.

J. Wen, H. Dai, J. He, M. Xi, S. Xiao, and J. Yang. Federated offline reinforcement learning with multimodal data. *IEEE transactions on consumer electronics*, 70(1):4266–4276, 2023.

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.

B. E. Woodworth, K. K. Patel, and N. Srebro. Minibatch vs local sgd for heterogeneous distributed learning. *Advances in Neural Information Processing Systems*, 33:6281–6292, 2020.

F. Wu, Z. Li, Y. Li, B. Ding, and J. Gao. Fedbiot: LLM local fine-tuning in federated learning without full model. In R. Baeza-Yates and F. Bonchi, editors, *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, 2024a. doi: 10.1145/3637528.3671897. URL https://doi.org/10.1145/3637528.3671897.

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.

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

Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversation framework. *CoRR*, abs/2308.08155, 2023. doi: 10.48550/ARXIV.2308.08155. URL https://doi.org/10.48550/arXiv.2308.08155.

Z. Xi, Y. Ding, W. Chen, B. Hong, H. Guo, J. Wang, D. Yang, C. Liao, X. Guo, W. He, S. Gao, L. Chen, R. Zheng, Y. Zou, T. Gui, Q. Zhang, X. Qiu, X. Huang, Z. Wu, and Y. Jiang. Agentgym: Evolving large language model-based agents across diverse environments. *CoRR*, abs/2406.04151, 2024. doi: 10.48550/ARXIV.2406.04151. URL https://doi.org/10.48550/arXiv.2406.04151.

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.

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.

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. *CoRR*, abs/2602.14721, 2026. doi: 10.48550/ARXIV.2602.14721. URL https://doi.org/10.48550/arXiv.2602.14721.

B. Xiong, X. Yang, Y. Song, Y. Wang, and C. Xu. Pilot: Building the federated multimodal instruction tuning framework. In T. Walsh, J. Shah, and Z. Kolter, editors, *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. AAAI Press, 2025a. doi: 10.1609/AAAI.V39I20.35476. URL https://doi.org/10.1609/aaai.v39i20.35476.

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, 2025b. URL https://openreview.net/forum?id=BfUDZGqCAu.

F. F. Xu, Y. Song, B. Li, Y. Tang, K. Jain, M. Bao, Z. Z. Wang, X. Zhou, Z. Guo, M. Cao, et al. Theagentcompany: benchmarking llm agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161*, 2024.

M. XU, X. Chen, Z. Chen, S. Wang, and J. Wang. Offline federated deep reinforcement learning with awareness of expected returns and policy inconsistency. 2025. URL https://openreview.net/forum?id=GWjvweJDnG.

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.

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. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 38: 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.

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. ijcai.org, 2025. doi: 10.24963/IJCAI.2025/1196. URL https://doi.org/10.24963/ijcai.2025/1196.

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*, 2022a. URL http://papers.nips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. 2022b.

S. Yao, D. Yu, J. Zhao, I. Shafran, T. 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.

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):79:1–79:44, 2024a. doi: 10.1145/3625558. URL https://doi.org/10.1145/3625558.

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 R. Baeza-Yates and F. Bonchi, editors, *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, 2024b. doi: 10.1145/3637528.3671582. URL https://doi.org/10.1145/3637528.3671582.

X. Yi, Y. Liu, B. Yang, and J. Zhang. Federated continual learning via orchestrating multi-scale expertise. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=AlSHcopwHi.

P. Yu, L. Wynter, and S. H. Lim. Federated reinforcement learning for portfolio management. In H. Ludwig and N. Baracaldo, editors, *Federated Learning - A Comprehensive Overview of Methods and Applications*, pages 467–482. Springer, 2022. doi: 10.1007/978-3-030-96896-0_21. URL https://doi.org/10.1007/978-3-030-96896-0_21.

Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, T. Fan, G. Liu, L. Liu, X. Liu, H. Lin, Z. Lin, B. Ma, G. Sheng, Y. Tong, C. Zhang, M. Zhang, W. Zhang, H. Zhu, J. Zhu, J. Chen, J. Chen, C. Wang, H. Yu, W. Dai, Y. Song, X. Wei, H. Zhou, J. Liu, W. Ma, Y. Zhang, L. Yan, M. Qiao, Y. Wu, and M. Wang. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476, 2025a. doi: 10.48550/ARXIV.2503.14476. URL https://doi.org/10.48550/arXiv.2503.14476.

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, 2025b. doi: 10.48550/ARXIV.2506.00320. URL https://doi.org/10.48550/arXiv.2506.00320.

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.

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.

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. P. 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. *Trans. Mach. Learn. Res.*, 2026, 2026a. URL https://openreview.net/forum?id=RY19y2RI1O.

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, 2025a. doi: 10.48550/ARXIV.2510.04206. URL https://doi.org/10.48550/arXiv.2510.04206.

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, 2024a. doi: 10.1109/ICASSP48485.2024.10447454. URL https://doi.org/10.1109/ICASSP48485.2024.10447454.

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. *CoRR*, abs/2510.08558, 2025b. doi: 10.48550/ARXIV.2510.08558. URL https://doi.org/10.48550/arXiv.2510.08558.

K. Zhang, K. Tian, R. Liu, S. Zeng, X. Zhu, G. Jia, Y. Fan, X. Lv, Y. Zuo, C. Jiang, et al. Marti: A framework for multi-agent llm systems reinforced training and inference. In *The Fourteenth International Conference on Learning Representations*, 2025c.

Y. Zhang, Z. Qin, Z. Wu, J. Hou, and S. Deng. Personalized federated fine-tuning for llms via data-driven heterogeneous model architectures. pages 5099–5110, 2026b.

Z. Zhang, J. Zhang, J. Huang, L. Qu, H. Zhang, Q. Wang, X. Zhou, and Z. Xu. Fewfedpit: Towards privacy-preserving and few-shot federated instruction tuning. *arXiv preprint arXiv:2403.06131*, 2024b.

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

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.

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.

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. A. 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 / OpenReview.net, 2024a. URL https://proceedings.mlr.press/v235/zhou24r.html.

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, 2024b. URL https://openreview.net/forum?id=oKn9c6ytLx.

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. A. 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 / OpenReview.net, 2024c. URL https://proceedings.mlr.press/v235/zhou24t.html.

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.

K. Zhu, H. Du, Z. Hong, X. Yang, S. Guo, Z. Wang, Z. Wang, C. Qian, R. 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.

L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. volume 32, 2019.

Y. Zhu and X. Gong. Single-loop federated actor-critic across heterogeneous environments. In T. Walsh, J. Shah, and Z. Kolter, editors, *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. AAAI Press, 2025. doi: 10.1609/AAAI.V39I21.34469. URL https://doi.org/10.1609/aaai.v39i21.34469.

Appendix

A Author Contributions 38

B Notation 39

C Reproducibility Statement 42

D Broader Impact 44

E Limitations 46

F More Experiment Results 47

F.1 Hyperparameter Sensitivity of FedAgent's Federated Configuration 47

F.2 PPO Results: Effectiveness under Uniform Client Distribution 48

F.3 PPO Results: Pattern A under Task-Level Heterogeneity 50

G Extended Related Work 53

G.1 Supervised Federated Learning 53

G.2 Federated Reinforcement Learning 54

G.3 Federated Learning for LLMs 56

G.4 LLM Agent Reinforcement Learning 58

H FedAgent Algorithm 62

H.1 Pseudocode 62

H.2 Intrinsic Privacy Properties 63

H.3 Communication Cost and Deployment Considerations 65

I Experimental Setup Details 67

I.1 Hardware and Software 67

I.2 Models and Tokenization 67

I.3 Federation Protocol 67

I.4 Policy Optimization Hyperparameters 68

I.5 Rollout Protocol 69

I.6 Heterogeneity Construction Protocol 69

I.7 Evaluation Protocol . . . 70
I.8 Reproducibility Checklist . . . 71

J Task-Level Heterogeneity Characterization **72**
J.1 Inter-Client Dispersion Measures . . . 72
J.2 Primary Properties of the Three Projections . . . 73
J.3 Definitional Orthogonality and First-Order Sufficiency . . . 74

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

L Environment-Level Heterogeneity Variants **84**
L.1 A Four-Stage Transition Pipeline . . . 84
L.2 The WebShop Pipeline as a Concrete Instantiation . . . 85
L.3 Deterministic Per-Client Variant Assignment . . . 86
L.4 Variant 1: Catalog Split (Stage 1) . . . 87
L.5 Variant 2: Field-Subset Index (Stage 2) . . . 88
L.6 Variant 3: BM25 Reweighting (Stage 3) . . . 90
L.7 Variant 4: Lookalike Injection (Stages 1+3) . . . 91
L.8 Variant 5: Rank Wrapper (Stage 4) . . . 92

M Asymmetric Robustness Mechanism: Theory and Proofs **94**
M.1 Realizability Assumptions for LLM Agents . . . 94

M.2 Proofs of Theorems 1 and 2 (Task-Level Robustness) . . . 95
M.3 Proofs of Theorems 3 and 4 (Env-Level Non-Robustness) . . . 97
M.4 Recovery Theorem Proof ((C1)–(C3)) . . . 100
M.5 Convergence Implications via ζ^2 . . . 103

N Empirical Patterns: Extended Discussion **106**
N.1 The Unified Causal Chain . . . 106
N.1.1 The Single Architectural Fact: Input-Dynamics Asymmetry . . . 107
N.1.2 The Two Sides of Asymmetry . . . 107
N.1.3 The Full Causal Chain . . . 108
N.2 Pattern A: Plateau Identity vs. Trajectory Drift . . . 109
N.3 Pattern B vs. Pattern A: Same Plateau, Different Mechanism . . . 109
N.4 The B/C/D Spectrum and Its Slack Parameters . . . 110
N.5 Pattern D Mechanism: Decomposing Oscillation and Forgetting . . . 110
N.6 ζ^2 as the Mechanism Bridge: Rate vs. Quality . . . 111
N.7 Diagnostic Decision Tree . . . 112
N.8 Operational Mitigation Protocol . . . 113
N.8.1 Pre-Deployment Tests for (C1), (C2), (C3) . . . 113
N.8.2 The Quantitative Per-Client Gap Bound . . . 114
N.8.3 Mitigations Indexed by the Dominant Slack . . . 114
N.8.4 The Uncertain Regime: When All (C) Tests Fail but Collapse Does Not Occur 115
N.9 Predictive and Falsifiable Use of the Mechanism . . . 115
N.9.1 Connecting Theory and Experiment . . . 116
N.9.2 Falsifiability per Pattern . . . 116
N.9.3 Predictive Use Given Deployment Characteristics . . . 117

O Why the Mechanism Is Sharper on LLM Agents **121**
O.1 Task-as-Input: Pre-Training as Implicit Realizability . . . 121
O.2 Env-as-Implicit-Knowledge: Pre-Training Prior Degradation (Informal Claim) . . . 122
O.3 (C3) Explicit Form: Env-Aware Prompting and Corollary 1 . . . 123
O.4 (C3) Implicit Form: In-Context Posterior Inference . . . 125

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. It 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.

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.

Table 2 | 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 J–K; the construction constants C_n, C_h are defined in Propositions 2 and 3 of Appendix K.2.

Symbol	Description
Federation setup	
N	total number of clients
M	clients selected per round
T	total communication rounds
E	local optimization epochs per round
$\lvert X_i \rvert$	per-client task pool size (sample budget)
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, $\lvert S_t \rvert = M$
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 space of $\mathcal{M}_i$
P_i	client i's transition kernel $P_i(s' \mid s, a)$
R_i or R	reward function (shared under Definitions 1–2)
γ	discount factor
H	episode horizon (max steps when $\gamma = 1$; $1/(1-\gamma)$ when $\gamma < 1$)
T_{hor}	decision rounds per episode in Theorem 4's chain construction (taken = H)
$R_{\max}$	per-step reward bound, $r_t \in [0, R_{\max}]$
μ_0	initial state distribution
τ	task descriptor (prompt input)
s_t, a_t, r_t	state, action, reward at step t
h_t	trajectory prefix (history) up to step t
$\mathcal{H}$	history (trajectory-prefix) space, $h_t \in \mathcal{H}$
χ	full trajectory $(\tau, s_0, a_0, r_0, \ldots, s_H)$; unrelated to the χ^2 divergence
Distributions and policies	
$\mathcal{D}_{\tau_i}$	per-client task distribution over $\mathcal{T}$
$\mathcal{D}_\tau$	shared task distribution under env-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 $\{\pi_\theta(a \mid s, \tau, h_t)\}$
$\pi^\star$	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; $V_{\max} := R_{\max} H$
$d^\pi_{\mathcal{M}_i}$	discounted state-action occupancy of π on $\mathcal{M}_i$

continued on next page

(Table 2 continued)

Symbol	Description
Objectives	
$\mathcal{J}(\pi;\tau,\mathcal{M})$	trajectory return of π on task τ in MDP $\mathcal{M}$
$\mathcal{J}_\tau(\theta)$	shorthand for $\mathcal{J}(\pi_\theta;\tau,\mathcal{M}_{\text{env}})$ (task-level proofs)
$\mathcal{J}_i(\theta)$	per-client expected return $\mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}\mathbb{E}_\chi[\sum_u \gamma^u r_u]$
$\mathcal{J}_{\text{fed}}(\theta)=\frac{1}{N}\sum_i \mathcal{J}_i(\theta)$	federated objective
$\theta^\star$	realizing parameter under (R) or (R′)
$\theta^\star_{\text{fed}}$	federated optimum, $\arg\max_\theta \mathcal{J}_{\text{fed}}(\theta)$
$\hat\theta_{\text{fed}}$	ϵ_{opt}-approximate federated maximizer
Heterogeneity measures (task-level)	
Δ^2_{pref}	preference dispersion, $\frac{1}{N}\sum_i \|p_i-\bar p\|_2^2$
Δ^2_{cov}	coverage dispersion, $\widehat{\text{CV}}^2(\{n_i\})$
Δ^2_{hard}	hardness dispersion, $\widehat{\text{Var}}_i(\rho_i)$
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 $\|X_i\|$
$L,\ L_{\min},\ L_{\max}$	per-client pool size and its bounds ($n_i\in[L_{\min},L_{\max}]$)
r	global overlap: average number of clients holding each task (distinct from r_t)
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 thresholded success rate $\Pr_{\tau\sim\mathcal{D}_{\tau_i}}[\rho(\tau)\ge\eta]$
$\bar p,\bar n,\bar\rho$	global means of p_i,n_i,ρ_i
ω,ξ,ξ'	partition hyperparameters (preference, coverage, hardness)
Heterogeneity measures (environment-level)	
δ	worst-case env 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
$\mathcal{V}$	env-variant pool of $\|\mathcal{V}\|$ configurations (Appendix L)
`env_div`	Catalog Split distractor-dispersion hyperparameter
`keep_ratio`	Catalog Split retention-fraction hyperparameter
Theory slacks	
ϵ_{approx}	capacity/realizability slack on the policy class
ϵ_{opt}	federated optimization slack of $\hat\theta_{\text{fed}}$
$\chi^2(P\,\|\,Q)$	chi-squared divergence between P and Q
ζ^2	gradient heterogeneity (cross-client gradient variance)
ζ^2_∞	$\lim_{t\to\infty}\zeta^2(\theta_t)$ along the training trajectory (Appendix N)
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	
$t^\star$	env-reveal step (integer), after which the classifier is reliable
π_0	default prefix policy generating $h_{t^\star}$ (followed for $t<t^\star$)
$\hat\imath$	env-identity classifier $\hat\imath:\mathcal{H}\to[N]$
$\hat P_i$	kernel estimator for client i's transition kernel P_i
η_{cls}	classifier slack, post-reveal misclassification rate

continued on next page

(Table 2 continued)

Symbol	Description
η_{ker}	kernel-modeling slack, $\sup_{s,a} D_{\mathrm{TV}}(\hat{P}_i, P_i)$
c_0	absolute constant in the recoverability condition (C3-rec)

C. Reproducibility Statement

We have made every effort to make FEDAGENT fully reproducible. The following paragraphs map each contribution to the appendix that substantiates it.

Code and data release. A public repository containing the full implementation of FEDAGENT (built on top of HuggingFace `transformers` and the `verl-agent` framework (Feng et al., 2025)), dataset preprocessing scripts for WebShop and ALFWorld, and all per-experiment configuration files is available at `https://github.com/sunblaze-ucb/FedAgent`. The repository does not bundle large benchmark data, trained checkpoints, or post-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**', $\mathbf{R}_{\text{env}}$, $\mathbf{R}_{\text{aug}}$, **LR**) are formally stated in Appendix M.1. Theorem 1 and Theorem 2 (task-level robustness, idealized and approximate) are proved in Appendix M.2. Theorem 3 and Theorem 4 (env-level non-robustness, existence and quantitative $\Omega(R_{\max} H \delta)$ lower bound) are proved in Appendix M.3. Theorem 5 (recovery via (C1)–(C3)) is proved in Appendix M.4. The cross-MDP simulation lemma (Lemma 2) and the convergence implications via the gradient-heterogeneity quantity ζ^2 are in Appendices M.3 and M.5 respectively. The formal foundation for the partition-driven heterogeneity experiments used in §3.2 is in Appendices J and K: 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 J. The three partition algorithms with single-hyperparameter monotone control (Propositions 1 to 3) and the four design properties (D1)–(D4) (Theorems 9 to 11) are in Appendix K.

Heterogeneity construction. Pseudocode for the three task-level partition procedures (PREFERENCEPARTITION, COVERAGEPARTITION, HARDNESSPARTITION) is in Appendix K.1. Per-client distribution figures verifying that each hyperparameter cleanly controls its dispersion measure are in Appendix K.3. The five WebShop env-level variants (Catalog Split, Field-Subset Index, BM25 Reweighting, Lookalike Injection, Rank Wrapper) are formalized at the MDP level in Appendix L, 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 I. 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 F.1. PPO counterparts of all main-text GRPO results are in Appendix F.2.

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 mean and standard deviation across 3 random seeds. The standard deviation is shown as a subscript, and 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, the U.S. HIPAA and 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 is often more identifying than preference labels alone. It thereby aligns agent training with the data minimization and purpose limitation principles that underlie modern privacy law. Appendix H.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 forbidden outright. 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 provides a path to collaborative agent training in these settings by replacing the data-sharing requirement with a parameter-aggregation requirement, and our structural guarantees (Theorems 1 and 4) give operators principled predictions, ahead of training, 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 (C1)–(C3) hold, to its environment) without their data ever leaving local storage. The Asymmetric Robustness Mechanism additionally makes personalization *principled* rather than ad hoc: under task-level heterogeneity and assumption (R), Theorem 1's mixture-equivalence implies that federated averaging need not average away small or minority preference 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 explicit per-client personalization or hierarchical aggregation is required, and the diagnostic identifies which conditions failed, giving downstream personalization methods a precise structural target.

Scalability and accessibility. The modular client structure of FEDAGENT means 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 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. It also broadens who shapes modern agent capabilities, reaching application domains (rare diseases, low-resource languages, regional public-sector workflows) that rarely attract centralized investment. Combined with parameter-efficient fine-tuning (LoRA,

adapters, zeroth-order methods), the per-client cost can be brought down 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), conditionally robust to environment-level heterogeneity when (C1), (C2), (C3) hold (Patterns B, C), and worst-case non-robust when they fail (Pattern D). Pattern D's flat or collapsing training curve is itself an actionable signal: it indicates that aggregation may be degrading the LLM's pre-trained behavior and suggests that environment-aware prompting, per-client personalization, or hierarchical aggregation should replace plain federated averaging. The deployment decision thus becomes observable from the training curve, before the cost of large-scale rollout is incurred.

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 structural 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 to either 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 delineate the scope of FEDAGENT along three axes.

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 axis along which to organize Agent Heterogeneity. It 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 M: 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 LLM-tailored and designed to be plausible for modern instruction-tuned LLMs operating on bounded-reward, episodic agent benchmarks (Appendix M.1 discusses when and how tightly they hold). (The gradient-heterogeneity quantity ζ^2 enters only the convergence-rate discussion of Appendix M.5, which inherits its boundedness assumption from the cited convergence literature rather than from our theorems.) The sufficient conditions (C1), (C2), (C3) provide a checklist that turns Theorem 4's worst-case $\Omega(R_{\max} H\delta)$ floor into an actionable diagnostic. We expect (C3) in particular to be the condition that applies for LLM agents in many real deployments, thanks to natural-language environment cues.

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, 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 embodied or multi-modal agents, larger model scales, and additional RL algorithms remains to be verified. The framework makes each of these settings a natural next direction.

F. More Experiment Results

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

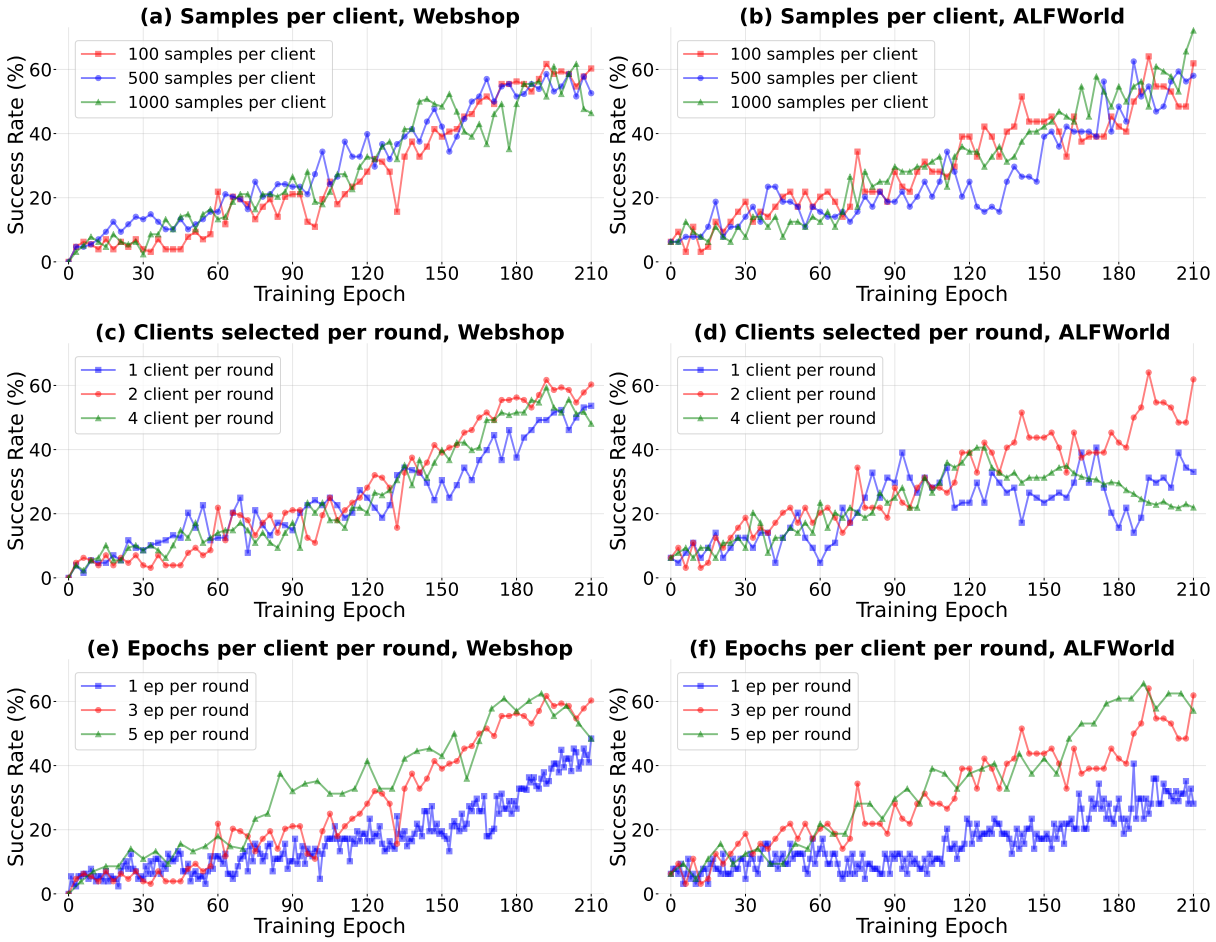

Figure 7 | **Sensitivity of FEDAGENT to federated configuration hyperparameters** on WebShop and ALFWorld. Panels (a,b): samples per client $|X_i|$; (c,d): clients per round M; (e,f): local epochs per round E. Across the tested values, E and M affect final performance most, while $|X_i|$ has a smaller, benchmark-dependent effect.

F.1. Hyperparameter Sensitivity of FedAgent's Federated Configuration

This subsection examines how FEDAGENT responds to the three standard federated optimization hyperparameters that govern its behavior *independently of the heterogeneity analysis* of §3–§4: per-client sample budget $|X_i|$, clients-per-round M, and local-epochs-per-round E. The goal is twofold: (i) justify the $M = 2$, $E = 3$, $|X_i| = 100$ configuration used throughout the main text, and (ii) provide design guidance for practitioners deploying FEDAGENT on new agent benchmarks.

Setup. We vary three dimensions: (i) **samples per client** $|X_i| \in \{100, 500, 1000\}$, controlling each client's local 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 GRPO as the policy optimizer, with a uniform task partition (no heterogeneity), so any sensitivity is attributable to the federated configuration alone.

Findings. Figure 7 reveals three sensitivity patterns. **(1) Local epochs E matters substantially.** Moving from $E = 1$ to $E \in \{3, 5\}$ yields significant performance gains, particularly after epoch ~100, indicating that shallow local updates do not fully exploit FEDAGENT's local-SGD-style optimization and that the aggregation cost amortizes well over deeper local epochs. $E = 3$ and $E = 5$ reach comparable converged plateaus on both benchmarks (panels e, f), with $E = 5$ showing at most a transient edge, so $E = 3$ already captures essentially all of the benefit at 40% less local compute per round. **(2) Clients-per-round M performs best at an intermediate value.** On ALFWorld, $M = 2$ outperforms both $M = 1$ (insufficient diversity in aggregated gradients) and $M = 4$ (excessive client drift weakens the global signal). The effect is milder on WebShop. **(3) Samples-per-client $|X_i|$ has the smallest, benchmark-dependent effect.** On WebShop the three budgets $|X_i| \in \{100, 500, 1000\}$ stay within ~ 3–5 pp (with 100 best), whereas on ALFWorld larger budgets help somewhat (1000 best by ~ 5–8 pp). About 100 tasks already provide solid per-client exploration on both benchmarks. We default to $|X_i| = 100$ because it is strong and inexpensive, not because performance strictly saturates. These findings justify the $M = 2$, $E = 3$, $|X_i| = 100$ configuration used in the main text and recommend it as a default starting point for new FEDAGENT deployments.

F.2. PPO Results: Effectiveness under Uniform Client Distribution

This subsection reports the PPO counterpart of §5.1, addressing (Q1) when the local policy optimizer is PPO (Schulman et al., 2017) (token-level clipped surrogate objective with a value baseline) instead of GRPO (Shao et al., 2024) (group-relative advantage that drops the critic). The goal is to verify that the central finding of the main text is not specific to GRPO's group-relative formulation but extends to PPO's clipped-update style: FEDAGENT matches centralized agent training and substantially outperforms single-client local training.

Setup. We mirror the configuration of §5.1 exactly except for the local optimizer. The dataset (WebShop / ALFWorld) is partitioned into 100 clients with $|X_i| = 100$ task instructions per client (potential overlap), $M = 2$ clients are selected per communication round, each selected client trains for $E = 3$ local epochs, and the protocol runs for $T = 70$ communication rounds for 210 total local epochs (this $(M, E, |X_i| = 100)$ configuration is justified by the hyperparameter sensitivity ablation in Appendix F.1). Each local epoch samples a minibatch of 64 tasks iteratively with replacement from local data (= 64 rollout trajectories under PPO's one rollout per task, matching GRPO's per-epoch trajectory budget of §5.1). 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. Following (Liu et al., 2024a), we compare against two baselines: **Centralized Agent Training** (full dataset, same total epochs, PPO local optimizer) and **Local Agent Training** (a single client's dataset; we report three random indices 21, 42, 84). Federated and centralized rows are averaged over three random seeds with standard deviations reported in subscript. 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.

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	62.5	26.6
Local (Client 42)	21.4	37.5	16.7	22.2	27.3	30.0	25.0	70.3	34.4
Local (Client 84)	7.1	12.5	33.3	11.1	9.1	10.0	14.1	60.9	40.6
Centralized	$54.8_{\pm 4.1}$	$\mathbf{83.3}_{\pm 7.2}$	$\mathbf{86.1}_{\pm 4.8}$	$18.5_{\pm 12.8}$	$15.2_{\pm 10.5}$	$\mathbf{86.7}_{\pm 5.8}$	$57.3_{\pm 5.0}$	$77.1_{\pm 3.9}$	$55.2_{\pm 3.9}$
FEDAGENT	$\mathbf{83.3}_{\pm 10.9}$	$45.8_{\pm 14.4}$	$44.4_{\pm 9.6}$	$\mathbf{74.1}_{\pm 6.4}$	$\mathbf{84.8}_{\pm 5.2}$	$46.7_{\pm 11.5}$	$\mathbf{64.6}_{\pm 1.8}$	$\mathbf{77.6}_{\pm 3.3}$	$\mathbf{59.9}_{\pm 2.4}$
Qwen2.5-3B-Instruct									
Local (Client 21)	7.1	12.5	16.7	11.1	18.2	10.0	12.5	70.3	39.1
Local (Client 42)	42.9	25.0	41.7	44.4	45.5	40.0	40.6	60.9	29.7
Local (Client 84)	21.4	37.5	25.0	22.2	36.4	30.0	28.1	73.4	45.3
Centralized	$\mathbf{85.7}_{\pm 7.1}$	$50.0_{\pm 12.5}$	$\mathbf{75.0}_{\pm 14.4}$	$\mathbf{81.5}_{\pm 6.4}$	$33.3_{\pm 13.9}$	$70.0_{\pm 10.0}$	$67.2_{\pm 6.2}$	$\mathbf{82.8}_{\pm 4.7}$	$\mathbf{60.9}_{\pm 4.7}$
FEDAGENT	$78.6_{\pm 12.4}$	$\mathbf{79.2}_{\pm 7.2}$	$41.7_{\pm 8.3}$	$51.9_{\pm 12.8}$	$\mathbf{72.7}_{\pm 9.1}$	$\mathbf{83.3}_{\pm 5.8}$	$\mathbf{67.7}_{\pm 0.9}$	$82.3_{\pm 7.0}$	$58.9_{\pm 7.0}$
Qwen2.5-7B-Instruct									
Local (Client 21)	42.9	25.0	50.0	22.2	27.3	40.0	35.9	70.3	42.2
Local (Client 42)	50.0	37.5	16.7	44.4	45.5	50.0	40.6	60.9	48.4
Local (Client 84)	21.4	12.5	41.7	11.1	9.1	10.0	18.8	75.0	37.5
Centralized	$88.1_{\pm 8.2}$	$\mathbf{79.2}_{\pm 14.4}$	$\mathbf{77.8}_{\pm 9.6}$	$\mathbf{77.8}_{\pm 11.1}$	$\mathbf{87.9}_{\pm 5.2}$	$73.3_{\pm 11.5}$	$\mathbf{81.2}_{\pm 6.8}$	$\mathbf{89.1}_{\pm 3.1}$	$69.8_{\pm 3.3}$
FEDAGENT	$\mathbf{90.5}_{\pm 4.1}$	$75.0_{\pm 12.5}$	$55.6_{\pm 4.8}$	$74.1_{\pm 6.4}$	$78.8_{\pm 10.5}$	$\mathbf{80.0}_{\pm 10.0}$	$76.0_{\pm 5.5}$	$88.5_{\pm 5.5}$	$\mathbf{70.8}_{\pm 5.5}$
Llama-3.2-3B-Instruct									
Local (Client 21)	35.7	37.5	41.7	**44.4**	9.1	50.0	35.9	68.8	40.6
Local (Client 42)	7.1	12.5	8.3	11.1	36.4	10.0	14.1	67.2	25.0
Local (Client 84)	14.3	25.0	16.7	22.2	18.2	20.0	18.8	60.9	32.8
Centralized	$59.5_{\pm 14.9}$	$\mathbf{45.8}_{\pm 14.4}$	$\mathbf{80.6}_{\pm 12.7}$	$22.2_{\pm 11.1}$	$\mathbf{63.6}_{\pm 9.1}$	$26.7_{\pm 5.8}$	$52.1_{\pm 2.4}$	$78.1_{\pm 7.8}$	$\mathbf{55.7}_{\pm 8.6}$
FEDAGENT	$\mathbf{71.4}_{\pm 14.3}$	$20.8_{\pm 7.2}$	$58.3_{\pm 14.4}$	$37.0_{\pm 12.8}$	$60.6_{\pm 13.9}$	$\mathbf{83.3}_{\pm 11.5}$	$\mathbf{57.8}_{\pm 4.1}$	$\mathbf{78.6}_{\pm 4.8}$	$54.2_{\pm 4.8}$

Table 3 | **Performance Comparison on ALFWorld and WebShop under PPO**, counterpart of Table 1. We report the averaged performance and the corresponding standard deviation for Centralized Training and FEDAGENT over three random seeds. For ALFWorld, the **Success Rate** (%) is reported. For WebShop, both the **Task Score** (0–100, the WebShop convention of 100× mean episode reward) and the **Success Rate** (%) are reported.

Findings. Across all four model scales and both benchmarks, Table 3 shows that **FEDAGENT with PPO matches centralized PPO within a few points in every configuration and outperforms single-client local PPO throughout**, mirroring the GRPO conclusion of §5.1. Concrete numbers: 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% (the one configuration where Centralized leads, and the 5.2 pp gap is 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), which 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

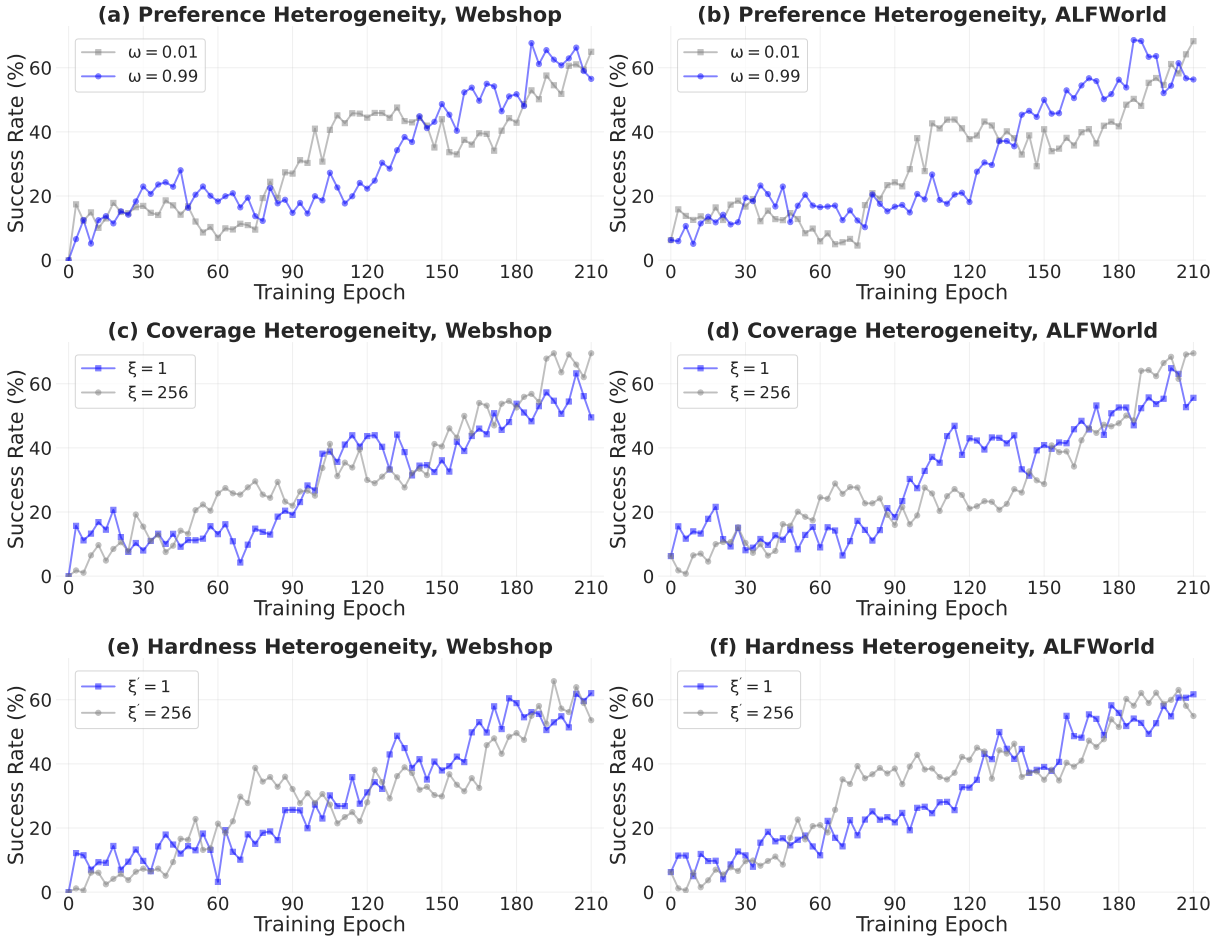

Figure 8 | **Training dynamics of FEDAGENT (PPO) under task-level heterogeneity**, counterpart of Figure 5. The federated curve under extreme task heterogeneity (blue) converges close to the near-uniform baseline (grey) across all three sub-types (preference, coverage, hardness) and both benchmarks (WebShop, ALFWorld). The residual differences, largest on coverage, stay within the run-to-run variability of agent RL. A transient slow-start appears under extreme hardness (ξ'=1, panels e, f) before closing the gap at convergence (around epoch ~100–140 depending on the benchmark), confirming **Pattern A** also holds under PPO.

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 read Table 3 at the aggregate level, where the two paradigms agree within a few points, rather than as either paradigm dominating specific skills. The effectiveness result of §5.1 is therefore not specific to GRPO: on every aggregate column of Table 3, Centralized leads FEDAGENT by at most 5.2 pp, while FEDAGENT leads by up to 7.3 pp.

F.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 Pattern A is predicted by Theorem 1 as a structural property of the task-augmented MDP under input-channel heterogeneity, independently of the optimizer: the population objective collapses to a single mixture expectation over $\bar{\mathcal{D}}_\tau = \frac{1}{N}\sum_i \mathcal{D}_{\tau_i}$, so the loss landscape faced by any policy-gradient estimator is

invariant to how tasks are split across clients. The present experiment empirically verifies this prediction for PPO, complementing the GRPO results in Figure 5. Together with the env-level PPO results already reported in panel (b) of Figure 6, this completes the PPO coverage of the four-pattern spectrum.

Setup. We mirror §5.2 exactly except for the local optimizer. The three task-level sub-types from §3.2 are varied independently via PREFERENCEPARTITION(ω), COVERAGEPARTITION(ξ), and HARDNESSPARTITION(ξ'), each moving from a near-uniform setting (grey: $\omega = 0.01$, $\xi = 256$, $\xi' = 256$) to an extreme heterogeneous setting (blue: $\omega = 0.99$, $\xi = 1$, $\xi' = 1$). See Appendix K.3.1, K.3.2, K.3.3 for the resulting per-client distributions on WebShop and ALFWorld. The federated configuration is otherwise identical to the previous subsection: 100 clients, $M = 2$ per round, $E = 3$ local epochs, $T = 70$ rounds, $|X_i| = 100$, batch of 64 tasks per local epoch, `Qwen2.5-1.5B-Instruct` backbone. The only change relative to §5.2 is that PPO replaces GRPO as the local policy-gradient estimator. The heterogeneity construction and partition algorithms are unchanged, so any difference relative to the GRPO curves of Figure 5 is attributable purely to the local-optimizer choice.

Findings. Figure 8 shows that **Pattern A holds under PPO across all three sub-types and both benchmarks**: at convergence, the blue curve (extreme heterogeneity) tracks the grey curve (near-uniform) closely on the preference and hardness panels and to within ~10–15 points on coverage. On WebShop (panels a, c, e), both curves plateau around 55–65% success rate (the extreme-hardness curve in panel e reaching ~ 62% by the final epochs). On ALFWorld (panels b, d, f), both converge to around 55–65% All-task success. Even the coverage separation, the largest of the three sub-types, is comparable to the per-checkpoint variability of agent RL rather than a systematic penalty, exactly the within-noise behavior predicted by Theorem 1. Beyond the convergence-level match, two finer features of the PPO curves are worth noting. **(1) Transient slow-start under extreme hardness.** For the hardness sub-type at $\xi' = 1$ (panels e, f), the extreme-heterogeneity curve lags behind the uniform baseline during roughly the first 90–150 epochs before closing the gap at convergence. This is consistent with the intuition behind HARDNESSPARTITION: when the success-rate quotas spread toward the extremes, clients at the hard end hold mostly unsuccessful tasks, so their early rollouts are zero-reward more often. PPO's clipped-surrogate update with a value baseline then takes longer to bootstrap a positive learning signal than in the near-uniform setting, where some clients receive easier tasks that yield non-zero advantages early. The slow-start does not affect the converged plateau, which is what Theorem 1 predicts (Pattern A is a statement about the population objective, not the optimization trajectory). **(2) Reduced training-curve variability relative to GRPO.** 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). We attribute the difference to PPO's clipping mechanism, which bounds the per-step policy displacement by construction. This suppresses the exploration-induced collapse that GRPO's group-relative advantage normalization can amplify when a batch happens to be all-zero-reward (advantages collapse to noise) or all-one-reward (advantages collapse to zero).

Combined with the env-level PPO results in panel (b) of Figure 6, the present experiment strengthens the optimizer-robustness reading beyond what GRPO alone can support: **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 shifts positions only *within* the B/C/D family**. Within the

env-level regime, the optimizer's local robustness modulates which env-variants fall into B vs. C vs. D. As discussed in §5.3, PPO's clipping-induced robustness margin moves the GRPO Pattern D variant (Rank Wrapper) back to Pattern C and raises most Pattern C plateaus (*e.g.* Lookalike Injection; BM25 Reweighting is the exception). The underlying B/C/D structure and the task-level Pattern A are preserved across both optimizers.

G. Extended Related Work

G.1. Supervised Federated Learning

Federated learning (FL) was introduced by McMahan et al. (2017) with the Federated Averaging (FedAvg) algorithm, which enables multiple clients to collaboratively train a shared model by communicating only model updates rather than raw data. 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 a subset of 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 $\boldsymbol{w}_k^{t+1}$; and the server 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 (Kairouz and McMahan, 2021; Li et al., 2020a). 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 (Hsu et al., 2019), feature distribution shift (Li et al., 2021c), and quantity imbalance (Gao et al., 2022; Ye et al., 2024a). Convergence analyses under non-i.i.d. data have been developed for FedAvg (Li et al., 2020c; Stich, 2018; Woodworth et al., 2020), establishing how heterogeneity slows convergence and motivating algorithm improvements. To address client drift, various methods have been proposed: FedProx (Li et al., 2020b) adds a proximal regularization term to the local objective. SCAFFOLD (Karimireddy et al., 2020) corrects for client drift using control variates. FedNova (Wang et al., 2020b) normalizes local updates to handle heterogeneous local computation. FedDyn (Acar et al., 2021) augments each device's objective with dynamic linear and quadratic penalty terms. MOON (Li et al., 2021a) 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. Chen et al. (2022) proposed FedOpt, a general framework applying adaptive server-side optimizers (FedAdam, FedAdagrad, FedYogi) to federated learning. FedMA (Wang et al., 2020a) aligns neural network layers via matched averaging, while knowledge distillation-based approaches such as FedDF (Lin et al., 2020) aggregate client models via ensemble distillation on the server. FedMD (Li and Wang, 2019) 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. PerFedAvg (Fallah et al., 2020) formulates personalized FL as a MAML-style meta-learning problem. pFedMe (Dinh et al., 2020) decouples personalized model optimization from global model learning using Moreau envelopes. FedRep (Collins et al., 2021) and FedBABU (Oh et al., 2022) learn a shared representation (body) while maintaining client-specific heads. Ditto (Li et al., 2021b) balances global and local objectives for fair and robust personalization. Other approaches include FedPer (Arivazhagan et al., 2019) with personalization layers and

APFL (Deng et al., 2020) with adaptive mixture weights. More recently, Hot-Pluggable FL (Shen et al., 2025a) proposes a plug-in module framework that bridges general and personalized FL via dynamic module selection from a shared modular store, and FOL (Huang and Shu, 2025) introduces a practical one-shot personalized FL framework achieving strong personalization with only a single round of model exchange. Clustered FL has also advanced with HCFL (Guo et al., 2025c), which proposes a four-tier framework that systematically integrates and extends clustering strategies. MultiFCL (Yi et al., 2026) 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 (Konečný et al., 2016; Lin et al., 2018), quantization (Alistarh et al., 2017), and sketching (Rothchild et al., 2020) can reduce communication cost by orders of magnitude. Privacy-preserving techniques such as differential privacy (Geyer et al., 2017; McMahan et al., 2018) 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 remedies on the optimizer (client-drift correction, normalization, contrastive alignment) or the model (personalization). Our setting 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 is no longer about label distributions but about *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 pattern characteristic of this asymmetry. Existing supervised-FL heterogeneity tools do not directly target this asymmetry.

G.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 (Qi et al., 2021). 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 depends on the evolving policy. This introduces several distinctive difficulties beyond standard FL, including (i) *non-stationarity* of the data distribution as the policy changes during training, (ii) *environment heterogeneity* where clients may have different transition dynamics P_k or 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 (Gao et al., 2025; Liu et al., 2025a).

Foundations and Theory. Early work by Yu et al. (2022) proposed sharing experiences through an encrypted channel for collaborative multi-task RL. Jin et al. (2022) studied federated RL in linear MDPs, providing sample complexity analysis. Khodadadian et al. (2022) analyzed the convergence of federated temporal-difference learning with linear speedup under Markovian sampling. Woo et al. (2023) further studied the sample complexity of heterogeneous FRL and showed that heterogeneity can in fact help exploration. More recently, Zheng et al. (2024)

proposed FedQ-Hoeffding and FedQ-Bernstein algorithms for tabular episodic MDPs that achieve linear regret speedup with logarithmic communication cost, and Zheng et al. (2025) further achieved almost minimax-optimal regret via reference-advantage decomposition. Labbi et al. (2025) extended UCBVI to the federated setting with communication-efficient regret minimization.

Federated Policy Gradient Methods. Several works have developed federated policy optimization algorithms. Yang et al. (2024) developed federated natural policy gradient (NPG) and natural actor-critic (NAC) methods with the first global convergence guarantees for federated multi-task RL. Lan et al. (2025) proposed AFedPG, an asynchronous FRL framework that achieves linear speedup under heterogeneous computing. Zhu and Gong (2025) introduced a single-loop federated actor-critic method that jointly considers both federated policy evaluation and improvement. Fan et al. (2021) proposed the first Byzantine fault-tolerant FRL framework, proving linear speedup even when less than half of agents are adversarial.

Heterogeneous Environments. Handling environment heterogeneity across clients is a key challenge. Wang et al. (2024a) proposed momentum-based federated policy gradient methods that converge to a stationary point regardless of the magnitude of environment heterogeneity. Xiong et al. (2025b) introduced personalized FRL with shared representations, proving the first linear speedup result in the personalized FRL setting. Liu et al. (2019) explored lifelong federated RL for cloud robotic systems, where heterogeneous robots share knowledge to accelerate learning.

Applications. FRL has been applied to diverse domains including autonomous driving (Liang et al., 2019), network resource management (Wang et al., 2020c), recommendation systems (Li et al., 2024), and federated offline RL for healthcare (Wen et al., 2023). Hwang and Hong (2025) proposed FedRQ for tabular FRL with proven asymptotic convergence under environment heterogeneity, extended to continuous state spaces via expectile loss. Recent work has also explored offline federated deep RL (XU et al., 2025), which aggregates multiple client-side offline RL models by considering both Q-values and policy inconsistency when assigning client aggregation weights. A comprehensive survey by Cheruiyot et al. (2025) 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 in FRL (long natural-language action sequences, sparse episode-level rewards, token-level credit assignment), our work differs from prior FedRL in two structural ways. First, we separate client heterogeneity into a *two-level* object (task-level, observable through the policy's input channel; environment-level, implicit in the dynamics), rather than treating all non-i.i.d.-ness uniformly. Second, while prior FedRL heterogeneity analyses (Jin et al., 2022; Khodadadian et al., 2022; Wang et al., 2024a) target the *convergence rate* of the federated optimizer under bounded gradient heterogeneity ζ^2, we target the *quality of the federated solution* as a structural property: 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 contributions are orthogonal and compose.

G.3. Federated Learning for LLMs

The intersection of federated learning and large language models (LLMs) has recently attracted growing attention (Kuang et al., 2024; Ye et al., 2024b; Zhang et al., 2024a). 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 infeasible, necessitating 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 spurred rapid development of federated LLM training methods across multiple research communities (Wei et al., 2025b).

Federated Instruction Tuning. A natural first step is to apply FL to supervised instruction tuning. Zhang et al. (2024a) proposed FedIT for federated instruction tuning with LoRA adapters on heterogeneous instruction datasets. Ye et al. (2024b) introduced OpenFedLLM, a training pipeline supporting federated SFT and instruction tuning across diverse datasets. A key challenge in federated instruction tuning is *data quality and selection*: FedHDS (Qin et al., 2025) selects representative subsets at both intra- and inter-client levels without sharing raw data, achieving 10.72% improvement in Rouge-L on unseen tasks while using less than 1.5% of the data. FedDQC (Du et al., 2025) 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*, FewFedPIT (Zhang et al., 2024b) uses LLMs' in-context learning capability to self-generate synthetic data, combined with parameter-isolated training that separates public parameters from private ones to prevent data extraction attacks. On the *communication efficiency* front beyond LoRA-based methods, FedKSeed (Qin et al., 2024) 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 (Ling et al., 2024) provides the first convergence analysis of memory-efficient zeroth-order optimization in federated LLM tuning, demonstrating faster convergence than first-order FedAvg with remarkably reduced GPU memory requirements. FedAMoLE (Zhang et al., 2026b) 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 (Xiong et al., 2025a) extends federated instruction tuning to the multimodal setting, building a comprehensive framework for federated vision-language model fine-tuning that handles heterogeneous multimodal data across clients.

Parameter-Efficient Federated Fine-Tuning. Since communicating full LLM parameters is prohibitively expensive, parameter-efficient methods have become central. Sun et al. (2024) 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. (2024c) introduced FLoRA, a stacking-based aggregation method that achieves noise-free aggregation and supports heterogeneous LoRA ranks. Bai et al. (2024b) proposed FlexLoRA, which dynamically adjusts local LoRA ranks to harness diverse

client resources via SVD-based weight redistribution. Guo et al. (2025b) proposed FedSA-LoRA, which shares only the A matrices for server aggregation while keeping B matrices local, and Yan et al. (2025) introduced FRLoRA, performing global updates in a higher-rank parameter space via residual low-rank adaptation. Cho et al. (2024) proposed HetLoRA, allowing heterogeneous LoRA ranks across clients matching individual system resources. Hyeon-Woo et al. (2021) re-parameterizes weight layers using low-rank weights followed by the Hadamard product, achieving 3–10× lower communication cost. More recently, FedEx-LoRA (Singhal et al., 2024) achieves exact LoRA aggregation by adding a residual error term to frozen weights, eliminating the inexactness of traditional federated averaging of LoRA adapters. FedALT (Bian et al., 2025) departs from FedAvg by maintaining individual LoRA adapters alongside a shared Rest-of-World LoRA component with an adaptive MoE-style mixer. FedLEASE (Wang et al., 2025b) adaptively clusters clients based on representation similarity to allocate domain-specific LoRA experts with a top-M MoE mechanism. pFedGPT (Shen et al., 2025b) hierarchically optimizes LoRA aggregation weights for personalized federated GPT models. EcoLoRA (Liu et al., 2025b) uses round-robin segment sharing where each client uploads only a complementary LoRA segment per round, reducing communication time by up to 79%. FedICU (Liao et al., 2025) introduces importance-aware splitting and updating to dynamically balance LoRA components based on their contribution to global model performance, preventing catastrophic forgetting and domain overfitting. A comprehensive survey on federated LoRA methods was provided by Yang et al. (2025).

Federated Preference Optimization and Alignment. For alignment, FedRLHF (Fan et al., 2024) 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) rather than multi-step agent training. Wu et al. (2025) 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 (2025) evaluated Kahneman-Tversky Optimization (KTO) against DPO in federated settings, finding KTO more robust to data heterogeneity. Srewa et al. (2025) proposed PluralLLM, using federated learning for pluralistic LLM alignment that preserves group fairness.

Federated LLM Pre-Training and Knowledge Distillation. Sani et al. (2025) presented Photon, the first complete system for federated end-to-end LLM pre-training from scratch, training models up to 7B parameters. Wu et al. (2024a) 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. (2025b) organizes federated learning for reasoning LLMs by training signal types, covering federated pre-training, instruction tuning, prompt learning, adapter learning, and value alignment.

Federated Agent Systems. Very recently, a few works have begun to explore the intersection of federated learning and LLM agents. Fed-SE (Chen et al., 2025c) proposes a federated self-evolution framework where agents perform parameter-efficient fine-tuning on filtered high-return trajectories locally and aggregate globally, improving average task success rates by ~18% over FedAvg across five heterogeneous environments. FICAL (Wu et al., 2024b) takes a different approach by transmitting knowledge compendiums rather than model parameters via in-context learning, combined with RAG-based tool learning. EPEAgent (Shi et al., 2025) 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.

Positioning of Our Work. Federated LLM training is largely confined to supervised fine-tuning, single-turn RLHF, or preference optimization. The few concurrent works on federated agent learning (Chen et al., 2025c; Shi et al., 2025; Wu et al., 2024b) either fine-tune on filtered trajectories without systematic RL, rely on in-context learning instead of policy gradient updates, or address retrieval-side rather than training-side privacy. 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*, which maps four observable training-curve patterns to their underlying realizability and structural conditions.

G.4. LLM Agent Reinforcement Learning

LLM-based agents can perform complex interactive tasks requiring multi-step reasoning, tool use, and environment interaction (Xi et al., 2024). 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 applied to diverse domains including web navigation (Qi et al., 2025; Zhou et al., 2024b), software engineering (Jimenez et al., 2024; Wei et al., 2025c), embodied household tasks (Shridhar et al., 2021), mobile device control (Bai et al., 2024a; Rawles et al., 2023), and scientific research. 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 (Yao et al., 2022b), Reflexion (Shinn et al., 2023), Chain-of-Thought (Wei et al., 2022), and Tree of Thoughts (Yao et al., 2023) improve agent reasoning without parameter updates, but their performance is bounded by the underlying model's capabilities. Schick et al. (2023) demonstrated that LLMs can learn to use external tool APIs in a self-supervised manner. Zhou et al. (2024a) unified reasoning, acting, and planning via Monte Carlo Tree Search with LM-powered value functions. Multi-agent systems such as AutoGen (Wu et al., 2023) 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 (Chen et al., 2023; Zeng et al., 2024). Song et al. (2024) proposed ETO, which learns from contrastive pairs of successful expert and failed agent trajectories using DPO. Zhang et al. (2025b) proposes "early experience" as a middle ground between expert demonstrations and full RL, using interaction data generated by the agent's own actions where 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 (Schulman et al., 2017) 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 (Guo et al., 2025a) showed that RL can incentivize complex reasoning in LLMs, spurring rapid follow-up work. GRPO (Shao et al., 2024) eliminates the need for a separate critic model by using group-relative advantages, and has become one of the most widely adopted algorithms for agent RL. DAPO (Yu et al., 2025a) introduces clip-higher and dynamic sampling techniques that achieve strong reasoning at scale. GiGPO (Feng et al., 2025) further improves upon GRPO with group-in-group advantage estimation for finer-grained credit assignment. RLOO (Ahmadian et al., 2024) proposes a parameter-free REINFORCE Leave-One-Out baseline that outperforms PPO while using substantially less GPU memory. Zhou et al. (2024c) proposed ArCHer, a hierarchical multi-turn RL framework, and Putta et al. (2024) introduced Agent Q with advanced reasoning for autonomous agents. Wang et al. (2025c) introduced StarPO, a general trajectory-level agent RL framework, identifying the "Echo Trap" failure mode. Agent-R1 (Cheng et al., 2025) extends single-turn RL to multi-turn agent settings by formalizing the LLM agent MDP.

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 (Lightman et al., 2023; Wang et al., 2024b) provide step-level supervision that significantly outperforms outcome-only rewards. iStar (Liu et al., 2025c) proposes implicit step rewards that jointly optimize an implicit process reward model with the policy model via multi-turn DPO, achieving state-of-the-art on WebShop and SOTOPIA. Agentic Reward Modeling (Peng et al., 2025) combines human preference rewards with verifiable correctness signals from factuality and instruction following. For multi-turn credit assignment, Wei et al. (2025a) provides the first systematic study of turn-level reward design, extending GRPO and PPO to multi-turn variants. Wang and Ammanabrolu (2025) offers a practitioner's guide that systematically analyzes the design space across environment, reward, and policy dimensions. SWEET-RL (Zhou et al., 2025) uses a critic model with access to training-time information to provide step-level rewards for collaborative reasoning tasks. HiPER (Peng et al., 2026) 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 (Chen et al., 2025b) cold-starts the policy via a self-play SFT stage to internalize a world model, then uses it to simulate future states prior to policy optimization. Dyna-Think (Yu et al., 2025b) integrates planning with an internal world model into the reasoning-acting loop, using Dyna-GRPO for online RL and achieving comparable best-of-n performance to R1-style reasoning with 2× fewer tokens. DynaWeb (Ding et al., 2026) applies the Dyna architecture to web agents, training a dedicated world model to generate multi-step imagined trajectories. WebWorld (Xiao et al., 2026) 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 (Wang

et al., 2026) 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 (Yu et al., 2026) proposes self-supervised action-conditioned world model learning using sim-to-real gap rewards, outperforming direct RL on ALFWorld and τ-Bench. On the experience efficiency side, DreamGym (Chen et al., 2025d) introduces the first unified framework for synthesizing diverse agent experiences at scale, distilling environment dynamics into a reasoning-based experience model, outperforming all baselines by over 30% on non-RL-ready tasks. ERL (Shi et al., 2026) augments trial-and-error RL with an explicit experience-reflection-consolidation loop, achieving gains of up to 81% in complex multi-step environments. SkillRL (Xia et al., 2026) 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. Agent RL has been successfully applied across diverse interactive domains. For *web agents*, WebRL (Qi et al., 2025) 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 (Wei et al., 2025e) demonstrates that simple end-to-end multi-turn RL with binary success rewards can boost web agent performance from 8.5% to 44.8%. For *device-control agents*, DigiRL (Bai et al., 2024a) proposes a two-stage offline-to-online RL approach. For *software engineering*, SWE-RL (Wei et al., 2025c) scales RL-based LLM reasoning for real-world software tasks, and Self-Play SWE-RL (Wei et al., 2025d) further extends it with a self-play paradigm for autonomous bug injection and repair without human labels. SWE-Gym (Pan et al., 2025) provides the first training environment for real-world software engineering agents with 2,438 executable task instances. DeepSWE trains a fully open-source RL-based coding agent achieving 59% on SWE-Bench-Verified. Golubev et al. (2025) combines rejection fine-tuning with DAPO to train a 72B SWE agent, increasing SWE-bench pass rate from 20% to 39%. For *reasoning and tool use*, Search-R1 (Jin et al., 2025) extends R1-style RL to search-augmented settings, improving performance by 41% over RAG baselines. ARTIST (Singh et al., 2025) unifies agentic reasoning and tool integration via RL, enabling autonomous tool invocation within multi-turn reasoning chains. ML-Agent (Liu et al., 2025d) applies RL to autonomous machine learning engineering, where a 7B agent trained on merely 9 ML tasks outperforms the 671B DeepSeek-R1 agent. AgentRefine (Fu et al., 2025) trains 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 (Zhang et al., 2025a) 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 (Chen et al., 2025a) proposes a data- and memory-efficient PPO variant for interactive digital agents, with a 32B agent outperforming OpenAI o1 on AppWorld. AgentGym-RL (Xi et al., 2025) proposes ScalingInter-RL that gradually shifts from exploitation to exploration across training, matching or surpassing commercial models on 27 tasks. ReMA (Wan et al., 2025) decouples reasoning into a meta-thinking agent and a low-level reasoning agent via multi-agent RL. MAGRPO (Liu et al., 2026) models LLM collaboration as cooperative multi-agent RL with centralized group-relative advantages. MARTI (Zhang et al., 2025c) provides an open-source framework for training multi-agent LLM systems with graph-based workflows and both rule-based and generative rewards. VAGEN (Wang et al., 2025a) introduces world modeling RL for multi-turn VLM agents.

Benchmarks and Evaluation. Several benchmarks have been established to evaluate agent capabilities, including WebShop (Yao et al., 2022a) for web-based shopping tasks, ALFWorld (Shridhar et al., 2021) for embodied household tasks, AndroidInTheWild (Rawles et al., 2023) for mobile device control, SWE-bench (Jimenez et al., 2024) for real-world software engineering, WebArena (Zhou et al., 2024b) for realistic web navigation, AgentBench (Liu et al., 2024b) for multi-dimensional agent evaluation, and GAIA (Mialon et al., 2024) for general AI assistants. More recently, TheAgentCompany (Xu et al., 2024) simulates a full software company environment with integrated development and communication tools, where the best agents solve only 30% of tasks autonomously. AgentGym (Xi et al., 2024) provides a unified framework featuring 7 scenarios, 14 environments, and 89 tasks for concurrent agent interaction and self-evolution. MultiAgentBench (Zhu et al., 2025) 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 agent RL (spanning policy optimization (Shao et al., 2024; Yu et al., 2025a), reward design (Liu et al., 2025c; Wei et al., 2025a), world-model augmentation (Chen et al., 2025b; Xiao et al., 2026), and scalable infrastructure (Xi et al., 2025; Zhang et al., 2025a)) rests on a shared assumption that training data (task queries and trajectories) can be aggregated centrally. This 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 when it does not (Patterns B/C/D under environment-level heterogeneity), each tied to a precise structural condition on the LLM agent's MDP.

H. FEDAGENT Algorithm

This appendix complements §2 with three pieces: (i) a line-by-line description of the FEDAGENT training protocol of Algorithm 1 (Appendix H.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 H.2), and (iii) communication cost and deployment considerations (Appendix H.3). The protocol follows the classical FEDAVG (McMahan et al., 2017) base protocol (sample → broadcast → local update → average), but instantiates the local update with *policy-gradient RL on LLM-induced policies* rather than supervised SGD. This single change produces both the new heterogeneity surface analyzed in §3 and the privacy properties discussed in Appendix H.2.

H.1. Pseudocode

The FEDAGENT training protocol is stated as Algorithm 1 in the main text (§2), with client-side and server-side computation color-coded for readability. This subsection goes through it line by line. The local objective is no longer a fixed loss over labeled examples but an expected return over rollouts that depends on the per-client task distribution $\mathcal{D}_{\tau_i}$ and the per-client environment $(\mathcal{S}_i, \mathcal{A}_i, P_i, R_i)$, 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 ($M \le N$), and E is the number of local optimization epochs each sampled client performs before returning (each epoch samples a fixed batch of tasks from the local pool. Exact batch sizes are listed in Appendix I.6 and the surrounding subsections). 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 this.

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 uniformly without replacement). A sample-size-weighted variant 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.

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 a minibatch of trajectories $B_{i,t,e}$ 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 $\theta_{i,t,e+1} \leftarrow \theta_{i,t,e} + \eta_{\text{lr}}\, g_{i,t,e}$. The pseudocode is deliberately optimizer-agnostic: Line 9 abstracts the gradient estimator behind the 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** (Shao et al., 2024) (the group-relative variant that drops the critic and normalizes by per-task success rate) and **PPO** (Schulman

et al., 2017) (with token-level clipped updates and a value baseline), and we observe in §5.3 that the choice of local optimizer substantially affects env-level robustness even though the FedAvg outer loop is identical.

H.2. Intrinsic Privacy Properties

A central motivation for FEDAGENT is privacy: centralizing user task queries and agent trajectories at scale is increasingly at odds with regulation (Kairouz and McMahan, 2021) and with the sensitivity of natural-language interaction logs. Below we make this informal motivation precise by laying out the threat model, identifying what does and does not cross the client/server boundary, and arguing 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. Rather, we identify the structural reasons why standard attacks are weakened, and point to orthogonal defenses (DP, 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 (Kairouz and McMahan, 2021): 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 χ_i 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 (often thousands of tokens), 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. Raw task descriptors, rollout trajectories, intermediate token sequences, environment observations, action logits, value-network states, advantage estimates, per-task reward signals, and per-client success rates all stay on the client. This is the same data-minimization principle that FEDAVG (McMahan et al., 2017) 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 (Geiping et al., 2020; Huang et al., 2021; Zhu et al., 2019) 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 update breaks 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 rounds of stochastic policy optimization across many minibatches, so the linear correspondence between individual training samples and the returned delta $\Delta\theta_{i,t}$ is broken. Huang et al. (2021) document that this turns the inversion landscape badly non-convex with many spurious minima even in

the supervised-classification setting. The phenomenon is more severe under policy-gradient updates, which are themselves stochastic estimators of an expectation over trajectories rather than deterministic gradients of a loss on a labeled example.

- *No discrete label channel for label-leakage attacks.* The label-leakage attack of Zhao et al. (2020) recovers ground-truth class labels by inspecting the sign and magnitude 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-leakage path simply does not exist in the FEDAGENT protocol.
- *Long-sequence inversion is fundamentally hard.* Token-level GIAs against language-model gradients (Balunovic et al., 2022; Petrov et al., 2024) have made progress, but their effectiveness degrades sharply with sequence length: DAGER (Petrov et al., 2024) guarantees exact recovery only when the total token count stays below the model's embedding dimension, and prior approximate methods are unreliable beyond input lengths of a few tens of tokens. Agent rollouts in WebShop and ALFWorld routinely span thousands of tokens of intermediate reasoning and tool-call outputs, several orders of magnitude beyond the regime where these attacks have been demonstrated effective on supervised text data.

RL-specific gradient inversion is weakened by environmental dynamics. A nascent line of work (He, 2025) adapts GIAs to federated RL by jointly reconstructing (s, a, r, s') tuples from policy-gradient updates. He (2025) document a unique *pseudo-solution problem*: candidate reconstructions that match the gradient values often violate the underlying environment dynamics (P_i, R_i), so an attacker without dynamics access cannot disambiguate genuine trajectories from infeasible ones. Their proposed RGIA mitigates this by adding regularizers that constrain reconstructions to plausible state distributions, plausible reward ranges, and consistent transition dynamics. In FEDAGENT the dynamics $(\mathcal{S}_i, \mathcal{A}_i, P_i, R_i)$ are local to each client, and our environment-level heterogeneity construction (§3.3) explicitly varies them across clients. The attacker therefore lacks the most informative side-channel needed to make the inversion well-posed for any specific client. Even granting the attacker a perfect simulator of the average environment, per-client divergences along the env-level axis (catalog subsets, BM25 reweightings, lookalike injections, rank wrappers in our experiments) leave the inversion under-determined.

Membership inference is similarly weakened. Membership-inference attacks (Nasr et al., 2019; Shokri et al., 2017) ask whether a specific example was in the training set, exploiting model overfitting on individual samples. Agent RL departs from this setup in two ways. First, the natural unit of training data is a task descriptor τ, and a single τ produces a distribution of trajectories under the stochastic policy rather than a single labeled example. 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 global model never observes per-example reward signals or per-task success indicators. The standard signal channels for MIA, 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 them from $\Delta\theta_{i,t}$, which 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, rather than

establishing 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 tighten the argument quantitatively: (i) differential privacy applied to $\Delta\theta_{i,t}$ (Abadi et al., 2016; Geyer et al., 2017), (ii) secure aggregation (Bonawitz et al., 2017) so the server only observes the post-average parameters θ_{t+1} rather than individual client updates, and (iii) communication-efficient transmission via parameter-efficient methods such as FLoRA (Wang et al., 2024c) or zeroth-order seed-based protocols (Qin et al., 2024), which both reduce bandwidth (cf. Appendix H.3) and shrink the inversion surface, since lower-rank or coarsely quantized updates carry strictly less per-sample information. A formal privacy analysis tailored to LLM agent RL, in particular extending the pseudo-solution analysis of He (2025) to long-horizon LLM rollouts, is a worthwhile direction for future work.

H.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: at the `Qwen2.5-1.5B-Instruct` backbone we use in the main experiments, this is ≈ 3 GB per direction in FP16 per participating client per round, comparable to the per-round cost reported in the LLM-FedAvg literature (Sun et al., 2024; Ye et al., 2024b) for full-parameter fine-tuning. We deliberately use full-parameter transmission throughout the experiments to keep the communication channel an isolated variable so that any observed effect on training quality is attributable purely to the heterogeneity axes of §5.2–5.3, not to compression-induced loss. In a practical deployment, the parameter-efficient federated fine-tuning literature provides several drop-in replacements that retain the base FEDAGENT protocol while drastically reducing per-round bandwidth: LoRA-based variants (Bai et al., 2024b; Sun et al., 2024; Wang et al., 2024c) transmit only adapter matrices. Segment-sharing variants (Liu et al., 2025b) amortize the cost over several rounds. Zeroth-order variants (Qin et al., 2024) reduce the per-round payload to kilobytes. None of these change the algorithmic structure analyzed in §3–4, and they compose cleanly with the privacy mechanisms of Appendix H.2.

Partial participation and client drift. Two practical considerations from supervised FL (Kairouz and McMahan, 2021; Karimireddy et al., 2020) carry over to FEDAGENT. *Partial participation* ($M < N$) introduces extra variance into the aggregated update: with M clients per round, the population of N clients is covered in expectation every N/M rounds, 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 env-level results in §5.3 show that drift alone is not the source of Pattern D collapse either: uniform task partitions do not exhibit the collapse.

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 steps* 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 M and E is reported in Appendix F.

Deployment scenarios. Two natural deployment patterns motivate FEDAGENT. (i) *Cross-silo:* a small number of organizations (*e.g.*, enterprises with their own user query logs and tool catalogs) pool agent training 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 H.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 path to a deployable instantiation. The empirical results in §5.2–5.3 use N=100 with M=2 per round, which models the cross-silo regime closely while remaining tractable on a research-scale cluster.

I. 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 are verified against the released configuration files in the public code repository (§C).

I.1. Hardware and Software

All experiments are run on 4× 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 (Feng et al., 2025) (itself built on `verl` 0.3.1.dev). Our federated extension is released alongside the paper at the public repository in §C. Software stack: Python 3.10 (3.10.0 for the WebShop conda environment, 3.10.18 for the ALFWorld environment), PyTorch 2.6.0+cu124, CUDA toolkit 12.4, and vLLM 0.8.5 for inference rollout. 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. WebShop is run against our in-memory BM25 backend (§L). ALFWorld uses the standard textual interface from Shridhar et al. (2021), with the data cache located at `$ALFWORLD_DATA/json_2.1.1/train` (3553 game files, 550 spec files of which 311 are multi-scene).

I.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 (§F.1). The four backbones together populate the multi-backbone effectiveness rows of Table 1 and Table 3. All models are fine-tuned with full fine-tuning (no LoRA), using each model's released chat template and tokenizer. No system-prompt modification is applied. Maximum prompt length is 4096 tokens for WebShop and 2048 tokens for ALFWorld. Maximum response length (max-new-tokens per agent action) is 512 tokens.

I.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: $|X_i| = 100$. The first 500 goals `goals[0:500]` are reserved as the validation pool and never seen during training. Each client's pool is drawn from the remaining `goals[500:]` via `uniform_partition()` with `min_samples_per_client`= 100 and `data_sharding.seed`= 42. Adjacent clients overlap on ~36 goals due to contiguous slicing with boundary expansion.

- Per-epoch sampling: training rollouts use `ppo_mini_batch_size` = 64 trajectories.
- 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 ($\mu = 0.01$ kept as default for ablation).
- 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 choice of ($M = 2$, $E = 3$, $|X_i| = 100$) is justified by the sensitivity analysis in Appendix F.1.

I.4. Policy Optimization Hyperparameters

We instantiate the local optimizer with either GRPO (Shao et al., 2024) (main text default) or PPO (Schulman et al., 2017) (Appendix F.2). Common 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).
- PPO epochs per FedAvg client step: 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 is a small additive engineering term applied at the optimizer side to discourage malformed actions. The formal reward of the underlying MDP R_i remains the non-negative success-based component $r_t \in [0, R_{\max}]$ 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. Group size $G = 8$ (`env.rollout.n`); advantage estimator: group-relative baseline (`algorithm.adv_estimator`= `grpo`); KL coefficient $\beta_{\text{KL}} = 0.01$ against the reference policy (`actor.kl_loss_coef`, with `kl_loss_type`= `low_var_kl`, the low-variance k_3 estimator of Schulman); entropy coefficient 0.001; mini-batch size 64, micro-batch size per GPU 8; train batch size 8 on both benchmarks; within-group advantage normalization `norm_adv_by_std_in_grpo`= True.

PPO-specific. Advantage estimator: GAE (`algorithm.adv_estimator`= `gae`) with $\lambda = 1.0$ (verl default, equivalent to Monte-Carlo advantage). Clip ratio: symmetric $\epsilon_{\text{clip}} = 0.2$ (`actor.clip_ratio_low`=`actor.clip_ratio_high`= 0.2); dual-clip parameter `clip_ratio_c` = 3.0. Value-loss clipping `cliprange_value` = 0.5. Entropy coefficient 0.001. Mini-batch size 64, actor micro-batch size per GPU 8, critic micro-batch size per GPU 4; train batch size 64 on both benchmarks. We do *not* fold the KL term into the reward

(use_kl_in_reward = False). The in-loss actor KL coefficient is $\beta_{\text{KL}} = 0.01$ (same as GRPO). Critic head is initialized from the actor's hidden states (verl default value-head wrapper).

I.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 it gave no rollout speedup and triggered NaN crashes under 7B PPO. Per-task rollout settings:

- Number of rollouts per task: $G = 8$ (training, env.rollout.n). PPO uses the same group-size for advantage estimation infrastructure consistency.
- 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.

I.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), $\xi \in \{1, 256\}$ (coverage: extreme spread vs. near-uniform pool sizes), and $\xi' \in \{1, 256\}$ (hardness: extreme success-rate skew vs. near-uniform). Pseudocode and the formal isolation guarantees (D1)–(D4) are in Appendix K (specifically Appendix K.1 and K.2). The formal dispersion measures and intrinsic-property results that ground these guarantees are in Appendix J. Per-client distribution figures at both endpoints are in Appendix K.3. The hardness reference checkpoint used to pre-label tasks as "successful"/"unsuccessful" is the released Qwen2.5-1.5B-Instruct backbone evaluated zero-shot on the training pool.

Environment-level. The five WebShop env-variants (Catalog Split, Field-Subset Index, BM25 Reweighting, Lookalike Injection, Rank Wrapper) are described in §3.3 with full implementation details in Appendix L. 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 rather than additional reported experiments. The variant pools and verbatim hyperparameters are:

- **Catalog Split**: env_div $\in \{0.0, 0.3, 0.7, 1.0\}$ at keep_ratio = 0.7 (803 products per client).
- **Field-Subset Index**: $|\mathcal{V}| \in \{4, 8\}$ field subsets drawn from {name, Title, description, features, BulletPoints} and pairs/triples thereof.
- **BM25 Reweighting**: $|\mathcal{V}| \in \{4, 8\}$ extreme (k_1, b) combinations with $(1.2, 0.75)$ as default and $(0.3, 0.75)$, $(5.0, 0.75)$, $(1.2, 0.0)$ as the three extreme combinations 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 pools (price: 1,000 lookalikes, color: 286, size: 268, price+color: 286; the 2-pool setting is a reduced configuration 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 to prevent target dropouts under aggressive catalog filtering. 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`). Task partition is held *uniform* across all env-level experiments so any divergence is attributable to the transition perturbation alone.

I.7. Evaluation Protocol

Test sets. WebShop is evaluated on 64 goals drawn from the held-out validation pool `goals[0:500]` (disjoint from training). ALFWorld is evaluated on 64 trials drawn from the held-out `valid_seen/valid_unseen` splits (scenes not in the training pool), spanning the six task types (Pick, Look, Clean, Heat, Cool, Pick2). For env-level experiments, evaluation is performed on the *standard* (un-perturbed) WebShop environment so the metric isolates the post-aggregation generalization quality.

Metrics. Following Liu et al. (2024a), we report two complementary metrics: **Success Rate** (binary task completion under the WebShop / ALFWorld convention) and **Task Score** (WebShop only: the continuous per-episode reward $r \in [0, 1]$ averaged over evaluation tasks and scaled by 100, following the WebShop convention; a 0–100 score, not a percentage). On ALFWorld, we additionally break down success rate by the six task types and report the All-task aggregate (cf. Table 3).

Evaluation schedule. Two independent switches govern evaluation, and together they 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 at both its first and its last local epoch.

(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 and tracing 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 these per-client values appear individually as the colored circle marks rather than being averaged. The separate setting `test_freq` = 5 is `verl`'s step-level schedule *inside* a single client's job, not a round-level control: it would launch a validation whenever the internal global-step index is divisible by 5, but since each sampled client runs only $E = 3$ local epochs (= `verl` steps) per round ($3 < 5$), 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 (deterministic client-to-goal mapping), 42 + `client_id` (variant assignment for BM25 / Lookalike / Search Variant constructions), 42 + 1000 · `client_id` (Catalog Split per-client distractor RNG with global seed 42 for the shared distance threshold $u[\cdot]$), and 99999 (offline lookalike synthesis). The federated and centralized rows in Table 1 and Table 3 report mean and standard deviation across 3 random seeds. The standard deviation is shown as a subscript. Local-training rows use the indicated client indices (21, 42, 84).

I.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 (built on the `verl-agent` framework), dataset preprocessing scripts, all per-experiment configuration files, and scripts for regenerating the post-training outputs are linked in §C.
- **Compute estimate.** Per-run wall-clock times: 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). Cumulative compute spent on the experiments reported in §5 and the PPO appendix is approximately 1,800 H100 GPU-hours.
- **Hyperparameters.** All federation, optimization, and rollout hyperparameters are reported above with values verified against the released configuration files.
- **Statistical reporting.** Mean and standard deviation across 3 seeds. Training-dynamics figures show single-seed curves with the per-client population overlaid 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 L) are generated programmatically from the WebShop catalog and inherit its license.

J. Task-Level Heterogeneity Characterization

This appendix supports §3.2 with the formal measures used to quantify task-level heterogeneity at the population level. We introduce the three inter-client dispersion measures $\Delta^2_{\text{pref}}, \Delta^2_{\text{cov}}, \Delta^2_{\text{hard}}$ (Definitions J.1–J.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), together with the converse statement that fails (Theorem 8). The algorithmic side, namely how single-hyperparameter partitions realize each measure and the four design properties (D1)–(D4) connecting algorithms to measures, is deferred to Appendix K.

J.1. Inter-Client Dispersion Measures

Throughout this section, we work with a federation of N clients, each holding a local task pool X_i of size $n_i = |X_i|$ drawn from an underlying task distribution $\mathcal{D}_{\tau_i}$ over a task space $\mathcal{T}$. 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, the pretrained checkpoint). 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 $p_{i,c} = \Pr_{\tau\sim\mathcal{D}_{\tau_i}}[\ell(\tau) = c]$ is the type marginal, n_i is the local pool size (a partition-protocol parameter rather than a distributional functional), and $\rho_i = \Pr_{\tau\sim\mathcal{D}_{\tau_i}}[\rho(\tau) \geq \eta]$ is the thresholded success probability. The empirical counterparts are $\hat{p}_{i,c} = (1/n_i)\sum_{\tau\in X_i}\mathbf{1}\{\ell(\tau) = c\}$ and $\hat{\rho}_i = (1/n_i)\sum_{\tau\in X_i} S_\eta(\tau)$, both unbiased estimators with $O(1/\sqrt{n_i})$ deviation. We reuse the symbols p_i, ρ_i for both population and empirical versions and disambiguate by context.

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

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

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

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. Range: $0 \leq \Delta^2_{\text{pref}} \leq 1 - 1/C$ when $N \geq C$, with the upper bound attained by spreading one-hot clients evenly across categories.

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

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

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 J.2 does not rely on it.

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

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

For tasks with bounded support $\rho(\tau) \in [0,1]$, ρ_i fully summarizes client i's thresholded success profile, and the sample variance $\widehat{\text{Var}}_i(\rho_i)$ is the standard inter-client dispersion summary. Range: $0 \le \Delta^2_{\text{hard}} \le \bar{\rho}(1-\bar{\rho}) \le 1/4$, the upper bound attained when each $\rho_i \in \{0,1\}$.

J.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)**.** *For each client i, the three projections satisfy:*

- ***(P1) Well-defined under sampling:*** *each projection is a first-order linear functional or operational descriptor that is invariant to permutations of task identities preserving ℓ and ρ, and admits an unbiased empirical estimator from any finite X_i with $O(1/\sqrt{n_i})$ deviation.*
- ***(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 $\text{CV}^2 \le N-1$).*
- ***(P3) Monotone-controllable:*** *each projection's inter-client dispersion is controlled by a single hyperparameter (Propositions 1, 2, 3) that monotonically traverses the realizable range of Theorem 11 (an open interval strictly inside the definitional bounds).*

Proof. For (P1), p_i is the marginal of $\mathcal{D}_{\tau_i}$ along ℓ, ρ_i is the marginal along the indicator S_η, and $n_i = |X_i|$ is a sampling-protocol output. All three are permutation-invariant. The empirical estimators $\hat{p}_{i,c}, \hat{\rho}_i$ 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 (an upper limit set by the construction regime, not an intrinsic constant; the intrinsic bound for nonnegative sizes is $\text{CV}^2 \le N-1$); 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 K). □

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 J.1 (which 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 J.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 rather than proved inequalities: 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.

J.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 (open intervals from the simplex, CV^2, and Bernoulli-variance constraints) are the outer feasibility bounds. The realizable region delivered by the 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, which lies inside the definitional bounds (e.g., $1 - \|p_0\|_2^2 \le 1 - 1/C$). Every target triple in $\mathcal{R}$ is pointwise achievable by varying $\{p_i\}, \{n_i\}, \{\rho_i\}$ independently, under the hypotheses of Theorem 11.

Proof. The three functionals are taken directly from Definitions J.1–J.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. Pointwise reachability over the realizable box $\mathcal{R}$ follows by independently varying the three projections, with concrete realizations given by Theorem 11 (under Assumption 1 and its balanced-pool condition) in Appendix K. 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. Direct from Theorem 6: each Δ^2_* is a function of one of the three projections, so equal projections imply equal measures. The second statement is the contrapositive: if a moment m_i of $\mathcal{D}_{\tau_i}$ affected, *e.g.*, Δ^2_{pref} but were not encoded in p_i, this 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 $p^A_1 = (0.6, 0.4)$, $p^A_2 = (0.4, 0.6)$, giving $\bar{p}^A = (0.5, 0.5)$ and $\Delta^2_{\text{pref}} = 0.02$; and partition B with $p^B_1 = (0.5, 0.5)$, $p^B_2 = (0.3, 0.7)$, 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 is why we adopt projection-based heterogeneity rather than full-distribution heterogeneity. □

K. 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 K.1 presents the pseudocode of PREFERENCEPARTITION, COVERAGEPARTITION, and HARDNESSPARTITION. Appendix K.2 states and proves the four design properties (D1)–(D4) connecting these algorithms to the dispersion measures of Appendix J. Appendix K.3 visualizes the per-client distributions induced on WebShop and ALFWorld at the near-uniform and most-heterogeneous settings used in §5.2.

K.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 §3.2). The three procedures are also compositionally orthogonal (D4), so any feasible target triple can be reached by chaining them. Below we describe each algorithm in turn.

K.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 $p_0 \cdot (1-\omega)/\omega$, a one-parameter family with mean fixed at p_0 for all ω and variance monotonically increasing in ω: as $\omega \to 0$ all clients see 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 draw asks for more items than a category contains, so $\bar{n}$ remains constant across all values of ω. The leftover-redistribution rule reassigns surplus draws proportionally to q over classes with spare capacity, which preserves the per-client category marginal in expectation up to the capacity constraint and avoids systematic bias toward larger pools.

K.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 pinning 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 q_j clients, where q_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 moves only Δ^2_{cov} and leaves the global category and difficulty marginals invariant at leading order (Theorem 9).

Algorithm 2 PREFERENCEPARTITION

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

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

$p_{0,c} \leftarrow n_c / \sum_{j=1}^{C} n_j$ for $c = 1 \ldots C$ ▷ global category marginal (anchor)

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

  $(q_1, \ldots, q_C) \sim \text{Dirichlet}\left(p_{0,1}\frac{1-\omega}{\omega}, \ldots, p_{0,C}\frac{1-\omega}{\omega}\right)$ ▷ larger $\omega \Rightarrow$ higher variance; $\mathbb{E}[q] = p_0$ for all ω

  $(a_1, \ldots, a_C) \sim \text{Multinomial}(L; q_1, \ldots, q_C)$ ▷ category counts for client i

  if any $a_c > n_c$ **then** set $a_c \leftarrow \min(a_c, n_c)$ and redistribute leftover by q to classes with spare capacity ▷ capacity fix within a set

  $X_i \leftarrow \bigcup_{c=1}^{C} \text{SAMPLEWITHOUTREPLACEMENT}(\mathcal{I}_c, a_c)$

end for

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

K.1.3. HARDNESSPARTITION

HARDNESSPARTITION (Algorithm 4) controls the spread of per-client success-rate quotas $\{\rho_i\}$, defined as the fraction of "successful" tasks (those with reference checkpoint success rate $\rho(\tau) \geq \eta$) 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) they spread toward $\kappa_{\min}$ and $\kappa_{\max}$ so ρ_i ranges from near-zero (no positive learning signal) to near-one (no room for improvement). 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 (Theorem 9). The isolation holds at leading order, under the approximate threshold-independence discussed in Assumption 1.

K.2. Design Properties (D1)–(D4)

This subsection states and proves the four design properties (D1)–(D4) connecting the three partition algorithms of Appendix K.1 to the dispersion measures introduced in Appendix J: each algorithm's hyperparameter monotonically controls one measure (D1, Propositions 1–3), varying that hyperparameter leaves the other two measures invariant in expectation under a mild threshold-independence assumption (D2, Theorem 9), all first-order means are preserved as the hyperparameters vary (D3, Theorem 10), and the three partitions compose to reach any feasible target triple (D4, Theorem 11).

K.2.1. Single-Hyperparameter Monotone Control (D1)

We now show that each partition algorithm exposes 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

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 $X_1, \ldots, X_N$

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

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 $|X_i| = L$

$\{Y_i\}_{i=1}^N \leftarrow \text{CoveragePartition}(|\mathcal{Y}|, N, (\kappa_{\min}, \kappa_{\text{avg}}, \kappa_{\max}), \xi', r)$ ▷ larger $\xi' \Rightarrow$ lower variance
for $i = 1$ to N **do**
  $\nu_i \leftarrow L - |Y_i|$; $F_i \leftarrow \text{SampleWithoutReplacement}(\mathcal{U}, \nu_i)$
  $X_i \leftarrow Y_i \cup F_i$
end for
return $\{X_i\}_{i=1}^N$

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} pins $\bar{n} = L_{\text{avg}}$; 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 PrefPart, CovPart, and HardPart. All limits are stated under $N, L \to \infty$, with finite-sample corrections of order $O(1/\sqrt{L}) + O(1/\sqrt{N})$.

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* $X_i \sim \text{Multinomial}(L; q_i)$,

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

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 $\mathrm{tr}(\mathrm{Cov}(q_i)) = (1 - \|p_0\|_2^2)/(\alpha+1) = \omega(1-\|p_0\|_2^2)$. 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, $\mathbb{E}[\Delta^2_{\mathrm{pref}}] \to \mathbb{E}[\|q_i - p_0\|_2^2] = \omega(1-\|p_0\|_2^2)$. 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* $x_i \sim \mathrm{Beta}(\mu\xi, (1-\mu)\xi)$, $u_i = L_{\min} + x_i(L_{\max} - L_{\min})$, *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 $\mathrm{Var}(x_i) = \mu(1-\mu)/(\xi+1)$. Linear transformation yields $\mathbb{E}[u_i] = L_{\mathrm{avg}}$ and $\mathrm{Var}(u_i) = (L_{\max} - L_{\min})^2\mu(1-\mu)/(\xi+1)$. Renormalization enforces $\bar{n} = L_{\mathrm{avg}}$ exactly (the rescaling factor concentrates to 1 as $N \to \infty$), so $\mathbb{E}[\Delta^2_{\mathrm{cov}}] \to \mathrm{Var}(u_i)/L^2_{\mathrm{avg}} = C_n/(\xi+1)$. 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 $O(1/L^2_{\mathrm{avg}})$ residuals. Both are negligible at the experimental scale ($N = 100$, $L_{\mathrm{avg}} = 500$; cf. the coverage distributions of Figure 9). □

Proposition 3 (Hardness Control: ξ' for Δ^2_{hard})**.** *Under* HardnessPartition *with* $\{|Y_i|\}$ *drawn 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 $\{|Y_i|\}$: $\mathrm{Var}(|Y_i|) = (\kappa_{\max} - \kappa_{\min})^2\mu'(1-\mu')/(\xi'+1)$. Since $\rho_i = |Y_i|/L$, dividing by L^2 gives $\mathrm{Var}(\rho_i) = C_h/(\xi'+1)$. The construction $|X_i| = L$ enforces $\bar{\rho} = \kappa_{\mathrm{avg}}/L$ 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. □

K.2.2. Cross-Measure Isolation (D2)

To formalize that varying one hyperparameter does not contaminate the other measures, we need a mild assumption coupling type and difficulty in the global pool. In the thresholded interpretation of Definition J.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 $\ell \perp \rho$. It is also incomparable with the mean-independence variant $\mathbb{E}[\rho \mid \ell = c] \equiv \mathbb{E}[\rho]$ (the global mean of the continuous ρ, not the inter-client mean $\bar{\rho}$), which

constrains the conditional mean but not the thresholded marginal. We caution that we do *not* report such a test on our experimental task pools, and TI is likely to hold only approximately in practice. On ALFWorld, for example, success rates visibly differ across the six task types (cf. the per-type columns of Table 1), so for the reference checkpoint s_c is at best near-constant. The results below degrade gracefully under approximate TI: if $\max_c |s_c - \bar{s}| \le \epsilon_{\mathrm{TI}}$, then every cell of Theorems 9 and 10 that invokes TI holds up to an additional additive $O(\epsilon_{\mathrm{TI}})$ term, and the TI-free cells are unaffected. The isolation guarantees should therefore be read as leading-order approximations, with an $O(\epsilon_{\mathrm{TI}})$ cross-contamination residual on the affected cells.

Theorem 9 (Cross-Measure Isolation, D2)**.** *Under Assumption 1, for each algorithm $\mathcal{A}_\theta \in \{\textsc{PrefPart}_\omega, \textsc{CovPart}_\xi, \textsc{HardPart}_{\xi'}\}$ with target measure $\Delta^2_{\mathcal{A}}$, every other measure $\Delta^2_{\mathcal{B}}$ ($\mathcal{B} \neq \mathcal{A}$) satisfies*

$$\mathbb{E}[\Delta^2_{\mathcal{B}}(\theta)] = \text{const} + O(1/\sqrt{L}) + O(1/\sqrt{N}) \qquad \forall\theta,$$

in particular, at leading order, $\partial_\omega \mathbb{E}[\Delta^2_{cov}] = \partial_\omega \mathbb{E}[\Delta^2_{hard}] = \partial_\xi \mathbb{E}[\Delta^2_{pref}] = \partial_\xi \mathbb{E}[\Delta^2_{hard}] = \partial_{\xi'} \mathbb{E}[\Delta^2_{pref}] = \partial_{\xi'} \mathbb{E}[\Delta^2_{cov}] = 0$.

Proof. We sketch the six cross-cells. *(I.a)* PrefPart $\times\Delta^2_{\mathrm{cov}}$: the algorithm enforces $|X_i| = L$ exactly, so $\Delta^2_{\mathrm{cov}} \equiv 0$ for all ω. *(I.b)* PrefPart $\times\Delta^2_{\mathrm{hard}}$: 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 $\mathbb{E}[\rho_i \mid q_i] = \bar{s} \sum_c q_{i,c} = \bar{s}$, decoupling from q_i (hence from ω). The remaining inter-client variance is the sampling-noise floor $\bar{s}(1-\bar{s})/L + O(1/N)$. *(II.a, II.b)* CovPart $\times(\Delta^2_{\mathrm{pref}}, \Delta^2_{\mathrm{hard}})$: the algorithm performs weighted-without-replacement sampling with weights proportional to remaining capacity, which is independent of ℓ and S_η, so per-client $\hat{p}_i$ and $\hat{\rho}_i$ are unbiased estimates of the global marginals with $O(1/\sqrt{n_i})$ noise. No TI is needed here. *(III.a)* HardPart $\times\Delta^2_{\mathrm{pref}}$: 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)* HardPart $\times\Delta^2_{\mathrm{cov}}$: the algorithm enforces $|X_i| = L$ exactly via $X_i = Y_i \cup F_i$ with $|F_i| = L - |Y_i|$, so $\Delta^2_{\mathrm{cov}} \equiv 0$ for all ξ'. The leading-order partial derivatives in the statement follow from these constants. □

K.2.3. Factor Invariance (D3)

Theorem 10 (Factor Invariance, D3)**.** *Under Assumption 1, for every algorithm $\mathcal{A}_\theta$ and every first-order mean $\bar{f} \in \{\bar{p}, \bar{n}, \bar{\rho}\}$, we have $\bar{f}(\theta) = \bar{f}^*$ exactly (by construction), exactly in expectation (by Dirichlet/Beta first moments), 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 θ, where the reference value $\bar{f}^*$ is fixed at p_0, L (or L_{avg}), and $\bar{s}$ (or κ_{avg}/L) respectively. The 9-cell preservation table is:*

Algorithm	$\bar{n}(\theta)$	$\bar{p}(\theta)$	$\bar{\rho}(\theta)$
PrefPart(ω)	$= L$ *exactly*	$= p_0$ *in expectation*	$= \bar{s}$ *leading-order (TI)*
CovPart(ξ)	$= L_{avg}$ *exactly*	$= p_0$ *leading-order*	$= \bar{s}$ *leading-order*
HardPart(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, with no drift in any of the three first-order means.

Proof. We verify the cells in turn. *Exact-by-construction cells:* PrefPart fixes $|X_i| = L$ via the Multinomial$(L; q_i)$ step. CovPart enforces $\sum_i n_i = NL_{\mathrm{avg}}$ via the renormalization step. HardPart fixes $|X_i| = L$ via $X_i = Y_i \cup F_i$ and $\sum_i |Y_i| = N\kappa_{\mathrm{avg}}$ 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 $\mathbb{E}[\bar{p}] = p_0$ for all ω. *TI cells:* for PREFPART $\times\bar{\rho}$, $\mathbb{E}[\rho_i \mid q_i] = \sum_c q_{i,c} s_c = \bar{s}$ under TI. For HARDPART $\times\bar{p}$, 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 COVPART, weighted-without-replacement sampling with capacity weights is independent of type and success indicator, so per-client $\bar{p}, \bar{\rho}$ inherit the global marginals at leading order. The sampling-noise residuals are $O(1/\sqrt{L}) + O(1/\sqrt{N})$. □

K.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 reachable 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 $\Phi(\{X_i\}) := (\Delta^2_{pref}, \Delta^2_{cov}, \Delta^2_{hard})$. 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 $\mathbb{E}[\Phi(\{X_i\})] = (\delta_p, \delta_c, \delta_h)$ in expectation, 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):* draw $\{n_i\}$ from COVPART at ξ^*, fixing Δ^2_{cov} to δ_c by Proposition 2. *Layer 2 (type):* draw 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):* draw a per-client target success *fraction* ρ_i^{q} 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*, $|Y_i| = \text{round}(\rho_i^{\text{q}} n_i)$. Drawing the fraction rather than the raw count is essential once $\{n_i\}$ varies (Layer 1 with $\delta_c > 0$): it makes $\rho_i = |Y_i|/n_i \approx \rho_i^{\text{q}}$, so $\Delta^2_{\text{hard}} = \text{Var}(\rho_i) = C_h/(\xi'^* + 1) = \delta_h$ by Proposition 3, decoupled from the Layer-1 size dispersion, and it guarantees $0 \le |Y_i| \le n_i$ so the filler $\nu_i = n_i - |Y_i| \ge 0$ is always well-defined. *Joint sampling:* given $(n_i, q_i, |Y_i|)$, draw 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), which is $\Theta(L)$ by the balanced-pool assumption and is drawn from 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_i^{q} (independent of the type split by uniform within-cell sampling, and independent of $\{n_i\}$ by the size-proportional draw). Sample-level concentration $O(1/\sqrt{L}) + O(1/\sqrt{N})$ follows from finite-sample multinomial residuals. □

Theorem 11 completes the (D1)–(D4) chain: (D1) target control (Propositions 1–3), (D2) cross-measure invariance (Theorem 9), and (D3) factor invariance (Theorem 10) compose into joint configurability over the open reachable region $\mathcal{R}$, providing the formal foundation for the partition-driven heterogeneity experiments used throughout §3.2 and Appendix K.3.

K.3. Per-Client Distribution Figures

We visualize the per-client task distributions induced by the three partition algorithms of Appendix K.1 at the two extreme dispersion settings used in §5.2: a near-uniform setting (small dispersion, grey curves in the main paper) and an extreme heterogeneous setting (large dispersion, blue curves). The first endpoint serves as the near-uniform baseline, while the second sets each axis to its operational maximum within feasibility bounds. Two qualitative observations are consistent across all three sub-types and both benchmarks. First, the global marginals are preserved between the two endpoints 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 §3.2. Second, the per-client variability is much larger at the heterogeneous endpoint, with individual clients departing far from the global average. Together these two patterns confirm that each hyperparameter cleanly controls its dispersion measure, modifying Δ^2 along its target axis without shifting the first-order means.

K.3.1. Preference Heterogeneity

Figure 3 (main text, §3.2) shows per-client category compositions (stacked bars over client IDs) for WebShop and ALFWorld respectively, at $\omega = 0.01$ (near-uniform) versus $\omega = 0.99$ (extreme heterogeneous). At $\omega = 0.01$, each client's histogram closely traces 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 the centralized pool, so the cross-client spread of p_i is small. At $\omega = 0.99$, by contrast, each client concentrates on one or two categories, 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 by orders of magnitude. The two benchmarks have different category structures (WebShop has five top-level product categories with two dominant heads, ALFWorld has six fairly balanced household task types), but the resulting concentration patterns are qualitatively similar at both endpoints, confirming that ω controls dispersion comparably on both benchmarks (and, by construction, on any benchmark with a categorical task structure).

K.3.2. Coverage Heterogeneity

Figure 9 shows the empirical distribution of per-client pool sizes $\{n_i\}$ at $\xi = 1$ (extreme heterogeneous) and $\xi = 256$ (near-uniform). At $\xi = 256$, the per-client bars are tightly concentrated around L_{avg}, so all clients face essentially the same per-epoch exploration breadth, and the corresponding Δ^2_{cov} is close to zero. At $\xi = 1$, the sizes spread out to cover the full $[L_{\min}, L_{\max}]$ interval, with a non-trivial fraction of clients at or near each boundary. This is the regime where some clients repeatedly traverse a narrow region while others explore broadly, and it is the regime where iterative-with-replacement RL sampling is most likely to expose the per-client breadth bottleneck discussed in §3.2. The global replica budget and the per-client mean L_{avg} are held identical across the two endpoints, so any cross-setting comparison isolates the dispersion effect rather than confounding it with total exposure. Because the partition operates only on item 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 rather than one per benchmark.

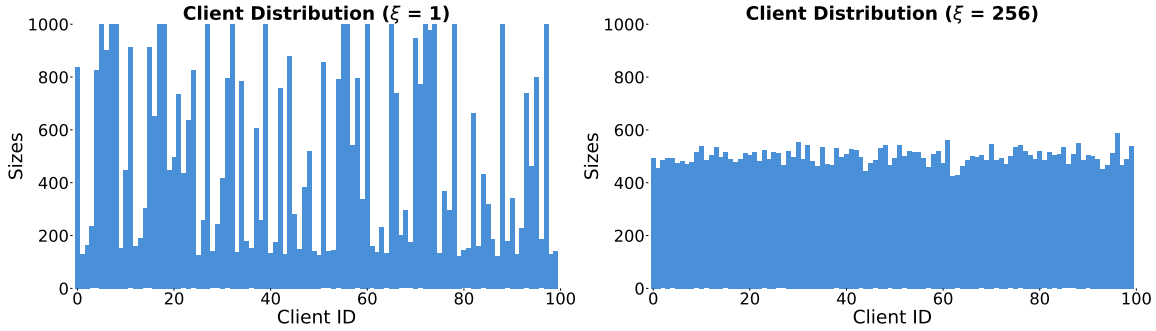

Figure 9 | **Client Distribution under Coverage Heterogeneity ($\xi = 1$ vs. $\xi = 256$).** Bars show each client's pool size n_i (client IDs on the horizontal axis). Left: $\xi = 1$ (extreme heterogeneous), right: $\xi = 256$ (near-uniform). Note that smaller ξ is *more* dispersed, opposite to ω in Figure 3. The per-client mean and the global replica budget are pinned across the two settings, so only the spread changes. The same construction is applied identically to WebShop and ALFWorld, as it operates purely on task indices.

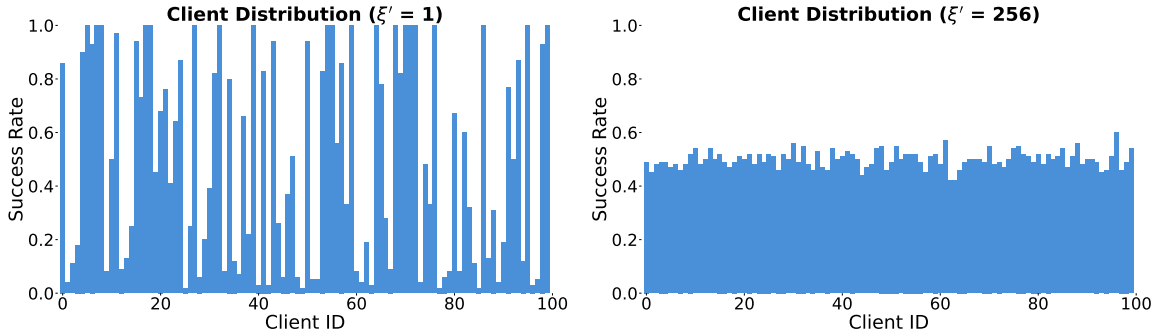

Figure 10 | **Client Distribution under Hardness Heterogeneity ($\xi' = 1$ vs. $\xi' = 256$).** Bars show each client's success-rate quota ρ_i (client IDs on the horizontal axis). Left: $\xi' = 1$ (extreme heterogeneous), right: $\xi' = 256$ (near-uniform). The per-client size and the global success rate are pinned, so only the spread changes. The same construction is applied to both benchmarks, with per-benchmark success pre-labels from the reference checkpoint π_{ref}.

K.3.3. Hardness Heterogeneity

Figure 10 shows the empirical distribution of per-client success-rate quotas $\{\rho_i\}$ at $\xi' = 1$ (extreme heterogeneous) and $\xi' = 256$ (near-uniform). At $\xi' = 256$, every client receives essentially the same proportion of "successful" tasks (those above the reference threshold η) and "unsuccessful" tasks, so the policy-gradient advantage signal is comparable across clients. At $\xi' = 1$, the success-rate quota spreads toward both extremes: some clients are dominated by easy tasks where the agent already saturates and produces no improvement signal, 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 tracks within-group success statistics. Although the two endpoints differ sharply in inter-client signal availability, the per-client size $|X_i| = L$ and the global success rate $\bar{\rho}$ are identical for every value of ξ' by construction (the unsuccessful pool $\mathcal{U}$ supplies the filler items), so the federated training comparison in §5.2 can attribute any divergence to Δ^2_{hard} alone.

L. Environment-Level Heterogeneity Variants

This appendix expands the construction sketched 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 not a single scalar quantity but the composition of several distinct mechanisms, and perturbing different mechanisms produces structurally different forms of $\pi^\star$ divergence and therefore different positions on the Pattern B/C/D spectrum of §4.4. Section L.1 formalizes this view as a four-stage transition pipeline and locates several common agent benchmarks within it. Section L.2 makes the four stages concrete on WebShop. Section L.3 specifies the deterministic per-client variant assignment procedure shared across all five variants (Algorithms 5–6). The remaining subsections (L.4–L.8) instantiate one variant per stage (plus a joint perturbation), giving in each case the motivation, an MDP-level formalization of what is changed, an implementation summary that 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 axis used in Figure 6 and isolate which sufficient condition fails along each axis.

L.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, code-base 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 retrieves differently and optimal query crafting becomes per-client. Stage 3 differences induce a *score-function shift*: identical (encoded) queries return different rankings, so optimal click order becomes per-client. Stage 4 differences induce *observation distortion*: identical rankings are presented through different rules, so optimal navigation through the observation becomes per-client. Stage 1 differences (the state-coverage shift) 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 break (C2) and force the policy to specialize on quantities (encoding, score, rendering) that are not directly observable, so satisfying any of (C1)–(C3) requires either an off-support shared optimum or in-context env identification.

Table 4 shows that every retrieval-style or simulator-backed agent benchmark we are aware

Table 4 | The four-stage transition pipeline instantiated on representative agent benchmarks. Each row enumerates the natural perturbation surface for one benchmark, illustrating that the decomposition extends naturally beyond WebShop.

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

of admits an analogous factorization, and federated training across clients with different content, indexing, scoring, or rendering yields four structurally distinct forms of env-level heterogeneity. We therefore treat the decomposition as a *systematic taxonomy* of env-level heterogeneity constructions rather than a WebShop-specific one. We instantiate at least one variant per stage to test whether the Pattern B/C/D classification of §4.4 tracks the structural form of the perturbation rather than any single variant's idiosyncrasies. (All five variants are instantiated on WebShop. Cross-benchmark instantiation is a natural follow-up.)

L.2. The WebShop Pipeline as a Concrete Instantiation

On WebShop, the agent's primary external action $a = \texttt{search}\,\langle q \rangle$ flows through 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 onto page 1 (ten products per page) for the agent to inspect.

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 re-tuned by (k_1, b) (Component 3), or the post-ranking step can be wrapped to alter the surface 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 specific subterm of the WebShop reward. 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 any observed degradation along a variant axis is attributable to its specific transition perturbation.

L.3. Deterministic Per-Client Variant Assignment

The five variants below share a common dispatch principle: 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 env-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 is trying to average. The implementation falls into two templates. Variant 1 (Catalog Split, §L.4) needs per-client catalog construction with two continuous hyperparameters (`env_div`, `keep_ratio`). We present this as Algorithm 5. Variants 2–5 share a discrete *variant-pool* template: each defines a finite pool of $|\mathcal{V}|$ alternatives, and each client is mapped to one pool entry by a deterministic seed. We present this 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.

Algorithm 5 CATALOGSPLITPARTITION (Variant 1)

Require: Global ASIN pool $C = \{c_1, \ldots, c_{|C|}\}$ with $|C| = 1000$; global target set $\mathcal{G} \subset C$ of all instruction-bearing ASINs ($|\mathcal{G}| = 415$), retained in *every* client's catalog; 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 $C_1, \ldots, C_N$ of identical size $|C_i| = |\mathcal{G}| + \text{round}(\texttt{keep_ratio} \cdot |\mathcal{R}|) = 803$ (at `keep_ratio` $= 0.7$)

$\mathcal{R} \leftarrow C \setminus \mathcal{G}$, then remove the 30 ASINs held out for OOD evaluation ▷ global distractor pool, $|\mathcal{R}| = 555$

$n_{\text{keep}} \leftarrow \text{round}(\texttt{keep_ratio} \cdot |\mathcal{R}|)$ ▷ fixed across clients; = 388 at `keep_ratio` $= 0.7$

$u_c \sim \text{Uniform}[0, 1]$ via RandomState(σ_0) for each $c \in \mathcal{R}$ ▷ global retention anchor (shared across clients)

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

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

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

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

  $C_i \leftarrow \mathcal{G} \cup \mathcal{R}_i^{\text{keep}}$ ▷ client catalog: all targets retained, distractors filtered

end for

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

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 use $|\mathcal{V}| = 4$, which assigns the 100 clients to the four pool entries with counts $(21, 20, 32, 27)$ under $\sigma_0 = 42$ (uniform in expectation, with the finite-sample imbalance expected from 100 independent randint(4) draws). 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 SimServer constructor accepts a separate `val_env_kwargs` that clears every entry `key` written by training-time variants, so the per-client perturbation does not affect the reported success-rate metric. This

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 $\text{cfg}_1, \ldots, \text{cfg}_N$

```
for i = 1 to N do
    j_i ← RandomState(σ_0 + i).randint(|V|)        ▷ deterministic pool index, identical across rounds
    cfg_i ← v_{j_i+1}   ▷ randint returns j_i ∈ {0, ..., |V| − 1}; one variant per client, frozen for all T rounds
end for
return {cfg_i}_{i=1}^N                              ▷ SimServer for client i reads env_kwargs[key] ← cfg_i
```

Table 5 | Per-variant instantiation of Algorithm 6. Each row specifies the pool $\mathcal{V}$, the supported pool sizes, and the SimServer override key `key`. The pool entries are listed explicitly in §L.5–§L.8.

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 default fields	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`.

keeps the variant comparisons consistent with one another and comparable to the homogeneous-environment baseline of §5.1.

L.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 env 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 task pattern "search for the goal attributes, click the matching ASIN, buy with the requested options" transfers verbatim across catalog subsets, so we expect this variant to leave $\pi^\star$ invariant on the shared support and probe whether the shared-optimum conditions (C1)/(C2) of §4.3 alone are sufficient for env-level robustness.

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

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

On the shared initial and search states the returned pages thus differ across clients through their distractor composition (and the re-indexed corpus statistics), 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 in $\mathcal{G}$ is retained in 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 $\pi^\star$) rather than literal (C1). (C1) holds only after abstracting states to their reward-relevant content (goal-product presence and position), under which the residual kernel disagreement is confined to reward-irrelevant page content.

Implementation. The per-client catalog C_i is constructed by Algorithm 5 (§L.3). The procedure runs three stages. *Stage 1: global target core.* Of the 1,000-product pool, the $|\mathcal{G}| = 415$ ASINs that carry a non-empty WebShop instruction (*i.e.*, can be the target of a synthetic goal) form a global target set $\mathcal{G}$ that is retained in *every* client's catalog, independently of which goals that client trains on, so every reward-bearing product stays reachable for every client. The remaining 585 ASINs are pure distractors, of which 30 are held out for OOD evaluation, leaving a global distractor pool $\mathcal{R}$ with $|\mathcal{R}| = 555$. *Stage 2: retention scoring.* Each $c \in \mathcal{R}$ receives a global anchor score u_c drawn once from RandomState(σ_0) (shared across all clients) and a per-client perturbation $v_{c,i}$ drawn from RandomState($\sigma_0 + 1000\,i$) (independent across i). The retention score $e_{c,i} = (1 - \texttt{env_div})u_c + \texttt{env_div}\, v_{c,i}$ interpolates between fully shared distractor selection ($\texttt{env_div} = 0$, all clients keep the same lowest-anchor distractors) and fully independent selection ($\texttt{env_div} = 1$, each client keeps a distinct lowest-score slice). *Stage 3: catalog assembly.* For each client we take the round($\texttt{keep_ratio} \cdot |\mathcal{R}|$) = 388 distractors with smallest $e_{c,i}$ and union them with the global target core $\mathcal{G}$. We use $\texttt{keep_ratio} = 0.7$, giving $|\mathcal{G}| + 388 = 803$ products in every client's catalog. The size is identical across clients and across `env_div`, since `env_div` changes only *which* distractors are kept, not how many. The curve in Figure 6 uses the strongest dispersion $\texttt{env_div} = 1.0$ (`env_div` is configurable over $\{0.0, 0.3, 0.7, 1.0\}$). At $\texttt{env_div} = 1.0$ two clients' retained distractor sets overlap at Jaccard ≈ 0.54, so once the shared target core $\mathcal{G}$ is included the pairwise Jaccard between any two clients' full catalogs is ≈ 0.75. The catalog is enforced at the SimServer layer: the constructor receives C_i as a list of ASINs and filters its internal product list `self.products` to that subset before indexing, so the BM25 backend (still the default Lucene searcher) re-indexes only the surviving products and the rest of the WebShop pipeline (reward function, page rendering, action grammar) is untouched. Validation rolls out on the unfiltered 1,000-product catalog, isolating the training-time perturbation from 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 overlays the single-env 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 ($\texttt{env_div} = 1.0$, pairwise catalog Jaccard ≈ 0.75). This gives the negative result that motivates the remaining variants: catalog heterogeneity alone, even when extreme, did not produce env-level collapse in our experiments.

L.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 (`name+Title` only, `description` only, `BulletPoints` only, the full concatenation, etc.). 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, and the agent must learn a per-client query-crafting style: short keyword queries on `name`-only clients, descriptive natural-language queries on `description`-only clients, and bulletized feature queries on `BulletPoints`-only 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

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

Identical (s, q) produces 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 part of the agent's observation, satisfying (C2) requires $\arg\max_a Q_i^\star$ to coincide across i on every reachable (s, τ), which fails whenever the optimal query under F_i would be ranked low under F_j.

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 search 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. The per-client variant is dispatched through Algorithm 6 with override key `bm25_in_memory_config.fields`: the variant pool at $|\mathcal{V}| = 4$ is the ordered list $(F_{\text{full}}, F_{\text{name}}, F_{\text{desc}}, F_{\text{bullets}})$, where $F_{\text{full}} = \{\texttt{name}, \texttt{Title}, \texttt{description}, \texttt{features}, \texttt{BulletPoints}\}$, $F_{\text{name}} = \{\texttt{name}, \texttt{Title}\}$, $F_{\text{desc}} = \{\texttt{description}\}$, $F_{\text{bullets}} = \{\texttt{BulletPoints}\}$. For $|\mathcal{V}| > 4$ we extend $\mathcal{V}$ with $\{\texttt{features}\}$, $\{\texttt{name}, \texttt{BulletPoints}\}$, $\{\texttt{description}, \texttt{features}\}$, and the *no-name* entry $\{\texttt{Title}, \texttt{description}, \texttt{features}, \texttt{BulletPoints}\}$ up to a code-level cap of $|\mathcal{V}| = 8$. The per-client BM25 hyperparameters are fixed at $(k_1, b) = (1.2, 0.75)$ across pool entries 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-level switch (`variant_pool: fields_only`) that the federated dispatcher `fed_env_manager.py` 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 287 agent queries from a baseline rollout against each of the eight variant indices: pairwise top-10 Jaccard averages ≈ 0.78 across the pool, which is moderate disagreement relative to Variant 3's ≈ 0.65. (A Jaccard of J over two size-10 sets corresponds to an overlap fraction of $2J/(1+J)$, so ≈ 0.78 means roughly nine of every ten top-10 results are shared between any two variants.) Validation overrides `val_env_kwargs` to clear `bm25_in_memory_config`, falling back to the Lucene searcher with the original F_{full} index, so the metric is measured on the unperturbed environment.

Expected pattern. We expect Pattern C. (C1) is broken because the optimal query under one field subset is suboptimal under another, (C2) is approximately broken because optimal action sets coincide on a majority of states but not all, and (C3) holds only weakly because the field subset is not directly inferable from the observation. We therefore expect a finite, observable degradation rather than full collapse, consistent with the moderate gap reported in Figure 6.

L.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: $k_1 \to 0$ collapses 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), $b = 1$ 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 produces different rankings on identical (q, d) pairs.

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

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

On the four extreme combinations we use, the average pairwise top-10 Jaccard is ≈ 0.65 and the rank-1 disagreement is $\approx 70\%$ across 287 replayed agent queries: that is, on roughly 200 of the 287 queries the four clients return four different top-1 products. (C2) breaks structurally because the same agent-emitted query returns differently ordered result pages across clients, so the optimal continuation after issuing that query (which position to click) differs across clients at the same pre-search state.

Implementation. We share 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 variant is dispatched through Algorithm 6 with the same partition function as Variant 2 but with the default pool (`variant_pool` kwarg unset). At $|\mathcal{V}| = 4$ the pool is the ordered list of probe-validated (k_1, b) combinations: $(1.2, 0.75)$ (default), $(0.3, 0.75)$ (saturated TF, term occurrence acts like a binary indicator), $(5.0, 0.75)$ (near-linear TF, long documents dominate), and $(1.2, 0.0)$ (no length normalization, long documents favored). 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 $\text{cfg}_i = (F_{\text{full}}, k_1^{(i)}, b^{(i)})$ 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 probe statistics quoted in the MDP-formalization paragraph above are obtained by replaying the same 287 baseline-rollout queries used for Variant 2, and the (k_1, b) combinations are visibly more disruptive: they reshuffle the ranking on the same encoded documents far more aggressively than alternative field subsets reshuffle which documents survive into the candidate set. Validation again clears `bm25_in_memory_config` via `val_env_kwargs`, falling back to the Lucene searcher at default (k_1, b).

Expected pattern. We expect Pattern C. The argument mirrors Field-Subset Index: (C2) is broken because the score function is per-client, and (C3) is again weak because the BM25

hyperparameter is not part of the observation. The empirical curve in Figure 6 shows a finite, stable gap rather than the larger-magnitude collapse that we report for the Rank Wrapper variant.

L.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) score highly under BM25 against the same queries the agent uses to retrieve the goal product, 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 agent must learn to inspect that specific subterm before purchasing, and the inspection skill is per-client because the attacked subterm differs across clients.

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

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

where each $p \in \mathcal{L}_i$ has its document text aligned with the goal (so BM25 places p near the true target) but its non-text fields (`pricing`, `customization_options`) 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. We verify empirically that r_{type} and the attribute-match count n_{attr} remain unchanged: the WebShop reward computes attribute matches via fuzzy string matching against the document text, which the lookalike preserves to mislead BM25, so the injection is a clean attack on a single reward subterm. End-to-end verification tests on 100 replayed queries show ΔR 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 rather than producing noise-level changes.

Implementation. The pipeline has two phases. *Phase 1: offline lookalike synthesis.* A standalone script `synthesize_lookalike.py` produces, for each goal in the WebShop training split, between zero and several lookalike products by (a) cloning the goal's product document text with light token mutations (synonym substitution, descriptor reordering) so that the lookalike still scores highly under BM25 against the goal's natural-language query, and (b) perturbing a designated non-text attribute set (`pricing`, `customization_options.color`, `customization_options.size`, or jointly `pricing` and `customization_options.color`) so the lookalike fails the WebShop reward decomposition on the corresponding subterm. The script is driven by a single fixed seed `SEED`= 99999, so the four output JSON files $\mathcal{L}_{\text{price}}$ (1,000 entries), $\mathcal{L}_{\text{color}}$ (286), $\mathcal{L}_{\text{size}}$ (268), and $\mathcal{L}_{\text{price+color}}$ (286) are byte-identical across runs and are checked into the configuration tree before training begins. Each lookalike ASIN is named `LK_<dim>_<orig_asin>_<rand_suffix>` so the prefix unambiguously identifies it as a synthetic product. *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 env constructor auto-routes search from the default Lucene backend to `InMemoryBM25Searcher` (the original Lucene index is read-only and does not contain `LK_*` ASINs, so adding products to it would require a re-index at every constructor call), with the augmented catalog $C \cup \mathcal{L}_i$ (of size 1,268–2,000 depending on the pool) 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 seen by search is enlarged. Validation overrides `val_env_kwargs` to clear `extra_products`, falling back to the unmodified 1,000-product Lucene index for a direct comparison under identical evaluation conditions with the homogeneous-environment baseline.

Expected pattern. We expect Pattern C under both optimizers. Empirically the GRPO run stabilizes in the C interior (~34%, comparable to BM25 Reweighting) because the variant's inspectable cue prevents it from reaching the D end. (C1) is broken because the optimal policy must specialize on per-client subterm inspection. (C2) is broken because two clients with different attacked subterms have different optimal action sets at the same observed state. (C3) is partial but backed by a concrete cue: the lookalike presence is exposed once the agent inspects the attacked field, an inspection skill the policy can acquire even under GRPO. This usable residual signal keeps the per-client gap below Theorem 4's worst-case $\Omega(R_{\max} H \delta)$ floor, so under GRPO the client-wise gradient conflict degrades performance but does not collapse it. Under PPO the clipped trust-region constraint adds further margin, with both settings falling into Pattern C as observed in §5.3. That cue is the structural feature that separates Lookalike from Rank Wrapper, whose env-identifying signal is far harder to exploit.

L.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 point of this construction is to isolate *behavior-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_{shuffle} and W_{partial}, possibly the presented top-K membership itself) is altered, which breaks any "trust the top position" heuristic the policy might have learned.

MDP formalization. For client i with wrapper W_i,

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

We use four wrappers: W_{default} is the identity, W_{shuffle} randomly permutes the top-50 candidates before truncating to K, W_{invert} reverses the order so that BM25's lowest-scoring candidate appears first, 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 exclude the obvious fifth wrapper W_{random}, which would always return a uniform sample, because it produces near-zero reward for the affected client (the goal product enters the random sample with probability $\approx K/|C| = 10/1000 = 1\%$), and the resulting near-zero gradient adds only noise to the FedAvg update.

Implementation. Each non-trivial wrapper is implemented as a small Python class that holds a reference to the base `InMemoryBM25Searcher` (configured with F_{full} fields,

$(k_1, b) = (1.2, 0.75)$, and the unmodified 1,000-product catalog) and post-processes its top-K output. `ShuffledTopKSearcher` requests the top-`shuffle_k` hits from the base searcher (with `shuffle_k` = 50, comfortably above the $K = 10$ products WebShop renders per page), draws a random permutation from a per-client `numpy.random.RandomState` seeded with $\sigma_0 + i$, and returns the first K of the permuted list. `InvertedTopKSearcher` reverses the base searcher's top-K list deterministically, with no random state. `PartialRandomSearcher` makes an independent random draw per query (the same per-client RandomState produces an independent Bernoulli draw at every `search` call). With probability `random_prob` = 0.5 it returns K ASINs sampled uniformly without replacement from the catalog and assigns them score 0, otherwise it forwards the base searcher's top-K. 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 overrides `val_env_kwargs` to clear `search_engine_variant`, falling back to the default Lucene backend with no wrapper, so reported success rate is measured on the standard evaluation environment.

Expected pattern. We expect Pattern D under GRPO and Pattern C under PPO. (C1) is broken because the wrapper alters the optimal click order. (C2) is broken on the shuffle and invert wrappers, since the position-1 product is no longer the highest-relevance one but the policy has no way to know this 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. Lookalike Injection's attack exposes a concrete inspectable cue. The Rank Wrapper's signal is far harder to exploit. Under GRPO, Rank Wrapper therefore realizes the worst-case floor (Pattern D) while Lookalike stabilizes in Pattern C. The margin left by PPO's clipping then raises Rank Wrapper into Pattern C as well.

M. 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 M.1), the proofs of Theorems 1–5 (Appendices M.2–M.4), and the convergence implications via the gradient-heterogeneity quantity ζ^2 that distinguishes the three robustness regimes at the optimization level (Appendix M.5).

M.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 LLM-specific (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 Bayes 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.
- **(R_{env})** *Env-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.
- **(R_{aug})** *History-augmented realizability*: the augmented class $\Pi_\Theta^{\text{aug}} = \{\pi_\theta(a \mid s, \tau, h_t)\}$, in which the LLM consumes the trajectory prefix h_t via its context window, can represent env-conditional optimal policies (cf. (C3) in §4.3).
- **(LR)** *Local realizability for lower bounds*: in worst-case env constructions, Π_Θ contains ϵ-approximate per-env local optima.

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 M.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. They 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 rather than rate-dependent: the task-level slack vanishes as model capacity grows (R′) and as the optimizer slack shrinks (Theorem 2), while the env-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 this is the standard target of supervised fine-tuning on multi-task instruction data, which already trains the model to produce task-conditional optimal behavior given the prompt as input. (R′) relaxes this 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′) realistic, as opposed to 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$_{\text{aug}}$) is exactly what in-context inference does). *(R$_{aug}$) augments the policy input with the trajectory prefix $h_t = (s_0, a_0, r_0, \ldots, s_t)$, so that the policy can condition on observations seen so far in the current episode rather than only on the current state. Operationally, this is what LLM-induced policies natively do: 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 machinery suffices to represent env-conditional optima when the env identity becomes inferable from history (the (C3) regime).*

Remark 5 (Why (LR) is mild). *(LR) requires Π_Θ to contain ϵ-approximations of the simple per-env optimal decision rules used in the worst-case constructions of Theorems 3 and 4, such as the always-arm-1 and always-arm-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. Its sole role is to ensure that the lower-bound proofs are not vacuously avoided by a policy class that cannot represent its own per-env optima.*

M.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 §M.1: Theorem 1 invokes (R), while Theorem 2 relaxes (R) to (R′) and additionally tracks a federated-optimization slack ϵ_{opt}. Throughout, $\mathcal{J}_\tau(\theta) := \mathcal{J}(\pi_\theta; \tau, \mathcal{M}_{\text{env}})$ denotes the trajectory 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 collapses 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 trajectory 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 $\tau \in \text{supp}(\bar{\mathcal{D}}_\tau)$. Since each $\mathcal{D}_{\tau_i}$ is absolutely continuous with respect to $\bar{\mathcal{D}}_\tau$ (indeed $\bar{\mathcal{D}}_\tau \geq N^{-1}\mathcal{D}_{\tau_i}$ as measures, which also gives $w_i := d\mathcal{D}_{\tau_i}/d\bar{\mathcal{D}}_\tau \leq N$ and hence the bound $\chi^2(\mathcal{D}_{\tau_i} \,\|\, \bar{\mathcal{D}}_\tau) = \int w_i^2\, d\bar{\mathcal{D}}_\tau - 1 \leq N - 1$ quoted in §4.1), the pointwise inequality $\mathcal{J}_\tau(\theta^\star) \geq \mathcal{J}_\tau(\theta)$ holds for all $\theta \in \Theta$ and all τ in the support of any $\mathcal{D}_{\tau_i}$. Taking expectation under any distribution $\mathcal{D} \ll \bar{\mathcal{D}}_\tau$,

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

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 Bayes 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 slot, no single θ could be pointwise optimal across heterogeneous tasks.*

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 $i \in [N]$,

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

Proof. The proof bounds the per-task gap $\hat{f}(\tau) := \sup_\pi \mathcal{J}_\tau(\pi) - \mathcal{J}_\tau(\hat{\theta}_{\text{fed}})$ on the federated mixture, translates it to client i via importance-weighted Cauchy–Schwarz, and then connects it to the per-client objective gap. 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

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

Splitting through any $\theta^\star$ realizing (R′),

$$\mathbb{E}_{\bar{\mathcal{D}}_\tau}[\hat{f}(\tau)] \;=\; \underbrace{\mathbb{E}_{\bar{\mathcal{D}}_\tau}[\sup_\pi \mathcal{J}_\tau(\pi)] - \mathcal{J}_{\text{fed}}(\theta^\star)}_{\leq \epsilon_{\text{approx}} \text{ by (R′)}} \;+\; \underbrace{\mathcal{J}_{\text{fed}}(\theta^\star) - \mathcal{J}_{\text{fed}}(\hat{\theta}_{\text{fed}})}_{\leq \epsilon_{\text{opt}} \text{ by definition}} \;\leq\; \epsilon_{\text{approx}} + \epsilon_{\text{opt}}.$$

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 first factor evaluates to 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 $\hat{f}(\tau)^2 \leq R_{\max} H \cdot \hat{f}(\tau)$, whence

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

where the last step invokes Step 1. Combining,

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

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

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

Indeed, $\sup_{\theta\in\Theta}\mathcal{J}_i(\theta)=\sup_{\theta\in\Theta}\mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\mathcal{J}_\tau(\theta)]\le\mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\sup_\pi\mathcal{J}_\tau(\pi)]$ (the unrestricted pointwise optimal 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 with Step 2 closes the proof. □

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 importing a convergence rate from federated-optimization literature). A complementary linear bound also holds: since $w_i\le N$ (cf. Part (ii) of the proof of Theorem 1), one has $\sup_\theta\mathcal{J}_i(\theta)-\mathcal{J}_i(\hat{\theta}_{fed})\le\mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\hat{f}(\tau)]\le N\,(\epsilon_{approx}+\epsilon_{opt})$, so the theorem's bound tightens automatically to the minimum of the square-root and linear forms. Regarding tightness, the square-root form is the sharper of the two precisely when $\epsilon_{approx}+\epsilon_{opt}\gtrsim R_{\max}H/N^2$, and its constants are attained (Cauchy–Schwarz is tight when $\hat{f}\propto w_i$, the boundedness step when $\hat{f}$ takes only the values $\{0,R_{\max}H\}$) only on the curve $(1+\chi^2)(\epsilon_{approx}+\epsilon_{opt})\asymp R_{\max}H$. In the small-slack limit the linear bound becomes the tighter of the two, so the square-root form's sharpness is regime-restricted rather than universal.*

M.3. Proofs of Theorems 3 and 4 (Env-Level Non-Robustness)

This subsection proves the environment-level non-robustness results stated in §4.2: Theorem 3 establishes the existence of a configuration with 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) introduced in §M.1, which guarantees that Π_Θ contains ϵ-approximations of the simple deterministic decision rules used in our worst-case constructions. The results delimit the scope of Theorem 2: in contrast to the task-level slack, which vanishes as capacity and optimization improve, the env-level slack persists as a structural property of the federated solution.

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

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

for some $\Delta(\{P_i\})$ depending on the construction.

Proof. We exhibit the transition-swap bandit. 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 function is $R(s_+)=R_{\max}$, $R(s_-)=0$, $R(s_0)=0$, and the task is trivial (single-task) so that all heterogeneity is confined to the transition kernels. The two clients differ only in dynamics:

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

This satisfies Definition 2 cleanly: $P_1 \neq P_2$, but the state space, action space, reward function, and task space are all shared. Per-client local optima are deterministic: $\pi_1^\star$ always picks a_1 (value $R_{\max}$), $\pi_2^\star$ always picks a_2 (value $R_{\max}$). By (LR), Π_Θ contains ϵ-approximations of these two simple policies, so $\sup_{\theta_i \in \Theta} \mathcal{J}_i(\theta_i) = R_{\max}$ up to negligible local-realizability slack.

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 $\Delta(\{P_i\}) \geq R_{\max}/2 > 0$, completing the existence proof. □

Remark 8. *The construction is minimal: $|\mathcal{S}| = 3$, $|\mathcal{A}| = 2$, $N = 2$, horizon 1. Any further reduction yields a homogeneous problem. 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 variants, Reptile-style aggregators) can break the indeterminacy: the failure is a property of the objective, not the optimizer. Disambiguation requires either personalization (distinct θ_i per client) or augmenting the input with env identity (Theorem 5).*

To upgrade 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 does not rely on it (it is obtained by direct computation on the explicit construction), but the bound is used later in the (C3) recovery proof (Section M.4).

Lemma 2 (Cross-MDP simulation)**.** *For two MDPs $\mathcal{M}_1, \mathcal{M}_2$ that share state space, action space, reward function, and discount factor 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*

$$\big|\mathcal{J}(\pi; \mathcal{M}_1) - \mathcal{J}(\pi; \mathcal{M}_2)\big| \;\leq\; \frac{R_{\max}}{(1-\gamma)^2}\,\delta \;=\; R_{\max} H^2\,\delta.$$

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 in the worst case. Under the policy-induced occupancy, 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 smaller occupancy-weighted divergence.

This is a standard result (Kakade and Langford, 2002; Kearns and Singh, 2002). We restate it here only to make the horizon scaling explicit.

Theorem (Environment-Level Non-Robustness, quantitative (Theorem 4))**.** *Under Definition 2 with shared task, 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 configurations and a corresponding policy class satisfying (LR), whose observations reveal neither past rewards nor outcomes (so that (C3) is unavailable), 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 $\sup_i [\sup_{\theta_i \in \Theta} \mathcal{J}_i(\theta_i) - \mathcal{J}_i(\theta^\star_{fed})] \geq \Delta_{pol}$.

Proof. The argument proceeds in three pieces: a smoothed bandit yielding $\Omega(R_{\max}\delta)$, a horizon-amplified contextual-bandit chain yielding $\Omega(R_{\max}T_{\text{hor}}\delta)$ under a silent-observation model, and a sup $\geq$ 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 $\delta \in [0, 1]$:

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

with $R(s_+) = R_{\max}$, $R(s_-) = 0$. Then $\sup_{s,a} D_{\text{TV}}(P_1, P_2) = \delta$ and the per-client local optimum satisfies $\sup_{\theta_i \in \Theta} \mathcal{J}_i(\theta_i) \geq R_{\max}\frac{1+\delta}{2} - \epsilon$ by (LR), with the negligible local-realizability slack ϵ absorbed into the $\Omega(\cdot)$ constant below (we write the exact value $R_{\max}\frac{1+\delta}{2}$ 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 a silent multi-step contextual bandit chain. We now amplify the bandit construction over T_{hor} rounds while preserving $\sup_{s,a} D_{\text{TV}}(P_1, P_2) = \delta$. Each episode consists of T_{hor} independent rounds: in each round, the state resets to s_0, the policy receives an abstract observation o, picks $a \in \{a_1, a_2\}$, and transitions to the reward state s_+ with probability $\frac{1+\delta}{2}$ (resp. $\frac{1-\delta}{2}$) on the locally optimal (resp. suboptimal) action, earning the shared deterministic reward $R(s_+) = R_{\max}$ (and $R(s_-) = 0$). Only the transition kernel differs across $\mathcal{M}_1$ and $\mathcal{M}_2$ (the reward is shared, as required by Definition 2), and the per-round bandit roles are swapped between them as in the smoothed bandit above. Critically, the policy is restricted to a *silent observation model*: $\pi_\theta(a \mid o)$ is conditioned on o only, with no access to past rewards or outcomes (this restriction makes the construction trigger Theorem 4's worst case rather than the (C3) regime).

In this silent regime, the best stationary policy in Π_Θ is to pick the same action across all rounds, since outcome history cannot disambiguate the two environments. Per-client local optima are therefore $\pi_1^\star$ = always-a_1 and $\pi_2^\star$ = always-a_2, with episode return $T_{\text{hor}}R_{\max}\frac{1+\delta}{2}$ each. For any federated $\pi_\theta(a_1 \mid o) = p$,

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

Therefore

$$\Delta_{\text{pol}} \;=\; T_{\text{hor}}R_{\max}\tfrac{1+\delta}{2} - T_{\text{hor}}R_{\max}/2 \;=\; \tfrac{T_{\text{hor}}R_{\max}\delta}{2} \;=\; \Omega(R_{\max}T_{\text{hor}}\delta).$$

Since only the choice at s_0 constitutes an agent decision step (the outcome state is absorbed into the next round's reset and requires no action), an episode of H agent steps contains $T_{\text{hor}} = H$

rounds in the episodic-undiscounted accounting of §2. If one instead counts the outcome state as a step, $T_{\text{hor}} = H/2$ and only the absolute constant changes. Taking $T_{\text{hor}} = H$ yields $\Delta_{\text{pol}} = R_{\max} H\delta/2 = \Omega(R_{\max} H\delta)$, the stated bound. The reward is bounded in $[0, R_{\max}]$ throughout, the policy class is non-trivial (it contains the per-round local optima by (LR)), and the divergence is exactly δ.

Discounted analogue. The undiscounted construction translates to the standard discounted-MDP setting as follows. Set $\gamma = 1 - 1/H$ and replace the T_{hor}-round episode with a discounted infinite-horizon episode in which the per-round bandit roles repeat and each round explicitly occupies *two* discount steps: a decision step at s_0 (choosing $a \in \{a_1, a_2\}$, reward 0) followed by an outcome step at $s_\pm$ (state reward $R(s_+) = R_{\max}$, $R(s_-) = 0$, single transition back to s_0). This keeps the reward function shared and state-based, exactly as in the smoothed bandit, and keeps the construction observation-silent (the decision state is always s_0, which encodes no outcome history). The per-client locally optimal return is then $\sum_{k=0}^{\infty} \gamma^{2k+1} R_{\max} \frac{1+\delta}{2} = \frac{\gamma R_{\max}(1+\delta)}{2(1-\gamma^2)}$, while the federated objective evaluates to $\frac{\gamma R_{\max}}{2(1-\gamma^2)}$ (constant in p by the same symmetry argument as the undiscounted case). Subtracting, and using $\frac{1}{1-\gamma^2} = \frac{H}{1+\gamma}$,

$$\Delta_{\text{pol}} \;=\; \frac{\gamma R_{\max}\delta}{2(1-\gamma^2)} \;=\; \frac{\gamma}{1+\gamma} \cdot \frac{R_{\max} H\delta}{2} \;\geq\; \frac{R_{\max} H\delta}{6} \;=\; \Omega(R_{\max} H\delta) \qquad (\gamma \geq 1/2),$$

yielding the stated horizon-amplified bound with a slightly smaller absolute constant than the undiscounted case.

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

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

where the first inequality is sup $\geq$ avg and the equality uses the optimality of $\theta^\star_{\text{fed}}$ for the federated objective. Combining 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 env-level slack persists for any optimizer. The single-step gap* $\Omega(R_{\max}\delta)$ *holds for any* Π_Θ *satisfying (LR). History cannot help there, since none exists before the sole decision. The horizon-amplified* $\Omega(R_{\max} H\delta)$ *holds for any silent (history-blind)* Π_Θ *satisfying (LR). Only structural conditions on* $\{\mathcal{M}_i\}$ *(Theorem 5) can eliminate it. The silent-observation model is essential to the H-amplification. This is deliberate: if the policy can observe per-round rewards or outcomes via context, Bayesian inference recovers env identity in* $O(1/\delta^2)$ *steps, the construction enters the (C3) regime, and the lower bound becomes negligible after the inference period, falling well below the worst-case* $\Omega(R_{\max} H\delta)$*. For history-conditioned policy classes (including LLM policies, whose context window natively carries* h_t*; cf. (*R_{aug}*)), the multi-step floor is thus exactly the (C3)-unavailable case, matching the statement of Theorem 4.*

M.4. Recovery Theorem Proof ((C1)–(C3))

This subsection proves Theorem 5, the recovery result that identifies exactly when the worst-case lower bound of Theorem 4 no longer applies. The recovery theorem identifies three structurally distinct regimes ((C1) common-optimal-off-support, (C2) action-preserving with shared $\pi^\star$, (C3) self-revealing environment) under which the per-client gap collapses to zero or to a controllable horizon-scaled slack. Each case relies on a different realizability assumption from §M.1: (C1)

and (C2) use (R_{env}), and (C3) uses (R_{aug}). Throughout, $d^{\pi}_{\mathcal{M}_i}$ denotes the discounted occupancy of π on $\mathcal{M}_i$ and $\delta_{\text{eff}}(\pi; i, j) := \mathbb{E}_{(s,a)\sim d^{\pi}_{\mathcal{M}_i}}[D_{\text{TV}}(P_i(\cdot \mid s, a), P_j(\cdot \mid s, a))]$ is the policy-induced (effective) divergence between $\mathcal{M}_i$ and $\mathcal{M}_j$.

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

(C1) *Under (R_{env}), if there exists $\pi^\star \in \Pi_\Theta$ such that (a) $\pi^\star$ is globally optimal in every $\mathcal{M}_i$ and (b) the policy-induced disagreement $\sup_{i,j} \delta_{eff}(\pi^\star; i, j) = 0$, then $\Delta_{pol} = 0$.*

(C2) *Under (R_{env}), if there exists a shared $\pi^\star \in \Pi_\Theta$ such that $\pi^\star(\cdot \mid s)$ puts all mass on $\arg\max_a Q_i^\star(s, a)$ for every i and every $s \in \text{supp}(d^{\pi^\star}_{\mathcal{M}_i})$, then $\Delta_{pol} = 0$. (This is the form of the main-text (C2) that applies here: when the optimal action sets coincide on every reachable state, any common selector realizable in Π_Θ serves as this $\pi^\star$. The stated form is slightly weaker, requiring only a shared selection.)*

(C3) *Under (R_{aug}), if env identity is $(t^\star, \eta_{cls}, \eta_{ker})$-self-revealing with recoverable prefix ((C3-cls), (C3-ker), (C3-rec)) in the sense of §4.3, then*

$$\Delta_{pol} \;\le\; O(R_{\max} t^\star) + O(R_{\max} H \eta_{cls}) + O(R_{\max} H^2 \eta_{ker}).$$

In particular, $\Delta_{pol} \to 0$ as $(t^\star, \eta_{cls}, \eta_{ker}) \to (0, 0, 0)$.

Proof. The three implications are proved separately. Each plugs the corresponding realizability assumption into a different optimality argument.

Case (C1). Condition (a) does the mathematical work. Condition (b) is included as a practical detection signature for (a). For the proof of $\Delta_{\text{pol}} = 0$: let $\pi^\star \in \Pi_\Theta$ satisfy (a), so $\mathcal{J}(\pi^\star; \mathcal{M}_i) = \sup_\pi \mathcal{J}(\pi; \mathcal{M}_i)$ for every i. By (R_{env}), there exists $\theta^\star \in \Theta$ such that $\pi_{\theta^\star} = \pi^\star$, 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 $\Delta_{\text{pol}} = 0$.

Why include (b) at all? Because direct verification of (a) requires solving every $\mathcal{M}_i$ to global optimality, which is operationally infeasible. The occupancy-based condition (b), $\sup_{i,j} \delta_{\text{eff}}(\pi^\star; i, j) = 0$, is easily estimated from rollouts and serves as a verifiable proxy: if rollouts of a candidate $\pi^\star$ never visit (s, a) pairs where the per-client kernels disagree, (b) is empirically certified. Note that (b) alone is *not* sufficient for $\Delta_{\text{pol}} = 0$: one can construct examples where (b) holds but (a) fails (some client has a strictly better policy outside $\pi^\star$'s occupancy), and (a) is needed to close the gap. Conversely, (a) alone establishes (C1). (b) only contributes operationally.

Case (C2). Let $\pi^\star \in \Pi_\Theta$ be the shared policy guaranteed by (C2). For each i, $\pi^\star$ selects an action in $\arg\max_a Q_i^\star(s, a)$ at every reachable state $s \in \text{supp}(d^{\pi^\star}_{\mathcal{M}_i})$. 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 (R_{env}), $\pi^\star \in \Pi_\Theta$ implies the existence of $\theta^\star \in \Theta$ realizing it, 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 (R_{aug}), history-conditioned policies are realized by the *same* parameter class Θ (the LLM's context window already carries h_t, and no new parameter is added, cf. the remark after (R_{aug}) in §M.1), so augmented policies are admissible in both suprema of Δ_{pol}

and its definition is unchanged. Consider the parameter $\theta^{\mathrm{aug}} \in \Theta$ realizing the following env-conditional policy: it follows the default prefix policy π_0 of (C3) for $t < t^\star$, adopts at the reveal step the env label $\hat{\imath}(h_{t^\star})$ inferred from the prefix, and thereafter executes the policy $\hat{\pi}_{\hat{\imath}}$ that is optimal for the *estimated* kernel:

$$\pi^{\mathrm{aug}}_{\theta^{\mathrm{aug}}}(a \mid s, \tau, h_t) \;=\; \hat{\pi}_{\hat{\imath}(h_{t^\star})}(a \mid s, \tau) \quad (t \geq t^\star), \qquad \hat{\pi}_j := \arg\max_\pi V^\pi_{\hat{P}_j},$$

where $\hat{\imath} : \mathcal{H} \to [N]$ is the classifier of (C3). 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 $\sup_{\theta \in \Theta} \mathcal{J}_{\mathrm{fed}}(\theta) \geq \mathcal{J}_{\mathrm{fed}}(\theta^{\mathrm{aug}})$. Combining,

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

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

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

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

where V_i^{post} denotes the value in $\mathcal{M}_i$ of the adopted post-reveal policy. The first bracket accounts both for the reward lost during the prefix and for the *state drift* induced by π_0. It is *not* controlled by $t^\star$ alone in a general MDP: a single prefix step can irreversibly enter a low-value (*e.g.* zero-reward absorbing) region, costing $\Omega(R_{\max}(H - t^\star))$ even when $t^\star = 1$. This is exactly what the recoverability condition (C3-rec) rules out: under (C3-rec), $\mathbb{E}_{(\mathcal{M}_i, \pi_0)}[V_i^\star(s_{t^\star})] \geq V_i^\star - O(R_{\max}\, t^\star)$, so the first bracket, and hence the prefix contribution, is $O(R_{\max}\, t^\star)$.

Step (b): post-reveal misclassification regret. For $t \geq t^\star$, the policy adopts the label $\hat{\imath}(h_{t^\star})$ for the rest of the episode, which is wrong with probability at most η_{cls} by (C3-cls). Note that (C3-cls) is stated under the prefix distribution generated by the same π_0 that θ^{aug} follows, so it applies verbatim. On this misclassified event, the policy follows $\hat{\pi}_{\hat{\jmath}}$ in $\mathcal{M}_i$ for some $\hat{\jmath} \neq i$ 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. This gives a misclassification-induced regret of

$$\mathbb{E}\left[V_i^\star - V_i^{\pi^{\mathrm{aug}}_{\theta^{\mathrm{aug}}}}\right]\Big|_{\text{post-reveal}} \;\leq\; \eta_{\mathrm{cls}} \cdot R_{\max} H \;=\; O(R_{\max} H \eta_{\mathrm{cls}}).$$

The horizon scaling here is H rather than H^2 because the label is fixed once at the reveal step $t^\star$: 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 rather than accumulating step-by-step.

Step (c): kernel-modeling regret. On the correctly classified event, the post-reveal policy is $\hat{\pi}_i = \arg\max_\pi V^\pi_{\hat{P}_i}$: optimal for the *estimated* kernel rather than the true one. With $\sup_{s,a} D_{\mathrm{TV}}(\hat{P}_i, P_i) \leq \eta_{\mathrm{ker}}$ by (C3-ker), applying Lemma 2 (cross-MDP simulation) twice yields the standard model-error bound:

$$V^{\pi_i^\star}_{P_i} - V^{\hat{\pi}_i}_{P_i} \;=\; \left(V^{\pi_i^\star}_{P_i} - V^{\pi_i^\star}_{\hat{P}_i}\right) + \left(V^{\pi_i^\star}_{\hat{P}_i} - V^{\hat{\pi}_i}_{\hat{P}_i}\right) + \left(V^{\hat{\pi}_i}_{\hat{P}_i} - V^{\hat{\pi}_i}_{P_i}\right) \;\leq\; 2R_{\max} H^2\, \eta_{\mathrm{ker}},$$

since the middle term is ≤ 0 by optimality of $\hat{\pi}_i$ under $\hat{P}_i$ and each outer term is at most $R_{\max} H^2 \eta_{\ker}$ by Lemma 2, contributing $O(R_{\max} H^2 \eta_{\ker})$ to the regret. Here the H^2 scaling does emerge, 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 and invoking ($\mathrm{R}_{\mathrm{aug}}$) to certify that θ^{aug} is realizable in the augmented class,

$$\Delta_{\mathrm{pol}} \;\leq\; O(R_{\max} t^\star) + O(R_{\max} H \eta_{\mathrm{cls}}) + O(R_{\max} H^2 \eta_{\ker}),$$

as claimed. In particular, when $t^\star = 0$, $\eta_{\mathrm{cls}} = 0$, and $\eta_{\ker} = 0$, the bound collapses to $\Delta_{\mathrm{pol}} = 0$, agreeing with the (C1) and (C2) results. □

Remark 10. *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 env label at the reveal step $t^\star$, 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 honest bound would inflate 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 conflating the two double-counts the horizon. For LLM agents, the typical regime is $\eta_{cls} > 0$, $\eta_{ker} = 0$ (the LLM occasionally identifies the wrong env, but once it identifies an env correctly, the inferred dynamics match the true dynamics, since the policy invokes the env's actual API). In this regime, only the H term remains and the bound is mild.*

Remark 11. *The (C3) bound interpolates smoothly between the worst-case $\Omega(R_{\max} H \delta)$ of Theorem 4 and the vanishing gap of (C1) and (C2). This predicts when env-heterogeneous federated fine-tuning can work without modification on real LLM-agent deployments: wherever branding, tool messages, and DOM structure make $t^\star = O(1)$ and η_{cls} small from the very first step. In such deployments, env identity is effectively recast as part of the input, reducing the situation to task-level heterogeneity, where Theorem 2 guarantees vanishing slack. In the language of the four-pattern decomposition (§4.4), this is the structural reason to expect Pattern B to dominate Pattern D in deployments with verbose, identity-revealing observations, giving way to Pattern D mainly on adversarial constructions like the observation-silent multi-step contextual bandit chain of Theorem 4.*

M.5. Convergence Implications via ζ^2

The previous subsections established *quality-of-solution* statements: Theorem 2 controls the per-client sub-optimality of any federated maximizer by a vanishing slack, while Theorem 4 identifies a structural floor of order $\Omega(R_{\max} H \delta)$ in the worst-case env-level regime. The present subsection bridges these statements with the *convergence-rate* quantities that prior FedAvg / local-SGD analyses focus on. The bridge is provided by the standard gradient-heterogeneity quantity ζ^2, which is central to FedAvg, SCAFFOLD, and most other supervised-FL and FedRL convergence theories (and which even heterogeneity-independent rates such as FedSVRPG-M must confront at the solution-quality level, cf. the closing remark below). Our point is that the Asymmetric Robustness Mechanism implies a structurally different ζ^2 shape under task-level versus env-level heterogeneity: under task-level heterogeneity (and under env-level heterogeneity satisfying (C1), (C2), or (C3)), ζ^2 is *transient* and decays along the optimization trajectory. Under worst-case env-level heterogeneity, ζ^2 has an *irreducible floor* at the worst federated stationary point that no choice of optimizer, batch size, or local-step count can close. This ζ^2-shape statement is orthogonal to, and composes with, any specific FedRL convergence rate the reader may wish to apply.

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

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

where $\nabla \mathcal{J}_i$ is the per-client policy gradient and $\nabla \mathcal{J}_{\text{fed}} = \frac{1}{N}\sum_{i=1}^{N} \nabla \mathcal{J}_i$ is the federated gradient. The supremum $\sup_{\theta\in\Theta} \zeta^2(\theta) \le \zeta^2$ is the inter-client heterogeneity bound used throughout the federated optimization literature. Standard FedAvg / local-SGD convergence analyses (Karimireddy et al., 2020; Li et al., 2020c) yield bounds of the prototypical form

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

where Φ_{alg} is an algorithm-specific convergence measure (*e.g.*, $\frac{1}{T}\sum_t \mathbb{E}\|\nabla \mathcal{J}_{\text{fed}}(\theta_t)\|^2$ for stationary-point analyses, or the federated objective gap for objective-gap analyses), K is the per-round local-step count (into which the protocol's E local epochs unroll), σ^2 is the intra-client stochastic-gradient variance, L_{sm} is a smoothness constant for $\mathcal{J}_{\text{fed}}$ (unrelated to the pool-size symbol L of §3.2), D_0 is the initial sub-optimality, and $(\alpha_\sigma, \alpha_\zeta, \alpha_0) \in (0, 1]^3$ are exponents specific to the analysis (*e.g.*, $\alpha_\sigma = 1/2$ for vanilla FedAvg, $\alpha_\sigma = 1$ for SCAFFOLD with control variates). We do not assume any particular choice of these exponents: Equation (1) is invoked only to localize the role of ζ^2 as the universal additive heterogeneity term that appears in every such rate. We deliberately write the heterogeneity term as decaying in T alone: in several analyses it does not improve with the local-epoch count K or the client count N, and can even degrade with K through client drift (Karimireddy et al., 2020). We also note that Li et al. (2020c) is a strongly convex analysis whose heterogeneity measure is an objective-value gap rather than ζ^2. We cite it only for the overall shape of the rate. The *quality* of a returned $\hat{\theta}_T$ as a per-client solution is then governed by whether $\zeta^2(\hat{\theta}_T)$ is transient (decays along the trajectory) or has an irreducible floor.

Task-level: ζ^2 is transient. Under task-level heterogeneity (Definition 1, shared environment $\mathcal{M}_{\text{env}}$) and the pointwise realizability assumption (R), the mixture-equivalence used in the proof of Theorem 1 (§M.2) implies that the realizing $\theta^\star$ satisfies $\nabla \mathcal{J}(\pi_{\theta^\star}; \tau, \mathcal{M}_{\text{env}}) = 0$ pointwise on every $\tau \in \text{supp}(\bar{\mathcal{D}}_\tau)$. (This step uses standard regularity: $\theta^\star$ in the interior of Θ, $\mathcal{J}_\tau$ differentiable in θ, and a smooth, non-saturated parametrization.) Consequently, every per-client gradient vanishes at $\theta^\star$:

$$\nabla \mathcal{J}_i(\theta^\star) \;=\; \mathbb{E}_{\tau\sim\mathcal{D}_{\tau_i}}[\nabla \mathcal{J}(\pi_{\theta^\star}; \tau, \mathcal{M}_{\text{env}})] \;=\; 0 \quad \forall i,$$

which immediately gives $\zeta^2(\theta^\star) = 0$. Under the relaxed assumption (R') that drives Theorem 2, $\theta^\star$ is no longer an exact common stationary point: its per-client value-suboptimality is bounded by the vanishing capacity slack $\sqrt{(1+\chi^2)\, R_{\max} H\, (\epsilon_{\text{approx}} + \epsilon_{\text{opt}})}$ of Theorem 2, so $\zeta^2(\theta^\star)$ is *transient* rather than identically zero, shrinking to zero as $\epsilon_{\text{approx}}, \epsilon_{\text{opt}} \to 0$. We do not convert this value-level slack into a specific squared-gradient bound, as that would require a smoothness or PL bridge we deliberately avoid assuming (§M.1). The claim is only that the heterogeneity term in Equation (1) vanishes in the same limit. In either case, the heterogeneity term in Equation (1) is transient, and the federated rate matches centralized SGD on the mixture distribution up to an additive slack of the same order as that controlled by Theorem 2. Thus, on the rate side, FedAvg recovers the centralized rate in the limit. On the quality side, the federated solution recovers the centralized solution as the slack vanishes.

Env-level under (C1)–(C3): ζ^2 is transient (exact under C1/C2, slack-controlled under C3). Under (C1) common-optimal-off-support or (C2) action-preserving with shared $\pi^\star$ (paired with (R_{env})), the per-client objectives admit a common optimizer $\theta^\star \in \Theta$. At any such common optimum, $\nabla \mathcal{J}_i(\theta^\star) = 0$ for every i, so $\zeta^2(\theta^\star) = 0$ exactly, the heterogeneity term in Equation (1) is transient, and the federated convergence rate matches the corresponding centralized rate. Under (C3) self-revealing-environment (with (R_{aug})), the same common-optimum picture holds only when the slacks $(t^\star, \eta_{\text{cls}}, \eta_{\text{ker}})$ are exactly zero. For nonzero slacks, Theorem 5 gives only $\Delta_{\text{pol}} \le O(R_{\max} t^\star) + O(R_{\max} H \eta_{\text{cls}}) + O(R_{\max} H^2 \eta_{\text{ker}})$, and the corresponding $\zeta^2(\theta^\star)$ is controlled by the same slack quantities rather than driven to exactly zero. The three sufficient conditions modulate the *transient* shape of $\zeta^2(\theta_t)$ along the trajectory: under (C1) the policy avoids the disagreement region throughout training, so $\zeta^2(\theta_t)$ is small at every iterate. Under (C2) the value differences across envs make $\zeta^2(\theta_t)$ transiently large early in training, decaying toward zero. Under (C3) the LLM must first learn in-context environment identification, after which $\zeta^2(\theta_t)$ decays toward a slack-controlled residual that vanishes as $(t^\star, \eta_{\text{cls}}, \eta_{\text{ker}}) \to 0$.

Worst-case env-level: ζ^2 has an irreducible floor on the lower-bound family. Consider the transition-swap bandit family underlying Theorem 4 (§M.3), in which two clients have swapped optimal actions and the policy class Π_Θ is parameterized by a finite-dimensional softmax over the action set. Within this family, $\nabla \mathcal{J}_i(\theta)$ has a simple closed form. Because $\mathcal{J}_{\text{fed}}$ is identically constant on this construction (Theorem 4), *every* θ is a federated stationary point. At the symmetric interior point θ_{sym} the two clients' gradients are equal and opposite, each of magnitude $\Omega(R_{\max} H \delta)$, so they cancel in the average while leaving a large cross-client disagreement. Hence the gradient-heterogeneity is strictly positive somewhere on the federated stationary set,

$$\sup_{\theta:\, \nabla \mathcal{J}_{\text{fed}}(\theta) = 0} \zeta^2(\theta) \;\ge\; \zeta^2(\theta_{\text{sym}}) \;=\; \Omega\Big(R_{\max}^2\, H^2\, \delta^2\Big) \;>\; 0.$$

The floor is *attained* at θ_{sym} rather than holding at every stationary point: 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 establishes the *existence* of a ζ^2 floor on the federated stationary set under env-level heterogeneity, complementing the Theorem 4 objective-side gap.

Two convergence dimensions must be kept separate. First, on the *rate-to-stationary* side: the heterogeneity term $\zeta^2 / T^{\alpha_\zeta}$ in Equation (1) still tends to zero as $T \to \infty$ even when ζ^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 reachable $\hat{\theta}_T$ is bounded below by

$$\sup_i \Big[\sup_{\theta_i \in \Theta} \mathcal{J}_i(\theta_i) - \mathcal{J}_i(\hat{\theta}_T) \Big] \;\ge\; \Omega(R_{\max} H \delta),$$

regardless of T, the local-step count K, the number of clients N, the model capacity, or the optimizer. This is the direct objective-side guarantee of Theorem 4. This floor is inherent to the federated solution *structure*, not to the optimization *rate*: no fast-rate optimizer (variance reduction, control variates, momentum, FedSVRPG-M-style heterogeneity-independent rate) can close it, since the Theorem 4 bound is on the worst-client objective gap, which is invariant to the optimizer. This is the convergence-side counterpart to the structural distinction emphasized in §4 of the main text and in the Related Work positioning. In particular, our bound is consistent

with, and complementary to, fast-rate-to-stationary results such as FedSVRPG-M (Wang et al., 2024a): they bound the rate at which $\nabla \mathcal{J}_{\text{fed}} \to 0$ independently of ζ^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 shape	Quality of $\hat{\theta}_{\text{fed}}$ as per-client solution
Task-level (Definition 1, under (R'))	transient, $\zeta^2(\theta^\star) \to 0$	$\sqrt{(1+\chi^2)\, R_{\max}\, H\, (\epsilon_{\text{approx}} + \epsilon_{\text{opt}})} \to 0$
Env-level under (C1)–(C3)	transient; $\zeta^2(\theta^\star) = 0$ under (C1)/(C2), $\to 0$ as slacks $\to 0$ under (C3)	$= 0$ under (C1)/(C2); $\to 0$ as slacks $\to 0$ under (C3)
Worst-case env-level (under (LR))	attainable floor $\Omega(R_{\max}^2 H^2 \delta^2)$	$\geq \Omega(R_{\max}\, H\, \delta)$, irreducible

Table 6 | Three-regime convergence comparison. Under task-level heterogeneity and under env-level heterogeneity satisfying any of (C1)–(C3), FedAvg's local-SGD convergence is preserved up to a vanishing slack, with ζ^2 acting only as a transient bottleneck. Under worst-case env-level heterogeneity, the gradient-heterogeneity quantity ζ^2 attains a floor of order $R_{\max}^2 H^2 \delta^2$ at the symmetric federated stationary point, and, as a complementary objective-level statement, Theorem 4 places a per-client quality floor of order $R_{\max} H \delta$ at *every* federated stationary point, which no fast-rate optimizer can close.

Closing remark. Existing FedRL 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 $\sup_\theta \zeta^2(\theta) \leq \zeta^2$ treated as a black-box constant, while FedSVRPG-M-style analyses achieve rates independent of ζ^2 altogether. In the worst-case env-level regime, however, ζ^2 attains a floor of order $R_{\max}^2 H^2 \delta^2$ at the worst federated stationary point, so the rate-side bound remains valid but loses its force: the stationary point it certifies can be structurally biased against at least one client, so convergence no longer implies per-client quality. Our ζ^2-shape analysis complements those rate-side bounds: it tells the reader what *quality* the converged $\hat{\theta}_T$ has as a per-client solution, whichever rate-side bound one prefers. This bridges the quality-of-solution statements of Theorems 2/4 and the rate-of-convergence statements of prior FedRL theory.

N. Empirical Patterns: Extended Discussion

This appendix is a self-contained description of how the four empirical training-curve patterns of §4.4 arise from a single architectural fact about LLM agents. We start by restating that fact and following the unified causal chain from it to each pattern (§N.1); then examine each pattern's mechanism in turn (§N.2 through §N.5); bridge the four patterns to the optimization layer through the gradient-heterogeneity quantity ζ^2 (§N.6); convert the mechanism into two complementary operational tools, a post-hoc diagnostic decision tree (§N.7) and a pre-deployment mitigation protocol (§N.8); and conclude with a synthesis of the mechanism's predictive and falsifiable content (§N.9). A reader who has only read §4 of the main text should be able to follow this appendix without flipping back. An operator who only needs the deployment procedure can skip directly to §N.8.

N.1. The Unified Causal Chain

The mechanism in §4 can be restated as a single causal chain rooted at one architectural fact about LLM-induced policies. This subsection states the fact, follows its two qualitatively different consequences, and goes through the chain end-to-end. The remaining subsections treat individual links of the chain in detail.

N.1.1. The Single Architectural Fact: Input-Dynamics Asymmetry

The chain starts from how LLM agents are built in practice:

> **Asymmetry (A).** 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.

(A) is descriptive: the prompt to a deployed LLM agent contains τ (the user's task) but does not contain P (the env it is acting in). The env is whatever HTTP endpoint, terminal, or DOM tree the agent's tool calls reach, and is opaque to the policy except through observation streams. Every theorem and pattern in this appendix is a consequence of (A) acting on 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 §M.1.

N.1.2. The Two Sides of Asymmetry

Asymmetry (A) has qualitatively different consequences depending on which side of the input/dynamics distinction the heterogeneity lies on.

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 uses the prompt to switch behaviors. Concretely, under Definition 1 the federated objective collapses 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 optimizer-visible loss landscape depends only on the mixture, not on how task mass is split across clients. This is the "benign" side of (A): no matter how extreme the per-client task split, the federated optimization faces the same loss surface as i.i.d. mixture training.

Env-level side: P_i is in the dynamics. A single θ *cannot* encode the function $i \mapsto b_i$ from env 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 rolled out in. 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 are three structurally distinct ways for this question to admit a "yes" answer: (C1) shared optimum on shared support; (C2) action-preserving with value differences; (C3) self-revealing observations let an augmented class Π_Θ^{aug} realize per-env 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-env optima and the $\Omega(R_{\max} H\delta)$ floor applies. The conditions are sufficient rather than exhaustive, so robustness without any of (C1)/(C2)/(C3) remains possible outside that construction class (§N.8.4).

N.1.3. The Full Causal Chain

Combining §N.1.1 and §N.1.2, the chain from Asymmetry (A) to each empirical pattern factors into two branches.

Task-level branch (→ Pattern A).

(1) Asymmetry (A): τ is in the input, so a single θ can encode $\tau \mapsto b(\tau)$.
(2) Theorem 1(i): $\mathcal{J}_{\text{fed}}(\theta) = \mathbb{E}_{\bar{\mathcal{D}}_\tau}[\mathcal{J}(\pi_\theta; \tau, \mathcal{M}_{\text{env}})]$, the mixture-equivalence identity.
(3) The optimizer-visible loss landscape depends only on $\bar{\mathcal{D}}_\tau$, hence is identical across any task split with the same mixture.
(4) At the federated stationary point, the gradient-heterogeneity quantity $\zeta^2(\theta^\star) \to 0$ under (R′).
(5) The federated training plateau coincides with the near-uniform baseline (the i.i.d.-style split of §5.2) up to estimation noise, and the transient curve is close up to local-SGD drift. **Pattern A.**

Env-level branch (→ Patterns B, C, D).

(1) Asymmetry (A): P_i is in the dynamics, so a single θ must choose one fixed behavior independent of env 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 are sufficient rather than exhaustive: when all three fail, the answer *may* be negative, and on Theorem 4's construction class it provably is (§N.8.4 discusses the empirically robust exceptions). The slack quantities $(\eta_{\text{cls}}, \eta_{\text{ker}}, t^\star)$ of §N.4 parameterize how closely each (C) holds (with $\widehat{\delta}_{\text{eff}}$ as the empirical signature of residual (C1) failure).
(4) One of three branches applies:
   - *All-(C) branch:* a shared $\pi^\star$ is exact and $\zeta^2(\theta^\star) = 0$ at the federated stationary point. Provided the optimizer attains this shared optimum (ϵ_{opt} small), the federated curve coincides with the single-env baseline (**Pattern B**). A nonzero ϵ_{opt} leaves the run in Pattern C despite full structural (C) satisfaction (the Catalog Split/GRPO case of §5.3).
   - *Partial-(C) branch:* a shared $\pi^\star$ is approximate and ζ^2_∞, the trajectory limit $\lim_{t\to\infty} \zeta^2(\theta_t)$ (cf. §N.6), is small but positive, controlled by the slacks. The plateau is stable but biased below the single-env baseline, by an amount bounded by the Recovery Theorem (Theorem 5). **Pattern C.**
   - *No-(C) branch:* no shared $\pi^\star$ exists, and the plateau is structurally biased by $\Omega(R_{\max} H \delta)$ at the objective level (Theorem 4). At the gradient level, the federated stationary set of the worst-case construction contains points with $\zeta^2 = \Omega(R^2_{\max} H^2 \delta^2)$, attained at the symmetric point θ_{sym}. Appendix M.5 deliberately claims no universal trajectory-level floor. The local-SGD trajectory oscillates between conflicting client gradients, and prior degradation can overwrite the LLM's pre-trained world model. **Pattern D.**

The chain is therefore: *single architectural fact → two structural sides → 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 empirical behavior decomposes further into quality

bias, oscillating updates, and prior degradation, as detailed in §N.5. The remainder of this appendix elaborates each link of the chain in turn.

N.2. Pattern A: Plateau Identity vs. Trajectory Drift

The mixture-equivalence identity (Theorem 1(i), underlying Theorem 2)

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

makes the federated population objective identical across any task split sharing the same mixture $\bar{\mathcal{D}}_\tau$. The first-order consequence is that the optimizer faces the same loss landscape under any split, so the converged plateau is identical. The empirical interpretation of Pattern A is therefore a two-part statement:

(P1) **Plateau identity (theorem-level).** The long-horizon plateau matches the near-uniform baseline up to estimation noise. This is a direct consequence of population-objective identity and is what Theorem 2 actually predicts.

(P2) **Trajectory closeness (algorithm-level).** The transient training curve under extreme task heterogeneity is close to the uniform-baseline curve, but the two are not identical in an algorithm-independent way. The residual transient drift depends on the local-SGD step count E, the learning rate, and aggregation frequency, and is bounded by standard FedAvg drift terms in E and the inter-round step size.

This decomposition matters because Pattern A can be misread as predicting bit-identical training curves, with any visible transient gap treated as a falsification. The actual prediction is plateau identity, and the empirical evidence in Figure 5 confirms that across all three task-level sub-types (preference, coverage, hardness), the federated and uniform-mixture plateaus coincide within run-to-run noise.

Falsifiability. A single observation falsifies Theorem 2: a task split $\{\mathcal{D}_i\}$ whose mixture matches the near-uniform baseline but whose federated training plateau differs from the uniform-baseline plateau by more than algorithmic noise. None of our task-level experiments produce such a divergence, which is the empirical content of "task-level robust, unconditional".

N.3. Pattern B vs. Pattern A: Same Plateau, Different Mechanism

Patterns A and B both produce a federated plateau that tracks a "best baseline" (near-uniform baseline for A, single-env baseline for B), but the underlying mechanism differs in a way that has 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 in Π_Θ^{aug} under (C3). Per-client population objectives differ (because $P_i \neq P_j$ in general), but their arg max coincides.

The signature of the difference appears in the transient phase, not at the plateau. Under Pattern A, transient curves across different splits are essentially indistinguishable because the loss landscape is the same. Under Pattern B, transient curves may visibly differ in shape across

env-variants (the per-client objectives differ), yet they converge to the same shared optimum and therefore reach the same plateau. Observing plateau identity *plus* non-trivial transient divergence is itself a signature that one of (C1)/(C2)/(C3) holds rather than mixture collapse, and provides a diagnostic for which condition applies.

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 env in-context, recasting env as input and recovering a task-level-style mixture-equivalence argument inside Π_Θ^{aug}. Pattern B in this regime is therefore best understood as "(C3) reduces env to an input dimension, and Pattern A's mixture-equivalence runs through".

N.4. The B/C/D Spectrum and Its Slack Parameters

The Recovery Theorem (Appendix M.4) is parameterized by quantitative slacks, not binary "(C) holds vs. fails" predicates. Patterns B, C, and D therefore form a continuous spectrum. The position of any given env-variant 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 decides whether a structurally achievable plateau is actually attained.

- $\eta_{\text{cls}} \in [0, \frac{1}{2})$: classification slack on the env 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 estimator (governs (C3), jointly with η_{cls} and $t^\star$).
- $t^\star \in \mathbb{N}$: burn-in horizon required before the env 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 $\hat{\pi}^\star$, which is obtained from a short per-client policy-improvement probe (test (T1) of §N.8.1) (the empirical detection signature for (C1); it does not appear as a term in the Theorem 5 bound, cf. §N.8.2).

Pattern B is the limit $(\eta_{\text{cls}}, \eta_{\text{ker}}, t^\star/H, \widehat{\delta}_{\text{eff}}) \to (0, 0, 0, 0)$ along the relevant axes. Pattern D is the regime where one or more slacks are large enough that the per-client gap saturates the worst-case $\Omega(R_{\max} H \delta)$ floor of Theorem 4. Pattern C populates the interior. Varying a single experimental quantity, such as reducing how much the observations reveal about the env while holding task and reward fixed, moves the outcome continuously from B through C to D, which is the spectrum visible in Figure 6: Catalog Split lies near the B end structurally (PPO attains the B-end plateau; under GRPO the nonzero ϵ_{opt} leaves the run appearing as Pattern C); Field-Subset Index, BM25 Reweighting, and Lookalike Injection fall in the C interior; and only Rank Wrapper lies at the D end under GRPO.

N.5. Pattern D Mechanism: Decomposing Oscillation and Forgetting

Pattern D's empirical signature includes more than a stable biased plateau: it also includes 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 fed stationary point but does not by itself predict oscillation or forgetting. The full set of observed Pattern D behaviors decomposes into three simultaneous mechanisms:

(D-i) **Quality bias.** The fed stationary point $\theta^\star_{\text{fed}}$ is structurally suboptimal for at least one client because no single θ can simultaneously encode mutually contradictory transition

kernels in Π_Θ. This is the $\Omega(R_{\max} H\delta)$ floor of Theorem 4, and it holds for any optimizer.

(D-ii) **Oscillating local updates.** Even at $\theta^\star_{\text{fed}}$, individual client gradients have non-zero magnitude (each client locally pushes θ toward its own env's optimum), and their directions conflict. After averaging, the net gradient is approximately zero (stationary in expectation), but along the local-SGD trajectory the parameter is pulled between conflicting client gradients. Multi-step local SGD ($E > 1$) amplifies this pull and can produce visible round-to-round oscillation in the global curve (as we observe for Rank Wrapper under GRPO, §5.3).

(D-iii) **Prior degradation.** 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 overwrites the pre-training prior. Capabilities the base LLM originally possessed (factual recall, multi-turn coherence, instruction following on out-of-distribution prompts) degrade, manifesting as catastrophic forgetting on top of the structural plateau bias.

Pattern D is the combination of (D-i), (D-ii), and (D-iii): structural quality bias at the stationary point, oscillating local updates around it, and prior degradation when those updates are large enough to overwrite the LLM's pre-trained representation. This decomposition explains why Pattern D in our experiments (Rank Wrapper under GRPO, Figure 6) appears as a low *and* unstable plateau rather than a clean stable bias, and why PPO's clipped local updates partially mitigate Pattern D by attenuating (D-ii) and (D-iii) without changing (D-i).

N.6. ζ^2 as the Mechanism Bridge: Rate vs. Quality

The gradient-heterogeneity quantity ζ^2 formalized in Appendix M.5 unifies all four patterns at the optimization level by separating *rate to stationary* from *stationary-point quality*:

- Pattern A: $\zeta^2(\theta^\star) \to 0$ at the fed stationary point under (R'). Per-client quality at stationary is optimal. The plateau matches the i.i.d. uniform-mixture baseline.
- Pattern B: $\zeta^2(\theta^\star) = 0$ exactly, either in Π_Θ under (C1)/(C2) or in Π^{aug}_Θ under (C3). Mechanism differs from A (shared-optimum existence rather than mixture identity), but the optimization-level signature coincides.
- Pattern C: $\zeta^2_\infty := \lim_{t\to\infty} \zeta^2(\theta_t)$ is small but positive, controlled by $(\eta_{\text{cls}}, \eta_{\text{ker}}, t^\star)$, with $\widehat{\delta}_{\text{eff}}$ serving as the empirical signature of residual (C1) failure rather than 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: per-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 $\zeta^2 = \Omega(R^2_{\max} H^2\delta^2)$, attained at the symmetric point θ_{sym}. However, $\zeta^2(\theta_t)$ itself may decay along the trajectory as the softmax saturates. Appendix M.5 deliberately claims no universal trajectory-level ζ^2 floor.

Under the smoothness and step-size assumptions inherited from Section M.5, all four patterns exhibit standard convergence *rates* to the fed stationary point. The differences across patterns are entirely at the level of *stationary-point quality*. Pattern D is therefore not a "training does not converge" phenomenon (the optimizer does reach a stationary point) but a "the stationary point is bad" phenomenon (fed stationary ≠ per-client optima). Table 6 of §M.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.

N.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 env-level: proceed to Step 3.

Step 2. Verify Pattern A. Compare the fed plateau to the near-uniform baseline.

- Plateaus match within run-to-run noise: Pattern A confirmed. No mitigation required. Given (R′) and a shared environment, robustness holds for any task split (Theorem 2).
- Plateaus diverge: Theorem 2 or one of its assumptions is violated. Diagnose: 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), then check the local optimizer for systematic bias (*e.g.*, a per-client KL constraint that saturates differently across heterogeneity levels).

Step 3. Localize on the B/C/D spectrum. Compare the fed plateau to the best single-env baseline.

- Match within run-to-run noise: **Pattern B**. At least one of (C1)/(C2)/(C3) holds. Identify which:
  - Ablate the env identity from the observation history (*e.g.*, remove brand strings, normalize tool message templates). If the plateau collapses, (C3) was responsible.
  - Hold context window fixed and vary the action set's overlap across envs. If the plateau collapses when overlap is removed, (C1) or (C2) was responsible.
- Plateau lower than baseline by a finite stable margin: **Pattern C**. Either the relevant (C) is partially holding, or the conditions hold fully but the optimizer leaves a nonzero ϵ_{opt}. Identify the dominant slack from the curve shape:
  - Long burn-in before the plateau saturates: $t^\star$ is the dominant slack (env identity is inferable but only after a long history prefix). Mitigation: inject env descriptor early in the prompt so $t^\star \to 0$, recovering Pattern B.
  - Plateau bias roughly linear in the env-divergence level: the residual (C1) failure signaled by $\widehat{\delta}_{\text{eff}}$ is dominant (a shared optimum holds only approximately). Mitigation: per-client fine-tuning or LoRA adapter on the worst clients.
  - Plateau bias 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 env tag.
  - Plateau recovers under a clipped/trust-region local optimizer (*e.g.* a short PPO probe) while the slack measurements (T1)–(T3) of §N.8.1 pass: the gap is optimization-level ϵ_{opt}, not structural. Mitigation is optimizer-side (clipping, smaller local steps), not structural.
- Plateau collapse, oscillation, or visible capability degradation: **Pattern D**. All of (C1)/(C2)/(C3) fail and Theorem 4's floor applies. 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 env, aggregate within clusters). A clipped local update (PPO-style) additionally mitigates (D-ii) and (D-iii) but does not change the (D-i) structural floor.

The decision tree converts the deployment question "is FedAgent working for my deployment?" into the observable question "where on the A/B/C/D map does my training curve fall?", and points to the structural change predicted to recover robustness when the answer is not Pattern A. This is the operational form of the Asymmetric Robustness Mechanism's claim that the patterns are a diagnostic tool rather than a taxonomy.

N.8. Operational Mitigation Protocol

The diagnostic decision tree of §N.7 maps an *observed* training curve back to the pattern that actually appears: it is post-hoc (run an experiment first, then determine the regime). A practitioner planning a federated deployment also needs the converse: *before* spending compute on a full run, decide 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 11 summarizes the protocol as a decision tree.

N.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 $\widehat{\delta}_{\text{eff}}(\hat{\pi}^\star; i, j) := \widehat{\mathbb{E}}_{(s,a)\sim d_{\mathcal{M}_i}^{\hat{\pi}^\star}}\left[D_{\text{TV}}(P_i(\cdot \mid s,a), P_j(\cdot \mid s,a))\right]$ along the candidate $\hat{\pi}^\star$. 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 support is a finite-sample proxy rather than a certificate (cf. §N.8.4).

(T2) **Test for (C2) (action-preserving with shared $\pi^\star$).** Run the same per-client probe as in (T1) and compute $\arg\max_a \hat{Q}_i^\star(s,a)$ at every reachable state. (C2) is satisfied when these argmax sets coincide across clients on shared support, even if $\hat{V}_i^\star \neq \hat{V}_j^\star$. 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 env-identity classifier $\hat{\iota} : \mathcal{H} \to [N]$ on observation prefixes h_t (we expect a logistic regression on bag-of-tokens or a small transformer head to suffice). Vary $t = 1, 2, \ldots, H$ and record the smallest $t^\star$ at which the classifier accuracy on a held-out probe set saturates. The residual error rate at $t \geq t^\star$ is the empirical $\hat{\eta}_{\text{cls}}$. For the kernel slack $\hat{\eta}_{\text{ker}}$, fit a one-step transition predictor $\hat{P}_{\hat{\iota}(h_t)}(\cdot \mid s,a)$ on probe rollouts and report its TV distance to per-client kernels (or set $\hat{\eta}_{\text{ker}} = 0$ when the policy invokes the env's actual API at inference time, in which case $\hat{P} = P$ trivially). In environments without per-round resets, additionally probe the recoverability condition (C3-rec) of §4.3: roll out the default prefix policy for $t^\star$ steps and check that post-prefix success rates are not systematically depressed relative to fresh starts.

The tests are independent: a deployment may simultaneously satisfy (C1) and (C3), or (C3) only, etc. Their empirical satisfaction levels feed directly into the bound of §N.8.2.

N.8.2. The Quantitative Per-Client Gap Bound

The Recovery Theorem (Theorem 5, restated and proved in §M.4) gives a sharp per-client gap bound parameterized by the measured slacks:

- **Under (C1) or (C2):** $\Delta_{\text{pol}} = 0$. The federated optimum coincides with the per-client optimum, no further mitigation needed.
- **Under (C3) with $(t^\star, \eta_{\text{cls}}, \eta_{\text{ker}})$ slacks:**

$$\Delta_{\text{pol}} \;\le\; \underbrace{O(R_{\max}\, t^\star)}_{\text{prefix regret}} + \underbrace{O(R_{\max}\, H\, \eta_{\text{cls}})}_{\text{misclassification regret}} + \underbrace{O(R_{\max}\, H^2\, \eta_{\text{ker}})}_{\text{kernel-modeling regret}}.$$

  The three terms are structurally distinct (one trajectory-prefix, one trajectory-event, one per-step compounding), and their relative magnitudes determine which slack dominates the residual gap. The prefix term assumes the recoverability condition (C3-rec) of §4.3 (automatic in per-round-reset and back-navigable environments). Where the default prefix can enter irrecoverable states, the prefix term degrades toward $O(R_{\max} H)$ and the (C3-rec) probe of (T3) should be run alongside the classifier test.
- **Under no-(C) (worst-case):** $\Delta_{\text{pol}} \ge \Omega(R_{\max}\, H\, \delta)$ by Theorem 4, 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. This bound 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 §N.4: as $(t^\star, \eta_{\text{cls}}, \eta_{\text{ker}})$ shrink toward zero the upper bound collapses to the (C1)/(C2) regime (Pattern B). As they grow the upper bound saturates the worst-case floor (Pattern D). The interior is 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 env-conditional optima. 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 certified and the gap is exactly zero. (b) alone is *not* sufficient (cf. the remark in the (C1) proof of §M.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.

N.8.3. Mitigations Indexed by the Dominant Slack

The four operational slacks $(t^\star, \eta_{\text{cls}}, \eta_{\text{ker}}, \widehat{\delta}_{\text{eff}})$ each admit a structurally distinct mitigation. Table 7 summarizes the list. Each entry is justified by the bound term it targets.

The list is structural rather than algorithm-specific: it identifies which axis of the deployment to change ($t^\star$ via prompt, η_{cls} via context window, η_{ker} via API wrapping, $\widehat{\delta}_{\text{eff}}$ via per-client correction), not which optimizer or learning-rate schedule to use. The optimizer choice (GRPO vs. PPO, FedAvg vs. SCAFFOLD) is orthogonal to which mitigation is appropriate, although it can affect (D-ii) and (D-iii) once the structural mitigation is in place.

N.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 gap, and the practitioner protocol depends on which one applies:

- **Case 1: a (C)-class condition outside our framework holds.** (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 genuinely different structural mechanism may apply, or a condition may hold while its finite-sample test fails. For example, heterogeneity confined to reward-irrelevant transitions structurally satisfies (C1), yet (T1) can still fail if the probe policies visit the disagreement region that the true optimum avoids. Either way, 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 env constructions of the kind in §L.

Empirical fallback protocol. When all three tests of §N.8.1 fail, run a short federated probe (a few rounds at full federation but with a small token budget) and measure the worst-client gap $\widehat{\Delta}_{\text{pol}}$. Compare the measurement to the worst-case floor $R_{\max} H \widehat{\delta}$ (using the empirical $\widehat{\delta} = \sup_{i,j,s,a} D_{\text{TV}}(\hat{P}_i, \hat{P}_j)$ from the probe rollouts):

(F1) If $\widehat{\Delta}_{\text{pol}}$ is within a small constant factor of $R_{\max} H \widehat{\delta}$, the deployment is in Pattern D in the not-yet-triggered sense (Case 2 above). Switch to per-client personalization or hierarchical aggregation as recommended in Table 7.

(F2) If $\widehat{\Delta}_{\text{pol}}$ is substantially below the worst-case floor, the deployment is in Case 1: there is a recovery mechanism not captured by (C1)/(C2)/(C3). 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: every deployment receives either a (C1)/(C2)/(C3) certification with a quantitative bound, a Pattern D personalization recommendation, or a documented "Case 1" exception that flags a research opportunity. This makes the mechanism usable as a deployment diagnostic.

N.9. Predictive and Falsifiable Use of the Mechanism

This subsection synthesizes §N.1 through §N.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 Asymmetry (A) under varying degrees of (C1)/(C2)/(C3) satisfaction:

- **Task-level = the benign side of (A).** Because τ enters the input, a single θ can encode $\tau \mapsto b(\tau)$ (under (R′)), and Pattern A (federated curve $\approx$ uniform-mixture baseline) is the expected consequence of mixture-equivalence. Pattern A is unconditional in the sense that no choice of per-client task split can break it.
- **Env-level = the pathological side of (A).** 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) are three structurally distinct ways for "yes" to be the answer. The slacks $(\eta_{\text{cls}}, \eta_{\text{ker}}, t^\star, \delta_{\text{eff}})$ measure how closely each (C) is satisfied. The position on the Pattern B/C/D spectrum is determined by these slacks together with the optimizer's optimization-level slack ϵ_{opt} (§4.4).

N.9.1. Connecting Theory and Experiment

The mechanism links theory and experiment in both directions. A reviewer who observes a particular training-curve shape can follow it through §N.7 back to the specific theorem and condition responsible for the shape:

- Curve $\approx$ near-uniform baseline under task-level heterogeneity $\Leftrightarrow$ the mixture-equivalence identity (Theorem 1(i)) behind Theorem 2, Pattern A.
- Curve $\approx$ single-env baseline under env-level heterogeneity $\Leftrightarrow$ Theorem 5 shared optimum, Pattern B. Which (C) holds can be further identified via the ablation prescribed in Step 3 of §N.7.
- Stable plateau biased below single-env baseline $\Leftrightarrow$ partial (C) holding with non-zero $(\eta_{\text{cls}}, \eta_{\text{ker}}, t^\star)$ slacks (signaled empirically by $\widehat{\delta}_{\text{eff}}$), or full (C) with a nonzero optimizer slack ϵ_{opt}, Pattern C. The dominant slack determines which structural mitigation applies (Table 7).
- Plateau collapse with oscillation and capability degradation $\Leftrightarrow$ Theorem 4 $\Omega(R_{\max} H \delta)$ floor, Pattern D. Mitigation requires per-client personalization or hierarchical aggregation rather than vanilla FedAvg.

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 §N.8. The diagnostic and operational protocols make this correspondence explicit at the level of curve shapes rather than theorem statements.

N.9.2. Falsifiability per Pattern

Each pattern is paired with a precise observation that would falsify the corresponding theoretical statement. We list these explicitly so that future work can rule out our framework if any of the following is observed:

- **Pattern A (falsifies Theorem 2).** A task split $\{\mathcal{D}_{\tau_i}\}$ with mixture matching the uniform baseline whose federated training plateau differs from the uniform-baseline plateau by more than algorithmic noise, while satisfying (R′) on the actual Π_Θ.
- **Pattern B (falsifies Theorem 5).** A deployment in which (T1) or (T2) or (T3) is verifiably satisfied (*e.g.*, a small env-classifier reaches near-zero error on early observation prefixes) but the federated plateau nevertheless falls substantially below the single-env baseline, with no algorithmic explanation.
- **Pattern C (falsifies the slack-bound of Theorem 5).** A deployment in which the measured slacks $(t^\star, \eta_{\text{cls}}, \eta_{\text{ker}})$ are small but the per-client gap $\widehat{\Delta}_{\text{pol}}$ is large compared to $O(R_{\max} t^\star)$ +

$O(R_{\max} H \eta_{\text{cls}}) + O(R_{\max} H^2 \eta_{\text{ker}})$, indicating the bound is loose by more than a constant factor.

- **Pattern D (falsifies the exhaustiveness interpretation of (C1)–(C3)).** A deployment in which all of (T1)/(T2)/(T3) fail with arbitrarily large $\delta = \sup_{i,j,s,a} D_{\text{TV}}(P_i, P_j)$ yet no plateau collapse occurs and the per-client gap remains $o(R_{\max} H \delta)$. This would indicate that (C1)/(C2)/(C3) are not exhaustive in the sense of §N.8.4, Case 1. (It would *not* falsify Theorem 4 itself, which is an existence statement about its own construction class. Falsifying the theorem would require reproducing the explicit transition-swap construction and measuring a gap well below $R_{\max} H \delta / 2$.)

The asymmetry between Pattern A and Patterns B/C/D in falsifiability is itself informative: Pattern A's falsification is a single counter-example (one task split with a divergent plateau), while Patterns B/C/D's falsification requires a calibrated comparison between measured slacks and the predicted bound. This reflects the structural difference between mixture-equivalence (a strict equality) and the recovery slack-bound (a quantitative inequality with non-vacuous constants).

N.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 locus**: do clients differ in $\mathcal{D}_{\tau_i}$, in P_i, or in both? If only in $\mathcal{D}_{\tau_i}$, the prediction is Pattern A. If (also) in P_i, proceed to (2).

(2) **How much the observations reveal**: are env identity-revealing tokens (URLs, branding, error templates, tool message formats) present in the early observation prefix h_t for $t = O(1)$? If yes, (T3) is likely to be satisfied with $t^\star = O(1)$ and small η_{cls}, predicting Pattern B.

(3) **Optimal-action overlap**: does the optimal action set $\arg\max_a Q_i^\star(s, a)$ coincide across clients on shared support? Affirmative answers (action-preserving across envs) predict Pattern B via (C2), independent of (C3). We expect this in deployments where envs share an underlying API but differ in latency or error rates.

(4) **Adversarial env construction**: is the env perturbation engineered to be observation-silent (no env identity in observations) and action-flipping (optimal actions differ across envs on shared states)? If yes, all of (C1)/(C2)/(C3) fail and the prediction is Pattern D. The five WebShop variants of §L are calibrated probes for this regime.

(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 mis-specified deployment characteristic.

Closing remark. Together, the unified causal chain (§N.1), the per-pattern mechanism analyses (§N.2 through §N.5), the optimization-level ζ^2 bridge (§N.6), the post-hoc diagnostic (§N.7), and the pre-deployment mitigation protocol (§N.8) turn a single architectural fact about LLM agents into a deployment-time toolkit: a prediction before training, a diagnostic during training, and a mitigation after training. So far the toolkit is validated only on our WebShop/ALFWorld

testbeds. Every observation along the chain is paired with a falsifiable theoretical claim and a structural mitigation, and the chain is specific to LLM agents in the sense that Asymmetry (A) holds automatically for LLM-induced policies (tasks arrive as text in the prompt) rather than as an assumption to be engineered.

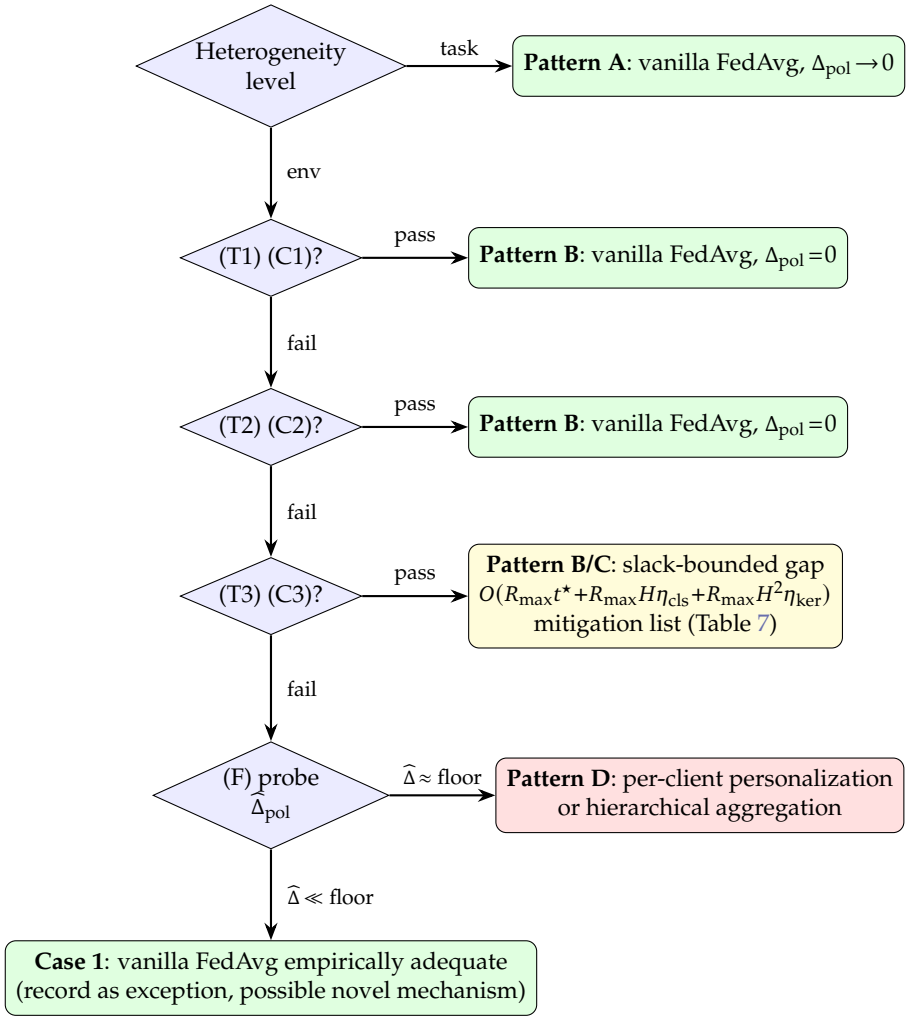

Figure 11 | **Mitigation decision tree for federated agent deployments.** Diamonds are tests defined in §N.8.1 (T1, T2, T3) and §N.8.4 (F). Rectangles are outcomes paired with the structural mitigation. Tests are evaluated sequentially top-to-bottom on a small probe rollout. The first passing test exits to the right with a vanilla-FedAvg or slack-bounded prescription. The empirical probe (F) at the bottom resolves the (C1)-(C3)-fail case into either Pattern D (personalization required, red) or Case 1 (vanilla FedAvg empirically adequate, green). "Floor" here refers to the worst-case lower bound $\Omega(R_{\max} H \delta)$ of Theorem 4. Color coding: green = $\Delta_{\text{pol}} = 0$ or $\to 0$; yellow = bounded but nonzero gap; red = structural floor applies.

Dominant slack	Empirical signature	Mitigation
$t^\star$ (burn-in horizon)	Long warmup before plateau saturates; final plateau eventually high	Inject env descriptor early in the prompt (*e.g.*, a one-line env tag prepended to the system prompt). The classifier $\hat{\iota}$ becomes trivial, $t^\star \to 0$, and the prefix regret term vanishes, recovering Pattern B.
η_{cls} (classifier slack)	Plateau bias non-monotone in the heterogeneity level; seed-dependent run-to-run gap	Enlarge the observation context window so $\hat{\iota}$ has access to a longer history. Alternatively add an explicit env 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)	Action-set normalization across envs (*e.g.*, wrap each env's API in a shared interface). Use env-specific tokenization so the policy does not need to model env-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 roughly linear in the env-divergence level; (C1) test fails but (C3) test succeeds	Per-client LoRA adapter on the worst clients (keep the global θ shared, attach a small per-client correction); or 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); or hierarchical FedAvg with env-clustered groups. PPO-style clipped local updates additionally mitigate (D-ii) and (D-iii) but do not change the (D-i) structural floor.

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 bound term in §N.8.2. Multiple slacks can be dominant simultaneously, in which case the mitigations compose.

O. Why the Mechanism Is Sharper on LLM Agents

The Asymmetric Robustness Mechanism (§4, with full proofs in §M) is stated for federated RL in general: it requires only Asymmetry (A) on the input/dynamics distinction (§3.1) and the realizability assumptions (R)/(R′)/(R_{env})/(R_{aug})/(LR) of §M.1. Theorems 2 and 4 would apply unchanged to a non-LLM agent satisfying the same input/dynamics geometry. This appendix argues that LLM agents make the mechanism *sharper* along three orthogonal directions: (i) the realizability assumptions on the task-level side hold almost automatically because pre-training already aligns Π_Θ with task-conditional optima (§O.1); (ii) the env-level side incurs an LLM-specific cost from pre-training prior degradation that is not captured by the formal $\Omega(R_{\max} H\delta)$ floor (§O.2); and (iii) the (C3) self-revealing-environment condition admits two LLM-specific instantiations, an explicit one captured by env-aware prompting (with a precise theorem-level reduction in §O.3) and an implicit one realized by in-context posterior inference (§O.4). Together these three directions explain why, although the mechanism is stated in MDP-level generality, we expect it to be sharpest for LLM agents specifically.

O.1. Task-as-Input: Pre-Training as Implicit Realizability

Theorem 2's task-level robustness bound carries a capacity slack ϵ_{approx} controlling how closely Π_Θ realizes the task-conditional Bayes-optimal policy. For LLM agents, the pre-training and instruction-tuning pipeline already makes 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 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′) asks that the federated mixture $\bar{\mathcal{D}}_\tau$ admits a $\theta^\star \in \Theta$ with bounded expected sub-optimality on it. Whenever the federated $\bar{\mathcal{D}}_\tau$ is a sub-distribution of $\bar{\mathcal{D}}_\tau^{\text{pretrain}}$, (R′) plausibly holds with ϵ_{approx} already small at θ_0, before any federated update. We expect this containment for typical downstream agent applications, since instruction-tuning datasets cover a vast range of task formats. The federated optimizer therefore 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.

Federated task-heterogeneous fine-tuning extends pre-training rather than working against it. 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. The gradient updates a federated client computes are (approximately) refinements within the instruction-following manifold of Θ that pre-training already places θ_0 on. As a result, we expect the cross-client gradient cosines to be biased positive 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 (improve instruction adherence) while differing on second-order content-specific corrections. The bounded gradient heterogeneity $\zeta^2(\theta)$ that limits generic FedRL convergence (Appendix M.5) should thereby be small in practice on LLM agents. We do not measure gradient alignment directly. 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 ϵ_{approx} is small at θ_0 for the pre-training-coverage reasons above, and the aligned gradients additionally keep the optimization-side cost small.

When $\bar{\mathcal{D}}_\tau$ exceeds the pre-training coverage. The "almost automatically" qualifier breaks down when the federated task distribution lies substantially outside $\bar{\mathcal{D}}_\tau^{\text{pretrain}}$. Niche-domain federations (*e.g.*, specialized clinical workflows, low-resource languages, idiosyncratic enterprise tools) push $\bar{\mathcal{D}}_\tau$ away from pre-training coverage, and 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 incurs a contribution that scales with how far the federated mixture deviates from pre-training coverage. We do not study this regime empirically (our WebShop and ALFWorld task spaces lie well within instruction-tuning coverage), but the theoretical bound continues to apply. The practical recommendation is to verify that ϵ_{approx} at θ_0 is small before launching a federated run, *e.g.*, by evaluating the released checkpoint zero-shot on a probe task pool.

O.2. Env-as-Implicit-Knowledge: Pre-Training Prior Degradation (Informal Claim)

The env-level side of Asymmetry (A) is symmetric to 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. Closing that gap is left open for future work.

LLMs carry 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 is the reason a freshly initialized LLM agent can already act sensibly in many environments without any RL fine-tuning. Concretely, θ_0 is not a from-scratch initialization in the sense classical FedRL theory assumes. It carries substantial environment-level knowledge 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 env-conditional dynamics for every $\mathcal{M}_i$ simultaneously. By Asymmetry (A), the policy has *no* input channel indicating which env it is currently in. The same prompt at s_0 must produce one action distribution. The LLM's only options are (i) dedicate each part of θ to one world model and incur error on the others, or (ii) average across world models and incur error on all of them. Either choice degrades the pre-trained prior in the regions of θ that encoded it.

Representation-level interference. The degradation above is structurally distinct from the gradient-level conflict already captured by Theorem 4: it is not just that the federated stationary point is suboptimal for any one client (the formal claim of Theorem 4), but that the parameters themselves drift away from the pre-trained prior into a region of parameter space that is optimal for no client. Empirically, this manifests as catastrophic forgetting of capabilities the base model originally possessed (factual recall, instruction following on out-of-distribution prompts, multi-turn coherence), which is the (D-iii) "prior degradation" component of Pattern D in §N.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: doing so requires a careful experimental design that quantifies catastrophic forgetting as ϵ_{approx} degradation at the parameter level, and the existing forgetting literature does not phrase the question in those terms. We therefore label the claim as informal: LLM agents incur an *additional* prior-degradation risk under env-level heterogeneity, beyond the formal $\Omega(R_{\max} H\delta)$ floor. We treat it as motivating why Pattern D in our LLM-agent experiments often appears as oscillation and capability degradation rather than a clean stable bias. 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 §N.5 is the formal floor that no optimizer removes. The (D-ii) oscillation and (D-iii) prior degradation components are the LLM-specific observed behaviors on top. 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. Table 7's "all slacks large" row implicitly composes both classes of mitigation. The present subsection clarifies why both are needed for LLM agents specifically.

O.3. (C3) Explicit Form: Env-Aware Prompting and Corollary 1

A key implication of Asymmetry (A) for LLM agents is that the input-vs-dynamics distinction is not absolute: env identity can be *moved* from the dynamics channel to the input channel by a deliberate prompt-engineering operation, namely prepending an env descriptor e_i to the agent's prompt. The (C3) condition of §4.3 is the formal statement that this move is feasible. It admits an explicit instantiation (env-aware prompting, this subsection) and an implicit one (in-context posterior inference, §O.4).

Env descriptors as moved-to-input env identity. Suppose the federation is augmented with an env 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:

- "*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; only /tmp is writable.*"

With e_i in the input, the policy class becomes the *augmented* class Π_Θ^{aug} of §M.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 (env-level) instance to a Definition 1 (task-level) instance over the augmented input distribution $\bar{\mathcal{D}}_{(\tau,e)}$. Theorem 2 then applies on the augmented mixture, with two residual slacks corresponding to the two ways the move can be imperfect: a *misclassification slack* η_{cls} if the env descriptor is approximate (*e.g.*, a partial OS version string instead of an exact one) and a *kernel-modeling slack* η_{ker} if the shared kernel estimator $\hat{P}_e$ matches the true per-client kernels only approximately (condition (i) of Corollary 1).

Corollary 1 (Env-Aware Prompting Reduces Env-Level to Task-Level Heterogeneity)**.** *Under Definition 2 with assumption (R_{aug}), suppose there exists an env descriptor map $e : [N] \to \mathcal{E}$ and a classifier $\hat{e} : \mathcal{H} \to \mathcal{E}$ with kernel estimator $\hat{P}$ such that:*

(i) *For every client i, every policy π, and every step $t \le H$, the trajectory-marginal misclassification rate is bounded:*

$$\Pr_{h_t \sim \mathcal{M}_i, \pi}\left[\hat{e}(h_t) \neq e_i\right] \le \eta_{cls},$$

and the kernel estimator satisfies $\sup_{s,a} D_{\mathrm{TV}}(\hat{P}_{e_i}(\cdot \mid s,a), P_i(\cdot \mid s,a)) \le \eta_{ker}$.

(ii) *The augmented policy class Π_Θ^{aug} on the input distribution $\bar{\mathcal{D}}_{(\tau,e)}$ satisfies the relaxed realizability (R′) of §M.1 on the augmented input, with capacity slack ϵ_{approx}^{aug}.*

Then for any federated optimizer output $\hat{\theta}_{fed}$ with sub-optimality ϵ_{opt}, every client i satisfies

$$\mathcal{J}_i(\hat{\theta}_{fed}) \;\ge\; \sup_{\theta\in\Theta} \mathcal{J}_i(\theta) \;-\; \sqrt{(1+\chi_i^{2,aug})\, R_{\max}\, H\, (\epsilon_{approx}^{aug} + \epsilon_{opt})} \\ -\; O(R_{\max}\, H\, \eta_{cls}) \;-\; O(R_{\max}\, H^2\, \eta_{ker}),$$

where $\chi_i^{2,aug} := \chi^2(\mathcal{D}_{(\tau_i,e_i)} \,\|\, \bar{\mathcal{D}}_{(\tau,e)})$ is the chi-squared divergence on the augmented input distribution.

Proof sketch. The argument is a two-step reduction. First, with e_i in the input, the federation reduces to a Definition 1 (task-level) instance over the augmented input (τ, e) in the augmented class Π_Θ^{aug}: conditioning on the observable descriptor e_i combines the family $\{P_i\}$ into a single shared surrogate kernel $\hat{P}$ on the augmented state space $\mathcal{S}\times\mathcal{E}$ (the descriptor selects the environment, so one common $\hat{P}(\cdot \mid s,a,e)$ represents every per-client kernel), which is exactly what makes the instance task-level rather than env-level. The runtime discrepancy between $\hat{P}$ and the true P_i is deferred to Step 2. Theorem 2 applied on this augmented Definition 1 instance over $\hat{P}$ gives

$$\mathcal{J}_i^{\mathrm{aug}}(\hat{\theta}_{\mathrm{fed}}) \;\ge\; \sup_{\theta} \mathcal{J}_i^{\mathrm{aug}}(\theta) \;-\; \sqrt{(1+\chi_i^{2,\mathrm{aug}})\, R_{\max}\, H\, (\epsilon_{\mathrm{approx}}^{\mathrm{aug}} + \epsilon_{\mathrm{opt}})}.$$

Second, write $\mathcal{J}_i^{\mathrm{aug}}(\theta)$ for the return of the augmented policy with the descriptor in its input and $\mathcal{J}_i(\theta)$ for the original descriptor-free return. The corollary's left-hand side $\mathcal{J}_i(\hat{\theta}_{\mathrm{fed}})$ is understood as the *deployed* return with the descriptor pipeline active, *i.e.* $\mathcal{J}_i^{\mathrm{aug}}(\hat{\theta}_{\mathrm{fed}})$ (we keep the symbol $\mathcal{J}_i$ because deployment always prepends the descriptor), while the right-hand side $\sup_\theta \mathcal{J}_i(\theta)$ is the descriptor-free per-client optimum of Theorem 4. Augmenting the input only enlarges the class, $\Pi_\Theta^{\mathrm{aug}} \supseteq \Pi_\Theta$, so $\sup_\theta \mathcal{J}_i^{\mathrm{aug}}(\theta) \ge \sup_\theta \mathcal{J}_i(\theta)$ and substituting the augmented optimum only helps. The augmented return $\mathcal{J}_i^{\mathrm{aug}}$ approximates the deployed $\mathcal{J}_i$ up to two correction terms accounting for the imperfect classifier and kernel estimator: the trajectory-level misclassification regret $O(R_{\max}H\eta_{\mathrm{cls}})$ from condition (i)'s first inequality (analogous to Step (b) of the proof of Theorem 5 in §M.4, the descriptor being fixed once per episode), and the per-step kernel-modeling regret $O(R_{\max}H^2\eta_{\mathrm{ker}})$ from the second inequality (analogous to Step (c), via the cross-MDP simulation lemma Lemma 2 applied to the $\hat{P} \to P_i$ discrepancy). Combining these residuals with the augmented Theorem 2 bound yields the displayed inequality. In the explicit env-aware-prompting form the descriptor is prepended at $t = 0$, so the reveal is immediate ($t^\star = 0$) and the prefix-recoverability condition (C3-rec) of §4.3 is vacuous. If the descriptor is instead inferred from a nonempty prefix ($t^\star > 0$, the implicit form of §O.4), the corollary additionally inherits (C3-rec) and the $O(R_{\max}\, t^\star)$ prefix term of Theorem 5. □

Interpretation: env-aware prompting as a structural mitigation. Corollary 1 is the precise mathematical content of "env-aware prompting reduces env-level heterogeneity to task-level heterogeneity". Under perfect env description ($\eta_{\mathrm{cls}} = \eta_{\mathrm{ker}} = 0$), the bound collapses 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 ($\mathcal{D}_{(\tau_i,e_i)} = \mathcal{D}_{\tau_i} \otimes \delta_{e_i}$), the augmented divergence is maximal, $\chi_i^{2,\mathrm{aug}} = N - 1$, regardless of how similar the raw task distributions are. Env-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 $\sqrt{(1+\chi_i^{2,\mathrm{aug}})\, R_{\max} H\, (\epsilon_{\mathrm{approx}}^{\mathrm{aug}} + \epsilon_{\mathrm{opt}})}$: the $\chi_i^{2,\mathrm{aug}}$ factor may be $\Theta(\sqrt{N})$-sized, but the slack still vanishes with the capacity and optimizer slacks. Coarsening the descriptors so that several clients share an e lowers $\chi_i^{2,\mathrm{aug}}$ at the cost of raising η_{ker} (a single shared estimator $\hat{P}_e$ must then approximate several distinct per-client kernels), shifting weight from the vanishing $\sqrt{(1+\chi_i^{2,\mathrm{aug}})(\cdot)}$ slack onto the $O(R_{\max} H^2 \eta_{\mathrm{ker}})$ term. This is the B/C/D interpolation. With imperfect description, the bound interpolates between the task-level form and the worst-case env-level floor in proportion to the slacks, mirroring the B/C/D spectrum of §N.4. This makes env-aware prompting the natural candidate structural mitigation for env-level heterogeneity in LLM-agent deployments where env identity can be reliably described in natural language. It is the structural target of the $t^\star$-row mitigation in Table 7. Its empirical effectiveness is not tested here.

Composability. Corollary 1 composes cleanly with the mitigations of §N.8: env-aware prompting drives $t^\star \to 0$ and $\eta_{\mathrm{cls}} \to 0$, recovering Pattern B in the bound; per-client LoRA adapters compensate the residual (C1) failure signaled by $\widehat{\delta}_{\mathrm{eff}}$; and PPO-style clipped local updates (in contrast to GRPO) attenuate the (D-ii) oscillation and (D-iii) prior-degradation components. Of these, only the third is validated in our runs. The first two are theory-motivated and untested here. The three classes of mitigation address structurally distinct slacks and can be applied independently or jointly.

O.4. (C3) Implicit Form: In-Context Posterior Inference

Env-aware prompting (§O.3) is the explicit form of (C3): a deliberate prompt-engineering move that injects env identity into the input. The implicit form is specific to LLM agents: *the LLM identifies the env from observation history without being told to*.

In-context posterior inference is what frontier LLMs already do. Frontier LLM agents in environments that reveal identity through observation are often observed, at least anecdotally, to perform actions consistent with maintaining a posterior over env identity:

- 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 forward-slash conventions),
- 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 realize $\hat{\imath} : \mathcal{H} \to [N]$ in (C3) implicitly: the classifier is not a separate module but is the LLM itself performing posterior updating about which env it is in, given the trajectory prefix h_t observed so far. ($\mathrm{R}_{\mathrm{aug}}$) of §M.1 is the formal statement that $\Pi_\Theta^{\mathrm{aug}}$ realizes env-conditional optima from such history. What makes the assumption realistic on LLMs is that pre-training already equips them with the natural-language understanding required to read tool error

messages, DOM templates, and protocol responses as env-identifying signals.

How this differs from classical RL agents. Classical RL agents must learn such posterior inference from trial-and-error on the federated task itself, even when augmented with recurrent policies that can in principle condition on history. This is exactly what meta-RL methods do. Their policy class lacks the natural-language understanding to interpret observations as env-identifying signals zero-shot. LLM agents instead inherit the inference capability from pre-training as a side effect of next-token prediction on heterogeneous text. This is the most LLM-specific component of the Asymmetric Robustness Mechanism: (C3-implicit) gives LLM agents a way to recover robustness that non-pretrained agents can access only at additional on-task learning cost. It is the structural basis for expecting env-heterogeneous federated fine-tuning of LLM agents to work without modification on cue-rich real web/tool deployments, while still admitting failure on adversarial constructions like the observation-silent transition-swap bandit of Theorem 4.

The (C3-implicit) holds vs. fails dichotomy yields a testable deployment-level prediction. Federated LLM-agent deployments should be robust to env-level heterogeneity when env identity is revealed through observation (as in most real web and tool settings) and fail when env identity is suppressed (observation-silent benchmarks engineered to remove identifying cues). Our WebShop variants realize the suppressed end of this dichotomy. The cue-rich end remains untested here. The two regimes correspond to "(C3-implicit) holds" versus "(C3-implicit) fails", and §N.8.1's (T3) test for (C3) operationalizes this dichotomy as a measurable diagnostic. A single benchmark cannot distinguish "FedAvg is robust" from "FedAvg is robust because (C3-implicit) holds", and that distinction matters because the latter does not generalize to observation-silent envs. The five WebShop variants of §L probe this axis alongside (C1)/(C2): Catalog Split preserves a shared optimum, since (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 retains a usable partial (C3) cue (the attacked reward subterm is inspectable) and degrades but stabilizes in Pattern C; only Rank Wrapper suppresses usable cues and moves to Pattern D under GRPO (§5.3).

Sharper than the generic bound. Putting (C3-explicit) and (C3-implicit) together: LLM agents have *two* distinct ways to recover from env-level heterogeneity, neither readily available to generic FedRL agents (the explicit mechanism requires engineered env featurization, and the implicit one requires on-task meta-learning). Either the federation engineers can inject env descriptors via prompt (and Corollary 1 certifies the resulting bound), or the LLM realizes (C3) implicitly via in-context inference (and (R_{aug}) certifies the existence of θ^{aug} achieving it). Both mechanisms turn the input-vs-dynamics distinction of Asymmetry (A) into a movable boundary, which is the main sense in which the mechanism is sharper on LLM agents than on agents without natural-language pre-training.