BYTE-MATCHED EVALUATION OF RANK-SPLIT FEDERATED ADAPTATION FOR TOOL-USING VISION-LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using language agents interleave task reasoning with tool-specific execution. In federated deployments clients own heterogeneous tool environments and cannot share raw trajectories, which makes it natural to give every client private adaptation capacity alongside a shared, uploaded partition. We test that premise under a fixed upload budget rather than a fixed parameter count. Rank-split federated adaptation (DREL) spends a client's per-event upload on a shared rank-32 reasoning partition while keeping a rank-64 execution partition private, so at the identical $87{,}293{,}952$-parameter (333 MiB) upload it deploys exactly three times the adapter parameters of a plain rank-32 FedAvg control. On a four-client GTA protocol with two seeds, the threefold capacity does not convert into accuracy: DREL does not show an accuracy advantage over a byte-matched rank-32 FedAvg control (58.4% versus 60.1% on the pre-specified eight-cell macro endpoint, -1.74 points, 95% CI $[-3.56, +0.22]$). We then ask where that capacity goes by intervention rather than argument: zeroing the private partition at inference, with the uploaded partition byte-identical, raised mean accuracy from 58.53% to 60.85% across the 3 cells evaluated (3 of 3 improving), where the byte-matched control scores 59.30%. We also expose a confounded comparison in our own earlier work that had motivated dropping the method's routing component, and report lower observed GAIA accuracy than local adaptation on the one client evaluated. The reusable lesson is a protocol one: federated rank comparisons are confounded unless upload is matched exactly.

1 INTRODUCTION

Large language models augmented with tool calling have become a powerful paradigm for autonomous agents that reason, plan and act in complex environments (Wei et al., 2022; Wang et al., 2023; Yao et al., 2023; Schick et al., 2023; Qin et al., 2024; Patil et al., 2024; Li et al., 2023b; Gao et al., 2024; Liu et al., 2025). These agents emit interleaved sequences of reasoning traces (*Thought*) and tool invocations (*Code/Action*), enabling multi-step problem solving over heterogeneous APIs. A growing benchmark literature shows that this capability remains brittle across API calling, web navigation, software engineering and computer-use environments (Liu et al., 2024b; Mialon et al., 2024; Guo et al., 2024; Yoran et al., 2024; Drouin et al., 2024; Deng et al., 2023; Zhou et al., 2024; Xie et al., 2024; Jimenez et al., 2024), and recent studies show that multi-step tool-calling pipelines remain difficult to evaluate reliably and reproduce consistently (Hallinan et al., 2026; Zhang et al., 2026; Chuang et al., 2026; Yagubyan, 2026).

Real deployments of such agents are inherently distributed. Consider personal AI assistants running on user devices. Each device hosts a distinct set of applications and private APIs: one user may rely on Google Calendar and Gmail while another uses Outlook and Todoist. Training agents collaboratively across such clients without sharing private tool usage data requires federated learning (FL) (McMahan et al., 2017; Kairouz et al., 2021). However, federated learning is usually studied under statistical heterogeneity of a shared label or output space, whereas in tool-augmented agent settings the *action* space itself differs: our clients hold heterogeneous, partly overlapping tool sets with different API signatures, call patterns and argument schemas. This setting also aligns with the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

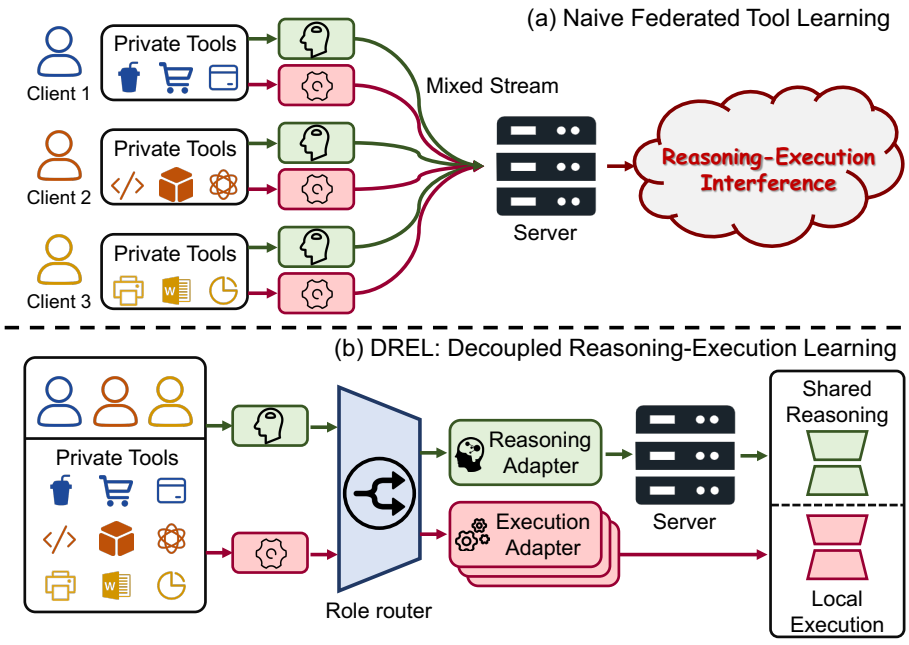

Figure 1: Reasoning–execution interference in federated tool-agent learning. Naive federated fine-tuning mixes general reasoning signals with client-specific tool execution signals, causing the shared model to encode heterogeneous execution patterns. DREL separates these roles so that reasoning can be aggregated while execution remains local.

privacy motivation behind decentralized optimization, where communication should expose as little user-specific behaviour as possible (Abadi et al., 2016).

The central design challenge is that a useful shared update should transfer task-level regularities without forcing every client to adopt another client's tool-grounding patterns. We call this conflict *reasoning–execution interference*: execution distributions $P_i(y_t \mid r_t = E)$ may differ across clients even when their trajectories share high-level task structure (Figure 1). Conventional full sharing mixes tool-grounded execution updates across clients, whereas DREL places the sharing boundary between role-specific adapter partitions.

This reframes the problem in a way that is, we believe, the crux for federated tool agents. Under a fixed per-round upload budget, a client can either *spend* the budget on a monolithic adapter that every client must share, or *split* it: publish a small part that carries transferable task structure and keep the rest private. The two choices transmit identical bytes but distribute capacity very differently, and the published rank-32 versus rank-96 comparisons do not separate them. We therefore make the matched-upload comparison the primary experiment in this paper: a plain rank-32 FedAvg baseline that uploads exactly the same number of bytes as DREL.

We propose DREL (**D**ecoupled **R**easoning-**E**xecution **L**earning), a federated adaptation framework that structurally separates reasoning and execution updates. DREL combines (1) **rank-split LoRA**, which partitions a fixed-rank update into reasoning and execution subspaces; (2) **role-selective aggregation**, which shares only the reasoning subspace while the execution subspace stays on the client; and (3) **semantic gradient routing** (SGR), which modulates backward updates by token role. Both subspaces are summed within the same adapted layer, so deployment requires one backbone forward pass.

Our contributions are:

- We formulate federated tool-agent adaptation as a *fixed-upload* question and instantiate it exactly: a shared rank-32 reasoning partition plus a private rank-64 execution partition against a plain rank-32 FedAvg adapter, both transmitting 87,293,952 parameters

(333 MiB) per client per aggregation event while the former deploys three times the adapter parameters (§3, §4).

- We show that the threefold deployed capacity does *not* convert into accuracy on the pre-specified eight-cell GTA endpoint, and we locate that null result rather than asserting a mechanism: a descriptive tool-availability split, a controlled inference-time partition intervention, and a single-seed training-corpus sensitivity analysis under rules frozen before any repaired model was scored (§5).
- We draw a methodological conclusion that outlives this method: rank comparisons in federated adaptation are confounded unless the *upload* is matched exactly, because a rank change alters the number of deployed parameters and the bytes transmitted at the same time. The routing component of the evaluated method remains unresolved and is not claimed (§5.3, §5.4).

2 Related Work

Tool-using LLM agents. Early reasoning-oriented work such as chain-of-thought and self-consistency improved latent deliberation, while ReAct made the reasoning–action interface explicit through interleaved traces (Wei et al., 2022; Wang et al., 2023; Yao et al., 2023). Toolformer, Gorilla, ToolLLM and API-Bank progressively broadened the tool-learning setting from self-supervised tool annotation to massive real-world API collections and benchmarked private tool pools (Schick et al., 2023; Patil et al., 2024; Qin et al., 2024; Li et al., 2023b), and ToolHop showed that multi-hop tool use remains challenging (Ye et al., 2025). More recent systems focus on plan–program separation, synthetic function-calling data and stronger execution protocols (Gao et al., 2024; Liu et al., 2025; Gim et al., 2024; Zhang et al., 2025). Evaluation has diversified in parallel, from large-scale API suites to realistic web, enterprise and software-engineering workflows (Guo et al., 2024; Liu et al., 2024b; Mialon et al., 2024; Yoran et al., 2024; Deng et al., 2023; Zhou et al., 2024; Xie et al., 2024; Jimenez et al., 2024). These works assume globally shared tool environments and study centralized training rather than federated learning under client-specific tool sets.

Multi-agent systems and role specialization. LLM-based multi-agent frameworks study how roles or software agents coordinate on complex tasks (Li et al., 2023a; Wu et al., 2023; Hong et al., 2024; Park et al., 2023), but assume centralized orchestration or shared tool access and so do not address parameter sharing when each client owns a private tool set. Our boundary is *parameter-level*: both partitions stay active in every adapted layer, and partial aggregation specifies which updates may leave a client.

Federated learning for LLMs. Parameter-efficient fine-tuning methods such as LoRA (Hu et al., 2022) and its variants (Liu et al., 2024a) have become standard for adapting LLMs under resource constraints, and federated LoRA, where only low-rank adapters are communicated, has been explored for text generation and domain adaptation (Zhang et al., 2023; Zhou et al., 2025; He et al., 2026). Personalized FL methods (Fallah et al., 2020; Collins et al., 2021) address client heterogeneity through local adaptation but do not handle heterogeneous action spaces, and FedAvg (McMahan et al., 2017) with its variants (FedProx (Li et al., 2020), SCAFFOLD (Karimireddy et al., 2020)) was designed for statistical heterogeneity, not for fundamentally different output spaces. Our setting is complementary: clients differ not only in data distribution but in the tool-action spaces available to their agents.

3 Problem Formulation

We consider N clients, each with a local dataset $\mathcal{D}_i = \{(x, y)\}$ of ReAct-format agent trajectories $y = (y_1, \ldots, y_T)$. Each token y_t carries a semantic role $r_t \in \{R, E\}$: R for reasoning tokens (inside `Thought:` blocks), E for execution tokens (inside `Code:` or `Action:` blocks).

The key heterogeneity is that execution distributions differ across clients,

$$P_i(y_t \mid r_t = E) \neq P_j(y_t \mid r_t = E), \quad i \neq j, \tag{1}$$

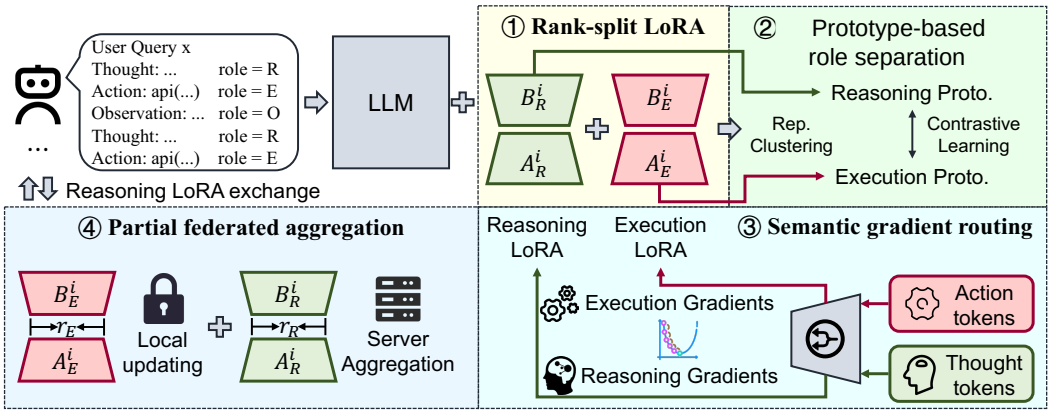

Figure 2: Overview of DREL. Client trajectories are decomposed into reasoning and execution roles. Semantic gradient routing sends role-specific updates into rank-split LoRA subspaces. Only reasoning adapters are aggregated by the server, while execution adapters remain local to each client's private tool environment.

because each client i owns a distinct tool set $\mathcal{T}_i$ with different API signatures and argument structures. Standard FL instead optimizes a single shared model,

$$\min_{\theta} \sum_{i=1}^{N} \frac{n_i}{n} \mathbb{E}_{(x,y)\sim\mathcal{D}_i}[-\log P_\theta(y \mid x)]. \tag{2}$$

When execution distributions differ, the shared representation $h_t = f_\theta(x, y_{<t})$ must encode both task-level reasoning semantics and tool-specific execution grounding, which we refer to as reasoning–execution interference.

We do not assume the two roles are perfectly independent: DREL simply provides separate adaptation capacity and restricts communication to the reasoning-token subspace, making the boundary explicit and testable.

4 Method: DREL

DREL separates reasoning and execution adaptation capacity and selectively shares it during federated aggregation. The deployed model contains both subspaces in the same LoRA update; it does not perform sequential inference through separate reasoning and execution models. Figure 2 summarises the lifecycle: role-labelled trajectories supervise a rank-split local adapter, semantic gradient routing assigns token-level updates to the two partitions, and the server aggregates only the reasoning partition.

4.1 Rank-Split LoRA with Orthogonality Regularization

We adopt LoRA (Hu et al., 2022), which learns low-rank matrices $A \in \mathbb{R}^{r\times d}$ and $B \in \mathbb{R}^{d\times r}$ so that $W' = W + BA$. In DREL we split the rank r into a reasoning partition r_R and an execution partition r_E, with $r_R + r_E = r$, partitioning the LoRA matrices column- and row-wise:

$$A_R \in \mathbb{R}^{r_R\times d}, \quad B_R \in \mathbb{R}^{d\times r_R}, \qquad A_E \in \mathbb{R}^{r_E\times d}, \quad B_E \in \mathbb{R}^{d\times r_E}. \tag{3}$$

For a hidden state h_t the role-specific representations are $z_t^R = B_R A_R h_t$ and $z_t^E = B_E A_E h_t$, with the full representation $z_t = z_t^R + z_t^E$. To encourage subspace separation we apply an orthogonality regularizer

$$\mathcal{L}_{\text{orth}} = \|(B_R A_R)^\top (B_E A_E)\|_F^2, \tag{4}$$

so that $\mathcal{L}_{\text{orth}} \to 0$ encourages approximately orthogonal reasoning and execution subspaces.

4.2 Semantic Gradient Routing

To reinforce role specialization during training we introduce semantic gradient routing (SGR). For each token t a lightweight classifier produces role probabilities

$$p_t^R = \sigma(w_{\text{role}}^\top h_t), \quad p_t^E = 1 - p_t^R, \tag{5}$$

where σ is the sigmoid and $w_{\text{role}} \in \mathbb{R}^d$ is learnable. During backpropagation, gradients to role-specific parameters are gated by these probabilities,

$$\frac{\partial \mathcal{L}}{\partial \theta_R} \leftarrow p_t^R \cdot \frac{\partial \mathcal{L}}{\partial \theta_R}, \qquad \frac{\partial \mathcal{L}}{\partial \theta_E} \leftarrow p_t^E \cdot \frac{\partial \mathcal{L}}{\partial \theta_E}, \tag{6}$$

biasing reasoning-token gradients toward $\theta_R = \{A_R, B_R\}$ and execution-token gradients toward $\theta_E = \{A_E, B_E\}$, while tokens at role boundaries receive blended gradients proportional to role ambiguity. A role classification loss $\mathcal{L}_{\text{role}}$ supervises the role predictor with ground-truth role labels. In the evaluated implementation the first training phase gates backward gradients only, while the second phase uses the legacy forward-gating path; at deployment the two partitions are summed in one backbone forward pass. Whether SGR contributes anything is left open here: the only available comparison against a no-routing variant also changes the auxiliary loss coefficients, and the matched-coefficient ablation did not complete (Section 5.3).

4.3 Partial Federated Aggregation

Client i maintains $\theta_i = (\theta_{\text{backbone}}, \theta_R^i, \theta_E^i)$, where θ_{backbone} is the frozen base model, $\theta_R^i = \{A_R^i, B_R^i\}$ are the reasoning LoRA parameters and $\theta_E^i = \{A_E^i, B_E^i\}$ the execution parameters. In each communication round, clients upload only their reasoning parameters and the server performs sample-weighted averaging,

$$\theta_R^{\text{global}} = \sum_{i=1}^{N} \frac{n_i}{n} \theta_R^i, \tag{7}$$

broadcasting θ_R^{global} back to all clients. Execution parameters θ_E^i are never shared and remain locally updated. This has two intended consequences: tool-specific execution updates do not enter the shared adapter, and only the fraction r_R/r of the rank budget is transmitted. Whether the first consequence improves task accuracy is an empirical question; the communication reduction follows directly from the transmitted tensor set.

4.4 Personalized State Across Federated Rounds

Selective aggregation requires an explicit state transition, since each client carries both a global and a private parameter block. Let $\bar{\theta}_R^{k-1}$ denote the reasoning adapter after round $k-1$ and $\theta_E^{i,k-1}$ client i's retained execution adapter. At the start of round k,

$$\theta_R^{i,k,0} \leftarrow \bar{\theta}_R^{k-1}, \qquad \theta_E^{i,k,0} \leftarrow \theta_E^{i,k-1}. \tag{8}$$

Local optimization updates both blocks using $\mathcal{D}_i$; the server then receives only $\theta_R^{i,k}$ and computes

$$\bar{\theta}_R^k = \sum_{i=1}^{N} \frac{n_i}{\sum_j n_j} \theta_R^{i,k}, \qquad \theta_E^{i,k} \text{ remains on client } i. \tag{9}$$

The final personalized state for client i is therefore the pair $(\bar{\theta}_R^K, \theta_E^{i,K})$. This differs from full-LoRA FedAvg, which replaces both types of adaptation with a single global average, and from local training, which shares neither block. Algorithm 1 (Appendix K) gives the full procedure.

4.5 Communication and Single-Pass Deployment

For an adapted matrix $W_\ell \in \mathbb{R}^{d_{\text{out}}^\ell \times d_{\text{in}}^\ell}$ the two partitions hold $P_R^\ell = r_R(d_{\text{in}}^\ell + d_{\text{out}}^\ell)$ and $P_E^\ell = r_E(d_{\text{in}}^\ell + d_{\text{out}}^\ell)$ parameters, so because DREL and FedAvg adapt the same modules the per-client upload ratio is $\sum_\ell P_R^\ell / \sum_\ell (P_R^\ell + P_E^\ell) = r_R/(r_R + r_E)$, which depends only on the chosen split.

At inference the client does not invoke separate reasoning and execution models; each adapted layer applies

$$W_i^{\text{deploy}} = W_0 + B_R^{\text{global}} A_R^{\text{global}} + B_E^i A_E^i, \tag{10}$$

so both low-rank updates are evaluated within the same layer and the same backbone generation call. The total LoRA rank is matched to the baselines; the decomposition changes parameter ownership and communication, not the number of backbone invocations.

4.6 What Selective Sharing Does to the Execution Block

The following is a parameter-displacement identity, not a prediction about accuracy. Consider one aggregation event and let client i's execution block be $\theta_E^i = \mu_E^i + \varepsilon_i$, with μ_E^i client-specific because the tool sets differ, and ε_i zero-mean noise of covariance Σ_E. Uniform aggregation would hand client i the mean $\bar{\theta}_E$ across clients, replacing its own block, whereas DREL leaves θ_E^i untouched. The expected squared displacement from the client's own block is then $\mathbb{E}\|\bar{\theta}_E - \theta_E^i\|^2 = \|\bar{\mu}_E - \mu_E^i\|^2 + (1 - \frac{1}{N})\,\mathrm{tr}\,\Sigma_E$ under uniform averaging and 0 under DREL (Appendix A). This is an identity about parameter displacement under uniform averaging; the reported runs use sample-weighted aggregation, and zero displacement is true by construction rather than evidence of lower task error. Whether it improves accuracy under a fixed upload budget is the empirical question Section 5 answers.

4.7 Overall Training Objective

Each client optimizes

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda_{\text{orth}}\mathcal{L}_{\text{orth}} + \lambda_{\text{role}}\mathcal{L}_{\text{role}}, \tag{11}$$

where $\mathcal{L}_{\text{LM}} = -\sum_{t=1}^{T} \log P_\theta(y_t \mid x, y_{<t})$ is the standard language-modelling loss. The concrete coefficients used in the evaluated configuration are given in Section 5.1.

5 Experiments

5.1 Setup

Training and federation. We use Qwen3-VL-8B-Instruct as a frozen backbone and four simulated clients holding 2,948, 1,058, 385 and 669 synthetic ReAct-style trajectories, each containing `Thought` and `Code`/`Action` spans for role supervision. Every client owns three tools and no tool is shared by all four clients (Table 5, Appendix H). We compare: *Local-96*, a client-specific rank-96 LoRA trained on local data with no communication; *FedAvg-96*, full rank-96 LoRA aggregation; *FedAvg-32*, a plain rank-32 LoRA with full aggregation, which transmits *exactly the same number of bytes* as DREL; *FullShare*, a matched rank-split control that aggregates both DREL partitions and is *composed post hoc* from the same DREL checkpoints rather than trained separately; and DREL, whose rank-96 update is split into a rank-32 reasoning and a rank-64 execution partition, of which only the reasoning partition is uploaded. All methods use the same target modules, data and two training seeds (42, 43).

Optimization. FedAvg-style methods follow a two-event protocol: three local epochs at 10^{-5}, one aggregation, two further epochs at 5×10^{-6}, and a second aggregation. DREL uses dropout 0.05, $\lambda_{\text{orth}} = 0.001$ and $\lambda_{\text{role}} = 0.03$. All methods adapt the same seven target modules at a total rank of 96, except the rank-32 controls. Communication is reported per client per aggregation event.

Evaluation. Every method×client×seed cell is evaluated on GTA, which reports answer accuracy, tool-selection accuracy and executable-code rate for tool-available (TA), tool-unavailable (TU) and all text tasks. A task is TA for client i only when every required tool belongs to $\mathcal{T}_i$. We macro-average client–seed cells with $N > 0$, so Client C is excluded from TA aggregation. Client A is additionally evaluated on the GAIA validation split (165 tasks) with a fixed full-coverage denominator, counting empty or null answers as incorrect.

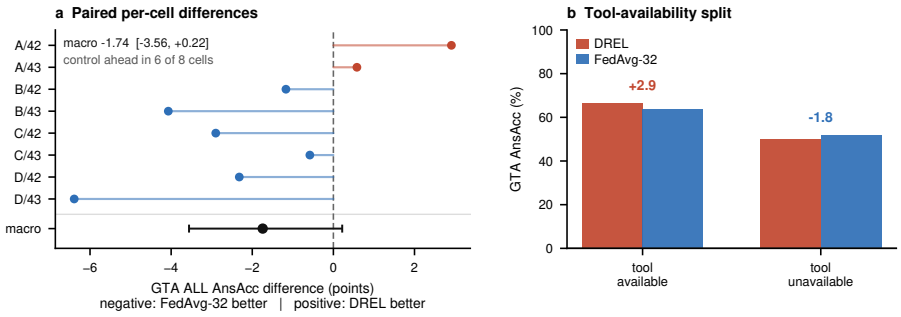

Figure 3: The byte-matched comparison, paired by construction. (**a**) Each cell's difference in GTA ALL answer accuracy, method minus the byte-matched rank-32 FedAvg control, for the eight client–seed cells and their macro summary with a paired cell bootstrap interval; the control leads in six of the eight cells. (**b**) The same contrast split by tool availability, which is where the aggregate hides two opposite effects: the method leads on tasks whose tools it owns and trails on tasks whose required tools are missing. Both panels use the pre-specified configuration; the routing component is not ablated here and remains unresolved. Panel **a** shows paired differences rather than overlapping marginal distributions because the comparison is within-cell.

Comparison logic. Local-96 and FedAvg-96 bracket the effect of sharing at full rank. The DREL–FullShare pair isolates the communication boundary at identical parameterization. The DREL–FedAvg-32 pair holds per-event upload fixed and is the pre-specified comparison, but it is a controlled *upload-budget* comparison rather than an isolation of any single factor: deployed capacity, partitioning and aggregation behaviour all change together. The match is exact rather than approximate: DREL's uploaded reasoning partition and FedAvg-32's entire adapter both contain exactly $87{,}293{,}952$ float32 parameters (333 MiB), while a full rank-96 upload contains $261{,}881{,}856$ (999 MiB). The two methods therefore differ only in how a fixed upload budget is spent: FedAvg-32 concentrates all 87.3M of a client's capacity in the shared adapter, whereas DREL ships the same 87.3M and couples it with a private execution partition, giving that client 261.9M of deployed LoRA capacity—$3.00\times$ as much—for the same transmitted bytes.

Primary endpoint and statistics. We fix one primary endpoint in advance: *macro-averaged GTA ALL AnsAcc over all eight client–seed cells, for* DREL *minus FedAvg-32*, the byte-matched control. TA and TU are secondary and are never used as fallback endpoints, nor are per-client slices. Where a cell is not yet evaluated we say so and report the comparison as incomplete rather than substituting a partial macro-average, since the finished cells are not a random subset of clients. We report two paired analyses and say which yields which number. (i) The primary interval is a percentile bootstrap over the eight paired client–seed cell differences ($10{,}000$ resamples); eight correlated clusters make it fragile. (ii) The p-value is an exact stratified sign test on the per-task records: within a cell the discordant pairs are exchangeable under the sharp null, so their count is $\mathrm{Binomial}(n_k, \frac{1}{2})$ and the stratified total $\mathrm{Binomial}(\sum_k n_k, \frac{1}{2})$, giving a two-sided p with no simulation. It is exact under exchangeability and within-cell independence, but the same 172 tasks recur across client views and seeds, so it is not cluster-aware. Neither is confirmatory. A non-significant difference is reported as *not resolved at this sample size*; we never call it equivalence, which would need a pre-specified margin.

5.2 Matched-Upload Comparison on GTA

Table 1 presents the primary comparison, macro-averaged over valid client–seed cells, and Figure 3 gives the cell-level paired evidence behind it; Appendix Table 3 lists the exact values and Appendix Figure 4 places every method on the communication–accuracy plane.

On the pre-specified endpoint — macro-averaged GTA ALL AnsAcc over all eight client–seed cells — the rank-split method scores 58.36% against 60.10% for the byte-matched FedAvg-32 control, an observed difference of -1.74 **points** (paired cell bootstrap 95% CI $[-3.56, +0.22]$; 6 of 8 cells favour the control).

Table 1: GTA AnsAcc (%) and communication, macro-averaged over valid client–seed cells. TA excludes Client C's two $N = 0$ cells; TU and ALL use all eight. "Shared" is the rank transmitted; upload is the float32 payload per client per aggregation event, relative to rank-96. † matched-upload control: FedAvg-32 and DREL transmit exactly the same tensor count (87,293,952 parameters), but FedAvg-32 deploys 87.3M of LoRA capacity while DREL deploys 261.9M (3.00×). FULLSHARE is composed post hoc.

Method	Total rank	Shared	TA	TU	ALL	Upload
Local-96	96	0	64.1	51.8	58.9	0%
FedAvg-96	96	96	**66.7**	51.6	59.2	100%
FULLSHARE	96	96	64.5	50.8	58.8	100%
FedAvg-32†	32	32	63.6	**51.9**	**60.1**	**33.3%**
DREL + SGR	32+64	32	66.5	50.0	58.4	**33.3%**

Of the 1376 paired task observations, 154 are discordant; 65 of those favour the rank-split method and 89 favour FedAvg-32 (exact stratified sign test $p = 0.063$).

The interval includes zero, so the difference is not resolved at this sample size; what the data do show is that the rank split conveys *no* accuracy advantage over the byte-matched control, and the point estimate and cell record both lean the other way. We report that plainly rather than describing the two as equivalent, which would require a pre-specified margin and an interval contained inside it.

DREL nevertheless deploys 261.9M of LoRA parameters against FedAvg-32's 87.3M, so three times the deployed capacity at an identical 333 MiB upload buys no measured accuracy here. For orientation, no-communication Local-96 reaches 58.94% and full-upload FedAvg-96 reaches 59.16%.

Tool availability, descriptive. The designs differ in opposite directions by task type: 66.47 versus 63.61% on tool-available tasks and 50.03 versus 51.88% on tool-unavailable tasks. Both differences are modest, TA and TU are different task subsets rather than randomised conditions, and we read no mechanism into them; both are reported so neither can be the headline.

Exploratory. FedAvg-32 averages 60.10% against FedAvg-96's 59.16% at one third of the upload, but only 4 of 8 cells favour the smaller adapter, the interval is $[-0.29, +2.18]$ and the exact sign test gives $p = 0.319$. The contrast was chosen after the primary comparison was seen, the adapters are not capacity-matched and neither rank was tuned, so we do not claim that lower rank improves accuracy.

Routing is unresolved rather than resolved against: the only completed comparison also changed both auxiliary loss coefficients, and the matched-coefficient ablation did not complete (Appendix E).

Repaired training condition. The corpus above supervises every client partly on tools it does not own. We repeated the comparison on a repaired corpus that drops those trajectories and changes nothing else, under rules frozen before any repaired model was scored. On the identical four seed-42 clients the repair moves the estimate $+1.89$ points in the method's favour, from -0.87 to $+1.02$, but neither four-cell interval excludes zero and neither per-task sign test approaches significance, so it shifts the point estimate without resolving it and the pre-specified eight-cell result is unchanged. The movement is a redistribution rather than a uniform gain: dropping the out-of-toolbox trajectories helps clients B, C and D and slightly hurts client A, so the corpus change is not a single-direction improvement. Per-cell values are in Appendix C.

Where the extra capacity goes. DREL deploys three times its control's adapter parameters, so the negative aggregate raises a narrower question: does that capacity do any work? Zeroing the private execution partition at inference—leaving the uploaded partition, the router and every other parameter byte-identical, so the active parameter count returns to the control's—raised accuracy in 3 of the 3 cells evaluated (A/42 $+0.58$, C/42 $+5.23$, D/42 $+1.16$), taking the macro average from 58.53% to 60.85% against 59.30% for the byte-matched control. The exact two-sided sign test gives $p = 0.25$; we do not rest the claim on a bootstrap interval here, because resampling

a set of consistently positive differences cannot produce a non-positive mean, so such an interval excluding zero would be an artefact of the observed signs rather than independent evidence. What the intervention supports is therefore narrow: for these checkpoints the trained private partition did not benefit the deployed model. It was chosen after the primary comparison was seen, and it does not overturn the pre-specified result above.

Provenance. Every cell is a real evaluated run, mapped to its archived evaluator output in the evidence manifest shipped with the code supplement.

5.3 Ablation: Does Semantic Gradient Routing Help?

We report routing as an ablation, not a contribution. The comparison that first motivated dropping it is *confounded*: it also raised both auxiliary coefficients (from $0.001/0.03$ to $0.005/0.1$, i.e. $5\times$ and $3\frac{1}{3}\times$), so we draw no causal conclusion from it. An unconfounded ablation (identical $\lambda_{\text{orth}} = 0.001$, $\lambda_{\text{role}} = 0.03$, routing flag off) did not complete before the deadline, so routing remains unresolved rather than resolved against.

5.4 The Boundary on GAIA

We use GAIA as a deliberately out-of-distribution boundary check, not as evidence for the main claim. On 165 Client-A validation tasks with a fixed full-coverage denominator (Table 8), Local reaches $16.4 \pm 2.6\%$, FedAvg reaches $13.9 \pm 3.4\%$, and DREL reaches $12.4 \pm 0.4\%$. DREL therefore does not improve this broader agent benchmark, consistent with a client-specific tool-adaptation mechanism rather than a general-purpose reasoning improvement. The evaluation covers one client and is reported as such.

6 Discussion

DREL aligns the communication boundary with the semantic boundary: reasoning updates are shared, execution updates stay local. That cuts per-client upload by 67% relative to full LoRA aggregation while a client still deploys the same total LoRA capacity, in a single inference pass. We therefore state the contribution as a characterisation rather than a win.

Limitations. The comparison against FedAvg-32 changes deployed capacity, partitioning and aggregation together, so even a resolved difference would identify the *configuration* rather than the causal contribution of private execution. Matching uploaded bytes does not match memory, optimizer state or training compute. The post-hoc FullShare control does not re-train under shared execution. Neither method was re-tuned against the other, so we claim no advantage for this configuration rather than for every rank split. The evaluation uses four simulated clients, two seeds (one for the corpus repair and the partition intervention), one backbone and one tool partition, and gives no evidence of broader reasoning improvement.

What this implies for reporting federated adaptation. Three reporting practices follow, and each is cheap. First, state per-event upload in parameters or bytes rather than as a rank, because a rank change moves the upload and the deployed capacity together; these two designs transmit identical bytes yet differ threefold in deployed parameters. Second, report deployed parameters separately from transmitted ones: the question is what a fixed budget buys, not how much of a client's adapter leaves the device. Third, ablate at inference as well as at training — the private partition was trained under the same objective as the shared one, and only removing it at evaluation revealed that it lowered accuracy. None of this needs new data or training.

Summary. Matched exactly on upload, a rank-split federated adapter deploying three times its control's parameters showed no accuracy advantage, and zeroing its private partition at inference improved it on every client measured, locating the loss in that partition rather than in the rank split. We report the pre-specified null, what it does and does not identify, and the protocol lesson that federated rank comparisons are confounded unless upload is matched exactly.

AI USE STATEMENT

In this work, we used generative AI tools to assist with code development (implementation and refactoring of the federated training, aggregation and evaluation pipeline), with exploratory analysis of experiment logs, and with language editing of the manuscript. We have not used generative AI tools to generate research ideas, to design the core method, or to produce experimental results: every number reported in this paper was produced by executing our own training and evaluation code on our own hardware. All AI-assisted code was reviewed, executed and validated by the authors, and every reported measurement was re-derived from archived evaluator outputs by an evidence-manifest script that records the source file and field for each value. Generative AI tools were not used to create, augment or alter the datasets. They were used to help draft and edit the manuscript, including these statements, whose content the authors reviewed and verified. Large language models are also the *object* of study in this work (we adapt a tool-using language agent), which is independent of the disclosure above. We take responsibility for the final content of this work, including all text, claims and artifacts.

ETHICS STATEMENT

This work studies federated adaptation of tool-using language agents. The "clients" in our evaluation are simulated partitions of a synthetic trajectory dataset, not real users, and no human subjects were involved in this research. Client tool sets are constructed from an existing public benchmark tool registry; no private user data, personal information, or sensitive records were collected, processed or released. Training trajectories are synthetic ReAct-style samples, and GTA and GAIA are used strictly as fixed evaluation sets—never for training, data filtering, checkpoint selection or hyperparameter tuning. We audited this rather than assuming it: all 206 extractable GTA evaluation questions were matched against the 22,584 human turns of the training corpora (original, repaired and intermediate) and no evaluation question occurs in any of them. We do not claim the same for the upstream synthetic trajectories whose provenance could not be reconstructed (Appendix I). The motivating application, personal assistants that hold private tool inventories, has an inherent privacy rationale: DREL uploads only the adapter tensors of its designated shared partition and keeps the rest local. We describe this as a parameter-sharing boundary rather than a privacy guarantee: we do not establish that the uploaded tensors are free of tool-specific information, and the method provides no formal privacy mechanism. We note one dual-use consideration: methods that improve tool-calling agents also improve agents that can invoke external APIs, including potentially harmful ones. This study does not release a new agent capability, new tool access, or a new deployed system; it characterises the communication boundary of an adapter-level adaptation method on a fixed public benchmark. The synthetic training artifacts were not generated by us, and we document in Appendix I exactly which aspects of their provenance could and could not be reconstructed, rather than asserting properties we cannot verify. We have read the ICLR Code of Ethics and conducted this work in accordance with it.

REPRODUCIBILITY STATEMENT

We describe the full training and evaluation protocol in Appendix K. All training configurations (LlamaFactory YAML files), the federated aggregation and adapter-composition scripts, per-client data manifests and the evaluation scorers are included in the supplementary material. Every number in the main tables and figures is mapped to an archived evaluator output by an evidence manifest that also records the checkpoint path and a digest of each adapter file. Appendix K lists the exact aggregation weights, LoRA target modules, learning-rate schedules and evaluation decoding settings, and Appendix L states the communication accounting convention (float32 tensor bytes of client-to-server uploads, excluding broadcast). Appendix I documents the provenance limitations of the training artifacts, including which upstream data-generation details could not be reconstructed. Anonymised source code for the federated pipeline and the evaluation harness is provided as supplementary material.

REFERENCES

Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pp. 308–318. ACM, 2016. doi: 10.1145/2976749.2978318. URL https://doi.org/10.1145/2976749.2978318.

Yun-Shiuan Chuang, Chaitanya Kulkarni, Alec Chiu, Avinash Thangali, Zijie Pan, Shivani Shekhar, Yirou Ge, Yixi Li, Uma Kona, Linsey Pang, and Prakhar Mehrotra. Toward scalable verifiable reward: Proxy state-based evaluation for multi-turn tool-calling LLM agents, 2026. URL https://doi.org/10.48550/arXiv.2602.16246.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *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*, pp. 2089–2099. PMLR, 2021. URL http://proceedings.mlr.press/v139/collins21a.html.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samual Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *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/5950bf290a1570ea401bf98882128160-Abstract-Datasets_and_Benchmarks.html.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vázquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks? In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *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.

Zhi Gao, Yuntao Du, Xintong Zhang, Xiaojian Ma, Wenjuan Han, Song-Chun Zhu, and Qing Li. CLOVA: A closed-loop visual assistant with tool usage and update. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 13258–13268. IEEE, 2024. doi: 10.1109/CVPR52733.2024.01259. URL https://doi.org/10.1109/CVPR52733.2024.01259.

In Gim, Seung-Seob Lee, and Lin Zhong. Asynchronous LLM function calling, 2024. URL https://doi.org/10.48550/arXiv.2412.07017.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 11143–11156. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.664. URL https://doi.org/10.18653/v1/2024.findings-acl.664.

Skyler Hallinan, Thejas Venkatesh, Xiang Ren, Sai Praneeth Karimireddy, Ashwin Paranjape, Yuhao Zhang, and Jack Hessel. Opaquetoolsbench: Learning nuances of tool behavior through interaction. In *Proceedings of the ACM Conference on AI and Agentic Systems, CAIS 2026, San Jose, CA, USA, May 26-29, 2026*, pp. 620–638. ACM, 2026. doi: 10.1145/3786335.3813182. URL https://doi.org/10.1145/3786335.3813182.

Shuaida He, Liwen Chen, and Long Feng. Federated lora fine-tuning for llms via collaborative alignment, 2026. URL https://doi.org/10.48550/arXiv.2605.21217.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for A multi-agent collaborative framework. 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=VtmBAGCN7o.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. 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=nZeVKeeFYf9.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik 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.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1-2):1–210, 2021. doi: 10.1561/2200000083. URL https://doi.org/10.1561/2200000083.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. SCAFFOLD: stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5132–5143. PMLR, 2020. URL http://proceedings.mlr.press/v119/karimireddy20a.html.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: communicative agents for "mind" exploration of large language model society. In *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*, 2023a. URL http://papers.nips.cc/paper_files/paper/2023/hash/a3621ee907def47c1b952ade25c67698-Abstract-Conference.html.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 3102–3116. Association for Computational Linguistics, 2023b. doi: 10.18653/V1/2023.EMNLP-MAIN.187. URL https://doi.org/10.18653/v1/2023.emnlp-main.187.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of the Third Conference on Machine Learning and Systems, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020. URL https://proceedings.mlsys.org/paper_files/paper/2020/hash/1f5fe83998a09396ebe6477d9475ba0c-Abstract.html.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32100–32121. PMLR / OpenReview.net, 2024a. URL https://proceedings.mlr.press/v235/liu24bn.html.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. Toolace: Winning the points of LLM function calling. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=8EB8k6DdCU.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie 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.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *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*, pp. 1273–1282. PMLR, 2017. URL http://proceedings.mlr.press/v54/mcmahan17a.html.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas 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.

Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA, 29 October 2023- 1 November 2023*, pp. 2:1–2:22. ACM, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/e4c61f578ff07830f5c37378dd3ecb0d-Abstract-Conference.html.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. 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=dHng2O0Jjr.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *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/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversation framework, 2023. URL https://doi.org/10.48550/arXiv.2308.08155.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

Abel Yagubyan. How consistent are LLM agents? measuring behavioral reproducibility in multi-step tool-calling pipelines, 2026. URL https://doi.org/10.48550/arXiv.2605.28840.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Junjie Ye, Zhengyin Du, Xuesong Yao, Weijian Lin, Yufei Xu, Zehui Chen, Zaiyuan Wang, Sining Zhu, Zhiheng Xi, Siyu Yuan, Tao Gui, Qi Zhang, Xuanjing Huang, and Jiecao Chen. Toolhop: A query-driven benchmark for evaluating large language models in multi-hop tool use. In *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*, pp. 2995–3021. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.150. URL https://doi.org/10.18653/v1/2025.acl-long.150.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. Assistantbench: Can web agents solve realistic and time-consuming tasks? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 8938–8968. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.505. URL https://doi.org/10.18653/v1/2024.emnlp-main.505.

Wei Zhang, Yi Zhang, Li Zhu, Qianghuai Jia, Feijun Jiang, Hongcheng Guo, Zhoujun Li, and Mengping Zhou. ADC: enhancing function calling via adversarial datasets and code line-level feedback. In *2025 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2025, Hyderabad, India, April 6-11, 2025*, pp. 1–5. IEEE, 2025. doi: 10.1109/ICASSP49660.2025.10888405. URL https://doi.org/10.1109/ICASSP49660.2025.10888405.

Zeyu Zhang, Guohao Li, Zhenchang Xing, Alexandros Apostolopoulos, Yu Lin Lee, and Liang Zheng. Gecko: A simulation environment with stateful feedback for refining agent tool calls, 2026. URL https://doi.org/10.48550/arXiv.2602.19218.

Zhuo Zhang, Yuanhang Yang, Yong Dai, Qifan Wang, Yue Yu, Lizhen Qu, and Zenglin Xu. Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, volume ACL 2023 of *Findings of ACL*, pp. 9963–9977. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-ACL.632. URL https://doi.org/10.18653/v1/2023.findings-acl.632.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Yajie Zhou, Xiaoyi Pang, and Zhibo Wang. Aflora: Adaptive federated fine-tuning of large language models with resource-aware low-rank adaption, 2025. URL https://doi.org/10.48550/arXiv.2505.24773.

A Derivation of the Aggregation-Error Decomposition

This appendix gives the derivation behind Section 4.6. The result is a one-step bias–variance decomposition, not a convergence guarantee; we state the assumptions explicitly so that the prediction it makes can be falsified by the matched-upload experiment.

Setting. Consider a single aggregation event. Client i's locally optimized execution block is

$$\theta_E^i = \mu_E^i + \varepsilon_i, \tag{12}$$

with $\mu_E^i \in \mathbb{R}^p$ the client-specific execution mean and ε_i a zero-mean random vector, $\mathbb{E}[\varepsilon_i] = 0$. We assume the $\{\varepsilon_i\}_{i=1}^N$ are independent with a common covariance $\mathrm{Cov}(\varepsilon_i) = \Sigma_E$, and that the reasoning and execution blocks are updated by the same local objective (no assumption is made about their statistical dependence).

FedAvg. Sample-weighted full aggregation that is uniform over clients hands client i

$$\bar{\theta}_E = \frac{1}{N}\sum_{j=1}^N \theta_E^j = \bar{\mu}_E + \bar{\varepsilon}, \qquad \bar{\mu}_E = \frac{1}{N}\sum_{j=1}^N \mu_E^j, \quad \bar{\varepsilon} = \frac{1}{N}\sum_{j=1}^N \varepsilon_j. \tag{13}$$

Hence

$$\bar{\theta}_E - \theta_E^i = \big(\bar{\mu}_E - \mu_E^i\big) + \big(\bar{\varepsilon} - \varepsilon_i\big), \tag{14}$$

and the cross term vanishes in expectation because $\mathbb{E}[\bar{\varepsilon} - \varepsilon_i] = 0$ and the two factors are independent of the noise. Therefore

$$\mathbb{E}\big\|\bar{\theta}_E - \theta_E^i\big\|^2 = \big\|\bar{\mu}_E - \mu_E^i\big\|^2 + \mathbb{E}\big\|\bar{\varepsilon} - \varepsilon_i\big\|^2. \tag{15}$$

For the second term, using independence and $\mathrm{Cov}(\varepsilon_i) = \Sigma_E$,

$$\mathbb{E}\|\bar{\varepsilon}\|^2 = \frac{1}{N^2}\sum_{j=1}^N \mathbb{E}\|\varepsilon_j\|^2 = \frac{1}{N}\operatorname{tr}\Sigma_E, \tag{16}$$

$$\mathbb{E}\langle\bar{\varepsilon}, \varepsilon_i\rangle = \frac{1}{N}\sum_{j=1}^N \mathbb{E}\langle\varepsilon_j, \varepsilon_i\rangle = \frac{1}{N}\operatorname{tr}\Sigma_E, \tag{17}$$

$$\mathbb{E}\|\varepsilon_i\|^2 = \operatorname{tr}\Sigma_E. \tag{18}$$

Substituting into $\mathbb{E}\|\bar{\varepsilon} - \varepsilon_i\|^2 = \mathbb{E}\|\bar{\varepsilon}\|^2 - 2\mathbb{E}\langle\bar{\varepsilon}, \varepsilon_i\rangle + \mathbb{E}\|\varepsilon_i\|^2$ gives

$$\mathbb{E}\big\|\bar{\varepsilon} - \varepsilon_i\big\|^2 = \Big(1 - \frac{1}{N}\Big)\operatorname{tr}\Sigma_E, \tag{19}$$

which together with equation 15 yields the stated decomposition

$$\mathbb{E}\big\|\bar{\theta}_E - \theta_E^i\big\|^2 = \big\|\bar{\mu}_E - \mu_E^i\big\|^2 + \Big(1 - \frac{1}{N}\Big)\operatorname{tr}\Sigma_E. \tag{20}$$

DREL. DREL never transmits θ_E, so client i's deployed execution block is exactly θ_E^i and the deviation is identically zero. The reasoning block θ_R is uploaded and averaged by both methods, so to first order the two methods differ only in the execution block, and DREL is weakly better for every client. The transmitted payload is identical in both cases, because only θ_R is uploaded either way.

Prediction and scope. The client-specific bias $\|\bar{\mu}_E - \mu_E^i\|^2$ is zero if and only if all clients share the same execution mean, and grows with execution heterogeneity; the noise term $(1 - 1/N)\operatorname{tr}\Sigma_E$ is irreducible at a single event but shrinks with more clients. The decomposition therefore predicts an advantage for selective sharing that is one-sided and that grows with tool heterogeneity, and no advantage in the homogeneous limit. It is a one-step statement: it does not model how the local optimizers move μ_E^i across rounds, nor the interaction between the aggregated reasoning block and the local execution block.

Table 2: Effect of the repair on the training corpus. Only R1 is used for the primary repaired comparison; R2 is reported for completeness. R2 removes a large share of client A's records because `visualizer` is over-represented there, and it leaves client B's thinnest tool untouched, which is why it is not the primary condition.

Client	Original	After R1	After R1+R2	Thinnest tool (after R1)
A	2,948	2,848	1,403	image_edit (488)
B	1,058	797	602	inspect_file_as_text (26)
C	385	339	333	ask_search_agent (64)
D	669	494	265	image_generator (81)

B ALGORITHM

Algorithm 1: DREL Federated Training

Input: N clients with local datasets $\mathcal{D}_i$ ($n_i = |\mathcal{D}_i|$), rounds K, local epochs E, rank split (r_R, r_E), frozen backbone θ_0

Initialize global θ_R^{global}, per-client θ_E^i and w_{role}^i;
for $k = 1$ *to* K **do**
  Server broadcasts θ_R^{global} to all clients;
  for *each client* i *in parallel* **do**
    $\theta_R^i \leftarrow \theta_R^{\text{global}}$;
    for $e = 1$ *to* E **do**
      Sample batch $(x, y) \sim \mathcal{D}_i$;
      Forward: $h_t = \theta_0(x, y_{<t})$, then $z_t^R = B_R A_R h_t$, $z_t^E = B_E A_E h_t$;
      $\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda_{\text{orth}}\mathcal{L}_{\text{orth}} + \lambda_{\text{role}}\mathcal{L}_{\text{role}}$;
      Apply the SGR rule configured for phase k to θ_R^i and θ_E^i;
      Update θ_R^i, θ_E^i, w_{role}^i;
    Upload θ_R^i to server;
  $\theta_R^{\text{global}} \leftarrow \sum_{i=1}^{N} \frac{n_i}{\sum_j n_j} \theta_R^i$;

Output: θ_R^{global}, $\{\theta_E^i\}_{i=1}^N$

C TRAINING-CONDITION REPAIR

The GTA training corpus was audited for a defect that is independent of any modelling choice: each client's trajectories call tools the client does not own. The share of recognised tool calls that are out-of-toolbox is 2.8% for client A, 17.8% for B, 12.8% for C and 14.0% for D. Coverage of each client's own three tools is also uneven; the worst case is client B, whose `segment` trajectory count exceeds its `inspect_file_as_text` count by more than an order of magnitude even though the latter is one of its three tools.

We therefore specified, *before scoring any repaired model*, a repaired corpus. The frozen rules are: (R1) drop any trajectory containing a call to a tool outside the owning client's toolbox, whole and unspliced; (R2) as a secondary condition, additionally cap each of a client's own tools at the median of its three per-tool record counts by downsampling only, never duplicating; (R3) no GTA evaluation task enters training; (R4) the byte match and the aggregation protocol are unchanged; (R5) every method trains on the identical repaired corpus; (R6) the original data, checkpoints and results are untouched. The primary repaired condition is R1 alone. The full specification, the audit, and the amendment that demoted R2 are shipped with the code supplement.

Table 2 summarises the effect on the corpus.

The repair is a training-condition ablation, not a claim that the corpus is now adequate: it can only remove records, so thin tools stay thin in absolute terms, and the clearest remaining hole is client

Table 3: Per-cell evidence for the frozen primary endpoint: macro GTA ALL AnsAcc for DREL (routing retained) against the byte-matched FedAvg-32 control, whose entire adapter is uploaded. Both transmit exactly $87{,}293{,}952$ parameters (333 MiB) per client per aggregation event. Values are not rounded before the difference is taken.

Client	Seed	DREL	FedAvg-32	difference
A	42	56.98	54.07	+2.91
A	43	56.98	56.40	+0.58
B	42	59.30	60.47	-1.17
B	43	55.81	59.88	-4.07
C	42	61.05	63.95	-2.90
C	43	62.21	62.79	-0.58
D	42	57.56	59.88	-2.32
D	43	56.98	63.37	-6.39
Macro	(8 cells)	58.36	60.10	-1.74

Table 4: Removing the private execution partition at inference. "Intact" is the method as specified; "E-off" is the same uploaded partition with the private 666M parameters zeroed. Higher is better. Client B is absent because its evaluation did not complete.

Client	Control (333 MiB)	Intact (999 MiB)	E-off (333 MiB)
A	54.07	56.98	57.56
C	63.95	61.05	66.28
D	59.88	57.56	58.72
Macro over A, C, D	59.30	58.53	**60.85**

B's `inspect_file_as_text` arm at 26 records. Generating additional trajectories for thin tools was out of scope before the deadline and is not attempted.

D Per-Cell Values and Auxiliary GTA Metrics

The exact per-cell values behind Figure 3 are listed here; the main text reports the paired evidence graphically and the interval once, in the prose.

E Status of the Routing Ablation

The method's semantic gradient routing component is left unresolved, and we state why rather than implying a result. The only completed comparison against a no-routing variant also raised both auxiliary loss coefficients, from $\lambda_{\text{orth}} = 0.001$ to 0.005 ($5\times$) and $\lambda_{\text{role}} = 0.03$ to 0.1 ($3\frac{1}{3}\times$), so it does not isolate routing. A matched-coefficient ablation, holding both coefficients fixed and disabling only the routing flag, was specified and partly trained but did not complete before the submission deadline; it is therefore not reported as a result, and the confounded comparison is not used to justify dropping the component.

F Inference-Time Private-Partition Intervention

Table 4 reports the full three-way comparison. We zero the private execution partition (`lora_E.lora_B`) at inference while leaving the uploaded shared partition, the role router, the prototypes and every other parameter byte-identical, then re-evaluate the same clients on the same tasks. Nothing is retrained, so the comparison is an inference-time sensitivity test on the deployed model.

Paired contrasts on those three clients: E-off minus intact $+2.32$ points (95% cell bootstrap CI $[+0.58, +5.23]$, no client degrading); E-off minus the byte-matched control $+1.55$

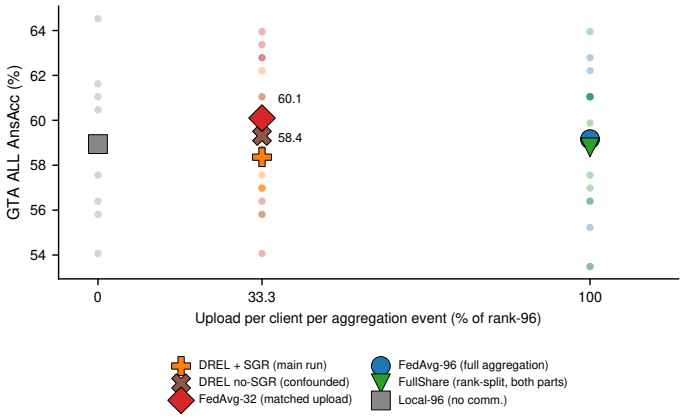

Figure 4: Communication–accuracy frontier on GTA. Large markers are macro means over eight client–seed cells; small markers are individual cells. The x-axis is the float32 adapter payload a client uploads per aggregation event, as a percentage of a full rank-96 payload. FedAvg-32 and DREL sit at exactly the same x position: they transmit identical bytes. FULLSHARE and FedAvg-96 both transmit the full payload.

($[-1.16, +3.49]$); intact minus the control -0.77 ($[-2.90, +2.91]$). Only the first interval excludes zero.

The result is deliberately read narrowly. It shows that the additional deployed parameters are not merely unused but harmful at inference in these client views; it does not show that the partition is semantically empty, that the same holds under other seeds or partitions, or that the intact model is inferior to the control (that interval still contains zero). Three clusters make the interval fragile, and the intervention was chosen after the primary comparison was seen.

G EXPLORATORY RANK CONTRAST

The main text omits this contrast because it was chosen after the primary comparison was seen. On the same endpoint FedAvg-32 averages 60.10% (87.3M deployed adapter parameters, 33.3% upload) against FedAvg-96's 59.16% (261.9M, 100% upload), a difference of $+0.94$ points. Only four of the eight cells favour the smaller adapter, the paired cell interval is $[-0.29, +2.18]$, and the exact stratified sign test gives $p = 0.319$. Because the contrast is post hoc, the two adapters are not capacity-matched and neither rank was tuned separately, we do not claim that a lower rank improves accuracy; we record the observation so that it is not silently discarded.

H AUXILIARY GTA METRICS AND CLIENT-LEVEL RESULTS

Table 12 reports the tool-selection and executable-code metrics, which the main text does not discuss because they do not enter the pre-specified endpoint.

The communication–accuracy frontier referenced from Section 5.2 is reproduced here.

Table 5 lists the audited client tool sets and the resulting client-dependent TA/TU counts; a task counts as tool-available only when every tool it requests is present in that client's toolbox.

Table 6 reports the auxiliary ToolAcc and CodeExec metrics for the full-aggregation comparison, Table 7 breaks ALL AnsAcc down by client, and Figure 5 shows the paired DREL$-$FULLSHARE differences with bootstrap intervals. All three are descriptive: with four clients and two seeds the intervals are wide, and we do not read any of them as a significance test.

Table 5: Audited GTA client tool sets. TA/TU counts are client-dependent because availability requires all tools requested by a task. No tool is shared by all four clients.

Client	Local tool set	TA N	TU N
A	image_edit, objectlocation, visualizer	132	40
B	image_generator, inspect_file_as_text, segmentation	68	104
C	ask_search_agent, facedetection, image_edit	0	172
D	ask_search_agent, image_generator, objectlocation	15	157

Table 6: GTA ToolAcc and CodeExec, macro-averaged over valid client–seed cells using the same $N > 0$ rule as Table 1. A dash marks a method whose cells have not been evaluated.

	ToolAcc			CodeExec		
Method	TA	TU	ALL	TA	TU	ALL
Local	75.1	32.9	45.0	76.0	73.1	72.8
FedAvg	**75.8**	33.5	44.6	76.1	72.0	71.2
FedAvg-32	72.2	32.6	43.7	**77.5**	**73.4**	**73.3**
FULLSHARE	73.3	33.0	44.3	74.2	70.4	70.1
DREL + SGR	74.8	**34.9**	**45.6**	76.4	71.2	71.1

Table 7: Client-level GTA ALL AnsAcc, averaged over seeds 42 and 43. Cells are the raw per-client means; the last column is the mean over clients that have data for every listed method.

Client	Local	FedAvg	FedAvg-32	FULLSHARE	DREL + SGR
A	55.2	54.4	55.2	55.2	**57.0**
B	56.7	58.7	**60.2**	57.0	57.6
C	62.8	61.9	**63.4**	61.0	61.6
D	61.0	61.6	61.6	**61.9**	57.3

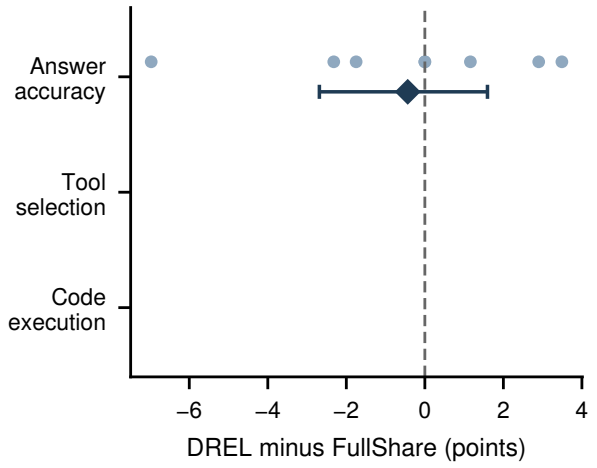

Figure 5: Matched GTA operational outcomes. Points are paired client–seed differences on ALL tasks (DREL−FullShare, $n = 8$); diamonds and bars denote means and nonparametric 95% bootstrap intervals from 10,000 resamples. Positive values favour DREL. FullShare differs only by aggregating the execution partition.

Table 8: GAIA full-coverage accuracy on Client A. Level columns pool seeds 42 and 43; S42/S43 use 165 tasks each. Missing, empty and null answers count as incorrect.

Method	L1 (106)	L2 (172)	L3 (52)	S42	S43	Mean (%) ↑
Local	23/106	30/172	1/52	30/165	24/165	16.4 ± 2.6
FedAvg	19/106	23/172	4/52	19/165	27/165	13.9 ± 3.4
DREL	18/106	22/172	1/52	21/165	20/165	12.4 ± 0.4

Table 9: Observed client data and GTA tool availability. TA requires every tool requested by a GTA task to be present in the client toolbox; TU denotes at least one missing required tool.

Client	Local tool set	Trajectories	TA N	TU N	GTA total
A	objectlocation, visualizer, image_edit	2,948	132	40	172
B	segmentation, inspect_file_as_text, image_generator	1,058	68	104	172
C	facedetection, ask_search_agent, image_edit	385	0	172	172
D	objectlocation, ask_search_agent, image_generator	669	15	157	172
Total	–	5,060	–	–	–

I Scope and Artifact Provenance

This supplement documents the implementation, data artifacts, evaluation protocol, and experimental records for the main paper. All quantities were reconstructed from the archived training configurations, checkpoints, evaluation outputs, and source scripts listed in the reproducibility manifest. We distinguish recorded facts from provenance no longer preserved in the available artifacts. We do not infer undocumented data-generation prompts, model versions, or filtering rules.

Protocol reconciliation note. The checkpoint lineage records two federated aggregation events: a three-epoch local stage, aggregation, a two-epoch local stage, and a second aggregation. Communication values are reported per client per event: DREL uploads 333 MiB per event and 666 MiB over two events, excluding broadcast. The training objective used $\lambda_{\text{orth}} = 0.001$ and $\lambda_{\text{role}} = 0.03$, as recorded in the archived per-client YAML files that the training runs consumed (for example `logs/main_federated/configs/drel_A_r1.yaml` and the corresponding seed-43 directory). These are the authoritative values; an earlier draft of this supplement stated 0.005/0.1, which does not match any archived training configuration and has been corrected.

J Data Artifacts and Client Partition

J.1 Observed Training Data Format

The training artifacts are ShareGPT-format JSON files containing synthetic ReAct-style trajectories with `Thought` and `Code`/`Action` spans. The DREL trainer consumes a token-level role mask (0 = reasoning, 1 = execution, -100 = ignored). The generation model, prompts, sampling settings, tool-execution environment, annotation procedure, and filtering implementation were not retained.

Loaded trajectory counts are reported in Table 9. Filenames for Clients B–D suggest larger historical totals than those loaded; we treat the loaded counts as authoritative. A random 10% validation split is configured; no task-overlap audit between training trajectories and GTA evaluation tasks was found.

Figure 6 makes the designed heterogeneity explicit. Each client has three tools. Pairwise Jaccard overlap is either 0 or 0.2 for distinct clients; no tool is shared by all four clients. These are simulated clients in a controlled benchmark, not real private user datasets.

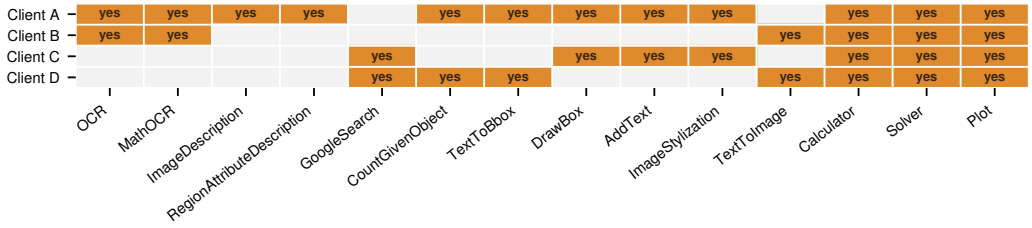

Figure 6: Client-specific tool availability in the simulated federated environment. Orange cells denote tools available to a client.

J.2 Unrecorded Provenance

The following could not be reconstructed: the synthetic-data generator and its prompt, sampling parameters, any human annotation, data license/origin, de-duplication and filtering rules, invalid-tool-call handling, answer verification during construction, and leakage checks. These omissions limit retrospective claims about data collection but do not affect the training files, client counts, evaluation scripts, or scored outputs documented below.

K Model, DREL Implementation, and Federated Protocol

K.1 Base Model and Optimization

All experiments use Qwen3-VL-8B-Instruct (frozen backbone) with the `qwen3_vl_nothink` template, cutoff length 4096, per-device batch size 1, gradient accumulation 8, bfloat16 precision, cosine scheduling, and eight warmup steps. The optimizer is AdamW via LlamaFactory; exact beta, epsilon, and weight-decay values were not retained. Training runs on four NVIDIA A800 80GB GPUs (CUDA 12.4, PyTorch 2.6.0+cu124, FlashAttention 2, LlamaFactory 0.9.5), with separate conda environments for training and evaluation.

K.2 DREL Parameterization and Routing

Each adapted linear layer has a frozen base output and two LoRA updates:

$$h' = h_{\text{base}} + B_R A_R x + B_E A_E x. \tag{21}$$

The reasoning and execution ranks are 32 and 64, respectively, for total rank 96. Their alphas are 32 and 64, and dropout is 0.05. A role router predicts p_R and $p_E = 1 - p_R$, trained with binary cross-entropy over valid role-mask tokens. Semantic gradient routing (SGR) uses a detach construction: the backward gradient to each partition is scaled by its role probability, while the forward adapter sum is unchanged. The first three-epoch phase uses this backward-only SGR; the second two-epoch phase uses legacy forward-gating. At deployment, both updates are summed in one backbone forward pass—the role gate is not a second inference stage.

The orthogonality term is implemented as a local cosine-similarity penalty between reasoning and execution LoRA weights. The training YAML consumed by the run sets $\lambda_{\text{orth}} = 0.001$ and $\lambda_{\text{role}} = 0.03$; both seeds use these values. The information-bottleneck coefficient is zero. Note that the composition initializer writes its own metadata fields into the composed adapter configuration, so the values stored in a composed checkpoint are not a reliable record of the training coefficients; the consuming YAML files are.

K.3 Federated State Transition

For client i, let θ_R^i and θ_E^i be the reasoning and execution adapter partitions. After local optimization, the server aggregates only reasoning tensors, weighted by sample sizes n_i:

$$\bar{\theta}_R = \sum_i \frac{n_i}{\sum_j n_j} \theta_R^i, \qquad (n_A, n_B, n_C, n_D) = (2948, 1058, 385, 669). \tag{22}$$

Table 10: Audited training and federation configurations. FULLSHARE is post-hoc composed from DREL checkpoints and is not independently optimized.

Property	Local	FedAvg	DREL	FULLSHARE
Backbone	Qwen3-VL-8B	same	same	same
Total LoRA rank	96	96	$32_R + 64_E$	$32_R + 64_E$
Local epochs	3	3	$3 + 2$	post-hoc
Aggregation events	0	2	2	post-hoc
SGR implementation	none	none	R1 backward-only; R2 forward-gated	inherited
Loss coefficients	–	–	$\lambda_{\text{orth}} = 0.001$, $\lambda_{\text{role}} = 0.03$	inherited
Aggregated tensors	none	full LoRA	reasoning only	reasoning + execution
Upload per round	0	999 MiB	333 MiB	999 MiB
Cumulative upload	0	1,998 MiB	666 MiB	1,998 MiB if transmitted
Evaluation decoding	greedy	greedy	greedy	greedy

The final personalized adapter combines the global reasoning tensors with each client's local execution and auxiliary tensors. The recorded lineage is: three local epochs, reasoning-only aggregation, two further local epochs, and a second reasoning-only aggregation.

L BASELINES, FAIRNESS, AND COMMUNICATION ACCOUNTING

Table 10 summarises the training and federation configurations for all methods. FULLSHARE is the matched control for selective aggregation: it uses the same DREL checkpoints, rank split, client data, aggregation weights, and composition procedure, but aggregates both reasoning and execution tensors—isolating the communication boundary within the DREL parameterization. Local and FedAvg use a single rank-96 LoRA with three local epochs (vs. DREL's rank split and five epochs), so comparisons against them are operational references; the DREL–FullShare comparison is the controlled contrast for the sharing boundary.

Table 11: Complete GTA answer accuracy by client, seed, and subset. Values are percentages. A dash denotes a client–seed cell with $N = 0$.

Subset	Client	Seed	N	Local	FedAvg	FullShare	DREL
TA	A	S42	132	59.85	61.36	64.39	64.39
TA	A	S43	132	62.12	59.85	59.85	63.64
TA	B	S42	68	72.06	72.06	70.59	73.53
TA	B	S43	68	70.59	73.53	72.06	70.59
TA	C	S42	0	–	–	–	–
TA	C	S43	0	–	–	–	–
TA	D	S42	15	46.67	60.00	60.00	60.00
TA	D	S43	15	73.33	73.33	60.00	66.67
TU	A	S42	40	35.00	35.00	32.50	32.50
TU	A	S43	40	37.50	32.50	32.50	35.00
TU	B	S42	104	48.08	53.85	47.12	50.00
TU	B	S43	104	46.15	45.19	48.08	46.15
TU	C	S42	172	64.53	61.05	61.05	61.05
TU	C	S43	172	61.05	62.79	61.05	62.21
TU	D	S42	157	63.06	61.15	59.87	57.32
TU	D	S43	157	59.24	61.15	64.33	56.05
ALL	A	S42	172	54.07	55.23	56.98	56.98
ALL	A	S43	172	56.40	53.49	53.49	56.98
ALL	B	S42	172	57.56	61.05	56.40	59.30
ALL	B	S43	172	55.81	56.40	57.56	55.81
ALL	C	S42	172	64.53	61.05	61.05	61.05
ALL	C	S43	172	61.05	62.79	61.05	62.21
ALL	D	S42	172	61.63	61.05	59.88	57.56

Table 11 (continued)

Subset	Client	Seed	N	Local	FedAvg	FullShare	DREL
ALL	D	S43	172	60.47	62.21	63.95	56.98

Table 12: Complete GTA tool accuracy by client, seed, and subset. Values are percentages. A dash denotes a client–seed cell with $N = 0$.

Subset	Client	Seed	N	Local	FedAvg	FullShare	DREL
TA	A	S42	132	83.52	79.40	76.40	79.03
TA	A	S43	132	81.65	77.90	77.90	84.27
TA	B	S42	68	60.16	60.16	60.16	60.16
TA	B	S43	68	60.16	60.16	60.16	60.16
TA	C	S42	0	–	–	–	–
TA	C	S43	0	–	–	–	–
TA	D	S42	15	84.62	84.62	88.46	84.62
TA	D	S43	15	80.77	92.31	76.92	80.77
TU	A	S42	40	38.10	42.86	37.14	50.48
TU	A	S43	40	45.71	45.71	45.71	46.67
TU	B	S42	104	12.70	12.70	12.70	12.70
TU	B	S43	104	12.70	12.70	12.70	12.70
TU	C	S42	172	36.83	36.02	36.02	37.10
TU	C	S43	172	36.83	36.83	37.37	37.37
TU	D	S42	157	39.60	39.88	41.04	40.75
TU	D	S43	157	40.46	41.04	41.62	41.33
ALL	A	S42	172	70.70	69.09	65.32	70.97
ALL	A	S43	172	71.51	68.82	68.82	73.66
ALL	B	S42	172	29.03	29.03	29.03	29.03
ALL	B	S43	172	29.03	29.03	29.03	29.03
ALL	C	S42	172	36.83	36.02	36.02	37.10
ALL	C	S43	172	36.83	36.83	37.37	37.37
ALL	D	S42	172	42.74	43.01	44.35	43.82
ALL	D	S43	172	43.28	44.62	44.09	44.09

Table 13: Complete GTA code-execution rate by client, seed, and subset. Values are percentages. A dash denotes a client–seed cell with $N = 0$.

Subset	Client	Seed	N	Local	FedAvg	FullShare	DREL
TA	A	S42	132	90.40	86.86	81.82	86.92
TA	A	S43	132	84.78	81.95	81.89	88.96
TA	B	S42	68	63.33	60.00	61.11	63.40
TA	B	S43	68	64.77	64.67	66.19	55.50
TA	C	S42	0	–	–	–	–
TA	C	S43	0	–	–	–	–
TA	D	S42	15	73.33	84.62	73.17	80.00
TA	D	S43	15	79.55	78.57	80.95	83.33
TU	A	S42	40	90.36	92.47	89.41	89.89
TU	A	S43	40	95.24	92.22	85.29	92.94
TU	B	S42	104	57.14	56.55	58.46	57.53
TU	B	S43	104	55.07	54.93	55.48	56.48
TU	C	S42	172	72.57	65.15	66.10	71.15
TU	C	S43	172	70.45	67.84	64.46	71.04
TU	D	S42	157	73.01	70.31	71.60	66.73

Table 13 (continued)

Subset	Client	Seed	N	Local	FedAvg	FullShare	DREL
TU	D	S43	157	70.58	76.35	72.29	64.05
ALL	A	S42	172	90.39	88.04	83.22	87.53
ALL	A	S43	172	86.87	84.05	82.65	89.76
ALL	B	S42	172	59.39	57.83	59.51	59.84
ALL	B	S43	172	58.69	58.76	59.96	56.07
ALL	C	S42	172	72.57	65.15	66.10	71.15
ALL	C	S43	172	70.45	67.84	64.46	71.04
ALL	D	S42	172	73.04	71.52	71.74	67.72
ALL	D	S43	172	71.35	76.56	73.09	65.32

Table 14: Complete GAIA full-coverage results for Client A. Empty, missing, and literal `None` answers are scored as incorrect.

Method	Seed	Level	Correct/Total	Valid	Empty	None	Accuracy (%)
Local	S42	L1	14/53	52	1	0	26.42
Local	S42	L2	16/86	81	5	0	18.60
Local	S42	L3	0/26	25	1	0	0.00
Local	S42	Overall	30/165	158	7	0	18.18
Local	S43	L1	9/53	52	1	0	16.98
Local	S43	L2	14/86	85	1	0	16.28
Local	S43	L3	1/26	21	5	0	3.85
Local	S43	Overall	24/165	158	7	0	14.55
FedAvg	S42	L1	9/53	51	2	0	16.98
FedAvg	S42	L2	8/86	83	2	1	9.30
FedAvg	S42	L3	2/26	23	3	0	7.69
FedAvg	S42	Overall	19/165	157	7	1	11.52
FedAvg	S43	L1	10/53	50	2	1	18.87
FedAvg	S43	L2	15/86	84	1	1	17.44
FedAvg	S43	L3	2/26	23	3	0	7.69
FedAvg	S43	Overall	27/165	157	6	2	16.36
DREL	S42	L1	9/53	51	2	0	16.98
DREL	S42	L2	11/86	84	2	0	12.79
DREL	S42	L3	1/26	24	2	0	3.85
DREL	S42	Overall	21/165	159	6	0	12.73
DREL	S43	L1	9/53	51	2	0	16.98
DREL	S43	L2	11/86	82	3	1	12.79
DREL	S43	L3	0/26	24	2	0	0.00
DREL	S43	Overall	20/165	157	7	1	12.12

Cumulative accounting. Table 15 reports the tensor inventory per aggregation event. With two events, DREL's cumulative per-client upload is 666 MiB; full sharing over the same schedule would transmit 1,998 MiB per client. Since FULLSHARE is post-hoc composed, this latter value is a matched communication counterfactual, not measured traffic.

Table 15: Communication inventory. Only client-to-server adapter uploads are counted; broadcast, metadata, router, and prototype tensors are excluded. The FedAvg-32 row is the matched-upload control: its entire adapter is uploaded, and it contains exactly the same number of elements as the partition DREL transmits (87,293,952), so the two methods are byte-matched by construction rather than by rounding. DREL and full-rank FedAvg carry the same total LoRA capacity (261,881,856); they differ only in which part is shared.

Method	Category	Tensors	Elements	MiB	Dtype
Local	total	0	0	0	N/A
FedAvg	LoRA r=96	504	261881856	999	float32
FedAvg-32	LoRA r=32 (all uploaded)	504	87293952	333	float32
DREL	R (r=32, aggregated)	504	87293952	333	float32
DREL	E (r=64, local)	504	174587904	666	float32
DREL	Other (router etc)	254	1335296	5	float32
FullShare	R+E (r=96)	1008	261881856	999	float32

M Evaluation Protocol

M.1 GTA

GTA contains 172 text-answerable tasks. Each client is evaluated with its three-tool local toolbox. A task is tool-available (TA) when all required tools are present; otherwise it is tool-unavailable (TU). Evaluation uses Qwen3-VL-8B-Instruct with temperature zero, one sampled trajectory, the `best_selector` verifier, and at most ten tool-use steps. Answer accuracy (AnsAcc) is normalized final-answer matching. Tool accuracy (ToolAcc) measures syntactically valid and semantically correct tool use. Code execution rate (CodeExec) measures Python code blocks that run without error. Empty, missing, or null outputs are scored as incorrect.

Macro summaries use the client–seed cell as the resampling unit. TA has six valid cells (Client C has no TA tasks); TU and ALL have eight. Bootstrap intervals use 10,000 percentile resamples (seed 42) and are descriptive, not significance-tested. Tables 11–13 report every evaluated cell for the four primary methods; the NoSGR ablation is reported separately (Tables 16–17).

M.2 GAIA

GAIA serves as an out-of-distribution boundary check on Client A. Full-coverage rescoring uses fixed denominators of 53, 86, and 26 for Levels 1–3 (165 tasks total). Empty, missing, and literal `None` answers are incorrect. Results are reported in Table 14; no claim of general-purpose reasoning improvement is made.

N Complete Results and Interpretation Boundaries

Complete GTA outcomes are listed in Tables 11–13. Figure 7 provides a three-panel visual summary: answer accuracy across tool-availability conditions (a), per-client DREL–FullShare AnsAcc deltas with bootstrap 95% CIs (b), and the communication–accuracy landscape (c). The GAIA record is in Table 14. The communication inventory (Table 15) counts only client-to-server uploads of aggregated float32 tensors, excluding broadcast, metadata, router/prototype tensors, latency, and peak memory.

The results support a bounded communication–performance observation on GTA, particularly through the DREL–FullShare control. The NoSGR ablation does not show an overall AnsAcc advantage for SGR. The results do not establish universal client-level gains, tool-unavailable gains, representation-level disentanglement, or GAIA improvement. The limited client count, two training seeds, simulated tool partition, two aggregation events, and absent upstream data-generation provenance are material limitations.

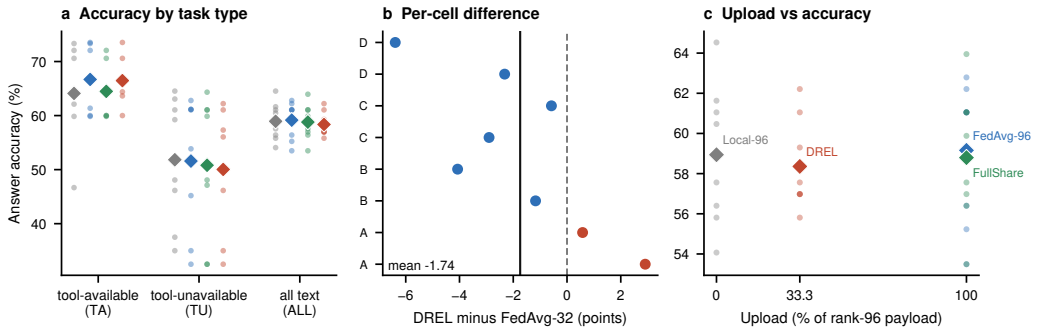

Figure 7: GTA multi-panel evidence grid. **(a)** Answer accuracy (AnsAcc) by method and tool-availability condition. Light points are individual client–seed cells ($N = 6$ TA, $N = 8$ TU/ALL); diamonds are macro means. **(b)** Per-client DREL − FullShare AnsAcc deltas for ALL text tasks. Light circles: paired cells; orange diamonds: per-client means; red square: overall mean with bootstrap 95% CI (shaded band). **(c)** Communication–accuracy landscape. Light points: individual ALL AnsAcc values; diamonds: macro means. Grey line connects the matched DREL–FullShare pair; bracket annotates the 67% upload reduction. All values are descriptive and derived from audited evaluator records.

Table 16: Primary GTA ablation results, macro-averaged over valid client–seed cells. TA uses six valid cells because Client C has $N = 0$; TU and ALL use eight. Both DREL variants communicate the same reasoning partition (333 MiB per client per aggregation event). Values are descriptive, not significance-tested.

Method	TA AnsAcc	TU AnsAcc	ALL AnsAcc
DREL	66.47	50.03	58.36
DREL–NoSGR	64.86	52.00	59.30
NoSGR − DREL	−1.61	+1.97	+0.94

O DREL–NoSGR Ablation

DREL–NoSGR uses the same client data, rank split, local-stage schedule, aggregation weights and reasoning-only aggregation as DREL, and disables SGR (`drel_enable_sgr=false`); however, this ablation also raises the auxiliary loss coefficients to $\lambda_{\text{orth}} = 0.005$ and $\lambda_{\text{role}} = 0.1$, whereas the main DREL runs use 0.001 and 0.03. The originally reported DREL–NoSGR contrast therefore changes two factors at once and cannot be attributed to routing. We report the unconfounded ablation—SGR disabled at the main run's coefficients—separately, and we do not draw a causal conclusion about semantic gradient routing from the confounded pair. A runtime gradient test confirms the NoSGR branch removes role-probability gradient gating while retaining the additive forward adapter structure, role loss, and orthogonality loss. All 29 queue stages completed without failure. We evaluate with the same GTA protocol and report AnsAcc for every client–seed–subset cell.

Table 16 compares NoSGR with DREL at 333 MiB per-client upload per aggregation event. Removing SGR raises ALL AnsAcc from 58.36% to 59.30% (+0.94 pp) and TU AnsAcc from 50.03% to 52.00% (+1.97 pp), while reducing TA AnsAcc from 66.47% to 64.86% (−1.61 pp). Across the eight paired ALL cells, NoSGR is higher in five, tied in one, and lower in two. These differences are descriptive (TA: six valid cells; TU/ALL: eight) and do not establish a causal benefit for either routing choice. The result is consistent with a task-availability-dependent trade-off in this small simulated setting.

Table 17: Complete DREL–NoSGR GTA AnsAcc by client and seed (%). A dash denotes Client C's $N = 0$ TA cells.

Client	Seed	TA	TU	ALL
A	S42	65.15	42.50	59.88

Table 17 (continued)

Client	Seed	TA	TU	ALL
A	S43	62.12	35.00	55.81
B	S42	76.47	47.12	58.72
B	S43	72.06	49.04	58.14
C	S42	–	61.05	61.05
C	S43	–	62.79	62.79
D	S42	60.00	58.60	58.72
D	S43	53.33	59.87	59.30

P REPRODUCIBILITY CHECKLIST

- Training files: ShareGPT JSON artifacts (counts in Table 9).
- Configuration: LlamaFactory YAML files, federated aggregation and composition scripts, adapter manifests.
- Evaluation: GTA result JSONs, the `eval_ta_tu.py` scorer, GAIA SQLite outputs, and the fixed-denominator GAIA rescorer.
- Accounting: tensor inventory of communicated float32 parameters in MiB ($1\,\mathrm{MiB} = 1{,}048{,}576$ bytes).
- Traceability: the evidence manifest maps each numerical result to an archived source file and field. Internal absolute paths should be replaced with repository-relative paths or a permanent artifact identifier before release.