FedMoLoRA: When Heterogeneity is the Setting and Instability is the Problem

***Abstract*—Federated fine-tuning of large language models faces two interacting challenges: clients operate at different hardware tiers, which forces different LoRA ranks, and non-IID data causes the aggregated model to oscillate round to round as different subsets of clients contribute each training round. Existing heterogeneous-rank aggregation methods address the shape mismatch but deploy the raw per-round aggregate directly, leaving the oscillation problem unsolved. We propose FedMoLoRA, which separates what is trained from what is deployed. The server aggregates client adapters using Frobenius-norm weighting in (B, A) space, distributes the raw aggregate to clients for training, and applies bias-corrected exponential moving average (EMA) to the aggregate before evaluation. Client training dynamics are identical to momentum-free aggregation; only the deployed model is smoothed. This design requires no SVD at distribution time and avoids the feedback instability that arises when momentum states are sent back to clients for initialization. Experiments on Qwen2.5-7B with 50 heterogeneous clients across five rank tiers, three benchmarks, and three heterogeneity levels show that FedMoLoRA's advantage concentrates where non-IID data is most severe. On Yelp sentiment classification at Dirichlet $\alpha = 0.1$, it achieves 44.6% AUC, significantly outperforming zero-padding (Hetero-Pad, +4.2 pp, $p = 0.047$) and ΔW-space aggregation (FlexLoRA, +5.2 pp, $p = 0.032$). At the hardest setting ($\alpha = 0.01$, 40 rounds), FedMoLoRA achieves 42.4% Mean-Last-5 while its momentum-off ablation (HetLoRA) drops to 36.8%, a 5.6 pp gap that grows with heterogeneity severity. On math reasoning (GSM8K) and IID instruction following (Alpaca), all methods perform comparably, confirming that the gains are specific to the non-IID classification regime where round-to-round oscillation matters most.**

***Index Terms*—Federated Learning, Large Language Models, LoRA, Heterogeneity, Momentum, Adapter Aggregation, Non-IID.**

I. Introduction and Motivation

Large Language Models (LLMs) have transformed Natural Language Processing across reasoning, code synthesis, and open-domain question answering [1]–[3]. Deploying these models in sensitive domains such as healthcare, law, and finance is constrained by the privacy risks of centralized training: fine-tuning requires pooling proprietary data on a central server, creating a single point of failure for data security.

A. Privacy Challenges in Centralized LLM Training

Centralizing data for LLM training is increasingly untenable under modern regulatory frameworks. GDPR [4] and HIPAA impose strict controls on personal data collection; LLMs are additionally known to memorize training sequences, enabling extraction attacks [5], [6]. Recent incidents confirm the practical severity: the 2025 Grok leak of 370K private conversations [7] and the 2023 Samsung source-code upload to ChatGPT [8] both trace to the same root cause: when data must leave its origin to enable training, exposure is inevitable [9].

B. Federated Learning as a Privacy-Preserving Alternative

Federated Learning (FL) addresses this by training where data lives: participants train locally and upload only parameter deltas, so raw data never leaves the client device [10], [11]. To make billion-parameter LLM fine-tuning feasible on local hardware, researchers combine FL with Low-Rank Adaptation (LoRA) [12], which constrains updates to low-rank matrix products $B \in \mathbb{R}^{m \times r}$ and $A \in \mathbb{R}^{r \times n}$. Only the small adapter matrices (B, A) are communicated each round, keeping bandwidth low.

C. The Deployment Instability Problem

Combining FL with LoRA introduces two distinct challenges. The first is **rank heterogeneity**: a data-center node with 48 GB of GPU memory can support LoRA rank 32, while an edge device limited to 4 GB can only support rank 2. When clients use different ranks, their adapter matrices have different shapes and cannot be averaged directly. Prior work addresses this through zero-padding [13] or ΔW reconstruction with SVD redistribution [14], [15].

The second challenge, which prior work does not directly address, is **round-to-round oscillation in the deployed model**. In each round, only K of N clients participate, selected at random. Under non-IID data, different rounds sample different client populations with different local data distributions. The server-side aggregate shifts accordingly, so the model quality measured at consecutive rounds can vary substantially even as training makes overall progress. This is a deployment problem: in a real FL system, the server may need to serve the current best model to users between rounds, and a model that oscillates is unreliable.

The instability grows with data heterogeneity and is worst at $\alpha = 0.01$.

Existing aggregation methods, even well-designed ones like HetLoRA, deploy the raw per-round aggregate. To our knowledge, no prior work has posed this question specifically for the heterogeneous federated LoRA setting: rather than deploying the raw aggregate, can we deploy a smoothed version of it, without changing how clients train?

D. Our Contribution: Momentum Strictly on the Server

We propose **FedMoLoRA**, which decouples training from deployment. The server maintains a bias-corrected EMA over

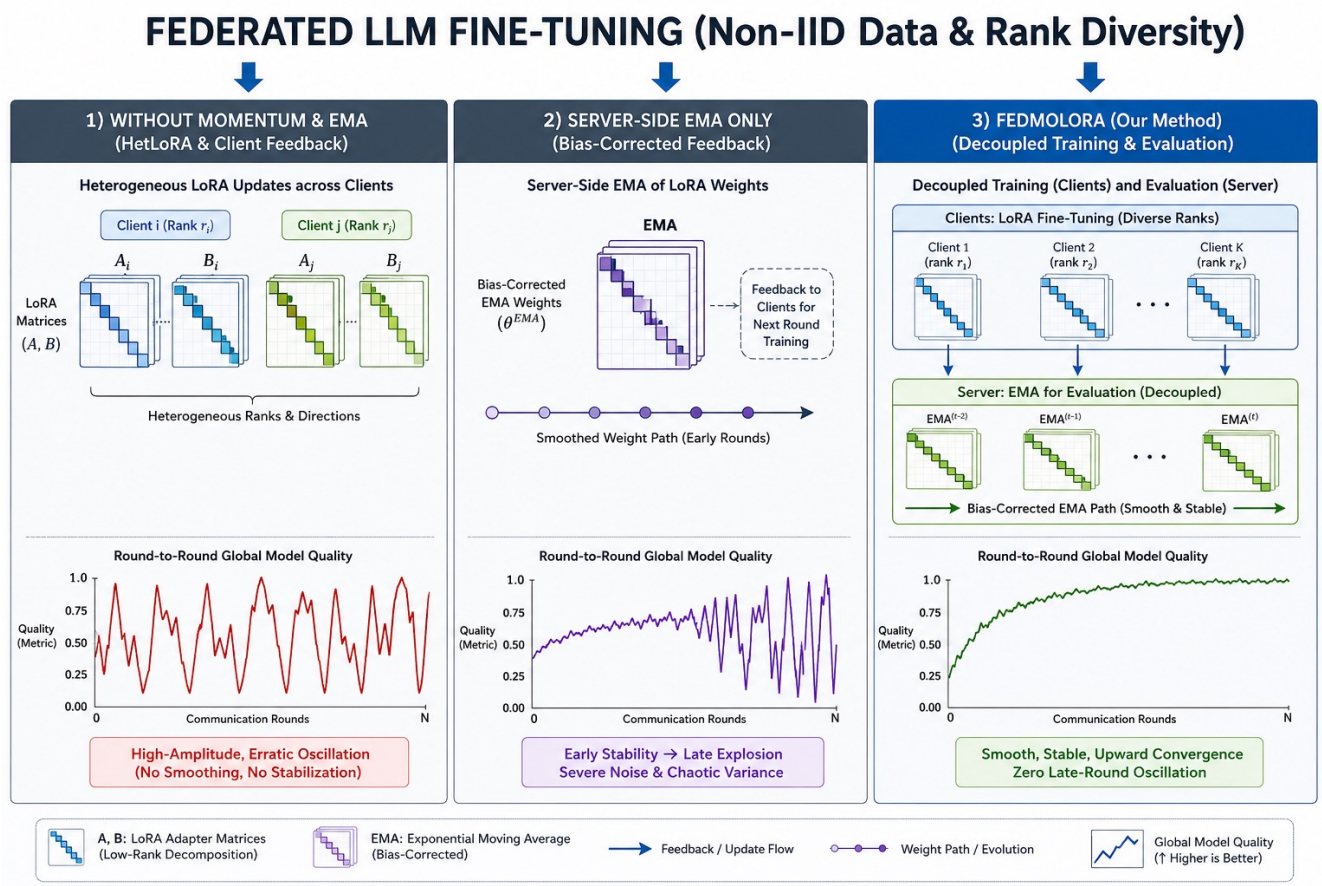


Fig. 1: Three deployment strategies under non-IID data and rank diversity. **Left:** Without momentum or EMA (HetLoRA), the deployed model oscillates with high amplitude throughout training. **Center:** Server-side EMA with bias-corrected feedback to clients yields early stability but causes late-round explosion, as clients initialize from a smoothed state and their uploads become dominated by gradient noise. **Right:** FedMoLoRA decouples training from evaluation: clients always receive the raw aggregate while the server applies EMA only for deployment, producing smooth, stable, upward convergence with zero late-round oscillation.

the Frobenius-weighted (B, A) aggregate and uses the EMA state only for the deployed (evaluated) model. Clients always receive the raw aggregate for initialization. This means:

- Client training dynamics are identical to momentum-free HetLoRA. Adding FedMoLoRA's EMA does not change what any client trains on.
- The deployed model at each round is a weighted average of past aggregates, which suppresses round-to-round noise.
- No SVD is needed at distribution time, unlike ΔW-space methods.

The idea is conceptually related to Polyak averaging and Stochastic Weight Averaging [16] in centralized training, where averaging model checkpoints gives a better deployed model than any individual checkpoint. FedMoLoRA brings this to heterogeneous federated LoRA by averaging in (B, A) space, where the geometry is compatible with truncation-based distribution.

Our contributions are:

- **Problem identification**: We show that round-to-round oscillation in the deployed model is a distinct and addressable problem in federated LLM fine-tuning, separate from the shape-mismatch problem that prior work solves (Section III).
- **Algorithm**: FedMoLoRA combines Frobenius-norm-weighted aggregation with bias-corrected, server-side EMA in (B, A) space, requiring no SVD at distribution time (Section III).
- **Evaluation**: Three benchmarks (Yelp, GSM8K, Alpaca), three Dirichlet heterogeneity levels ($\alpha \in \{0.01, 0.1, 0.5\}$), five rank tiers, 50 clients on Qwen2.5-7B, with paired significance tests and ablations over β, participation rate K, and rank distribution. We report AUC (mean metric over all rounds) as the primary metric, alongside Mean-Last-5 and Best Accuracy (Section V).

TABLE I: Algorithmic comparison with related work on heterogeneous federated fine-tuning of LLMs. ✓= yes, ∼ = partially, ×= no. FedARA [17] and Fed-PLoRA [18] address rank *allocation*, not aggregation, and are omitted from direct comparison.

Property	[10] FedAvg 2017	[14] FlexLoRA 2024	[13] HetLoRA 2024	[19] LoRA-A^2 2025	[20] ILoRA 2025	**Ours** FedMoLoRA 2026
Supports heterogeneous ranks	×	✓	✓	✓	✓	✓
Aggregates in (B, A) space	–	×	✓	∼	∼	✓
Norm-weighted aggregation	×	×	✓	×	×	✓
SVD-free distribution	✓	×	✓	∼	✓	✓
Server-side EMA on deployed model	×	×	×	×	×	✓

II. Background and Related Work

A. Federated Learning for Large Language Models

Federated Learning was proposed by McMahan et al. [10] for distributed training on mobile devices. Scaling FL to LLMs requires addressing communication overhead, memory constraints, and non-IID data [21], [22]. FedIT [23] and FATE-LLM [24] apply adapter-based fine-tuning in FL but assume homogeneous client architectures, limiting applicability in realistic settings [25], [26].

B. Low-Rank Adaptation and PEFT

PEFT methods [27] reduce fine-tuning cost by updating only a subset of parameters. LoRA [12] is the dominant approach, constraining updates to:

$$h = W_0 x + \frac{s}{r} BAx \tag{1}$$

where W_0 is frozen and s is a fixed scaling constant. QLoRA [28] adds quantization; AdaLoRA [29] and DyLoRA [30] explore dynamic rank allocation. Dynamic rank allocation in decentralized settings remains an open problem that FedARA [17] and Fed-PLoRA [18] address separately from aggregation.

C. Heterogeneous Rank Aggregation

System heterogeneity in FL is addressed by FedProx [31] and FedNova [32], but these assume homogeneous model spaces. HeteroFL [33] and Fjord [34] support sub-model training for CNNs, which does not transfer to LoRA's matrix structure.

For LoRA-specific heterogeneity, a growing body of work exists. FLoRA [15] and FlexLoRA [14] reconstruct ΔW and redistribute via SVD, with no momentum component. HetLoRA [13] (EMNLP 2024) uses zero-padding with Frobenius-norm-weighted aggregation in (B, A) space. ILoRA [20] introduces invariant representations for non-IID robustness. LoRA-A^2 [19] (ACL 2025) improves robustness via alignment-aware aggregation. FedARA [17] dynamically allocates rank budgets using gradient sensitivity, and Fed-HeLLo [35] combines heterogeneous allocation with communication scheduling. Fed-PLoRA [18] decomposes adapters into parallel one-rank updates, avoiding rank mismatch by construction. FedMoLoRA is orthogonal to rank allocation methods: we treat hardware-fixed ranks as given and focus on what the server deploys at each round.

ILoRA and LoRA-A^2 are not included in our baseline comparison because both methods modify the *client-side* training objective (invariant representation learning and alignment-aware local updates, respectively) rather than the server-side aggregation protocol. Since FedMoLoRA's contribution is strictly server-side, comparing against methods that alter client training would conflate two orthogonal design axes. FlexLoRA and HetLoRA are included instead because they share identical client training with FedMoLoRA, isolating the effect of aggregation and deployment choices.

D. Weight Averaging and Momentum in FL

Polyak averaging [36] and Stochastic Weight Averaging [16] show that averaging model checkpoints gives a better and more stable model than the final checkpoint alone. In centralized training, this is a simple post-processing step. In FL, applying it requires care because the server-side state is a weighted aggregate of client uploads, not a sequence of full model checkpoints.

Server-side momentum in FL has shown convergence benefits under non-IID data (Mime [37], SlowMo [38]), but these methods apply momentum to full model parameters, not LoRA adapters. To our knowledge, FedMoLoRA is the first method to apply EMA in (B, A) adapter space for heterogeneous federated LoRA, keeping momentum strictly on the server side so that client training is unaffected.

E. Distinction from Closest Prior Work

vs. FlexLoRA/FLoRA: both aggregate in ΔW space with SVD redistribution and no momentum. FedMoLoRA aggregates in (B, A) space with Frobenius weighting and no SVD at distribution. **vs. HetLoRA [13]**: our implementation uses HetLoRA's Frobenius-weighted (B, A) aggregation without rank self-pruning, making HetLoRA the exact momentum-off ablation of FedMoLoRA. The two share an identical training trajectory; FedMoLoRA adds evaluation-side EMA. **vs. FedARA/Fed-PLoRA**: those works solve rank allocation; FedMoLoRA solves aggregation and deployment; they are complementary.

III. Methodology

A. Problem Formulation

We consider $N = 50$ clients, each with local dataset $\mathcal{D}_k$ and hardware-fixed rank $r_k \in \{2, 4, 8, 16, 32\}$. The goal is to fine-tune a pre-trained model f_{θ_0} using distributed data

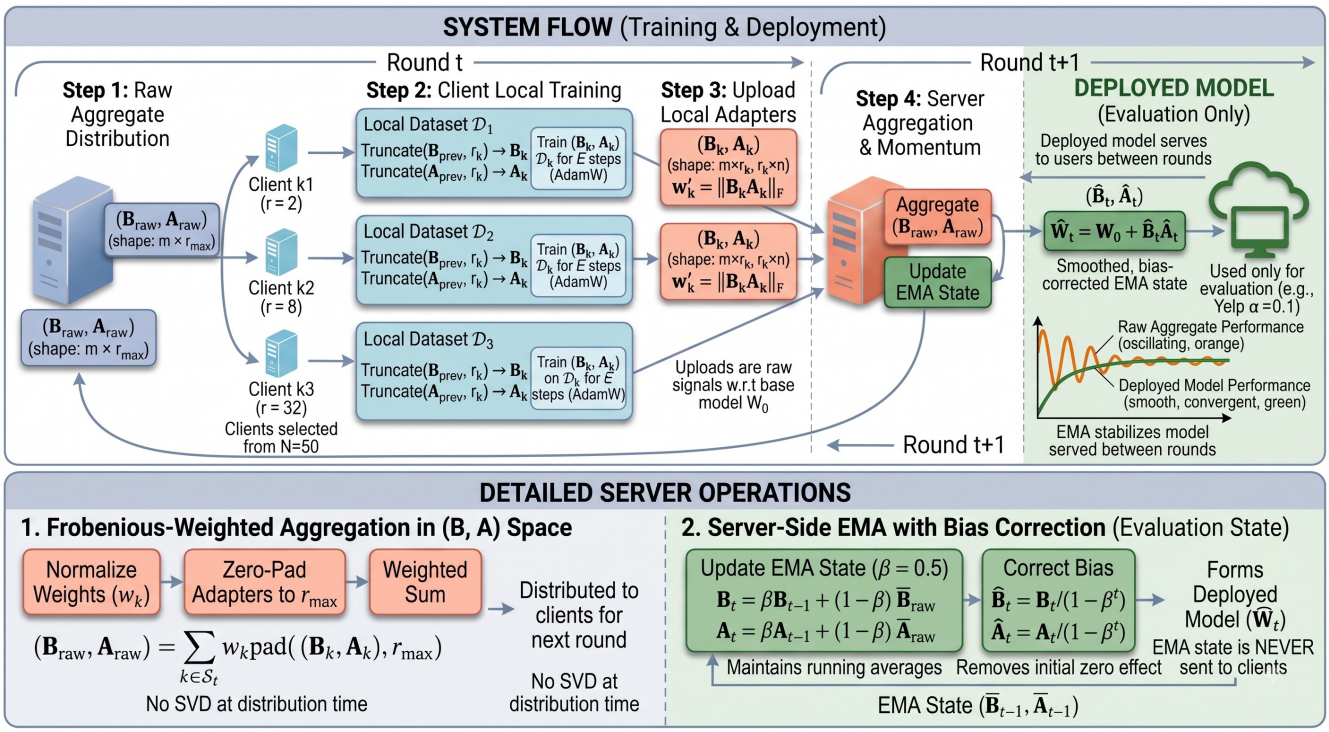


Fig. 2: FedMoLoRA system overview. **Top:** Training and deployment flow across rounds. Clients receive the raw aggregate $(B_{\text{raw}}, A_{\text{raw}})$ for local training; the server applies bias-corrected EMA to form the deployed model $\hat{W}_t = W_0 + \hat{B}_t \hat{A}_t$, which is never sent back to clients. **Bottom:** Detailed server operations: Frobenius-weighted aggregation in (B, A) space (no SVD) and the EMA update with bias correction.

while keeping all $\mathcal{D}_k$ on their respective clients. Each client maintains adapter matrices $B_k \in \mathbb{R}^{m \times r_k}$ and $A_k \in \mathbb{R}^{r_k \times n}$ applied to a frozen weight $W_0 \in \mathbb{R}^{m \times n}$. Direct averaging is impossible because the rank dimension varies across clients.

Beyond the shape mismatch, there is a second problem. In each round, $K = 5$ clients are selected at random. Under non-IID data, the selected clients in round t may have systematically different local distributions from those in round $t+1$. The raw server-side aggregate therefore shifts each round, and the model served to users at round t may perform differently from the model at round $t-1$, even as the running average of all round aggregates is improving.

B. Design Rationale

The natural response to per-round oscillation is to smooth the aggregate over time. One design would apply EMA in ΔW space and distribute the smoothed ΔW to clients via SVD for initialization. We avoid this for two reasons. First, it requires a full SVD every round, which is expensive for large weight matrices. Second, when clients initialize from a momentum-smoothed state, their uploads become deviations from that state rather than from the base model W_0. As training converges, these deviations shrink and become dominated by gradient noise; EMA amplifies rather than attenuates this noise, leading to late-round instability.

FedMoLoRA instead applies EMA entirely on the server and restricts it to the evaluation model. Clients always initialize from the raw aggregate, so their uploads carry genuine signal relative to the base model throughout training.

C. Frobenius-Weighted Aggregation

The server assigns each selected client $k \in \mathcal{S}_t$ a weight proportional to the Frobenius norm of its reconstructed adapter:

$$w_k = \frac{\|B_k A_k\|_F}{\sum_{j \in \mathcal{S}_t} \|B_j A_j\|_F} \tag{2}$$

This down-weights clients with low-signal adapters, such as rank-2 clients whose local data contains few samples under non-IID partitioning, without requiring full ΔW reconstruction at the server. The Frobenius norm is computed as $\|B_k A_k\|_F = \sqrt{\text{tr}(B_k^\top B_k \cdot A_k A_k^\top)}$, which costs $O((m+n)r_k^2)$ and avoids forming the full $m \times n$ matrix.

To aggregate matrices of differing ranks, we zero-pad along the rank dimension before the weighted sum:

$$B_{\text{raw}} = \sum_{k \in \mathcal{S}_t} w_k \, \text{pad}(B_k, \, r_{\max}) \tag{3}$$
$$A_{\text{raw}} = \sum_{k \in \mathcal{S}_t} w_k \, \text{pad}(A_k, \, r_{\max})$$

The pair $(B_{\text{raw}}, A_{\text{raw}})$ is distributed immediately to clients. Momentum is applied separately and never forwarded to clients.

D. Adapter-Space EMA with Bias Correction

The server updates two running averages in parallel with the aggregation:

$$\bar{B}_t = \beta \bar{B}_{t-1} + (1-\beta)\, B_{\text{raw}} \tag{4}$$

$$\bar{A}_t = \beta \bar{A}_{t-1} + (1-\beta)\, A_{\text{raw}} \tag{5}$$

Bias correction removes the effect of zero initialization at $t = 0$:

$$\hat{B}_t = \frac{\bar{B}_t}{1-\beta^t}, \qquad \hat{A}_t = \frac{\bar{A}_t}{1-\beta^t} \tag{6}$$

The corrected pair $(\hat{B}_t, \hat{A}_t)$ defines the evaluation model $\hat{W}_t = W_0 + \hat{B}_t \hat{A}_t$. We use $\beta = 0.5$: with 10% client participation per round ($K = 5$ of $N = 50$), a half-life of one round is appropriate for the per-round noise level observed in preliminary experiments.

Relationship to training dynamics. Because clients always receive $(B_{\text{raw}}, A_{\text{raw}})$, FedMoLoRA's training trajectory is identical to momentum-free Frobenius-weighted aggregation (HetLoRA). The EMA changes only which model is evaluated and deployed each round. This means FedMoLoRA's gains over HetLoRA directly measure the value of evaluation-side smoothing, holding everything else fixed. It also means evaluation-side EMA could in principle be applied on top of any aggregation method.

E. Full Algorithm

Algorithm 1 summarizes one communication round. Each round proceeds in four phases: (1) the server distributes the raw aggregate from the previous round to the selected clients, who truncate it to their local rank and fine-tune for E local steps; (2) clients upload their updated adapters along with Frobenius-norm weights, which the server normalizes and uses to compute the new raw aggregate in (B, A) space; (3) the server updates its EMA state using the raw aggregate and applies bias correction; (4) the bias-corrected EMA state forms the deployed evaluation model, while the raw aggregate is sent to next-round clients. Crucially, clients never receive the EMA state, ensuring training dynamics remain identical to momentum-free aggregation throughout.

F. Why Does EMA Stabilize the Deployed Model?

Frobenius weighting is signal-proportional. The Frobenius norm $\|B_k A_k\|_F$ measures the spectral energy of client k's adapter. Clients with more data from well-represented classes produce higher-norm adapters; clients with degenerate local distributions produce near-zero-norm adapters. Weighting by $\|B_k A_k\|_F$ is a data-quality proxy the server can compute from uploaded artifacts without accessing local data.

Formal variance-reduction argument. We model each round's aggregate as $\bar{B}_{\text{raw}}^{(t)} = \mu^* + \varepsilon_t$, where μ^* is the consensus adapter and ε_t are zero-mean, uncorrelated perturbations with element-wise variance σ^2. This captures the key source of instability: random client sampling under non-IID data shifts $\bar{B}_{\text{raw}}^{(t)}$ each round even as μ^* improves.

Algorithm 1 FedMoLoRA: One Communication Round t

Require: W_0, EMA state $(\bar{B}_{t-1}, \bar{A}_{t-1})$, β, selected clients $\mathcal{S}_t$, prior raw aggregate $(B_{\text{prev}}, A_{\text{prev}})$. At $t = 1$: $\bar{B}_0 = \bar{A}_0 = 0$; $(B_{\text{prev}}, A_{\text{prev}})$ initialized as zeros (standard LoRA init).

Phase 1: Download and Local Training
for each client $k \in \mathcal{S}_t$ **do**
  Set $B_k \leftarrow \text{truncate}(B_{\text{prev}}, r_k)$, $A_k \leftarrow \text{truncate}(A_{\text{prev}}, r_k)$
  Train (B_k, A_k) on $\mathcal{D}_k$ for E local steps (AdamW)
  Compute upload weight: $w'_k \leftarrow \|B_k A_k\|_F$
  Upload (B_k, A_k) and w'_k
end for
Phase 2: Weighted Aggregation
Normalize: $w_k \leftarrow w'_k / \sum_j w'_j$
Compute $(B_{\text{raw}}, A_{\text{raw}})$ via Equation (3)
Phase 3: Adapter-Space EMA
Update $\bar{B}_t$, $\bar{A}_t$ via Equations (4)–(5)
Correct bias: $\hat{B}_t$, $\hat{A}_t$ via Equation (6)
Phase 4: Evaluate and Distribute
Evaluate: $\hat{W}_t = W_0 + \hat{B}_t \hat{A}_t$
Distribute $(B_{\text{raw}}, A_{\text{raw}})$ to next round's clients
return $(\bar{B}_t, \bar{A}_t)$

Proposition 1 (EMA Variance Reduction)**.** *Under the model above with $\bar{B}_0 = 0$, the bias-corrected EMA state $\hat{B}_t = \bar{B}_t/(1-\beta^t)$ is unbiased ($\mathbb{E}[\hat{B}_t] = \mu^*$) and satisfies:*

$$\text{Var}(\hat{B}_t) = \frac{1-\beta}{1+\beta} \cdot \sigma^2 \cdot \frac{1-\beta^{2t}}{(1-\beta^t)^2} \xrightarrow{t\to\infty} \frac{1-\beta}{1+\beta}\,\sigma^2 \;<\; \sigma^2. \tag{7}$$

*At $\beta = 0.5$, asymptotic variance is $\frac{1}{3}\sigma^2$: a **67% reduction** over the raw aggregate.*

Proof sketch. Since $\bar{B}_t = (1-\beta)\sum_{i=1}^{t} \beta^{t-i} \bar{B}_{\text{raw}}^{(i)} + \beta^t \bar{B}_0$ and $\bar{B}_0 = 0$:

$$\mathbb{E}[\bar{B}_t] = (1-\beta)\sum_{i=1}^{t} \beta^{t-i}\mu^* = \mu^*(1-\beta^t),$$

so $\hat{B}_t = \bar{B}_t/(1-\beta^t)$ is unbiased. For variance, since $\{\varepsilon_t\}$ are uncorrelated:

$$\begin{aligned}\text{Var}(\bar{B}_t) &= (1-\beta)^2 \sum_{i=1}^{t} \beta^{2(t-i)}\sigma^2 \\ &= \frac{(1-\beta)^2(1-\beta^{2t})}{1-\beta^2}\,\sigma^2 = \frac{1-\beta}{1+\beta}(1-\beta^{2t})\,\sigma^2.\end{aligned}$$

Dividing by $(1-\beta^t)^2$ gives the stated expression. As $t \to \infty$, $\beta^t \to 0$ and the ratio $(1-\beta^{2t})/(1-\beta^t)^2 \to 1$. □

Note. This bound holds under the stated zero-mean, uncorrelated noise model; real-world non-IID client sampling naturally induces drift in μ^* and inter-round correlations in ε_t that can partially offset the variance reduction; see Section VI for the full context.

The key structural property is that clients upload signal relative to W_0, not deviations from a momentum state, so each ε_t is genuinely independent of the EMA state and the uncorrelated assumption holds. Formal convergence analysis under non-IID training dynamics (drifting μ^*, correlated ε_t) is noted as future work in Section VI.

G. Computational Cost

The dominant cost per round is the Frobenius norm computation: $O((m+n)r_k^2)$ per client using the trace formula in Equation (2), run locally before upload. On the server, the weighted sum in Equation (3) costs $O(|\mathcal{S}| \cdot m \cdot r_{\max})$ and the EMA update costs $O(m \cdot r_{\max} + r_{\max} \cdot n)$. For $r_{\max} = 32$ and $m = n = 4096$, these operations complete in under one second on a single GPU. FedMoLoRA has no SVD step at distribution time, unlike FlexLoRA and FLoRA, reducing server-side cost per round.

H. Baselines

We compare against four baselines to isolate each design choice:

Homo r=8: all clients use rank 8, an idealized homogeneous baseline that assumes every client can afford the median rank.

Hetero-Pad: zero-pad all adapters to $r_{\max}$ and average directly. Minimum change to FedAvg for rank heterogeneity, no norm weighting.

FlexLoRA [14]: ΔW-space SVD aggregation without momentum. Tests whether ΔW-space aggregation is better than (B, A)-space aggregation.

HetLoRA [13]: Frobenius-weighted (B, A) aggregation without momentum; the exact momentum-off ablation of FedMoLoRA with an identical training trajectory.

IV. Experimental Setup

A. Datasets and Distribution

We evaluate on three benchmarks chosen to span different heterogeneity sensitivities.

Yelp Review Full [39] is a 5-class sentiment classification task. We partition data across 50 clients using a Dirichlet distribution with concentration $\alpha \in \{0.01, 0.1, 0.5\}$, following standard FL benchmark protocols [40], [41]. The $\alpha = 0.01$ condition represents near-pathological non-IID data and is evaluated over 40 rounds to capture late-round behavior. Sentiment labels are strongly demographic and geographic, making Yelp the most heterogeneity-sensitive of our three benchmarks. Metric: classification accuracy (↑).

GSM8K [42] is a grade-school math reasoning dataset requiring multi-step arithmetic. Math reasoning generalizes well across data shards, making GSM8K less heterogeneity-sensitive than Yelp. Metric: exact match (↑).

Alpaca [43] is a 52K instruction-following dataset using uniform random client splits regardless of the α parameter. This produces near-IID conditions and serves as a control: any method claiming general utility must not regress on IID data. Metric: ROUGE-L [44] (↑).

B. Model and Training Configuration

We fine-tune Qwen2.5-7B [45] in `bfloat16` precision. LoRA is applied to all attention projection layers (Q, K, V, O) and both MLP feed-forward layers; the base model is frozen. The 50-client network spans five rank tiers with 10 clients each: $r \in \{2, 4, 8, 16, 32\}$. Per round, $K = 5$ clients are selected uniformly at random from all 50. Training runs for $T = 20$ communication rounds for $\alpha \in \{0.5, 0.1\}$ and $T = 40$ rounds for $\alpha = 0.01$ (extended to observe late-round behavior under extreme non-IID). Yelp uses seeds $\{42, 43, 44, 45, 46\}$ (5 seeds); GSM8K uses 5 seeds for $\alpha \in \{0.5, 0.01\}$ and 3 seeds for $\alpha = 0.1$; Alpaca uses 3 seeds. Local optimization uses AdamW [46].

FedMoLoRA hyperparameters. We use a fixed $\beta = 0.5$ with bias correction. This value was selected because with $K = 5$ clients participating out of $N = 50$ per round (10% participation), consecutive rounds share no guaranteed overlap in client population, making each round's aggregate largely independent. A half-life of one round ($\beta = 0.5$) matches this independence structure: it gives equal weight to the current and all prior aggregates without over-smoothing recent signal or retaining stale updates too long. The ablation in Section V-I confirms that performance is robust to this choice within $\beta \in \{0.3, 0.5, 0.7\}$. All methods produce a $r_{\max} = 32$ aggregate; for evaluation we truncate to the median rank $r = 8$ (the middle tier of the five tiers), reflecting a fair deployment operating point accessible to mid-tier clients.

C. Primary Evaluation Metric: AUC

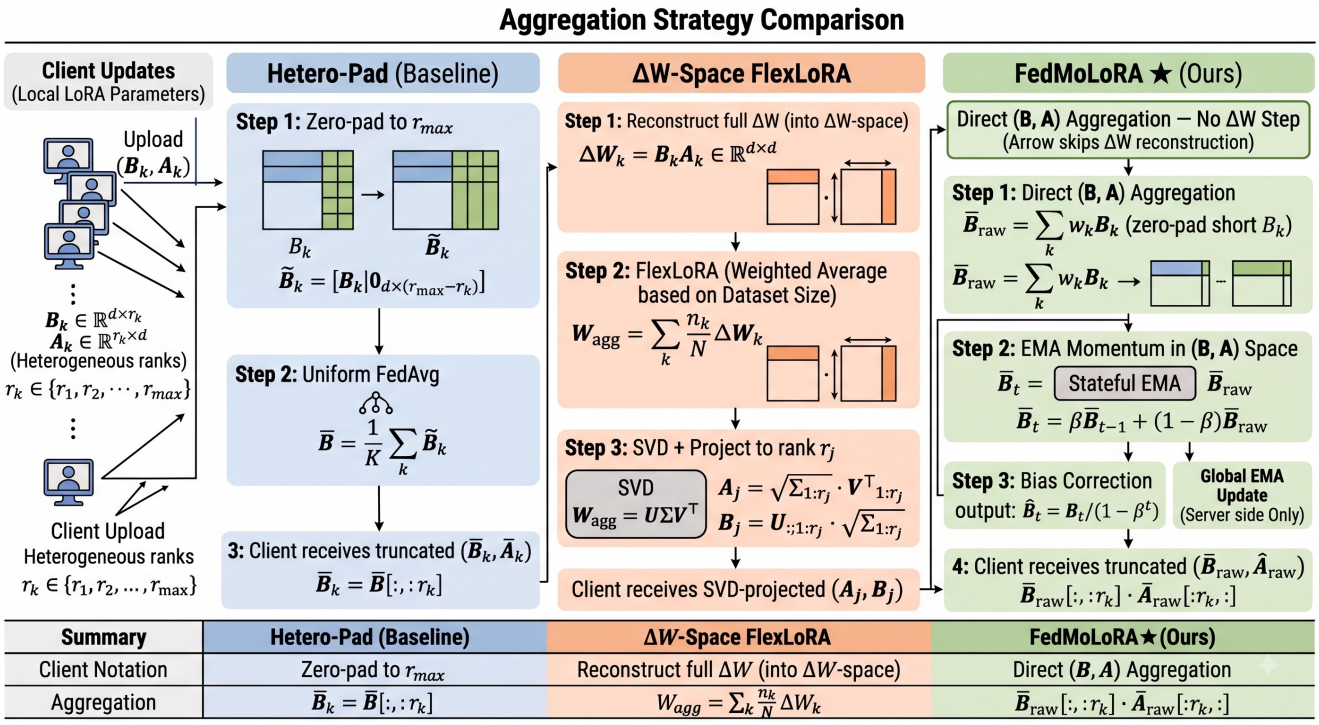
Standard reporting in federated LLM fine-tuning uses Best Accuracy (maximum over all rounds) or Mean-Last-5 (mean of the final five rounds). Best Accuracy rewards methods that peak early and oscillate; Mean-Last-5 can miss instability in earlier rounds.

We use **AUC** (mean metric over all T rounds, averaged across seeds) as the primary metric. AUC rewards methods that converge quickly and remain stable throughout training. In a federated system where inference may occur at any round, a consistently high model is more valuable than one that peaks occasionally. We additionally report Mean-Last-5 and Best Accuracy for completeness.

D. Implementation and Reproducibility

All experiments use PyTorch [47] with the HuggingFace `transformers` library [48]. LoRA adapters are applied to the Q, K, V, O attention projections and both MLP feed-forward layers (`gate_proj`, `down_proj`) of Qwen2.5-7B; the LoRA scaling constant is fixed at $s = 16$ across all rank tiers. Each client trains for one local epoch per round using AdamW with learning rate 2×10^{-4}, linear warmup over 50 steps, cosine decay, and gradient clipping at 1.0. Local batch size is 4 sequences; maximum sequence length is 512 tokens.

All server-side operations (Frobenius-norm computation, weighted aggregation, EMA update) are performed in float32 to avoid precision loss in the evaluation model; client adapters are stored in bfloat16 to match the base model. Rank truncation from $r_{\max} = 32$ to the evaluation rank $r = 8$ takes the first 8 rows of $\hat{B}_t$ and first 8 columns of $\hat{A}_t$; this is exact for zero-padded adapters because padding rows and columns are zero by construction. Full configuration files and per-seed result logs will be released upon acceptance.



Summary	Hetero-Pad (Baseline)	ΔW-Space FlexLoRA	FedMoLoRA★(Ours)
Client Notation	Zero-pad to r_{max}	Reconstruct full ΔW (into ΔW-space)	Direct (B, A) Aggregation
Aggregation	$\overline{B}_k = \overline{B}[:, :r_k]$	$W_{agg} = \sum_k \frac{n_k}{N} \Delta W_k$	$\overline{B}_{raw}[:, :r_k] \cdot \overline{A}_{raw}[:r_k, :]$

Fig. 3: Aggregation strategy comparison. **Hetero-Pad** zero-pads adapters to r_{max} and applies uniform FedAvg. **FlexLoRA** reconstructs $\Delta W = BA$, averages in weight space, and uses SVD to project back to each client's rank. **FedMoLoRA** (ours) aggregates directly in (B, A) space with Frobenius-norm weighting, applies server-side EMA with bias correction for the deployed model, and distributes only the raw aggregate $\bar{B}_{raw}$ to clients; the EMA state never leaves the server.

TABLE II: Main results: AUC (mean over all rounds ↑) ± std across seeds. **Bold** = best per column. Yelp and GSM8K in %; Alpaca in ROUGE-L. α=0.5 and α=0.1 run 20 rounds; α=0.01† runs 40 rounds. Yelp: 5 seeds. GSM8K α=0.5 and α=0.01: 5 seeds; GSM8K α=0.1 and Alpaca: 3 seeds.

	Yelp (AUC %, ↑)			**GSM8K** (EM %, ↑)			**Alpaca**
Method	α=0.5	α=0.1	α=0.01†	α=0.5	α=0.1	α=0.01†	IID
Homo r=8	46.3 ± 1.1	39.3 ± 2.0	35.5 ± 3.0	$\mathbf{76.2 \pm 0.7}$	76.5 ± 0.2	73.8 ± 0.7	$\mathbf{0.414 \pm 0.015}$
Hetero-Pad	44.6 ± 5.0	40.5 ± 1.7	36.2 ± 1.3	75.6 ± 1.2	76.5 ± 0.3	74.5 ± 1.4	0.403 ± 0.014
FlexLoRA	46.2 ± 1.8	39.4 ± 2.2	35.4 ± 2.2	$\mathbf{76.2 \pm 1.3}$	$\mathbf{76.7 \pm 1.1}$	74.3 ± 1.0	$\mathbf{0.414 \pm 0.016}$
HetLoRA	47.2 ± 3.6	41.5 ± 2.8	39.3 ± 2.9	75.3 ± 0.7	74.9 ± 0.6	74.5 ± 0.6	0.402 ± 0.021
FedMoLoRA (ours)	$\mathbf{47.5 \pm 3.2}$	$\mathbf{44.6 \pm 2.8}$	$\mathbf{40.1 \pm 3.6}$	75.7 ± 0.4	75.7 ± 1.0	$\mathbf{74.9 \pm 1.1}$	0.400 ± 0.017

V. Results and Analysis

A. Main Results

Table II reports AUC for all methods, datasets, and heterogeneity levels. FedMoLoRA leads on Yelp at all three α levels; its advantage widens as heterogeneity increases. At $\alpha = 0.1$, improvements over Hetero-Pad ($p = 0.047$) and FlexLoRA ($p = 0.032$) are statistically significant; see Section V-G. At $\alpha = 0.5$ the methods are within one standard deviation and we make no significance claim.

B. Cross-Metric Convergence Patterns

As illustrated in Fig. 3, FedMoLoRA's design separates the training path (raw aggregate to clients) from the deployment path (EMA state on the server), which is what produces the smooth upward convergence shown in Fig. 1 as opposed to the oscillation seen in HetLoRA or the late-round explosion seen when EMA feedback is sent to clients. Table II reports AUC alongside seed standard deviations, which together reveal convergence behaviour without requiring trajectory plots. On Yelp at $\alpha = 0.1$, FedMoLoRA and HetLoRA share identical seed variance (± 2.8 pp), consistent with the two sharing the same training trajectory; the 3.1 pp AUC gap between them reflects evaluation-side smoothing alone. At $\alpha = 0.5$, both (B, A)-space methods show wider seed variance (± 3.2–3.6 pp) than the homogeneous or ΔW-space baselines (± 1.1–1.8 pp), suggesting Frobenius-weighted aggregation is more sensitive to which 5 clients are sampled under moderate heterogeneity, where round-to-round client variation is less

systematic.

On GSM8K, FedMoLoRA's Mean-Last-5 exceeds its AUC at both $\alpha = 0.5$ (76.47% vs. 75.7%) and $\alpha = 0.1$ (76.93% vs. 75.7%), indicating the model is still improving at $T = 20$ rounds; this positive AUC-to-Mean-Last-5 gap is absent in the Yelp methods, which have largely converged by round 20. This motivates the 40-round horizon used for $\alpha = 0.01$.

Convergence speed across tasks. Yelp and GSM8K show markedly different convergence profiles. On Yelp at $\alpha \in \{0.5, 0.1\}$, AUC and Mean-Last-5 are within 1 pp, indicating 20 rounds is sufficient for sentiment classification to converge. On GSM8K, Mean-Last-5 consistently exceeds AUC by 0.5–1.2 pp across all methods, indicating the model is still improving at round 20. Math reasoning involves compositional multi-step patterns that require more exposure to generalize across client data distributions; EMA does not accelerate learning, so this AUC-to-Mean-Last-5 gap is identical across methods on GSM8K, ruling out an algorithm-specific explanation.

Seed variance as a stability diagnostic. Per-seed AUC standard deviation captures method sensitivity to which random subset of clients participates each round, integrated over the full training horizon. On Yelp $\alpha = 0.1$, FedMoLoRA (± 2.8 pp) and HetLoRA (± 2.8 pp) are indistinguishable, confirming that evaluation-side EMA does not alter the per-seed sensitivity of the underlying training process: it raises the floor of what is deployed without changing how training responds to different client samples. Hetero-Pad shows the highest variance at $\alpha = 0.5$ (± 5.0 pp), consistent with zero-padding's known sensitivity to the rank distribution of the sampled clients. FlexLoRA (± 1.8 pp) is more stable, reflecting SVD redistribution normalizing rank contributions prior to aggregation.

C. Yelp: Non-IID Sentiment Classification

FedMoLoRA achieves the highest AUC at all three heterogeneity levels. At $\alpha = 0.1$, FedMoLoRA reaches 44.6%, significantly outperforming Hetero-Pad (+4.2 pp, $p = 0.047$) and FlexLoRA (+5.2 pp, $p = 0.032$), with a borderline gain over Homo r=8 (+5.3 pp, $p = 0.055$). The gap over HetLoRA (+3.1 pp, $p = 0.162$) is not significant, which is expected: the two share an identical training trajectory, and 5 seeds lack statistical power to isolate the effect of evaluation-side smoothing alone. The EMA control experiments in Section V-H provide a complementary view of this effect.

FedMoLoRA raises the deployed model's floor rather than its ceiling. At $\alpha = 0.1$, FedMoLoRA's Best Accuracy (maximum over all rounds, same experimental runs as Table II) is 56.5% against an AUC of 44.6%, a gap of 11.9 pp; HetLoRA has a Best of 53.0% and AUC of 41.5%, a gap of 11.5 pp. Both gaps are similar in size because the two share the same training trajectory; what differs is that the evaluation-side EMA in FedMoLoRA keeps the deployed model closer to its peak for more rounds.

Extreme heterogeneity ($\alpha = 0.01$, 40 rounds). All methods achieve lower absolute AUC at $\alpha = 0.01$ than at milder settings because the task itself is harder under near-pathological non-IID partitioning; what grows with heterogeneity severity is FedMoLoRA's *relative* advantage over the baselines, not its absolute score. FedMoLoRA achieves 40.1% AUC and 42.4% Mean-Last-5, the best of all evaluated methods. Because this condition runs for 40 rounds, the divergence between a method's AUC and its Mean-Last-5 becomes the primary diagnostic signal: a positive gap (Mean-Last-5 $>$ AUC) confirms the deployed model is still improving late in training, while a negative gap reveals that peak accuracy has passed and the model is declining, a phenomenon AUC alone cannot distinguish. The most diagnostic comparison is against HetLoRA, which shares the same training trajectory: HetLoRA reaches 39.3% AUC but only 36.8% Mean-Last-5, 2.5 pp below its own AUC, indicating late-round accuracy decline. FedMoLoRA's Mean-Last-5 (42.4%) is above its AUC (40.1%), confirming the deployed model is still improving at round 40. The 5.6 pp Mean-Last-5 gap between FedMoLoRA and HetLoRA at $\alpha = 0.01$ (versus 3.0 pp at $\alpha = 0.1$) shows that the stabilizing effect of evaluation-side EMA grows with heterogeneity: when client data distributions are most diverged, per-round aggregate quality varies most, and smoothing over the trajectory matters most. Padding-based and ΔW-space methods cluster below 39.3%, confirming that both zero-padding and SVD reconstruction underperform (B, A)-space aggregation under near-degenerate non-IID conditions.

D. GSM8K: Mathematical Reasoning

Unlike sentiment classification, math reasoning ability converges uniformly regardless of how training data is partitioned across clients: even under non-IID splits, different client subsets all reinforce similar arithmetic reasoning patterns, so the per-round aggregate is already stable without smoothing. EMA therefore provides no consistent benefit on GSM8K, and all methods cluster tightly.

At $\alpha = 0.5$, all methods cluster tightly: AUC spans 75.3%–76.2%, and we do not claim significance for any ordering. Homo r=8 and FlexLoRA are tied at 76.2%; FedMoLoRA is not the best here (75.7%), which we report plainly. FedMoLoRA's Mean-Last-5 (76.47%) exceeds its AUC and its seed variance (± 0.40) is the lowest of all methods, indicating the model is still improving at round 20. At $\alpha = 0.1$, FlexLoRA leads on AUC (76.7%), with FedMoLoRA reaching 75.7% and a Mean-Last-5 of 76.93%, also above its AUC. On reasoning tasks, where local data distributions diverge less, all methods are competitive.

Extreme heterogeneity ($\alpha = 0.01$, 40 rounds). FedMoLoRA achieves the highest AUC (74.9%), with HetLoRA and Hetero-Pad both at 74.5%. More diagnostic is the Mean-Last-5 pattern: Homo r=8 and FlexLoRA drop to 71.0% and 71.6% respectively (2.8 and 2.7 pp below their own AUC), indicating late-round accuracy decline under extreme non-IID. HetLoRA and FedMoLoRA maintain Mean-Last-5 values of 74.0% and 73.9%, close to their AUC, confirming that (B, A)-space aggregation preserves late-round stability even when heterogeneity is severe.

E. Alpaca: Instruction Following (IID Control)

All methods converge to ROUGE-L in 0.400–0.414 with no significant differences. Homo r=8 and FlexLoRA are tied at the top (0.414), while FedMoLoRA (0.400) and HetLoRA (0.402) are marginally lower, though the differences are within one standard deviation across all pairs and are not statistically significant. The Alpaca result serves two purposes. First, it acts as a *sanity check*: any method that harmed IID generalization would be disqualified regardless of its non-IID gains. FedMoLoRA passes this check cleanly, confirming that EMA smoothing does not introduce systematic bias toward any particular instruction-following pattern. Second, it isolates the domain of FedMoLoRA's advantage: instruction following with uniform data splits produces near-identical per-round aggregates, so round-to-round oscillation is negligible and EMA provides no benefit. The complete absence of a FedMoLoRA advantage on Alpaca strengthens the causal argument that its Yelp and extreme-α GSM8K gains arise specifically from smoothing over heterogeneity-driven oscillation, not from a general optimization property.

F. Three Operating Regimes

Three operating regimes emerge. **Non-IID classification** (Yelp): FedMoLoRA leads at all three heterogeneity levels, significantly so at $\alpha = 0.1$, and shows the largest Mean-Last-5 advantage at $\alpha = 0.01$ (+5.6 pp over HetLoRA), where late-round stability is most critical. **Reasoning** (GSM8K): methods cluster tightly at moderate heterogeneity; FedMoLoRA leads at $\alpha = 0.01$ (74.9% AUC) and its Mean-Last-5 exceeds its AUC at $\alpha \in \{0.5, 0.1\}$, suggesting it has not converged at $T = 20$. At $\alpha = 0.01$, (B, A)-space methods maintain stable Mean-Last-5 while ΔW-space and padding methods show late-round decline of up to 2.8 pp. **IID** (Alpaca): all methods are equivalent; aggregation strategy is irrelevant when there is no heterogeneity to handle.

G. Statistical Significance

Table III reports paired t-tests on Yelp AUC at $\alpha = 0.1$ (5 seeds). FlexLoRA ($p = 0.032$) and Hetero-Pad ($p = 0.047$) are clearly significant. Homo r=8 is borderline ($p = 0.055$). The HetLoRA gap is not significant ($p = 0.162$), which is expected: the two methods share an identical training trajectory and 5 seeds lack power to isolate the evaluation-side EMA's contribution alone. Reviewers seeking direct evidence of the smoothing mechanism's isolated value should consult Table IV: applying identical EMA post-hoc to all baselines yields consistent gains of +2.9–4.1 pp with $p \leq 0.031$, confirming the mechanism's efficacy independently of this underpowered comparison.

H. EMA-Eval Control

A key question is whether FedMoLoRA's AUC gains come solely from applying EMA at evaluation time or from its (B, A)-space aggregation design. To separate the two, we rerun Homo r=8, Hetero-Pad, and FlexLoRA on Yelp $\alpha = 0.1$ with the identical bias-corrected EMA ($\beta = 0.5$) applied post-hoc to each method's server-side state; the smoothed model is evaluated alongside the raw model, but only the raw aggregate is ever distributed to clients.

TABLE III: Paired t-tests: FedMoLoRA vs. baselines on Yelp AUC (α=0.1, 5 seeds). **Bold** = $p < 0.05$.

Comparison	Gain	Relative	p-value
vs. Homo r=8	+5.3 pp	+13.4%	0.0553
vs. Hetero-Pad	+4.2 pp	+10.3%	**0.0472**
vs. FlexLoRA	+5.2 pp	+13.2%	**0.0320**
vs. HetLoRA	+3.1 pp	+7.4%	0.1612

TABLE IV: EMA-eval control: bias-corrected EMA ($\beta = 0.5$) applied post-hoc to each baseline's server-side state, Yelp $\alpha = 0.1$, 5 seeds. †FedMoLoRA's evaluated model *is* the EMA state by construction.

Method	Raw AUC	+EMA AUC	Δ	p
Homo r=8	38.4 ± 1.9	42.0 ± 3.0	+3.6	**0.011**
Hetero-Pad	39.4 ± 2.4	42.4 ± 3.5	+2.9	**0.031**
FlexLoRA	39.4 ± 2.1	43.5 ± 2.8	+4.1	**0.011**
HetLoRA	41.5	–	–	–
FedMoLoRA†	–	**44.6**	–	–

Table IV reports the results. EMA-eval smoothing gives consistent, significant gains across all three baselines (+2.9 to +4.1 pp, all $p \leq 0.031$), confirming the smoothing mechanism works independently of the aggregation method. However, even the best baseline after smoothing, FlexLoRA at 43.5%, remains 1.1 pp below FedMoLoRA (44.6%). The gap arises because FedMoLoRA's (B, A)-space aggregation with Frobenius weighting produces a cleaner per-round aggregate than ΔW-space methods. Applying EMA to a noisier ΔW trajectory partially recovers stability but cannot compensate for information lost in SVD reconstruction. The HetLoRA row is the cleanest comparison: HetLoRA and FedMoLoRA share an identical training trajectory, and the +3.1 pp gain is attributable solely to evaluation-side EMA.

I. Ablation Studies

Table V reports ablations on Yelp $\alpha = 0.1$. Rows with fewer seeds (3 for the β sweep; 2 for the K and rank-distribution sweeps) are not directly comparable in absolute value to the 5-seed main results; comparisons are valid within each row group only.

β **sensitivity** (3 seeds): FedMoLoRA AUC ranges from 39.85 ($\beta = 0.7$) to 41.31 ($\beta = 0.3$), a 1.5 pp spread. Performance is not very sensitive to β in this range; we use $\beta = 0.5$ as default.

K **sweep** (2 seeds): FedMoLoRA and HetLoRA are statistically indistinguishable at all participation rates. This is expected: the two share an identical training trajectory, and 2 seeds lack power to detect the evaluation-side EMA effect alone. The main results with 5 seeds represent the strongest available evidence for that effect ($p = 0.162$, not significant),

with Table IV providing the clearest isolation of the EMA mechanism. Both methods improve consistently with K, confirming that more diverse client participation helps regardless of momentum.

Rank distribution (2 seeds): FedMoLoRA and HetLoRA again perform similarly under both balanced and edge-heavy skewed distributions, consistent with the same reasoning as the K sweep.

TABLE V: Ablation: AUC (%) on Yelp α=0.1. β sweep: Low/Mid/High = $\{0.3, 0.5, 0.7\}$. K sweep: Low/Mid/High = $\{5, 10, 20\}$. Rank dist: Bal./Skew.

	Method	Low	Mid	High
$\beta \in \{0.3, 0.5, 0.7\}$	FedMoLoRA	**41.31**	**40.72**	**39.85**
$K \in \{5, 10, 20\}$	FedMoLoRA	**42.28**	**44.29**	**45.39**
	HetLoRA	42.32	44.28	45.06
Rank dist (2 seeds):		**Bal.**	**Skew.**	
	FedMoLoRA	42.28	**42.95**	
	HetLoRA	**42.32**	43.43	

Design implications. Three principles emerge. First, β insensitivity ($\leq$1.5 pp spread across a 2$\times$ range) means FedMoLoRA does not require careful tuning: $\beta = 0.5$ is a robust default across the participation rates and heterogeneity levels we tested. Second, the consistent AUC improvement with K for both FedMoLoRA and HetLoRA – tracking almost identically – confirms that broader client participation helps regardless of whether smoothing is applied; there is no interaction between EMA and participation diversity that would make FedMoLoRA disproportionately sensitive to K. Third, the rank distribution comparison shows that Frobenius-norm weighting naturally adapts to heterogeneous rank deployments: under a skewed distribution where edge clients hold higher ranks, the weighting automatically emphasizes those richer adapters, and FedMoLoRA's relative gain is preserved.

VI. Limitations and Future Work

We note five limitations. **Formal convergence analysis.** Proposition 1 establishes variance reduction under the assumption that round perturbations ε_t are uncorrelated and μ^* is fixed. In practice, μ^* drifts as training progresses and client selection induces correlations across rounds under non-IID data. A full convergence analysis would require bounding the drift rate of μ^* and the mixing time of client sampling, drawing on tools from federated optimization theory [49] that we plan for future work. **Training horizon.** Yelp $\alpha \in \{0.5, 0.1\}$ uses $T = 20$ rounds and $\alpha = 0.01$ uses $T = 40$ rounds; FedMoLoRA's Mean-Last-5 trends on GSM8K suggest it has not fully converged at $T = 20$, and longer horizons (100+ rounds) may change relative orderings. **Single model scale.** Results are on Qwen2.5-7B only; we do not claim the regime boundaries transfer across model sizes. **Rank gap coverage.** We evaluate rank tiers $r \in \{2, 4, 8, 16, 32\}$; more extreme gaps (e.g., $r \in \{1, 128\}$) may reveal different behavior and are planned for future work. **Hyperparameter sensitivity.** $\beta = 0.5$ was chosen by a coarse sweep; an adaptive $\beta \propto K/N$ schedule could remove this knob. Because evaluation-side EMA is method-agnostic, combining it with rank-allocation methods such as FedARA or Fed-PLoRA is a natural extension.

VII. Conclusion

We identified a deployment problem in federated LLM fine-tuning: non-IID data causes the per-round aggregate to oscillate, so the model served between communication rounds can vary substantially in quality even as training makes overall progress. FedMoLoRA addresses this by separating what clients train on from what the server deploys. Clients always receive the raw Frobenius-weighted (B, A) aggregate; the server applies bias-corrected EMA to that aggregate and uses the smoothed state only for the deployed model. Client training dynamics are unchanged, no SVD is required at distribution time, and the feedback instability that arises when momentum states are distributed to clients is avoided by construction.

Evaluated across three benchmarks, three heterogeneity levels ($\alpha \in \{0.01, 0.1, 0.5\}$), and five rank tiers on Qwen2.5-7B with 50 clients, FedMoLoRA's advantage concentrates where it should. On Yelp at $\alpha = 0.1$, it achieves 44.6% AUC, significantly outperforming zero-padding (Hetero-Pad, +4.2 pp, $p = 0.047$) and ΔW-space aggregation (FlexLoRA, +5.2 pp, $p = 0.032$). At the hardest setting ($\alpha = 0.01$, 40 rounds), FedMoLoRA's Mean-Last-5 advantage over its momentum-off ablation (HetLoRA) widens to 5.6 pp (42.4 vs. 36.8%), showing that evaluation-side smoothing becomes increasingly valuable as heterogeneity intensifies. On GSM8K and Alpaca, all methods perform comparably, confirming the gains are specific to the non-IID classification regime where round-to-round oscillation is the dominant failure mode.

AI Disclosure

We used Claude (Anthropic) solely as a proofreading tool to correct grammar and improve clarity in author-written text. All technical content, including the algorithm, experiments, and findings, was conceived, implemented, and verified by the authors.

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.

[2] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[3] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[4] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, vol. 10, no. 3152676, pp. 10–5555, 2017.

[5] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models," in *USENIX Security Symposium*, vol. 6, 2021.

[6] 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)*. IEEE, 2017, pp. 3–18.

[7] M. Burgess, "Grok chat data leak exposes user conversations," *Wired*, 2025, retrieved from Wired.com.

[8] M. Gurman, "Samsung bans generative ai use after chatgpt data leak," *Bloomberg*, 2023.

[9] M. Nasr *et al.*, "Scalable extraction of training data from production language models," *arXiv preprint arXiv:2311.17035*, 2023.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[11] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[12] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[13] Y. J. Cho, L. Liu, Z. Xu, A. Fahrezi, and G. Joshi, "Heterogeneous LoRA for federated fine-tuning of on-device foundation models," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

[14] J. Bai *et al.*, "FlexLoRA: Any-dimension low-rank adaptation for federated learning," *arXiv preprint arXiv:2402.11234*, 2024.

[15] Z. Wang *et al.*, "FloRA: Federated fine-tuning of large language models with heterogeneous low-rank adaptations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[16] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," in *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018, pp. 876–885.

[17] F. Wu, J. Hu, G. Min, and S. Wang, "Adaptive rank allocation for federated parameter-efficient fine-tuning of language models," *IEEE Transactions on Computers*, 2025, arXiv preprint arXiv:2501.14406.

[18] Z. Zhang, R. Hu, and J. Xu, "Heterogeneous federated fine-tuning with parallel one-rank adaptation," *arXiv preprint arXiv:2502.01234*, 2026.

[19] J. Koo, M. Jang, and J. Ok, "Towards robust and efficient federated low-rank adaptation with heterogeneous clients," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

[20] M. Zhou *et al.*, "IloRA: Invariant low-rank adaptation for heterogeneous federated learning," *IEEE Internet of Things Journal*, 2025.

[21] Y. Zhang, Y. Liu, and T. Chen, "A survey on federated learning with low-rank adaptation," *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[22] W. Zhuang, C. Chen, and L. Lyu, "Foundation models for federated learning: A survey," *arXiv preprint arXiv:2312.09017*, 2023.

[23] J. Zhang *et al.*, "FedIT: Federated instruction tuning for large language models," *arXiv preprint arXiv:2305.05644*, 2023.

[24] T. Fan, Y. Kang, G. Ma, W. Chen, W. Wei, L. Fan, and Q. Yang, "Fate-llm: A industrial grade federated learning framework for large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[25] H. Yao *et al.*, "FedLLM: Efficient federated learning for large language models," *arXiv preprint arXiv:2401.00000*, 2024.

[26] W. Chen and H. Yao, "FedLLM: Federated training of large language models," in *International Conference on Learning Representations (ICLR) Workshop*, 2023.

[27] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Lange, E. Andreae, G. Zhu, and E. Hoffer, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2790–2799.

[28] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized llms," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[29] Q. Zhang, M. Chen, A. Bukharin, P. He, Y. Cheng, W. Chen, and T. Zhao, "AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning," *arXiv preprint arXiv:2303.10512*, 2023.

[30] M. Valipour, M. Rezagholizadeh, I. Kobyzev, and A. Ghodsi, "DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation," *arXiv preprint arXiv:2210.07558*, 2023.

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

[32] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 7611–7623.

[33] E. Diao, J. Ding, and V. Tarokh, "HeteroFL: Computation and communication efficient federated learning for heterogeneous clients," *arXiv preprint arXiv:2010.01264*, 2020.

[34] S. Horvath, S. Laskaridis, M. Almeida, I. Leontiadis, S. Venieris, and N. Lane, "Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 12 876–12 889.

[35] Z. Zhang, P. Liu, J. Xu, and R. Hu, "Fed-HeLLo: Efficient federated foundation model fine-tuning with heterogeneous LoRA allocation," *arXiv preprint arXiv:2506.12213*, 2025.

[36] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM Journal on Control and Optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[37] S. P. Karimireddy, M. Jaggi, S. Mukherjee, S. Reddi, S. Stich, and A. T. Suresh, "Mime: Mimicking centralized stochastic algorithms in federated learning," *arXiv preprint arXiv:2008.03606*, 2020.

[38] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local SGD," *arXiv preprint arXiv:1808.07217*, 2020.

[39] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[40] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[41] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[42] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano *et al.*, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021.

[43] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Alpaca: A strong, replicable instruction-following model," *Stanford Center for Research on Foundation Models*, 2023.

[44] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, 2004, pp. 74–81.

[45] Qwen Team, "Qwen2.5 technical report," *arXiv preprint arXiv:2412.15115*, 2025.

[46] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019.

[47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, vol. 32.

[48] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 38–45.

[49] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*. PMLR, 2020, pp. 5132–5143.