SEEDFLOOD: A STEP TOWARD SCALABLE DECENTRALIZED FINE-TUNING OF LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

This work presents SeedFlood, a new approach to decentralized LLM fine-tuning designed to scale across large models, large collaborations, and complex network topologies while achieving global consensus with negligible communication overhead. Traditional methods suffer from high communication costs that grow with model size, while information decay over network hops renders global consensus inefficient. SeedFlood takes a significant departure from these practices by exploiting the seed-reconstructible structure of zeroth-order gradients and effectively making the messages to transmit near-zero in size, allowing them to be flooded to every client in the network, and thereby enhancing scalability of decentralized training. Consequently, SeedFlood enables training in regimes previously considered impractical, such as billion-parameter scale models or distributed across hundred of clients. Our experiments on decentralized LLM fine-tuning demonstrate that SeedFlood consistently outperforms the standard zeroth-order baselines in both communication efficiency and generalization performance, and even achieves results comparable to first-order gossip-based methods in large-scale settings, while requiring orders-of-magnitude less communication cost. We also provide theoretical analysis to formalize that SeedFlood avoids topology-dependent consensus terms in the convergence bound while retaining the acceleration enabled by increased client participation.

1 INTRODUCTION

Decentralized collaborative fine-tuning has recently emerged as a promising approach for adapting large language models under limited local compute and memory resources, enabling participants to collectively build a single model beyond what they could obtain individually [59, 37, 31, 16, 54]. To further advance this paradigm, recent efforts have begun to exploit the low intrinsic dimensionality of fine-tuning through techniques such as LoRA [21] and zeroth-order optimization [34, 7], and to explore various combinations that balance communication and optimization costs to identify configurations that better support scalable decentralized collaboration [16, 54], demonstrating promising communication–performance trade-offs.

However, while reduced communication loads have broadened the scalability of decentralized training, comparatively little attention has been paid to the deeper interplay beyond simply combining existing components: how optimization structure should shape communication design, and how communication strategies should, in turn, reshape optimization. This gap is particularly consequential given the longstanding limitations imposed by conventional treatments of communication constraints when designing decentralized optimization methods that scale to large and complex collaborations [25, 52, 28]. Moreover, just as advances in fine-tuning have broadened our understanding of what can be achieved through more efficient optimization, progress in decentralized fine-tuning may similarly reveal new principles for how decentralized training itself can be designed and optimized.

Motivated by this, we develop SeedFlood around a simple yet powerful recent idea: zeroth-order gradient estimates can be represented using only two scalars—a seed that reconstructs the random perturbation and a scaling coefficient—reducing high-dimensional gradient information to a nearly negligible payload independent of model scale [59, 37, 31]. Realizing this as a consistent end-to-end protocol calls for a new decentralized training framework that naturally accommodates this representation, since forcing it into the usual gossip-averaging paradigm generally destroys the very

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.

simplicity that makes seed–scalar communication attractive in the first place. Instead, we argue that the resulting compact payload makes flooding [29] a natural fit, while also enabling exact network-wide consensus by preserving individual contributions intact rather than diluting them through successive gossip averages [26], a major source of slow consensus at large collaboration scales in traditional decentralized frameworks.

Additionally, for the overall framework to scale effectively, local computation must remain balanced with the now-reduced communication burden: reconstructing high-dimensional perturbations from all other participants causes the per-node workload to grow linearly with the number of clients, eventually creating a severe computational bottleneck at large collaboration scales, particularly on modern accelerators. To resolve this, SeedFlood couples flooding with a novel aggregation-aware perturbation structure, constraining optimization to a periodically refreshed shared low-rank subspace whose size is independent of collaboration scale, while remaining sufficiently expressive in practice and allowing each client to aggregate updates directly in the compressed space before reconstructing them in the original parameter space.

Together, these designs enable SeedFlood to scale with both model size and collaboration scale, addressing three distinct challenges: model-size-dependent communication overhead (Section 3.1), consensus degradation in larger decentralized collaborations (Section 3.2), and the increasing local computation required to process flooded updates (Section 3.3). Empirically, SeedFlood delivers favorable efficiency–performance trade-offs and, at larger collaboration scales, consistently outperforms conventional framework designs—including diverse communication–optimization trade-offs—in both communication efficiency and generalization performance. Furthermore, our theoretical analysis establishes convergence bounds without topology-dependent consensus terms, while retaining the convergence acceleration enabled by increased client participation.

We highlight our contributions in the following:

- SeedFlood framework: A new decentralized training foundation that achieves topology-invariant consensus with minimal, constant-size, model-independent communication overhead.
- Scalable Aggregator: A scalable gradient aggregator that operates within a shared low-rank subspace, enabling an extreme speedup in aggregating large volumes of flooded updates.
- Theoretical guarantees: Topology-independent convergence bounds that retain the benefits of increased client participation through exact gradient aggregation.
- Empirical validation: Extensive evaluations scaling up to billion-parameter LLMs and networks of 128 clients, demonstrating a strong performance-cost trade-off against first-order gossip baselines, particularly as collaboration scale increases.

2 Background

2.1 Decentralized Training

Decentralized training aims to learn a common model from resources and data distributed across clients that communicate only with connected neighbors, without directly sharing private raw data [32, 45]. Formally, clients indexed by $\mathcal{V} = \{1, \dots, n\}$ communicate over a connected, undirected, static network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and update local parameters $\theta_i \in \mathbb{R}^d$ to solve

$$\min_{\theta_1,\dots,\theta_n} \sum_{i\in\mathcal{V}} f_i(\theta_i) \quad \text{s.t.} \quad \theta_i = \theta_j, \ \ \forall (i,j) \in \mathcal{E}, \tag{1}$$

where $f_i : \mathbb{R}^d \to \mathbb{R}$ is the local objective of client i. The core challenge is therefore to jointly design what information to exchange, how to exchange it, and how to perform local computation so that the local models rapidly reach consensus on a model that performs well across all clients as the collaboration scale n grows.

The existing paradigm has largely settled on a combination of gradient compression and gossip averaging [13]: techniques such as quantization [2], sparsification [44], and low-rank compression [51], often coupled with error-feedback or communication-aware mechanisms [25, 49, 63, 23, 1], reduce each round's payload. This information is usually exchanged via gossip averaging [32, 26, 45, 27], which recursively mixes neighboring updates into a single fixed-size message, *e.g.*,

$m_i \leftarrow \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} m_j$, allowing information to propagate across the network without forwarding an ever-growing collection of raw payloads. Although this substantially reduces communication, it still offers limited relief from payloads that scale with model dimensionality as model size grows; hence, the usual practice is to perform only a few rounds of gossip mixing over parameters θ per local optimization step, trading communication cost against the degree of consensus attained for global optimization.

However, local updates continually introduce model disagreement, while limited gossip mixing leaves residual consensus errors that can impair optimization [28]. This becomes particularly problematic in poorly connected or large network graphs, where achieving consensus requires more communication [52, 26, 28]. At LLM scale, exchanging high-dimensional models remains costly, while compression can further hinder consensus by introducing additional errors [32, 25, 51]. Thus, despite recent progress in collaborative LLM training [5, 11, 39], robust network-wide consensus at massive scale remains difficult to achieve without prohibitive communication [59, 16, 54].

2.2 Zeroth-order optimization

Zeroth-order optimization provides a gradient-free alternative to backpropagation and has shown strong effectiveness for modern language model fine-tuning [34, 7]. It estimates descent directions using only function evaluations [43], typically through a symmetric two-point estimator along a Gaussian perturbation $z \sim \mathcal{N}(0, I_d)$:

$$g = \alpha z = \frac{f(\theta + \epsilon z) - f(\theta - \epsilon z)}{2\epsilon} z, \tag{2}$$

where $\epsilon > 0$ is the perturbation scale, with averaging over multiple perturbations progressively improving the gradient estimate [15, 35]. Prior work has substituted such zeroth-order estimates for local gradients within standard decentralized templates [18, 50, 57, 22], but consequently inherits the consensus limitations of their first-order counterparts. Moreover, zeroth-order decentralized fine-tuning of LLMs over arbitrary network topologies remains largely unexplored.

Taken together, this motivates us to ask whether decentralized fine-tuning can instead achieve global consensus without making collaboration scale, network topology, or model dimensionality the limiting factor.

3 Method

We propose SeedFlood through three components that address communication overhead, consensus degradation, and the increasing local computation. The complete algorithm is provided in Appendix A.

3.1 Seed-scalar based dimension-free communication

Naively, communicating a zeroth-order estimate requires transmitting either the scalar directional derivative α together with the d-dimensional perturbation z, or simply the resulting d-dimensional estimate αz. In practice, however, z is typically generated deterministically from a compact seed s using a pseudorandom number generator $\texttt{PRNG} : \mathbb{R} \to \mathbb{R}^d$:

$$z = \texttt{PRNG}(s) \sim \mathcal{N}(0, I_d),$$

a standard approach for generating sufficiently diverse perturbations in computational statistics and machine learning [34, 37]. Consequently, any client using the same PRNG can exactly regenerate another client's perturbation from its seed:

$$\forall z_i := \texttt{PRNG}_i(s), \qquad \texttt{PRNG}_j = \texttt{PRNG}_i \implies \texttt{PRNG}_j(s) = z_i.$$

Thus, once the PRNG is shared in advance, the full zeroth-order estimate can be reconstructed from only the seed–scalar pair (s, α), effectively removing model dimensionality as the dominant communication bottleneck in centralized federated learning [37].

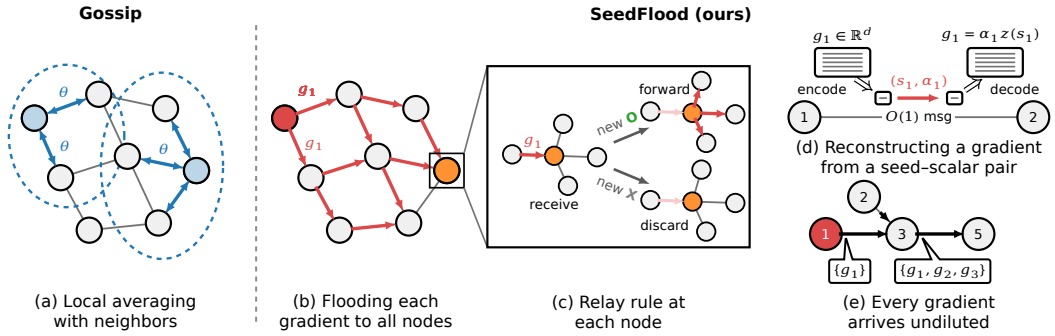

Figure 1: Comparison of gossip averaging and SeedFlood. (a) Gossip reaches consensus through repeated neighbor-wise model averaging. (b) SeedFlood instead floods every client's gradient to the entire network, so that each client updates with the average of all of them. (c) Flooding remains duplicate-free because each client forwards a message exactly once, upon its first reception. (d) Each zeroth-order gradient can be communicated using only a random seed and a scalar coefficient. (e) This lossless flooding enables rapid multi-hop propagation and exact consensus without the information dilution inherent in gossip averaging.

3.2 Flooding for Scalable Global Consensus

However, conventional gossip averaging is not well suited to seed–scalar communication for two reasons. First, an arbitrary message—such as gossip averaged—models generally cannot be represented by a single seed–scalar pair without solving an impractical inverse-search problem over the chosen PRNG, *i.e.*, $\arg\min_{s,\alpha} \|\alpha \cdot \mathtt{PRNG}(s) - \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} \theta_j\|$. Alternative approaches also compromise the simplicity or efficiency of seed–scalar communication (Appendix C). Second, even if such encoding were feasible, finite-round gossip mixing would still leave topology-dependent consensus errors that can degrade optimization, making its limitations difficult to justify when individual updates can already be communicated compactly [52].

We therefore propose adopting flooding [29] as the mechanism for seed-based communication in decentralized training, allowing compact seed–scalar representations to be disseminated intact across the network while naturally enabling exact global consensus. As illustrated in Figure 1b, each communication round consists of multiple *flooding iterations*, in which clients relay messages received in the preceding iteration. Upon receiving a previously unseen gradient, a client applies the corresponding update and forwards it to each neighbor only once (Figure 1c). Repeating this process for at least D flooding iterations, where D is the network diameter, ensures that every zeroth-order gradient g_i reaches every client. Since each gradient is transmitted only as a seed–scalar pair (Figure 1d), all clients receive the complete set of updates without the hop-wise dilution of gossip (Figure 1e), thereby applying identical updates and reaching exact consensus with communication that remains independent of model dimensionality.

3.3 Decoupling Gradient Aggregation from Numbers of Clients

While SeedFlood enables exact global consensus, local computation can still become a scalability bottleneck as collaboration scale grows.

To elaborate, naively aggregating n seed-encoded gradients into a model update requires regenerating and accumulating a dense perturbation $Z_i \in \mathbb{R}^{d_1\times d_2}$, with $d = d_1 d_2$, for each message, incurring $O(nd)$ computation that couples number of clients n with model dimensionality d. Beyond this linear complexity, how well the computation maps to modern accelerators is also an important practical consideration that can become a dominant factor under the large-scale workloads typical of modern LLMs. Here, the axpy-style accumulation of fully materialized perturbations is relatively low in arithmetic intensity and typically memory-bandwidth-bound compared to compute-dense matrix multiplication such as GEMM, adding an additional penalty on top of the linear $O(nd)$ scaling.

To address both aspects, we propose a particular solution that restricts perturbations to a finite set of rank-one directions $\{\, u_a v_b = U\mathbb{1}_{a,b}V^\top \mid (a,b) \in \mathbb{N}^2_{\le r} \,\}$ where each direction is specified by

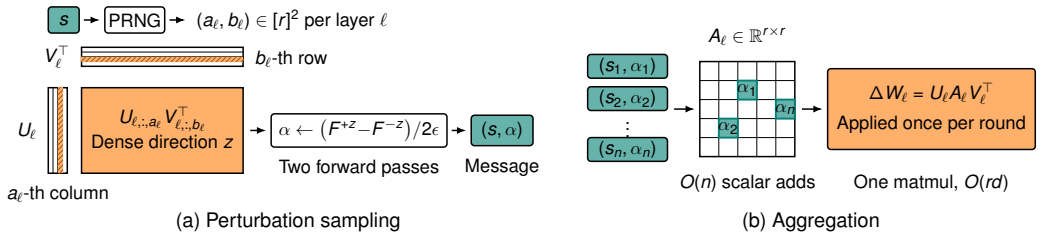

Figure 2: Subspace coordinate-wise perturbation of SeedFlood. (a) Instead of drawing perturbations from Gaussian distribution, SeedFlood prepares shared bases U_l, V_l, samples a random coordinate pair (a_l, b_l) at each iteration, which selects the dense rank-one direction $U_{\ell,:,a_\ell} V_{\ell,:,b_\ell}^\top$ as the perturbation. (b) The n received scalars are accumulated into A_ℓ and applied by a single matrix product.

a pair of low-dimensional indices (a, b), equivalently by a one-hot indexing matrix $\mathbb{1}_{a,b} \in \mathbb{R}^{r \times r}$, over periodically refreshed, globally shared random matrices $U \in \mathbb{R}^{d_1 \times r}$ and $V \in \mathbb{R}^{d_2 \times r}$. Across l layers, this yields r^2 candidate directions per layer, as illustrated in Figure 2, and hence r^{2l} possible model-wide perturbations z.

This construction does not fully materialize a d-dimensional perturbation for each message; instead, it first accumulates the n received scalars into their corresponding entries of a coefficient matrix $A = \sum_{i=1}^n \alpha_i \mathbb{1}_{a_i,b_i} \in \mathbb{R}^{r \times r}$, and only afterward materializes the aggregate update in parameter space through a single matrix product $UAV^\top$ (Figure 2b):

$$\underbrace{\sum_{i=1}^n \alpha_i Z_i}_{\text{Full matrix aggregation: O}(nd)} \quad \Longrightarrow \quad \sum_{i=1}^n \alpha_i U \mathbb{1}_{a_i,b_i} V^\top = U \underbrace{\left(\sum_{i=1}^n \alpha_i \mathbb{1}_{a_i,b_i} \right)}_{\text{Scalar aggregation: } O(n)} V^\top.$$

This reduces the aggregation cost from $O(nd)$ to $O(n + rd)$, decoupling collaboration scale from model dimensionality, while leaving a single dense matrix multiplication as the dominant operation—a workload substantially better suited to modern GPUs than repeated `axpy`-style updates. In our profiling run, applying thousands of received messages is up to $\sim 2000\times$ faster than naive reconstruction (Figure 7; implementation details in Appendix D).

Two practical considerations remain. First, the additional indices and shared matrices need not introduce substantial communication overhead: the $2l$ indices at each iteration and the periodically refreshed per-layer U, V, are likewise generated by a PRNG and can therefore be determined, communicated, and reconstructed from a single seed (Figure 2a). Second, confining to lr^2 directions within a refresh period may restrict the optimization dynamics despite periodic refreshing. We analyze this effect formally in Section 4, including the roles of the subspace size r and refresh period τ. Appendix B.2 empirically validates the runtime efficiency and task performance, including its match to unrestricted Gaussian estimation (Table 2).

4 Theoretical Analysis

In this section, we provide the convergence analysis of SeedFlood. The convergence of the underlying subspace coordinate-wise estimator is established separately in Proposition 2 in Appendix E. Full proofs, together with the formal statements of all assumptions (Assumptions 1–4), are provided in Appendices E and F.

Setup. Consider n clients with local objectives $f_i(X) := \mathbb{E}_{\xi \sim \mathcal{D}_i}[F_i(X; \xi)]$ and global objective $f(X) := \frac{1}{n} \sum_{i=1}^n f_i(X)$, where $X \in \mathbb{R}^{d_1 \times d_2}$. At period k, all clients share $U_k \in \mathbb{R}^{d_1 \times r_1}$ and $V_k \in \mathbb{R}^{d_2 \times r_2}$ for τ inner steps. Let $X^{(k)} := X_{k,0} = X_{k\tau}$, $B_{k,0} = 0$, and $q := r_1 r_2$. At inner step s, each client i independently samples $J_{i,k,s} \sim \text{Unif}\{1, \ldots, q\}$, sets $C_{i,k,s} = E_{J_{i,k,s}}$, and computes

$$a_{i,k,s} := \frac{F_i(X^{(k)} + U_k(B_{k,s} + \epsilon C_{i,k,s})V_k^\top; \xi_{i,k,s}) - F_i(X^{(k)} + U_k(B_{k,s} - \epsilon C_{i,k,s})V_k^\top; \xi_{i,k,s})}{2\epsilon},$$

$$B_{k,s+1} = B_{k,s} - \frac{\eta}{n}\sum_{i=1}^{n} a_{i,k,s} C_{i,k,s}, \qquad X_{k,s} = X^{(k)} + U_k B_{k,s} V_k^\top. \tag{3}$$

Under exact flooding, decentralized dynamics coincide with centralized data-parallel zeroth-order optimization over the shared subspace.

Under Assumption 4, each local objective satisfies the same smoothness and noise conditions, with minibatch variance at most σ^2/b, and $\frac{1}{n}\sum_i \|\nabla f_i(X) - \nabla f(X)\|_F^2 \le \zeta^2$. We further let $T := K\tau$, $\widetilde{L} := L\rho_U^2\rho_V^2$, and $\widetilde{\zeta}^2 := \rho_U^2\rho_V^2\zeta^2$.

Theorem 1 (Convergence of SeedFlood under exact flooding)**.** *Under Assumptions 2, 3, and 4 with exact flooding, let* $\Lambda_{q,n}(\tau) := \tau^2/q^2 + \tau/(nq)$. *If* $\eta \le \left(\widetilde{L}\sqrt{32\,\Lambda_{q,n}(\tau)}\right)^{-1}$ *and* $c_2\eta\widetilde{L}(\tau/q + 1/n) \le c_1/2$, *then*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla f(X_{k\tau})\|_F^2 \;\le\; \frac{2\Delta_0}{c_1\eta T} + a_\sigma\widetilde{L}\eta\frac{\sigma^2}{nb} + a_\zeta\widetilde{L}\eta\frac{\widetilde{\zeta}^2}{qn} + a_\epsilon\frac{c_\delta^2}{q}\left[1 + \widetilde{L}\eta\left(\frac{\tau}{q} + \frac{1}{n}\right)\right]\epsilon^2,$$

where $\widetilde{\zeta}^2 := \rho_U^2\rho_V^2\zeta^2$, $c_\delta := \sqrt{q}\,\widetilde{L}/2$, *and* $a_\sigma, a_\zeta, a_\epsilon > 0$ *are universal constants.*

The theorem highlights two key scaling properties of SeedFlood. First, the bound contains no topology-dependent factor such as a spectral gap: exact flooding makes the decentralized dynamics identical to centralized data-parallel optimization. Second, larger n reduces both the stochastic-noise term $\sigma^2/(nb)$ and the heterogeneity term $\widetilde{\zeta}^2/(qn)$, while relaxing the stepsize condition through $\Lambda_{q,n}(\tau)$. The remaining trade-off arises from subspace reuse: large τ relative to q amplifies the finite-difference error, making the refresh period a practically important hyperparameter.

5 Experiments

Our experiments examine two consequences of SeedFlood's design: whether dimension-free exact aggregation improves the communication–performance trade-off at LLM scale, and whether its advantage persists or grows with increasing client participation.

5.1 Experimental setup

We compare SeedFlood against nine baselines spanning three decentralized optimization methods—DSGD [32], communication-efficient ChocoSGD [25], and zeroth-order DZSGD [50], which isolates the benefit of flooding-based aggregation over naive gossip averaging under the same zeroth-order setting—each combined with FFT, LoRA [21], and FFA-LoRA [46], where only trainable parameters are communicated. To evaluate across diverse tasks and model families, we use a subset of SuperGLUE [53] and SST-2 [42] with OPT-1.3B [61], and additionally SQuAD [38] and DROP [12] with Qwen3-1.7B [56] to cover generative tasks and a more recent backbone. We report task performance using accuracy or F1 score, together with total communication volume per edge over training. To avoid overstating the communication cost of faster-converging first-order methods, we train first-order baselines for 500 local iterations and zeroth-order methods for $5,000$ which we observed to be sufficient for convergence without overfitting; full configurations and dataset details are provided in Appendix G.

5.2 Communication–performance trade-off at LLM scale

Across eight task–model pairs on a 16-client ring network, SeedFlood delivers a markedly better communication–performance trade-off within zeroth-order training (Figure 3). Compared with the zeroth-order gossip baselines (e.g., DZSGD and its LoRA/FFA-LoRA variants), SeedFlood transmits 10^4–$10^7\times$ fewer bytes, yet achieves higher task performance on nearly all tasks. By replacing

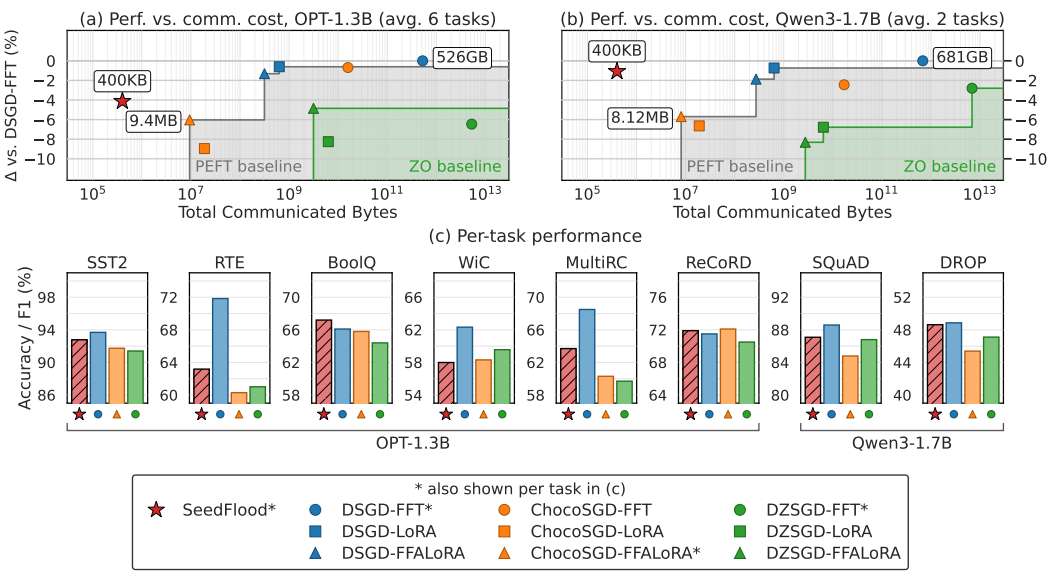

Figure 3: Performance–communication trade-off on a 16-client ring network. (a,b) SeedFlood versus 9 decentralized fine-tuning baselines on OPT-1.3B and Qwen3-1.7B-Base; performance is averaged over tasks relative to DSGD-FFT. (c) Per-task performance of SeedFlood and representative baselines at the communication-cost extremes: DSGD-FFT, DZSGD-FFT, and ChocoSGD-FFALoRA. SeedFlood outperforms zeroth-order and the most communication-efficient first-order PEFT baselines with model-size-independent communication and 10–$10^7\times$ fewer communicated bytes.

multi-hop gossip with flooding and communicating only seeds and scalars, SeedFlood reduces the total communication to just 400 KB over the entire training run.

Compared with the first-order baselines, DSGD-FFT provides the strongest but most communication-intensive reference, achieving the best performance in most settings. However, exchanging dense gradients costs 526 GB per edge for OPT-1.3B, corresponding to roughly half a day of transmission over a 100 Mbps link. Against this unconstrained baseline, SeedFlood incurs a 4% performance gap on OPT-1.3B and as little as 1.1% on Qwen3-1.7B, while reducing communication by $10^6\times$. In contrast, when FO methods are pushed toward a communication-efficient regime, their performance falls below SeedFlood: Choco-FFA, the most communication-efficient FO baseline at 9.4 MB, underperforms SeedFlood by 2–3%p.

Overall, SeedFlood combines near-zero, model-size-independent communication with performance close to unconstrained first-order training, while outperforming every baseline operating under a comparable communication budget. As we show next, this trade-off is specific to small networks: as the number of clients grows, gossip-based methods degrade while SeedFlood does not, turning the trade-off in SeedFlood's favor. Additional meshgrid topology results are provided in Appendix B.1.

5.3 Performance Scaling with Client Participation

Increasing client participation creates opposing effects in decentralized training. More clients provide additional optimization signal, but make global consensus increasingly difficult for gossip-based methods. We examine how SeedFlood behaves under this scaling regime.

We scale the network from 16 to 128 clients while keeping the total training set fixed and evenly partitioning it across clients. For computational feasibility, we evaluate four representative tasks with PEFT baselines.

Figure 4 reveals contrasting scaling behaviors. While gossip-based baselines degrade as the number of clients increases, SeedFlood maintains or even improves its performance. This suggests that exact aggregation not only removes topology-dependent consensus errors but also allows larger collaborations to benefit from the reduced variance of zeroth-order estimates.

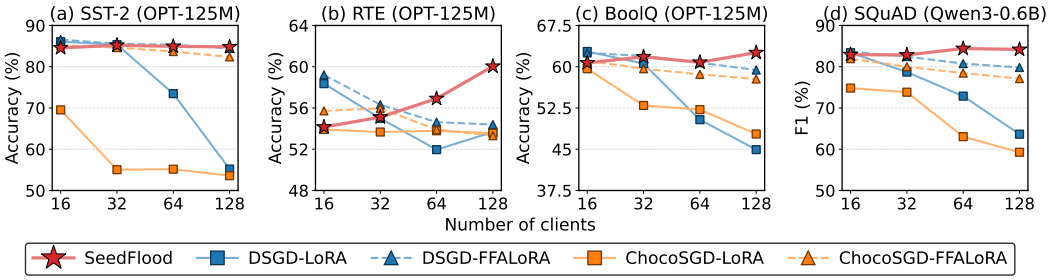

Figure 4: Performance scaling with increasing network size in a ring topology. Unlike first-order gossip baselines, SeedFlood maintains stable or improving performance as the number of clients increases, and surpasses the best-performing gossip baseline at larger client counts.

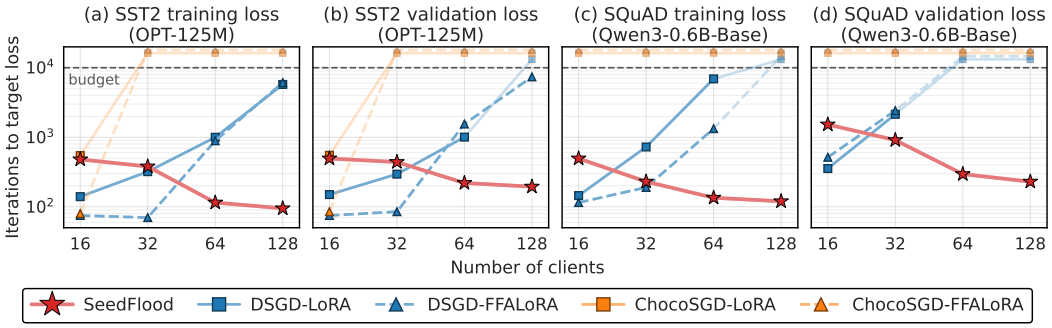

Figure 5: Iterations required to reach the target loss across 16–128 clients, measured on training (a, c) and validation (b, d) sets. While SeedFlood initially requires more iterations than convergent first-order gossip baselines, its iteration count decreases with increasing client participation. In contrast, first-order gossip baselines require more iterations or fail to reach the target within the 10,000-iteration budget (dashed horizontal line).

5.4 Iteration Scaling with Client Participation

At small scale, SeedFlood requires more iterations than first-order methods, raising the question of whether its communication advantage comes with a persistent optimization cost. We therefore examine how this iteration gap evolves with increasing client participation by measuring the number of iterations required to reach a target loss under the same setup as Section 5.3. The target is set to the best loss attained by SeedFlood in the 16-client setting; runs are capped at 10,000 iterations, and further protocol details are given in Appendix G.4.

The iteration narrows sharply as client participation increases (Figure 5). At 16 clients, SeedFlood requires roughly 3–5$\times$ more iterations than the first-order PEFT baselines to reach the same loss, confirming the expected convergence disadvantage of zeroth-order optimization. However, this gap narrows as the number of clients increases. For SeedFlood, the required iteration count decreases with more clients, as aggregating more perturbations in parallel reduces the variance of the zeroth-order estimate, consistent with the $1/n$ noise reduction in Theorem 1. In contrast, the first-order gossip baselines require increasingly more iterations as the number of clients grows and several configurations fail to reach the target. Consequently, the initial iteration gap between SeedFlood and the gossip baselines narrows substantially as the number of clients increases, and a crossover is observed under the target loss and tasks considered here.

6 Limitation

SeedFlood for general decentralized learning. While SeedFlood is not inherently restricted to fine-tuning and could, in principle, extend to more general decentralized learning tasks, several considerations make us cautious about drawing concrete conclusions in such settings. Notably, it would largely inherit the limitations of zeroth-order optimization, whose extension to broader

regimes such as pretraining remains an active area of research [6, 3], despite substantial progress in optimization efficiency [33, 14, 41, 7, 58, 62, 10]. However, decentralized training can be limited more directly by slow consensus than by the choice of zeroth- versus first-order local optimization, especially at large collaboration scales [28, 26, 19]; even with first-order updates, gossip-averaged models may require substantially more iterations to improve, extending into regimes where fully synchronized training has already passed its best test performance and begun to overfit [55, 65]. This suggests that the exact global consensus enabled by SeedFlood may touch on a critical bottleneck in decentralized learning, providing a basis for broader analysis and potentially inspiring approaches to narrow the gap with centralized learning in more general settings (see Appendix B.4 for additional small-scale pretraining results).

Dimension-free exact consensus beyond zeroth-order optimization. Establishing such a representation beyond zeroth-order optimization would be highly impactful: it could decouple seed–scalar communication from the optimization procedure itself, enabling more effective local optimization methods while preserving exact, dimension-free global consensus. Although the difficulty discussed in Section 3.2 suggests that such a representation is hard to retain for arbitrary updates, a natural compromise is to restrict optimization to a sufficiently expressive family of seeded updates and project first-order updates onto this family, as recently explored through shared-randomness projection [40]. Related work further suggests that the seeded family itself can be made more optimization-aware through selective seed dictionaries [37] or structured low-dimensional perturbations that better capture first-order gradients [7, 58].

Obtaining further scalability While flooding can reach exact consensus in finite time on any connected topology by propagating updates without dilution, its latency still scales with graph diameter and may become limiting on sufficiently large networks as each update must traverse more hops to reach all clients. One direct way to mitigate this cost is delayed flooding, where updates traverse only a limited number of hops per round rather than the full graph diameter, reducing per-round propagation latency while allowing global propagation to occur over multiple rounds. Beyond delayed flooding, system-level mechanisms such as topology-aware relay structures, hierarchical or tree-based routing, message batching, and pipelining across hops or local computation could further reduce effective propagation latency [29, 52, 48]. These design choices inevitably trade communication latency against the frequency and freshness of globally incorporated updates, and their impact on end-to-end learning efficiency warrants further study. We provide partial empirical results for delayed flooding in Appendix B.3, showing that moderate propagation delays can still preserve performance comparable to full flooding.

7 Conclusion

Decentralized training has long been hindered by two fundamental barriers: prohibitive communication overhead and the fragility of global consensus. SeedFlood bypasses these bottlenecks by leveraging the seed-reconstructible structure of zeroth-order updates alongside a robust flooding mechanism, and substantially reduces the communication overhead of decentralized LLM fine-tuning, while maintaining robust performance even in large and sparsely connected peer-to-peer networks. Our results further demonstrate that decentralized training and zeroth-order optimization can complement each other's weaknesses: communication-efficient zeroth-order updates with flooding alleviate the communication and consensus bottleneck of decentralized training, while large-scale collaboration helps offset the optimization inefficiency of zeroth-order optimization. These results suggest that SeedFlood provides a practical path toward decentralized LLM fine-tuning in low-bandwidth collaborative environments.

AI use statement

In this work, we used generative AI tools to assist with the proofs of our theoretical analysis, specifically to check the completed proofs for errors post hoc and to identify loose steps in the convergence bounds; and to assist with translation. We have not used generative AI tools to provide critical ingredients for proving mathematical claims, develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement the core framework of the method, or interpret

results. Generating synthetic datasets, cleaning or reformatting datasets, and conducting qualitative or thematic data analysis were not applicable to this work. Additionally, we used generative AI tools for editing software code (organizing variables, cleaning code, and helping interpret code written by co-authors), but not for implementing the method; for editing the paper to improve readability; for identifying relevant literature; and for creating or modifying Figures 1 and 2. We have reviewed all AI-assisted work. All theorem statements were formulated by the authors, every AI-assisted proof step, including the Grönwall-based argument, was verified line by line by the authors, all cited works were checked against the original sources, and all code edited with the aid of generative AI was reviewed by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement

This work does not involve human subjects, the collection or release of personal or sensitive data, or deployment in safety-critical settings. We use publicly available models and benchmark datasets and are not aware of specific ethical concerns beyond those generally associated with large language model research.

Reproducibility statement

We provide the SeedFlood algorithm in Section 3 and Appendix A. Experimental configurations, including datasets, hyperparameters, LoRA settings, iteration-budget protocols, and compute resources, are detailed in Section 5.1 and Appendix G, with implementation and runtime details in Appendix D and task-level numerical results in Appendix H. The assumptions and complete proofs for our theoretical results are provided in Appendices E and F. The source code will be made publicly available upon acceptance.

References

[1] Sai Aparna Aketi, Abolfazl Hashemi, and Kaushik Roy. Adagossip: Adaptive consensus step-size for decentralized deep learning with communication compression. *arXiv preprint arXiv:2404.05919*, 2024.

[2] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30, 2017.

[3] Nathan Allaire, Mahsa Ghazvini Nejad, Sébastien Le Digabel, and Vahid Partovi Nia. Zeroth order optimization for pretraining language models. In *ICPRAM*, pp. 113–121, 2025.

[4] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. *TAC*, 7(8):1, 2009.

[5] Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Max Ryabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. Petals: Collaborative inference and fine-tuning of large models. *arXiv preprint arXiv:2209.01188*, 2022.

[6] Aochuan Chen, Yimeng Zhang, Jinghan Jia, James Diffenderfer, Jiancheng Liu, Konstantinos Parasyris, Yihua Zhang, Zheng Zhang, Bhavya Kailkhura, and Sijia Liu. Deepzero: Scaling up zeroth-order optimization for deep model training. *arXiv preprint arXiv:2310.02025*, 2023.

[7] Yiming Chen, Yuan Zhang, Liyuan Cao, Kun Yuan, and Zaiwen Wen. Enhancing zeroth-order fine-tuning for language models with low-rank structures. *arXiv preprint arXiv:2410.07698*, 2024.

[8] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[9] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pp. 177–190. Springer, 2005.

[10] Sizhe Dang, yangyang Guo, Yanjun Zhao, Xiaodong Zheng, Guang Dai, Ivor Tsang, and Haishan Ye. Fzoo: Fast zeroth-order optimizer for fine-tuning large language models towards adam-scale speed. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 113760–113792, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/b8d4c82b365d9361863000f8d1e86f1b-Paper-Conference.pdf.

[11] Michael Diskin, Alexey Bukhtiyarov, Max Ryabinin, Lucile Saulnier, Anton Sinitsin, Dmitry Popov, Dmitry V Pyrkin, Maxim Kashirin, Alexander Borzunov, Albert Villanova del Moral, et al. Distributed deep learning in open collaborations. *Advances in Neural Information Processing Systems*, 34:7879–7897, 2021.

[12] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, 2019.

[13] Edoardo Gabrielli, Giovanni Pica, and Gabriele Tolomei. A survey on decentralized federated learning. *arXiv preprint arXiv:2308.04604*, 2023.

[14] Tanmay Gautam, Youngsuk Park, Hao Zhou, Parameswaran Raman, and Wooseok Ha. Variance-reduced zeroth-order methods for fine-tuning language models. *arXiv preprint arXiv:2404.08080*, 2024.

[15] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4):2341–2368, 2013.

[16] Sajjad Ghiasvand, Mahnoosh Alizadeh, and Ramtin Pedarsani. Decentralized low-rank fine-tuning of large language models. *arXiv preprint arXiv:2501.15361*, 2025.

[17] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pp. 1–9, 2007.

[18] Davood Hajinezhad, Mingyi Hong, and Alfredo Garcia. Zone: Zeroth-order nonconvex multiagent optimization over networks. *IEEE transactions on automatic control*, 64(10):3995–4010, 2019.

[19] Abolfazl Hashemi, Anish Acharya, Rudrajit Das, Haris Vikalo, Sujay Sanghavi, and Inderjit Dhillon. On the benefits of multiple gossip steps in communication-constrained decentralized federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):2727–2739, 2021.

[20] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[21] 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. *arXiv preprint arXiv:2106.09685*, 2021.

[22] Xiaolin Hu, Zixuan Gong, Gengze Xu, Wei Liu, Jian Luan, Bin Wang, and Yong Liu. Stability and generalization of zeroth-order decentralized stochastic gradient descent with changing topology. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 17342–17350, 2025.

[23] Rustem Islamov, Yuan Gao, and Sebastian U Stich. Towards faster decentralized stochastic optimization with communication compression. In *The Thirteenth International Conference on Learning Representations*, 2025.

[24] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 252–262, 2018.

[25] Anastasia Koloskova, Sebastian Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *International conference on machine learning*, pp. 3478–3487. PMLR, 2019.

[26] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized sgd with changing topology and local updates. In *International conference on machine learning*, pp. 5381–5393. PMLR, 2020.

[27] Chamin Hewa Koneputugodage, Thalaiyasingam Ajanthan, Sameera Ramasinghe, Hadi Mohaghegh Dolatabadi, Shamane Siriwardhana, Gil Avraham, Violetta Shevchenko, Karol Pajak, James Snewin, and Alexander Long. Factored gossip diloco: Reducing blocking communication in diloco. *ICML*, 2026.

[28] Lingjing Kong, Tao Lin, Anastasia Koloskova, Martin Jaggi, and Sebastian Stich. Consensus control for decentralized deep learning. In *International Conference on Machine Learning*, pp. 5686–5696. PMLR, 2021.

[29] Ajay D Kshemkalyani and Mukesh Singhal. *Distributed computing: principles, algorithms, and systems*. Cambridge University Press, 2011.

[30] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

[31] Zhe Li, Bicheng Ying, Zidong Liu, Chaosheng Dong, and Haibo Yang. Achieving dimension-free communication in federated learning via zeroth-order optimization. *arXiv preprint arXiv:2405.15861*, 2024.

[32] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in neural information processing systems*, 30, 2017.

[33] Sijia Liu, Bhavya Kailkhura, Pin-Yu Chen, Paishun Ting, Shiyu Chang, and Lisa Amini. Zeroth-order stochastic variance reduction for nonconvex optimization. *Advances in neural information processing systems*, 31, 2018.

[34] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.

[35] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

[36] Mohammad Taher Pilehvar and Jose Camacho-Collados. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Human language technologies, volume 1 (Long and short papers)*, pp. 1267–1273, 2019.

[37] Zhen Qin, Daoyuan Chen, Bingchen Qian, Bolin Ding, Yaliang Li, and Shuiguang Deng. Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes. *arXiv preprint arXiv:2312.06353*, 2023.

[38] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 2383–2392, 2016.

[39] Max Ryabinin, Tim Dettmers, Michael Diskin, and Alexander Borzunov. Swarm parallelism: Training large models can be surprisingly communication-efficient. In *International Conference on Machine Learning*, pp. 29416–29440. PMLR, 2023.

[40] Yao Shu, Wenyang Hu, See-Kiong Ng, Bryan Kian Hsiang Low, and Fei Richard Yu. Ferret: Federated full-parameter tuning at scale for large language models. *arXiv preprint arXiv:2409.06277*, 2024.

[41] Yao Shu, Qixin Zhang, Kun He, and Zhongxiang Dai. Refining adaptive zeroth-order optimization at ease. *arXiv preprint arXiv:2502.01014*, 2025.

[42] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

[43] James C Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341, 1992.

[44] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *Advances in neural information processing systems*, 31, 2018.

[45] Tao Sun, Dongsheng Li, and Bao Wang. Decentralized federated averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4289–4301, 2022.

[46] Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving lora in privacy-preserving federated learning. In *International conference on learning representations*, volume 2024, pp. 17978–17994, 2024.

[47] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

[48] Yuki Takezawa, Ryoma Sato, Han Bao, Kenta Niwa, and Makoto Yamada. Beyond exponential graph: Communication-efficient topologies for decentralized learning via finite-time convergence. *Advances in Neural Information Processing Systems*, 36:76692–76717, 2023.

[49] Hanlin Tang, Xiangru Lian, Shuang Qiu, Lei Yuan, Ce Zhang, Tong Zhang, and Ji Liu. DeepSqueeze: Decentralization meets error-compensated compression, 2019. URL https://arxiv.org/abs/1907.07346.

[50] Yujie Tang, Junshan Zhang, and Na Li. Distributed zero-order algorithms for nonconvex multiagent optimization. *IEEE Transactions on Control of Network Systems*, 8(1):269–281, 2020.

[51] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powergossip: Practical low-rank communication compression in decentralized deep learning. *arXiv preprint arXiv:2008.01425*, 2020.

[52] Thijs Vogels, Lie He, Anastasiia Koloskova, Sai Praneeth Karimireddy, Tao Lin, Sebastian U Stich, and Martin Jaggi. Relaysum for decentralized deep learning on heterogeneous data. *Advances in Neural Information Processing Systems*, 34:28004–28015, 2021.

[53] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

[54] Xiaoyu Wang, Xiaotian Li, Zhixiang Zhou, Chen Li, and Yong Liu. Adf-lora: Alternating low-rank aggregation for decentralized federated fine-tuning. *arXiv preprint arXiv:2511.18291*, 2025.

[55] Zesen Wang and Mikael Johansson. Controlled disagreement improves generalization in decentralized training. *arXiv preprint arXiv:2602.02899*, 2026.

[56] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[57] Xinlei Yi, Shengjun Zhang, Tao Yang, and Karl H Johansson. Zeroth-order algorithms for stochastic distributed nonconvex optimization. *Automatica*, 142:110353, 2022.

[58] Ziming Yu, Pan Zhou, Sike Wang, Jia Li, Mi Tian, and Hua Huang. Zeroth-order fine-tuning of llms in random subspaces. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4475–4485, 2025.

[59] Eric Zelikman, Qian Huang, Percy Liang, Nick Haber, and Noah D Goodman. Just one byte (per gradient): A note on low-bandwidth decentralized language model finetuning using shared randomness. *arXiv preprint arXiv:2306.10015*, 2023.

[60] Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*, 2018.

[61] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[62] Hanzhen Zhao, Ding Shihong, Cong Fang, and Zhouchen Lin. Pazo: Preconditioned accelerated zeroth-order optimization for fine-tuning llms. *Advances in Neural Information Processing Systems*, 38:111163–111195, 2026.

[63] Haoyu Zhao, Boyue Li, Zhize Li, Peter Richtárik, and Yuejie Chi. Beer: Fast $o(1/t)$ rate for decentralized nonconvex optimization with communication compression. *Advances in Neural Information Processing Systems*, 35:31653–31667, 2022.

[64] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

[65] Tongtian Zhu, Tianyu Zhang, Mingze Wang, Zhanpeng Zhou, and Can Wang. On the surprising effectiveness of a single global merging in decentralized learning. In *International Conference on Learning Representations*, volume 2026, pp. 40206–40241, 2026.

CONTENTS

1 Introduction **1**

2 Background **2**

2.1 Decentralized training 2

2.2 Zeroth-order optimization 3

3 Method **3**

3.1 Seed-scalar based dimension-free communication 3

3.2 Flooding for Scalable Global Consensus 4

3.3 Decoupling gradient aggregation from numbers of clients 4

4 Theoretical Analysis **5**

5 Experiments **6**

5.1 Experimental setup 6

5.2 Communication–performance trade-off at LLM scale 6

5.3 Performance Scaling with Client Participation 7

5.4 Iteration Scaling with Client Participation 8

6 Limitation **8**

7 Conclusion **9**

A Algorithm overview **17**

B Additional Experimental Results **17**

B.1 Additional Meshgrid Topology Results 17

B.2 Characterization of the subspace gradient estimator 17

B.3 Delayed flooding 18

B.4 Small-Scale Pretraining from Scratch 19

B.5 Supervised Instruction Tuning 19

C Alternatives for Integrating Seed–Scalar Communication with Gossip **20**

D Implementation details and runtime analysis of gradient estimator **21**

E Convergence Analysis of the Subspace Coordinate-Wise Estimator **22**

E.1 Period-wise subspace formulation 23

E.2 Basic identities for the subspace problem 24

E.3 Coefficient-space ZO estimator 26

E.4 Inner-loop drift bound 28

E.5 One-period descent 30
E.6 Recovery of the full gradient in expectation 32
E.7 Global convergence theorem 32

F SeedFlood as Data-Parallel Subspace Zeroth-Order Optimization **34**
F.1 Decentralized setup and exact flooding 34
F.2 Averaged estimator moments 35
F.3 SeedFlood convergence 36

G Experiment setup **38**
G.1 Dataset 38
G.2 Hyperparameters 38
G.3 LoRA configuration 38
G.4 Iteration-Budget Experiment Setup 38
G.5 Compute resource 39

H Supplementary Task-Level Results **40**

A ALGORITHM OVERVIEW

Algorithm 1: SeedFlood

Input : Initial models $\{\theta_i^0\}_{i=1}^n$ are identical; clients n; local data $\mathcal{D}_i$; diameter D; learning rate η_t; total steps T; perturbation scale ϵ; period τ; rank r; stochastic loss $F(\theta;\mathcal{B})$; $\mathcal{S}_i=\emptyset$, $\mathcal{R}_i=\emptyset$; global seed s^{glob}.

Output : Global model $\bar{\theta}=\frac{1}{n}\sum_{i=1}^n \theta_i^T=\theta_i^T$.

for $t=0,\dots,T-1$ *in parallel for all* $i\in[n]$ **do**

  /* **(A) Shared subspace setup** */

  if $t \bmod \tau = 0$ **then**

    $(\mathcal{U},\mathcal{V}) \leftarrow \texttt{SetupSubspace}(s^{\text{glob}},t,r)$

  /* **(B) Local ZO gradient estimation** */

  $(\alpha_{i,t},z_{i,t},s_{i,t}) \leftarrow \texttt{ZOEstimate}(\theta_i^t,\mathcal{D}_i,\mathcal{U},\mathcal{V},\epsilon)$

  $\theta_i^{t,0}=\theta_i^t-\eta_t\frac{1}{n}\alpha_{i,t}z_{i,t}$

  /* **(C) Flooding iter & aggregation** */

  $\mathcal{R}_i \leftarrow \mathcal{R}_i \cup \{(s_{i,t},\eta_t\alpha_{i,t}/n)\}$

  $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{(s_{i,t},\eta_t\alpha_{i,t}/n)\}$

  for $d=0,\dots,D-1$ **do**

    Send $\mathcal{R}_i$ to all neighbors $j\in\mathcal{N}(i)\setminus\{i\}$

    $\mathcal{R}_i \leftarrow$ messages received from neighbors

    $\mathcal{R}_i \leftarrow \mathcal{R}_i \setminus \mathcal{S}_i$

    $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \mathcal{R}_i$

    $\Delta \leftarrow \sum_{(s_k,v_k)\in\mathcal{R}_i} v_k\,\texttt{RNG_SF}(s_k,\mathcal{U},\mathcal{V})$

    $\theta_i^{t,d+1} \leftarrow \theta_i^{t,d}-\Delta$

  $\theta_i^{t+1} \leftarrow \theta_i^{t,D}$

Function $\texttt{SetupSubspace}(s^{\text{glob}},t,r)$**:**

  Initialize RNG with seed $s^{\text{glob}}+t$

  foreach *2D layer* $\theta_\ell \in \mathbb{R}^{n_\ell\times m_\ell}$ **do**

    $U_\ell \sim \mathcal{N}(0,1)^{n_\ell\times r}, \quad V_\ell \sim \mathcal{N}(0,1)^{m_\ell\times r}$

  $\mathcal{U}\leftarrow\{U_\ell\}_{\ell=1}^L, \quad \mathcal{V}\leftarrow\{V_\ell\}_{\ell=1}^L$

  return $(\mathcal{U},\mathcal{V})$

Function $\texttt{ZOEstimate}(\theta,\mathcal{D},\mathcal{U},\mathcal{V},\epsilon)$**:**

  Sample minibatch $\mathcal{B}\sim\mathcal{D}$ and random seed s

  $z\leftarrow\texttt{RNG_SF}(s,\mathcal{U},\mathcal{V})$

  $F^+\leftarrow F(\theta+\epsilon z;\mathcal{B}), \quad F^-\leftarrow F(\theta-\epsilon z;\mathcal{B})$

  $\alpha\leftarrow(F^+-F^-)/(2\epsilon)$

  return (α,z,s)

Function $\texttt{RNG_SF}(s,\mathcal{U},\mathcal{V})$**:**

  Initialize RNG with seed s

  foreach *layer* θ_ℓ **do**

    if θ_ℓ *is a 2D matrix* **then**

      Sample $i_\ell,j_\ell\sim\text{Unif}\{0,\dots,r-1\}$

      $z_\ell\leftarrow\text{vec}\left(U_\ell[:,i_\ell]V_\ell[:,j_\ell]^\top\right)$

    else

      Sample $z_\ell\sim\mathcal{N}(0,1)^{\text{shape}(\theta_\ell)}$

  return $z=[z_1,z_2,\dots,z_L]$

Algorithm 1 assembles the three primitives. (A) Subspace setup: every τ iterations, all clients re-derive the shared basis $(\mathbf{U},\mathbf{V})$ from the global seed. (B) Local gradient estimation: each client samples a minibatch and a local seed; the seed selects one coordinate of the shared basis per layer(RNG_SF), and the two-point finite difference along the induced rank-one direction yields the seed–scalar message, with the step size and the $1/n$ averaging factor folded into its scalar. (C) Flooding and aggregation: the message is flooded for D rounds following the relay rule of Section 3.2, and each client applies newly received messages through the coordinate accumulation of Section 3.3.

Table 1 contrasts the resulting costs, where n is the number of clients, d the parameter dimension, and r the subspace rank, with $d \gg n \geq r$. Traditional gossip pays $O(d)$ communication per edge without reaching exact consensus, whereas SeedFlood attains exact consensus with $O(n)$ communication—one compact message per client—and $O(n+rd)$ aggregation cost.

Table 1: Communication overhead and consensus properties.

Approach	Communicated Bytes	Applying Computation	Perfect Consensus
Traditional Gossip	$\mathcal{O}(d)$	$\mathcal{O}(d)$	–
SeedFlood (ours)	$\mathcal{O}(n)$	$\mathcal{O}(n+rd)$	✓

B ADDITIONAL EXPERIMENTAL RESULTS

In this section, we provide additional experimental results that complement the empirical evaluation in the main paper.

B.1 ADDITIONAL MESHGRID TOPOLOGY RESULTS

We additionally repeat the experiment in Section 5.2 under a denser meshgrid topology (Figure 6). As expected, several gossip-based baselines show modest performance improvements with increased connectivity, while SeedFlood remains competitive despite being largely insensitive to the underlying topology.

B.2 CHARACTERIZATION OF THE SUBSPACE GRADIENT ESTIMATOR

Setup. For the gradient-estimator ablation study, we conduct 20K training iterations on a single OPT-2.7B client. In the overall performance experiments, the subspace rank and update interval are fixed to 32 and 500, respectively, without additional search. For the hyperparameter sensitivity analysis,

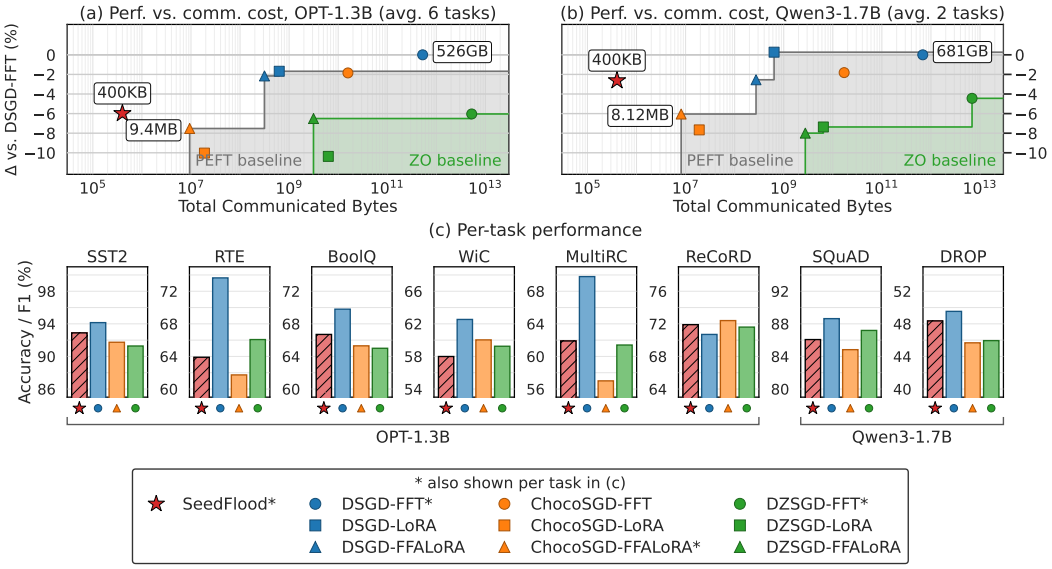

Figure 6: Performance–communication trade-off on a 16-client meshgrid network.

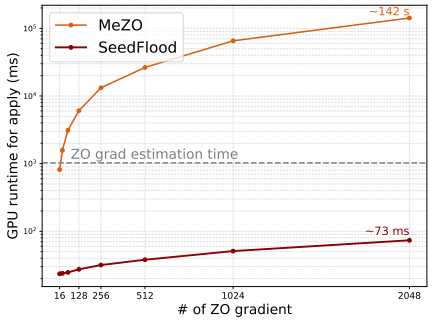

Figure 7: Runtime for applying multiple ZO messages to OPT-2.7B on a single A100 GPU with subspace size $R = 50$.

Table 2: Single-client finetuning of OPT-2.7B on SuperGLUE + SST-2 (%). "Avg" is normalized by MeZO and averaged.

Method	SST2	RTE	BoolQ	WiC	MultiRC	ReCoRD	Avg
MeZO	93.58	62.46	64.70	56.27	56.90	76.50	0.00%
SeedFlood	92.89	63.18	64.90	56.11	58.90	76.30	+0.62%

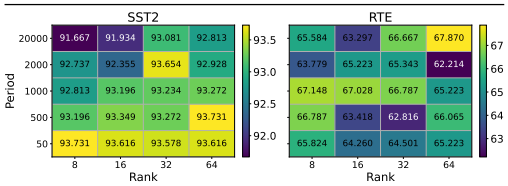

Figure 8: Sensitivity of SeedFlood to the subspace rank and refresh period on SST-2 and RTE.

we evaluate subspace ranks in 8,16,32,64 and update intervals in 50,500,1000,2000,20000, where an interval of 20000 corresponds to no subspace update during training. All other settings, including prompts, hyperparameters, and learning rate search procedures, follow the same experimental setup.

As shown in Figure 7, SeedFlood's gradient estimator maintains low and scalable aggregation overhead, remaining orders of magnitude faster than MeZO even with thousands of gradient updates. We provide implementation details and runtime analysis in Appendix D.

We further verify that this computational gain does not come from an ineffective estimator. In a single-client OPT-2.7B fine-tuning experiment, SeedFlood 's estimator matches MeZO-level performance on average as seen in Table 2. Hyperparameter ablation shows that larger rank and more frequent subspace refreshes generally improve performance. However, on RTE, excessively frequent refreshes at large ranks can degrade performance, likely because the optimizer moves to a new subspace before sufficiently exploiting the current one (see Figure 8).

B.3 Delayed Flooding

In practice, the exact network diameter may be unknown or full propagation before each update may be too restrictive. We thus evaluate a relaxed variant performing only k flooding steps per round. We evaluate delayed flooding by training OPT-125M on a ring topology with 32 clients, using $k \in \{1, 2, 4, 8, 16\}$. As shown in Figure 9, SeedFlood exhibits no observable performance degradation for moderate values of k ($k \geq 4$), maintaining performance comparable to full flooding

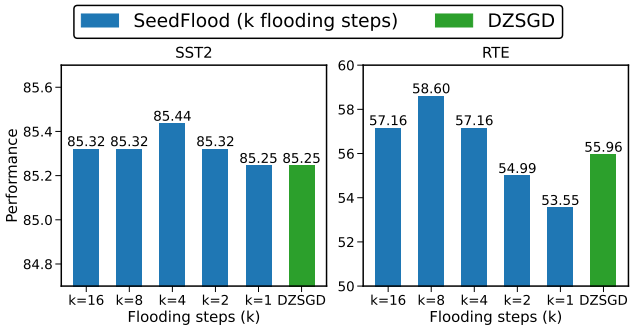

Figure 9: Task performances on SST2 and RTE under partial flooding with varying number of flooding steps.

Table 3: Training from scratch on MNIST with a 3-layer CNN under a 32-client ring topology. SeedFlood achieves competitive accuracy while requiring substantially less communication than dense decentralized baselines.

Method	IID Acc.	non-IID Acc.	Total Comm.	Comm. / Iter.
DSGD	97.57	96.08	260 MB	173 KB
ChocoSGD	90.47	69.01	5.24 MB	3.5 KB
SeedFlood	94.17	92.82	2.78 MB	0.3 KB
DZSGD	94.10	84.45	1.3 GB	173 KB

and consistently outperforming DZSGD. This indicates that the benefit of flooding-based global aggregation does not rely on strictly enforcing perfect consensus at every iteration, but rather on sufficiently fast global propagation of update information. Overall, SeedFlood remains effective without exact full flooding, as long as update propagation delay is sufficiently bounded.

B.4 Small-Scale Pretraining from Scratch

Although the main scope of this paper is decentralized LLM fine-tuning, this choice is intended to avoid overstating our claims as large-scale LLM pretraining. The underlying framework of SeedFlood, however, is not restricted to fine-tuning. When the model size is small enough for zeroth-order optimization to be practical, SeedFlood can also be applied to training from scratch.

We therefore consider a small-scale training-from-scratch setup previously studied in Li et al. [31]. To adapt this setup to decentralized learning, we replace the centralized communication structure with a 32-client ring topology and compare against decentralized baseline algorithms. Specifically, we train a 3-layer CNN from scratch on MNIST under both IID and non-IID data partitions. The results are summarized in Table 3.

The results show that SeedFlood is also effective beyond LLM fine-tuning. Compared with dense communication baselines such as DSGD and DZSGD, SeedFlood reduces communication by orders of magnitude while maintaining competitive accuracy. In particular, under the non-IID setting, SeedFlood substantially outperforms ChocoSGD and DZSGD, despite using the smallest per-iteration communication among all methods. These results suggest that the proposed seed-based flooding framework is applicable not only to communication-efficient LLM fine-tuning, but also to small-scale decentralized training from scratch.

B.5 Supervised Instruction Tuning

The main experiments evaluate SeedFlood on downstream fine-tuning tasks. This appendix asks whether the same estimator is usable for a supervised instruction tuning, under a same iteration budget. we report one model on one instruction-following corpus.

Setup. We instruction-tune Qwen3-8B-Base [56] on tulu-3-sft-personas-instruction-following [30], 29980 single-turn examples. We split 90/5/5 into 26982 / 1499 / 1499 for train / validation / test

Table 4: Instruction tuning Qwen3-8B-Base on Tulu-3 personas-IF on a 32-client ring under a matched budget of 3 epoch. Mean ± std over three seeds; all numbers in %. The first-order and DZO baselines train an FFA-LoRA adapter (r=8); SeedFlood trains the full parameter set through seed-encoded perturbations. Comm. is the total payload one client sends over the 159-iteration run. Learning rates were selected by validation loss from a per-method grid. Best result in bold.

		IFEval					
Method	Comm. (MB total)	Prompt strict	Inst. strict	Prompt loose	Inst. loose	BBH	MMLU
Qwen3-8B-Base (untuned)	–	46.4	58.5	52.1	63.2	47.4	69.1
DSGD-FFALoRA	182.9	51.1 ±0.3	61.5 ±0.5	55.3 ±0.3	65.3 ±0.5	**49.0** ±0.1	71.5 ±0.2
ChocoSGD-FFALoRA	5.5	50.3 ±1.7	61.0 ±1.6	55.0 ±1.2	64.9 ±1.2	48.9 ±0.3	**71.7** ±0.2
DZO-FFALoRA	182.9	46.7 ±0.7	58.4 ±0.2	52.1 ±0.7	62.7 ±0.3	47.1 ±0.1	69.0 ±0.2
SeedFlood (ours)	**0.04**	**51.8** ±0.8	**62.6** ±0.4	**56.1** ±1.0	**66.1** ±0.6	48.5 ±0.2	71.2 ±1.8

and format every example with the model's own ChatML template. All decentralized methods use FFA-LoRA [46] (r=8 on $\mathbf{q}_{\text{proj}}, \mathbf{v}_{\text{proj}}$; A frozen and shared, only B trained and communicated) on a 32-client ring. The token budget and iteration is identical for every method: 3 epochs, 159 iterations. For each method we sweep the learning rate over a three-point grid We evaluate on IFEval (541 generative prompts; prompt- and instruction-level, strict and loose) [64], BBH (18 subtasks, 0-shot, log-likelihood) [47] and MMLU (57 subjects, 0-shot, log-likelihood)[20]; the last two check that instruction tuning does not degrade knowledge or reasoning.

Result. SeedFlood, DSGD and ChocoSGD finish in one tier (Table 4): on IFEval prompt-level strict accuracy they reach 51.8 ± 0.8, 51.1 ± 0.3 and 50.3 ± 1.7 from an untuned 46.4, and every gap among the three is inside one seed's spread. What separates them is communication: SeedFlood sends 0.04 MB per client over the whole run against 182.9 MB for DSGD and 5.5 MB for ChocoSGD, while updating the full 8.2B parameters rather than an r=8 adapter. DZO does not train: its validation loss moves by less than 5×10^{-4} at every learning rate and every benchmark stays within one point of the untuned model.

C Alternatives for Integrating Seed–Scalar Communication with Gossip

A natural alternative to SeedFlood is to combine seed-based communication with existing gossip-based decentralized training frameworks. Since shared randomness allows a gradient update to be represented by a seed–scalar pair, one may try to avoid exchanging dense model parameters by representing each local model as an accumulation of past seed-reconstructible updates. Formally, let $\mathcal{M}_i^t$ denote the set of update messages known to client i at iteration t. Then the local model can be written as

$$\theta_i^t = \theta^0 - \sum_{m \in \mathcal{M}_i^t} c_{i,t}(m)\, \alpha(m)\, \mathrm{RNG}\big(s(m)\big), \tag{4}$$

where $s(m)$ and $\alpha(m)$ denote the seed and scalar coefficient of update message m, and $c_{i,t}(m)$ denotes the current gossip-induced mixing coefficient assigned to that update.

Under gossip-based model averaging, clients repeatedly mix their local models with neighbors. In the seed-based representation above, this corresponds to exchanging the update histories and averaging the coefficients of all known messages:

$$c_{i,t+1}(m) = \sum_{j \in \mathcal{N}(i)} w_{ij}\, c_{j,t}(m), \qquad \forall m \in \bigcup_{j \in \mathcal{N}(i)} \mathcal{M}_j^t. \tag{5}$$

This idea is closely related in spirit to prior centralized federated learning methods that represent model updates through seed-based histories [37, 31]. Because each message is only a constant-size seed–scalar pair, communicating all updates generated up to iteration t requires $O(tn)$ communication, which is much smaller than transmitting dense model parameters of dimension d.

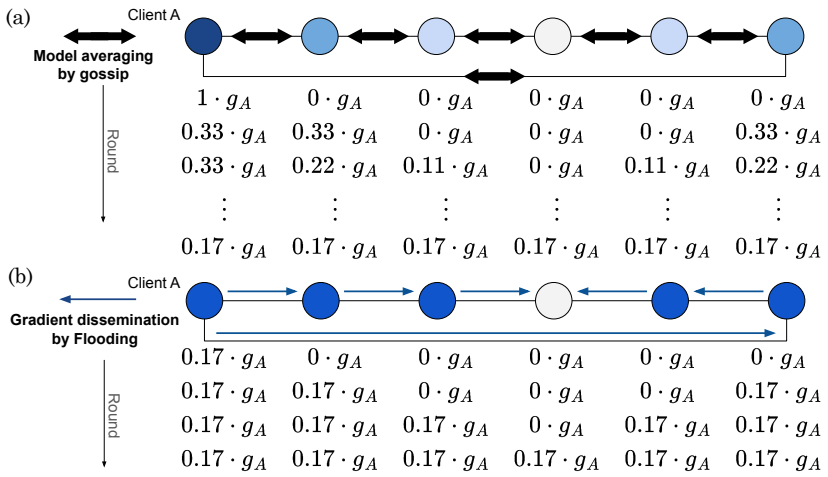

Figure 10: Consensus dynamics of a single gradient under gossip-based model averaging (a) and flooding-based gradient dissemination (b).

However, this communication saving does not translate into a scalable decentralized algorithm when combined with gossip. The key issue is that gossip does not merely disseminate an update; it repeatedly changes the effective coefficient of every past update as models are mixed over the network. Figure 10(a) illustrates this behavior. Even a single update generated by client A is gradually diffused through repeated local averaging, and its contribution to each client's model changes at every gossip step.

This creates a computational bottleneck under seed-based communication. Whenever the coefficient $c_{i,t}(m)$ of a past update changes, client i must realize the corresponding change in the dense model space. Equivalently, it must reconstruct the perturbation $\mathrm{RNG}(s(m))$ from its seed and apply the coefficient difference to the model. Since the number of stored update messages grows as $O(tn)$ and reconstructing each update costs $O(d)$, maintaining the gossip-averaged model can require $O(tnd)$ computation over time. Thus, the bottleneck is shifted from communication to reconstruction.

This limitation is structural rather than merely implementation-specific. Gossip is designed for iterative averaging of dense model states, where repeated reweighting is the mechanism by which consensus is reached. In contrast, seed-based communication is most effective when each transmitted object is an individual update that can be reconstructed and applied with a fixed coefficient. Repeatedly reweighting the same seed-reconstructible updates undermines this advantage, because it forces clients to revisit and reconstruct an ever-growing history of past updates.

This observation motivates the flooding-based design of `SeedFlood`. Instead of gradually diffusing model states through repeated averaging, flooding disseminates each update message itself across the network. Once a client receives a previously unseen update, it applies the update exactly once with its fixed coefficient and forwards the message to its neighbors. As illustrated in Figure 10(b), the role of communication is no longer to iteratively reweight local models, but to ensure that all clients eventually receive the same set of update messages. Therefore, under seed-based communication, flooding provides a more compatible consensus primitive: it preserves the communication benefit of shared randomness while avoiding the repeated reconstruction cost induced by gossip.

D Implementation Details and Runtime Analysis of Gradient Estimator

In this section, we provide implementation details of our subspace coordinate-wise gradient estimator and a detailed breakdown of its runtime components. In our implementation, each two-dimensional weight matrix maintains an additional buffer of the form $U_\ell A_\ell V_\ell^\top$. During training, all coordinate-wise updates are accumulated into A_ℓ, and the full update $U_\ell A_\ell V_\ell^\top$ is applied to the base weight on-the-fly during the forward pass. This design reduces the cost of applying received messages to $O(n)$ and shifts the $O(rd)$ matrix multiplication cost to the forward pass. Moreover, operations

required for generating and updating perturbation vectors—which account for approximately 25% of the gradient estimation cost in MeZO—are replaced with lightweight coordinate updates in our estimator.

Table 5 presents a detailed runtime comparison between our estimator and MeZO. For each zeroth-order method, the total runtime is decomposed into gradient estimation and message application phases. The cost of applying multiple messages further consists of random number generation and arithmetic operations for parameter updates.

Our estimator follows MeZO's procedure only for a small number of one-dimensional tensors, while for two-dimensional tensors it performs coordinate updates on the buffer A_ℓ. In the gradient estimation phase, the additional $O(rd)$ matrix multiplication required to inject buffer values into the model slightly increases the forward-pass time; however, this overhead is offset by the substantially reduced perturbation cost.

Overall, although our estimator has a theoretical computation complexity of $O(n + rd)$, our implementation effectively eliminates the computation overhead of message application in practice, with the additional $O(rd)$ computation largely hidden by the efficiency gains in perturbation.

Table 5: Detailed wall clock time per iteration of the SeedFlood framework with MeZO versus our subspace coordinate-wise gradient estimator. Results are averaged over 5 steps on OPT-2.7B with batch size 16 and 16 clients (i.e., 16 ZO-gradient messages generated per iteration). "GE" denotes the gradient estimation phase, and "MA" denotes the message-applying phase.

1. Overall runtime (ms)

Method	GE	MA	Total time
MeZO	1077	1432	2509
Ours	914	28	942

2. Gradient estimation phase (ms)

Method	Forward pass	Perturbation	Local update
MeZO	757	239	80
Ours	870	26	2

3. Message apply phase (ms)

Method	RV generation	Param update	Coordinate update
MeZO	223.2	1189	–
Ours	2.4	4.8	10.48

E Convergence Analysis of the Subspace Coordinate-Wise Estimator

In this section, we analyze subspace coordinate wise gradient estimator through a period-wise subspace optimization view. The main idea is to reinterpret this estimator as solving, over each outer period, a lower-dimensional stochastic zeroth-order subproblem induced by a fixed low-rank subspace. This avoids handling directly the dependence between the iterates and the reused subspace.

Actual update used in the analysis. We consider the following concrete form of the subspace coordinate-wise estimator. At the beginning of outer period k, a pair of low-rank basis matrices

$$U_k \in \mathbb{R}^{d_1 \times r_1}, \qquad V_k \in \mathbb{R}^{d_2 \times r_2}$$

is sampled according to Assumption 2 and fixed for the next τ inner steps. For each inner step $s = 0, 1, \ldots, \tau - 1$, we sample a coefficient-space random direction

$$C_{k,s} \in \mathbb{R}^{r_1 \times r_2},$$

form the full-space perturbation direction

$$D_{k,s} := U_k C_{k,s} V_k^\top, \tag{6}$$

and compute the two-point zeroth-order coefficient

$$a_{k,s} := \frac{F(X_{k,s} + \epsilon D_{k,s}; \xi_{k,s}) - F(X_{k,s} - \epsilon D_{k,s}; \xi_{k,s})}{2\epsilon}. \tag{7}$$

Then the actual update is

$$X_{k,s+1} = X_{k,s} - \eta\, a_{k,s} D_{k,s}, \qquad s = 0, 1, \ldots, \tau - 1, \tag{8}$$

where $X_{k,0} = X_{k\tau}$ and $X_{k+1,0} = X_{k,\tau}$.

Objective. We consider the stochastic optimization problem

$$\min_{X \in \mathbb{R}^{d_1 \times d_2}} f(X) := \mathbb{E}_\xi[F(X; \xi)]. \tag{9}$$

We present the proof in matrix form, using the Frobenius norm throughout.

E.1 Period-wise subspace formulation

Let

$$X^{(k)} := X_{k,0} = X_{k\tau} \tag{10}$$

denote the starting point of period k. Define the subspace objective

$$g_k(B) := f\big(X^{(k)} + U_k B V_k^\top\big), \qquad B \in \mathbb{R}^{r_1 \times r_2}. \tag{11}$$

The point of this definition is that, once U_k and V_k are fixed, the algorithm only moves inside the affine subspace

$$\{X^{(k)} + U_k B V_k^\top : B \in \mathbb{R}^{r_1 \times r_2}\}.$$

We now define the coefficient-space iterates by

$$B_{k,0} := 0, \tag{12}$$

and

$$B_{k,s+1} = B_{k,s} - \eta\, a_{k,s} C_{k,s}, \qquad s = 0, 1, \ldots, \tau - 1. \tag{13}$$

The next proposition shows that the full-space recursion (8) is exactly equivalent to the lower-dimensional recursion (13).

Proposition 1 (Period-wise subspace representation)**.** *Suppose that U_k and V_k are fixed during period k. Let $B_{k,s}$ be generated by (12)–(13). Then for every $s = 0, 1, \ldots, \tau$,*

$$X_{k,s} = X^{(k)} + U_k B_{k,s} V_k^\top. \tag{14}$$

Proof. We prove the claim by induction on s. For $s = 0$, the claim follows from the definitions $X_{k,0} = X^{(k)}$ and $B_{k,0} = 0$. Now suppose that

$$X_{k,s} = X^{(k)} + U_k B_{k,s} V_k^\top$$

holds for some $s \in \{0, 1, \ldots, \tau - 1\}$. Using the actual update (8) and the definition of $D_{k,s}$ in (6), we obtain

$$X_{k,s+1} = X_{k,s} - \eta a_{k,s} U_k C_{k,s} V_k^\top.$$

Substituting the induction hypothesis gives

$$X_{k,s+1} = X^{(k)} + U_k B_{k,s} V_k^\top - \eta a_{k,s} U_k C_{k,s} V_k^\top.$$

Factoring out U_k and $V_k^\top$ yields

$$X_{k,s+1} = X^{(k)} + U_k \big(B_{k,s} - \eta a_{k,s} C_{k,s}\big) V_k^\top.$$

By the coefficient-space update (13), the term in parentheses is precisely $B_{k,s+1}$. Hence

$$X_{k,s+1} = X^{(k)} + U_k B_{k,s+1} V_k^\top.$$

This closes the induction. □

By Proposition 1, the period-k analysis reduces to the analysis of a stochastic ZO method on the lower-dimensional problem (11).

E.2 Basic identities for the subspace problem

We first record the gradient of the subspace objective.

Lemma 1 (Gradient of the subspace objective)**.** *For the function $g_k(B) = f(X^{(k)} + U_k B V_k^\top)$, the gradient is*

$$\nabla g_k(B) = U_k^\top \nabla f\big(X^{(k)} + U_k B V_k^\top\big) V_k. \tag{15}$$

In particular,

$$\nabla g_k(0) = U_k^\top \nabla f(X^{(k)}) V_k. \tag{16}$$

Proof. Let $H \in \mathbb{R}^{r_1 \times r_2}$ be arbitrary. By the chain rule,

$$\begin{aligned}\langle \nabla g_k(B),\, H\rangle &= \frac{d}{dt} g_k(B + tH)\Big|_{t=0} \\ &= \frac{d}{dt} f\big(X^{(k)} + U_k(B + tH)V_k^\top\big)\Big|_{t=0} \\ &= \langle \nabla f(X^{(k)} + U_k B V_k^\top),\, U_k H V_k^\top\rangle.\end{aligned}$$

Using the Frobenius inner product identity

$$\langle G,\, U H V^\top\rangle = \langle U^\top G V,\, H\rangle,$$

we obtain

$$\langle \nabla g_k(B),\, H\rangle = \langle U_k^\top \nabla f(X^{(k)} + U_k B V_k^\top) V_k,\, H\rangle.$$

Since this holds for all H, equation (15) follows. Setting $B = 0$ yields (16). □

We next show that the smoothness of f induces smoothness of g_k.

Assumption 1 (Smoothness and bounded minibatch noise)**.** f is bounded below by $f^\star$, and every realization $F(\cdot;\xi)$ is L-smooth: $\|\nabla F(X;\xi) - \nabla F(Y;\xi)\|_F \le L\|X - Y\|_F$ (hence f is L-smooth). Minibatch stochastic gradients of size b are unbiased with variance at most σ^2/b.

For convenience, we restate this in the explicit form used in the proofs: L-smoothness reads

$$\|\nabla f(X) - \nabla f(Y)\|_F \le L\|X - Y\|_F, \qquad \forall X, Y, \tag{17}$$

and unbiased mini-batch stochastic gradients with bounded variance read

$$\mathbb{E}_\xi[\nabla F(X;\xi)] = \nabla f(X), \qquad \mathbb{E}_\xi\|\nabla F(X;\xi) - \nabla f(X)\|_F^2 \le \frac{\sigma^2}{b}, \qquad \forall X, \tag{18}$$

where ξ denotes a mini-batch of size b. In addition, Assumption 1 imposes *per-realization* smoothness: for every mini-batch ξ, the map $X \mapsto F(X;\xi)$ is differentiable and

$$\|\nabla F(X;\xi) - \nabla F(Y;\xi)\|_F \le L\|X - Y\|_F, \qquad \forall X, Y, \tag{19}$$

which implies (17) by Jensen's inequality. Per-realization smoothness is standard in convergence analyses of zeroth-order methods [34].

Assumption 2 (Sphere-normalized subspace sampling)**.** Columns of U_k and V_k are drawn i.i.d. from $\mathrm{Unif}(\sqrt{d_1}\,\mathbb{S}^{d_1-1})$ and $\mathrm{Unif}(\sqrt{d_2}\,\mathbb{S}^{d_2-1})$, independently across periods and of all inner-loop randomness, and satisfy $\|U_k\|_2 \le \rho_U := 2\sqrt{d_1}$ and $\|V_k\|_2 \le \rho_V := 2\sqrt{d_2}$; the discussion below explains why this operator-norm bound is benign.

We first make precise the ensemble underlying Assumption 2: the columns of $U_k = [u_{k,1}, \dots, u_{k,r_1}]$ and $V_k = [v_{k,1}, \dots, v_{k,r_2}]$ are sampled independently as $u_{k,a} \sim \mathrm{Unif}(\sqrt{d_1}\mathbb{S}^{d_1-1})$ and $v_{k,b} \sim \mathrm{Unif}(\sqrt{d_2}\mathbb{S}^{d_2-1})$, independently across periods and of $X^{(k)}$ and inner-loop randomness, and the joint draw (U_k, V_k) is retained only on the spectral event

$$\mathcal{E}_k := \big\{\|U_k\|_2 \le 2\sqrt{d_1}\big\} \cap \big\{\|V_k\|_2 \le 2\sqrt{d_2}\big\} \tag{20}$$

and redrawn otherwise, so that the operator-norm bounds $\|U_k\|_2 \le \rho_U = 2\sqrt{d_1}$ and $\|V_k\|_2 \le \rho_V = 2\sqrt{d_2}$ of Assumption 2 hold deterministically. This conditioning is benign: whenever $\max(r_1/d_1,\, r_2/d_2) \le 1/32$ and $\min(d_1, d_2) \ge 36$ (satisfied with in all our experiments,

where $r \le 64$ and $d_\ell \ge 2048$), Lemma 2 below shows that $\mathcal{E}_k$ fails with probability at most $(r_1{+}r_2{+}2)\, e^{-\min(d_1,d_2)/72}$, so the conditioned ensemble is statistically indistinguishable from the unconditioned one; moreover, the conditioning preserves the per-column isotropy $\mathbb{E}[u_{k,a}u_{k,a}^\top] = I_{d_1}$, $\mathbb{E}[v_{k,b}v_{k,b}^\top] = I_{d_2}$ (Lemma 3).

Lemma 2 (Spectral concentration of the sphere-normalized ensemble)**.** *Let $U \in \mathbb{R}^{d\times r}$ have columns drawn i.i.d. from $\mathrm{Unif}(\sqrt{d}\,\mathbb{S}^{d-1})$, with $r/d \le 1/32$ and $d \ge 36$. Then*

$$\Pr\big[\|U\|_2 > 2\sqrt{d}\big] \;\le\; (r+1)\, e^{-d/72}.$$

Proof. Write $u_a = \sqrt{d}\, g_a/\|g_a\|$ with $g_a \overset{\text{iid}}{\sim} \mathcal{N}(0, I_d)$, so that $U = \sqrt{d}\, G\,\mathrm{diag}(\|g_a\|^{-1})$ and hence $\|U\|_2 \le \sqrt{d}\,\|G\|_2/\min_a \|g_a\|$. Two standard Gaussian facts: $\Pr[\|G\|_2 > \sqrt{d}+\sqrt{r}+s] \le e^{-s^2/2}$, and, since $g \mapsto \|g\|$ is 1-Lipschitz with $\mathbb{E}\|g_a\| \ge \sqrt{d-1}$, $\Pr[\|g_a\| < \sqrt{d-1}-s] \le e^{-s^2/2}$. Taking $s = \sqrt{d}/4$ in the former and $s = \sqrt{d}/6$ in the latter with a union bound over $a \le r$, the event

$$\|G\|_2 \le \big(\tfrac{5}{4} + \sqrt{r/d}\big)\sqrt{d}, \qquad \min_{a\le r} \|g_a\| \ge \tfrac{4}{5}\sqrt{d}$$

holds with probability at least $1-(r+1)e^{-d/72}$, where we used $\sqrt{d-1} - \sqrt{d}/6 \ge \frac{4}{5}\sqrt{d} \iff \sqrt{d-1} \ge \frac{29}{30}\sqrt{d} \iff 59\,d \ge 900$, valid for $d \ge 36$ (indeed for all $d \ge 16$). On this event, since $\sqrt{r/d} \le 1/\sqrt{32}$,

$$\|U\|_2 \;\le\; \frac{5/4 + 1/\sqrt{32}}{4/5}\,\sqrt{d} \;<\; 2\sqrt{d}.$$

□

Lemma 3 (Properties of the conditioned ensemble)**.** *Under Assumption 2, (i) every column satisfies $\mathbb{E}[u_{k,a}u_{k,a}^\top] = I_{d_1}$ and $\mathbb{E}[v_{k,b}v_{k,b}^\top] = I_{d_2}$, and (ii) U_k and V_k are independent.*

Proof. (ii) The event (20) factorizes over U_k and V_k, and U_k and V_k are independent before conditioning, so they remain independent after conditioning on the product event. (i) The event (20) is invariant under the map $U_k \mapsto RU_k$ for any $R \in O(d_1)$, and so is the unconditioned law of U_k; hence the conditional law of U_k is rotation-invariant. Consequently $\mathbb{E}[u_{k,a}u_{k,a}^\top]$ commutes with every orthogonal matrix and equals cI_{d_1} for some $c \ge 0$; since $\|u_{k,a}\|^2 = d_1$ almost surely, taking traces gives $c = 1$. The same argument applies to V_k. □

Remark 1. In the actual implementation, the entries of U_k and V_k are sampled independently from $\mathcal{N}(0,1)$, as in [7]; since column norms concentrate at $\sqrt{d_\ell}$, the sphere normalization above is the standard idealization of this ensemble [7]. The conditioning on (20) is vacuous in practice: by Lemma 2 its failure probability is exponentially small in d_ℓ (negligible even after a union bound over all periods and layers), and it can be realized by *deterministic* rejection sampling—increment the seed and redraw on failure—which preserves seed-reconstructibility and never triggers in practice. Since (20) is rotation-invariant, the conditioning leaves the per-column isotropy used in the analysis intact (Lemma 3).

Lemma 4 (Smoothness of the subspace objective)**.** *Under Assumptions 1 and 2, the function g_k is $\widetilde{L}$-smooth with*

$$\widetilde{L} := L\rho_U^2\rho_V^2. \tag{21}$$

Under Assumption 2, this gives $\widetilde{L} = 16\,Ld_1d_2$. That is, for all $B_1, B_2 \in \mathbb{R}^{r_1\times r_2}$,

$$\|\nabla g_k(B_1) - \nabla g_k(B_2)\|_F \le \widetilde{L}\|B_1 - B_2\|_F. \tag{22}$$

Proof. Let $\Delta_{\nabla f} := \nabla f(X^{(k)} + U_kB_1V_k^\top) - \nabla f(X^{(k)} + U_kB_2V_k^\top)$. Using Lemma 1 and $\|AXB\|_F \le \|A\|_2\|X\|_F\|B\|_2$,

$$\begin{aligned}\|\nabla g_k(B_1) - \nabla g_k(B_2)\|_F &= \|U_k^\top \Delta_{\nabla f}\, V_k\|_F \le \rho_U\|\Delta_{\nabla f}\|_F\rho_V \\ &\le \rho_U \cdot L\|U_k(B_1-B_2)V_k^\top\|_F \cdot \rho_V \\ &\le L\rho_U^2\rho_V^2\|B_1 - B_2\|_F,\end{aligned}$$

where the last step uses L-smoothness of f and $\|UAV^\top\|_F \le \|U\|_2\|A\|_F\|V\|_2$. This proves (22). □

E.3 COEFFICIENT-SPACE ZO ESTIMATOR

Define the lifted stochastic objective

$$G_k(B;\xi) := F(X^{(k)} + U_k B V_k^\top; \xi). \tag{23}$$

Using Proposition 1, the two-point coefficient (7) can be rewritten as

$$a_{k,s} = \frac{G_k(B_{k,s} + \epsilon C_{k,s}; \xi_{k,s}) - G_k(B_{k,s} - \epsilon C_{k,s}; \xi_{k,s})}{2\epsilon}. \tag{24}$$

Therefore, the coefficient-space estimator is

$$\widehat{\nabla} g_k(B_{k,s}; \xi_{k,s}) := a_{k,s} C_{k,s}. \tag{25}$$

In other words, the coefficient update (13) is simply

$$B_{k,s+1} = B_{k,s} - \eta\, \widehat{\nabla} g_k(B_{k,s}; \xi_{k,s}). \tag{26}$$

We now specialize the estimator to the rule we actually use: at each inner step, a single coordinate is sampled uniformly at random, and only that coordinate's two-point finite difference is computed. This makes our method a *randomized coordinate descent with a one-coordinate finite-difference estimator*, not full coordinate finite-difference gradient descent.

Assumption 3 (Uniform coordinate sampling)**.** $J_{k,s} \sim \mathrm{Unif}\{1,\dots,q\}$ independently of $\xi_{k,s}$; both function evaluations in (3) use the same minibatch $\xi_{k,s}$.

We work in the period-wise coefficient space introduced above. Let $q := r_1 r_2$, and let $\mathcal{E} = \{E_1,\dots,E_q\}$ be the standard coordinate basis of $\mathbb{R}^{r_1\times r_2}$. The sampling rule of Assumption 3 can be written

$$C_{k,s} = E_{J_{k,s}}, \qquad J_{k,s} \sim \mathrm{Unif}\{1,\dots,q\}, \tag{27}$$

with $J_{k,s}$ independent of the mini-batch $\xi_{k,s}$ and the two function values in the central difference sharing the same mini-batch. For each coordinate $j \in \{1,\dots,q\}$, we further define

$$a_j(B;\xi) := \frac{G_k(B+\epsilon E_j;\xi) - G_k(B-\epsilon E_j;\xi)}{2\epsilon}, \qquad \bar{a}_j(B) := \mathbb{E}_\xi[a_j(B;\xi)],$$

and the population coordinate finite-difference operator

$$\nabla_\epsilon g_k(B) := \sum_{j=1}^{q} \bar{a}_j(B)\, E_j.$$

Lemma 5 (Standard CGE finite-difference property)**.** *Under Assumptions 1–3, the population coordinate finite-difference operator satisfies*

$$\nabla_\epsilon g_k(B) = \nabla g_k(B) + \delta_k(B), \qquad \|\delta_k(B)\|_F \le c_\delta \epsilon, \tag{28}$$

with $c_\delta = \sqrt{q}\,\widetilde{L}/2$.

Proof. This is the standard central coordinate finite-difference property used in CGE analyses. Indeed, for each coordinate j, define $h_j(t) := g_k(B+tE_j)$. Since g_k is $\widetilde{L}$-smooth by Lemma 4, the derivative $h_j'(t) = \langle \nabla g_k(B+tE_j),\, E_j\rangle$ is $\widetilde{L}$-Lipschitz. Therefore,

$$\left| \frac{g_k(B+\epsilon E_j) - g_k(B-\epsilon E_j)}{2\epsilon} - \langle \nabla g_k(B),\, E_j\rangle \right| \le \frac{1}{2\epsilon}\int_{-\epsilon}^{\epsilon} \widetilde{L}|t|\, dt = \frac{\widetilde{L}\epsilon}{2}.$$

Since $\bar{a}_j(B)$ is the corresponding population central difference, summing the squared coordinate errors over $j = 1,\dots,q$ gives $\|\delta_k(B)\|_F \le \sqrt{q}\,\widetilde{L}\epsilon/2$. □

The next lemma bounds the coordinate finite-difference noise. Bounding the noise pathwise at the current iterate would pick up the dimension factor $\rho_U^2\rho_V^2$ through the subspace-lifted variance; anchoring the noise at the period start $X^{(k)}$, where the minibatch noise is independent of the subspace draw, removes this factor from the stochastic-noise constant.

Lemma 6 (Anchored coordinate finite-difference noise)**.** *Suppose Assumptions 1–3 hold. Fix a period k and an inner step s, let*

$$\mathcal{F}_{k,s} := \sigma\big(X^{(k)}, U_k, V_k, \xi_{k,0}, J_{k,0}, \ldots, \xi_{k,s-1}, J_{k,s-1}\big),$$

let B be an $\mathcal{F}_{k,s}$-measurable random matrix in $\mathbb{R}^{r_1\times r_2}$, and let $\xi = \xi_{k,s}$ be the fresh minibatch of step s, independent of $\mathcal{F}_{k,s}$. Then

$$\frac{1}{q}\sum_{j=1}^{q}\mathbb{E}\big[\big(a_j(B;\xi)-\bar a_j(B)\big)^2 \,\big|\, X^{(k)}\big] \;\le\; \frac{3\sigma^2}{b} + \frac{12\widetilde{L}^2}{q}\,\mathbb{E}\big[\|B\|_F^2 \,\big|\, X^{(k)}\big] + 12\,\widetilde{L}^2\epsilon^2, \tag{29}$$

where $\bar a_j(B) := \mathbb{E}[a_j(B;\xi) \mid \mathcal{F}_{k,s}]$.

Proof. Write $\Delta(X;\xi) := \nabla F(X;\xi) - \nabla f(X)$, let $E_j = e_a e_b^\top$ denote the coordinate pair (a,b) corresponding to j, so that $\langle U_k^\top M V_k,\, E_j\rangle = u_{k,a}^\top M v_{k,b}$ for any $M \in \mathbb{R}^{d_1\times d_2}$, and set $X^{(j)}(t) := X^{(k)} + U_k(B+tE_j)V_k^\top$ for $t \in [-\epsilon,\epsilon]$. Since B is $\mathcal{F}_{k,s}$-measurable and ξ is independent of $\mathcal{F}_{k,s}$, we have, pathwise on $\mathcal{F}_{k,s}$, $\bar a_j(B) = \big(g_k(B+\epsilon E_j) - g_k(B-\epsilon E_j)\big)/(2\epsilon)$. Applying the fundamental theorem of calculus to $t \mapsto G_k(B+tE_j;\xi)$ and $t \mapsto g_k(B+tE_j)$, whose derivatives are $\langle U_k^\top \nabla F(X^{(j)}(t);\xi)V_k,\, E_j\rangle$ and $\langle U_k^\top \nabla f(X^{(j)}(t))V_k,\, E_j\rangle$ by the chain rule of Lemma 1

$$a_j(B;\xi) - \bar a_j(B) = \frac{1}{2\epsilon}\int_{-\epsilon}^{\epsilon} u_{k,a}^\top\,\Delta\big(X^{(j)}(t);\xi\big)\,v_{k,b}\,dt. \tag{30}$$

Step 1: three-term anchor decomposition. Let $X_B := X^{(k)} + U_k B V_k^\top$ and split the noise along the integration path into an anchor term at the period start, a drift term *common to all coordinates*, and a coordinate-specific perturbation residual:

$$\Delta\big(X^{(j)}(t);\xi\big) = \underbrace{\Delta(X^{(k)};\xi)}_{=:\,\Delta_0} + \underbrace{\Delta(X_B;\xi)-\Delta(X^{(k)};\xi)}_{=:\,\Gamma} + \underbrace{\Delta\big(X^{(j)}(t);\xi\big)-\Delta(X_B;\xi)}_{=:\,S_t^{(j)}}.$$

Only $S_t^{(j)}$ depends on the coordinate j and on t. By per-realization smoothness (19) and L-smoothness of f (17),

$$\|\Gamma\|_F \le 2L\|X_B - X^{(k)}\|_F \le 2L\rho_U\rho_V\|B\|_F, \qquad \|S_t^{(j)}\|_F \le 2L\|U_k\,tE_j\,V_k^\top\|_F \le 2L\rho_U\rho_V\,\epsilon$$

for all $|t|\le\epsilon$. Substituting into (30),

$$a_j(B;\xi)-\bar a_j(B) = A_j + D_j + C_j, \quad A_j := u_{k,a}^\top\Delta_0 v_{k,b}, \quad D_j := u_{k,a}^\top\Gamma\,v_{k,b}, \quad C_j := \frac{1}{2\epsilon}\int_{-\epsilon}^{\epsilon} u_{k,a}^\top S_t^{(j)} v_{k,b}\,dt,$$

and $(A_j+D_j+C_j)^2 \le 3A_j^2 + 3D_j^2 + 3C_j^2$.

Step 2: anchor term (isotropy). The term A_j involves neither B nor any past inner-loop randomness: Δ_0 is a function of $(X^{(k)},\xi)$ only, and (U_k,V_k) is independent of $(X^{(k)},\xi)$ by Assumption 2. Hence, conditionally on $(X^{(k)},\xi)$, the pair (U_k,V_k) retains the law of Assumption 2, and Lemma 3 (per-column isotropy and $U_k \perp V_k$) gives, exactly as in the single-pair computation of Lemma 10,

$$\mathbb{E}_{U_k,V_k}\big[(u_{k,a}^\top\Delta_0 v_{k,b})^2\big] = \mathbb{E}\big[u_{k,a}^\top\Delta_0\,\mathbb{E}[v_{k,b}v_{k,b}^\top]\,\Delta_0^\top u_{k,a}\big] = \mathrm{tr}\big(\Delta_0\Delta_0^\top\big) = \|\Delta_0\|_F^2.$$

Taking the expectation over ξ and using (18) at $X = X^{(k)}$,

$$\mathbb{E}\big[A_j^2 \,\big|\, X^{(k)}\big] = \mathbb{E}_\xi\big[\|\Delta_0\|_F^2 \,\big|\, X^{(k)}\big] \le \frac{\sigma^2}{b} \qquad \text{for every } j. \tag{31}$$

Step 3: drift term (Frobenius aggregation over coordinates). Since Γ is common to all coordinates, summing the squared projections over the full coordinate basis collapses to a Frobenius norm, pointwise in (U_k,V_k,B,ξ):

$$\frac{1}{q}\sum_{j=1}^{q} D_j^2 = \frac{1}{q}\|U_k^\top\Gamma V_k\|_F^2 \le \frac{\rho_U^2\rho_V^2}{q}\|\Gamma\|_F^2 \le \frac{4L^2\rho_U^4\rho_V^4}{q}\|B\|_F^2 = \frac{4\widetilde{L}^2}{q}\|B\|_F^2. \tag{32}$$

Step 4: perturbation residual. For every j and $|t| \le \epsilon$, $|u_{k,a}^\top S_t^{(j)} v_{k,b}| \le \rho_U \rho_V \|S_t^{(j)}\|_F \le 2L\rho_U^2\rho_V^2\,\epsilon = 2\widetilde{L}\epsilon$, hence $|C_j| \le 2\widetilde{L}\epsilon$ and $\frac{1}{q}\sum_j C_j^2 \le 4\widetilde{L}^2\epsilon^2$.

Step 5: combining. Averaging $(A_j + D_j + C_j)^2 \le 3A_j^2 + 3D_j^2 + 3C_j^2$ over $j \in \{1, \dots, q\}$, taking $\mathbb{E}[\cdot \mid X^{(k)}]$, and inserting (31), (32), and Step 4 yields (29). □

Remark 2. The coordinate-averaged form of (29) is essential: it is the aggregation over the full coordinate basis in (32) that yields the drift correction $\widetilde{L}^2\,\mathbb{E}\|B\|_F^2/q$, placing it at the same order in q as the signal terms of the inner-loop recursion (Lemma 8), where it is absorbed by the Grönwall argument with only a change of absolute constants. The discretization correction $12\widetilde{L}^2\epsilon^2$ is of the same scale as the finite-difference bias term $(2c_\delta^2/q)\epsilon^2 = (\widetilde{L}^2/2)\epsilon^2$ already present in Lemma 7.

Lemma 7 (Mean and second moment of the coordinate estimator)**.** *Under Assumptions 1–3, the actual one-step estimator we use,*

$$\widehat{\nabla} g_k(B; \xi, J) := a_J(B; \xi)\, E_J, \tag{33}$$

satisfies

$$\mathbb{E}\big[\widehat{\nabla} g_k(B; \xi, J) \mid X^{(k)}, U_k, V_k, B\big] = \frac{1}{q}\big(\nabla g_k(B) + \delta_k(B)\big). \tag{34}$$

Moreover, if B is $\mathcal{F}_{k,s}$-measurable and $(\xi, J) = (\xi_{k,s}, J_{k,s})$ is the fresh randomness of step s, as in Lemma 6, then

$$\mathbb{E}\big[\|\widehat{\nabla} g_k(B; \xi, J)\|_F^2 \mid X^{(k)}\big] \le \frac{2}{q}\,\mathbb{E}\big[\|\nabla g_k(B)\|_F^2 \mid X^{(k)}\big] + \frac{2c_\delta^2}{q}\epsilon^2 + \frac{3\sigma^2}{b} + \frac{12\widetilde{L}^2}{q}\,\mathbb{E}\big[\|B\|_F^2 \mid X^{(k)}\big] + 12\widetilde{L}^2\epsilon^2. \tag{35}$$

Proof. Conditioning on $X^{(k)}, U_k, V_k, B$ and averaging over the uniform coordinate J,

$$\mathbb{E}[\widehat{\nabla} g_k(B; \xi, J)] = \frac{1}{q}\sum_{j=1}^{q} \bar{a}_j(B) E_j = \frac{1}{q}\nabla_\epsilon g_k(B).$$

Applying Lemma 5 gives (34). For the second moment, conditionally on $\mathcal{F}_{k,s}$ (averaging over the fresh (ξ, J) only),

$$\mathbb{E}\big[\|\widehat{\nabla} g_k(B; \xi, J)\|_F^2 \;\big|\; \mathcal{F}_{k,s}\big] = \frac{1}{q}\sum_{j=1}^{q} \mathbb{E}_\xi[a_j(B;\xi)^2] = \frac{1}{q}\|\nabla_\epsilon g_k(B)\|_F^2 + \frac{1}{q}\sum_{j=1}^{q} \mathbb{E}_\xi[(a_j(B;\xi) - \bar{a}_j(B))^2].$$

Using $\nabla_\epsilon g_k(B) = \nabla g_k(B) + \delta_k(B)$, $\|M + M'\|_F^2 \le 2\|M\|_F^2 + 2\|M'\|_F^2$, and $\|\delta_k(B)\|_F \le c_\delta\epsilon$, the first term is at most $\frac{2}{q}\|\nabla g_k(B)\|_F^2 + \frac{2c_\delta^2}{q}\epsilon^2$, pointwise on $\mathcal{F}_{k,s}$. Taking $\mathbb{E}[\cdot \mid X^{(k)}]$ of both sides via the tower property and applying Lemma 6 to the averaged noise term yields (35). □

E.4 Inner-loop drift bound

We now control how far the coefficient iterate $B_{k,s}$ moves from the period start. Compared with a full-gradient estimator, the one-coordinate estimator only touches one of the q coordinates at each step, contributing a $1/q$ factor to both the conditional mean and the second moment. This is reflected in the bound through the quantity

$$\Lambda_q(\tau) := \frac{\tau}{q}\left(1 + \frac{\tau}{q}\right). \tag{36}$$

Lemma 8 (Inner-loop drift for the one-coordinate estimator)**.** *Under Assumptions 1–3, there exist absolute constants $C_1, C_2, C_3 > 0$ such that, for every $s \le \tau$,*

$$\mathbb{E}\big[\|B_{k,s}\|_F^2 \mid X^{(k)}\big] \le C_1\eta^2 q\,\Lambda_q(\tau)\|\nabla f(X^{(k)})\|_F^2 + C_2\eta^2\frac{\tau}{b}\sigma^2 + C_3\eta^2\big(c_\delta^2\,\Lambda_q(\tau) + 6\widetilde{L}^2\tau\big)\epsilon^2, \tag{37}$$

provided

$$\eta \;\le\; \frac{1}{\widetilde{L}\sqrt{32\,\Lambda_q(\tau)}}. \tag{38}$$

Proof. Starting from (26) and $B_{k,0} = 0$,

$$B_{k,s} = -\eta \sum_{j=0}^{s-1} \widehat{\nabla} g_k(B_{k,j}; \xi_{k,j}, J_{k,j}). \tag{39}$$

Let $\mathcal{F}_j$ be the σ-algebra generated by $(X^{(k)}, U_k, V_k, \xi_{k,0}, J_{k,0}, \ldots, \xi_{k,j-1}, J_{k,j-1})$. Decompose each estimator as

$$\widehat{\nabla} g_k(B_{k,j}; \xi_{k,j}, J_{k,j}) = \bar{Z}_j + N_j, \qquad \bar{Z}_j := \frac{1}{q}\big(\nabla g_k(B_{k,j}) + \delta_k(B_{k,j})\big),$$

where $\bar{Z}_j$ is the $\mathcal{F}_j$-conditional mean (by (34)) and N_j is the centered noise, which is a martingale-difference sequence with respect to $\{\mathcal{F}_j\}$. Throughout the rest of the proof, plain expectation signs abbreviate $\mathbb{E}[\cdot \mid X^{(k)}]$. Using $\|M + M'\|_F^2 \le 2\|M\|_F^2 + 2\|M'\|_F^2$,

$$\mathbb{E}\big[\|B_{k,s}\|_F^2 \mid X^{(k)}\big] \le 2\eta^2 \underbrace{\mathbb{E}\left[\|\sum_{j=0}^{s-1} \bar{Z}_j\|_F^2\right]}_{\text{signal}} + 2\eta^2 \underbrace{\mathbb{E}\left[\|\sum_{j=0}^{s-1} N_j\|_F^2\right]}_{\text{noise}}. \tag{40}$$

Signal part. By Cauchy–Schwarz and $\|\bar{Z}_j\|_F \le \frac{1}{q}\big(\|\nabla g_k(B_{k,j})\|_F + c_\delta \epsilon\big)$,

$$\|\sum_{j=0}^{s-1} \bar{Z}_j\|_F^2 \le \frac{2s}{q^2} \sum_{j=0}^{s-1} \big(\|\nabla g_k(B_{k,j})\|_F^2 + c_\delta^2 \epsilon^2\big). \tag{41}$$

By $\widetilde{L}$-smoothness of g_k (Lemma 4),

$$\|\nabla g_k(B_{k,j})\|_F^2 \le 2\|\nabla g_k(0)\|_F^2 + 2\widetilde{L}^2 \|B_{k,j}\|_F^2. \tag{42}$$

Both (41) and (42) hold pointwise. Substituting, using $s \le \tau$, taking $\mathbb{E}[\cdot \mid X^{(k)}]$, and applying Lemma 10 to the $X^{(k)}$-measurable matrix $\nabla f(X^{(k)})$, i.e. $\mathbb{E}\big[\|\nabla g_k(0)\|_F^2 \mid X^{(k)}\big] = \mathbb{E}\big[\|U_k^\top \nabla f(X^{(k)}) V_k\|_F^2 \mid X^{(k)}\big] = q\|\nabla f(X^{(k)})\|_F^2$,

$$\begin{aligned} 2\eta^2 \mathbb{E}\left[\|\sum_{j<s} \bar{Z}_j\|_F^2\right] \le{}& \frac{8\tau^2\eta^2}{q} \|\nabla f(X^{(k)})\|_F^2 + \frac{8\tau\eta^2\widetilde{L}^2}{q^2} \sum_{j=0}^{s-1} \mathbb{E}\big[\|B_{k,j}\|_F^2\big] \\ &+ \frac{4c_\delta^2 \tau^2 \eta^2}{q^2} \epsilon^2. \end{aligned} \tag{43}$$

Noise part. Since $\{N_j\}$ is a martingale-difference sequence, the cross terms vanish; indeed, for $i < j$, towering through $\mathcal{F}_j \supset \sigma(X^{(k)})$ gives $\mathbb{E}\big[\langle N_i, N_j\rangle \,\big|\, X^{(k)}\big] = \mathbb{E}\big[\langle N_i, \mathbb{E}[N_j \mid \mathcal{F}_j]\rangle \,\big|\, X^{(k)}\big] = 0$, so the orthogonality survives the coarser conditioning:

$$\mathbb{E}\left[\|\sum_{j=0}^{s-1} N_j\|_F^2\right] = \sum_{j=0}^{s-1} \mathbb{E}\big[\|N_j\|_F^2\big] \le \sum_{j=0}^{s-1} \mathbb{E}\big[\|\widehat{\nabla} g_k(B_{k,j}; \xi_{k,j}, J_{k,j})\|_F^2\big]. \tag{44}$$

Applying (35) to each term (with $B = B_{k,j}$, which is $\mathcal{F}_{k,j}$-measurable and faces the fresh randomness $(\xi_{k,j}, J_{k,j})$), then (42) and Lemma 10 as in the signal part,

$$\begin{aligned} 2\eta^2 \mathbb{E}\left[\|\sum_{j<s} N_j\|_F^2\right] \le{}& 8\tau\eta^2 \|\nabla f(X^{(k)})\|_F^2 + \frac{32\,\eta^2\widetilde{L}^2}{q} \sum_{j=0}^{s-1} \mathbb{E}\big[\|B_{k,j}\|_F^2\big] \\ &+ \frac{4c_\delta^2 \tau\eta^2}{q} \epsilon^2 + \frac{6\tau\eta^2}{b}\sigma^2 + 24\,\tau\eta^2\widetilde{L}^2\epsilon^2. \end{aligned} \tag{45}$$

Combining. Adding (43) and (45),

$$\mathbb{E}\big[\|B_{k,s}\|_F^2 \mid X^{(k)}\big] \le 8\eta^2\left(\frac{\tau^2}{q}+\tau\right)\|\nabla f(X^{(k)})\|_F^2 + \frac{6\tau\eta^2}{b}\sigma^2 + 4c_\delta^2\eta^2\left(\frac{\tau^2}{q^2}+\frac{\tau}{q}\right)\epsilon^2 + 24\,\tau\eta^2\widetilde{L}^2\epsilon^2 + 8\eta^2\widetilde{L}^2\left(\frac{\tau}{q^2}+\frac{4}{q}\right)\sum_{j=0}^{s-1}\mathbb{E}\big[\|B_{k,j}\|_F^2\big]. \tag{46}$$

Note $\frac{\tau^2}{q}+\tau = q\,\Lambda_q(\tau)$ and $\frac{\tau^2}{q^2}+\frac{\tau}{q} = \Lambda_q(\tau)$. The stochastic noise now enters at the dimension-free level σ^2/b and accumulates linearly in τ because the centered noise terms form a martingale-difference sequence; the drift correction inherited from Lemma 6 enters the recursive coefficient at the same $1/q$ order as the signal terms.

Grönwall step. Write $\phi_s := \mathbb{E}\big[\|B_{k,s}\|_F^2 \mid X^{(k)}\big]$. The recursion (46) takes the form

$$\phi_s \;\le\; \alpha + \beta\sum_{j=0}^{s-1}\phi_j,$$

where

$$\alpha := 8\eta^2 q\,\Lambda_q(\tau)\|\nabla f(X^{(k)})\|_F^2 + \frac{6\tau\eta^2}{b}\sigma^2 + 4\eta^2\big(c_\delta^2\,\Lambda_q(\tau) + 6\widetilde{L}^2\tau\big)\epsilon^2,$$
$$\beta := 8\eta^2\widetilde{L}^2\left(\frac{\tau}{q^2}+\frac{4}{q}\right).$$

Since $\beta\tau = 8\eta^2\widetilde{L}^2\big(\frac{\tau^2}{q^2}+\frac{4\tau}{q}\big) \le 32\,\eta^2\widetilde{L}^2\Lambda_q(\tau)$, condition (38) implies $\beta\tau \le 1$. By the discrete Grönwall inequality, for all $s \le \tau$,

$$\phi_s \;\le\; \alpha(1+\beta)^s \;\le\; \alpha(1+\beta)^\tau \;\le\; e^{\beta\tau}\alpha \;\le\; e\,\alpha.$$

Setting

$$C_1 := 8e, \qquad C_2 := 6e, \qquad C_3 := 4e, \tag{47}$$

we obtain (37). □

E.5 One-period descent

The next lemma proves that one full period decreases the objective by a quantity proportional to the projected gradient norm, up to higher-order drift and stochastic error terms.

Lemma 9 (One-period descent). *Under Assumptions 1–3, there exist absolute constants $c_1, c_2, c_3 > 0$ and a constant $c_4 > 0$ depending only on c_δ such that*

$$\mathbb{E}\big[f(X^{(k+1)}) - f(X^{(k)}) \mid X^{(k)}\big] \le -\left(c_1 - c_2\eta\widetilde{L}\left(1+\frac{\tau}{q}\right)\right)\eta\tau\,\|\nabla f(X^{(k)})\|_F^2 + c_3\widetilde{L}\,\frac{\eta^2\tau}{b}\,\sigma^2 + c_4\left(\frac{\eta\tau}{q}+\widetilde{L}\eta^2\Lambda_q(\tau)\right)\epsilon^2. \tag{48}$$

provided (38) holds.

Proof. By Proposition 1, the final point of period k satisfies

$$X^{(k+1)} = X^{(k)} + U_k B_{k,\tau} V_k^\top.$$

Therefore,

$$f(X^{(k+1)}) - f(X^{(k)}) = g_k(B_{k,\tau}) - g_k(0). \tag{49}$$

By the $\widetilde{L}$-smoothness of g_k,

$$g_k(B_{k,\tau}) - g_k(0) \le \langle \nabla g_k(0),\, B_{k,\tau}\rangle + \frac{\widetilde{L}}{2}\|B_{k,\tau}\|_F^2. \tag{50}$$

We now analyze the two terms on the right-hand side.

Step 1: the linear term. Using (39),

$$\langle \nabla g_k(0),\, B_{k,\tau} \rangle = -\eta \sum_{s=0}^{\tau-1} \langle \nabla g_k(0),\, \widehat{\nabla} g_k(B_{k,s}; \xi_{k,s}, J_{k,s}) \rangle. \tag{51}$$

Write $\mathbb{E}_k[\cdot] := \mathbb{E}[\cdot \mid X^{(k)}]$ for brevity. Taking conditional expectation, towering through $\mathcal{F}_{k,s}$ (with respect to which $\nabla g_k(0)$ is measurable), and using (34),

$$\begin{aligned}
\mathbb{E}_k\big[\langle \nabla g_k(0),\, B_{k,\tau} \rangle\big] &= -\frac{\eta}{q} \sum_{s=0}^{\tau-1} \mathbb{E}_k\big[\langle \nabla g_k(0),\, \nabla g_k(B_{k,s}) + \delta_k(B_{k,s}) \rangle\big] \\
&= -\frac{\eta\tau}{q}\, \mathbb{E}_k\big[\|\nabla g_k(0)\|_F^2\big] \\
&\quad - \frac{\eta}{q} \sum_{s=0}^{\tau-1} \mathbb{E}_k\big[\langle \nabla g_k(0),\, \nabla g_k(B_{k,s}) - \nabla g_k(0) \rangle\big] \\
&\quad - \frac{\eta}{q} \sum_{s=0}^{\tau-1} \mathbb{E}_k\big[\langle \nabla g_k(0),\, \delta_k(B_{k,s}) \rangle\big].
\end{aligned} \tag{52}$$

We bound the drift term using Cauchy–Schwarz and Lemma 4:

$$\big|\langle \nabla g_k(0),\, \nabla g_k(B_{k,s}) - \nabla g_k(0) \rangle\big| \le \widetilde{L}\, \|\nabla g_k(0)\|_F\, \|B_{k,s}\|_F. \tag{53}$$

Applying Young's inequality,

$$\widetilde{L}\, \|\nabla g_k(0)\|_F\, \|B_{k,s}\|_F \le \frac{1}{4} \|\nabla g_k(0)\|_F^2 + \widetilde{L}^2 \|B_{k,s}\|_F^2. \tag{54}$$

Similarly, the bias term is bounded by

$$\big|\langle \nabla g_k(0),\, \delta_k(B_{k,s}) \rangle\big| \le \frac{1}{4} \|\nabla g_k(0)\|_F^2 + c_\delta^2 \epsilon^2. \tag{55}$$

Bounding each inner-product term in (52) pointwise by (54) and (55), summing over $s = 0, \dots, \tau-1$, and applying Lemma 11 in the form $\mathbb{E}_k \|\nabla g_k(0)\|_F^2 = q \|\nabla f(X^{(k)})\|_F^2$,

$$\begin{aligned}
\mathbb{E}_k\big[\langle \nabla g_k(0),\, B_{k,\tau} \rangle\big] &\le -\eta\tau \|\nabla f(X^{(k)})\|_F^2 + \frac{\eta\tau}{2} \|\nabla f(X^{(k)})\|_F^2 + \frac{\eta \widetilde{L}^2}{q} \sum_{s=0}^{\tau-1} \mathbb{E}_k\big[\|B_{k,s}\|_F^2\big] \\
&\quad + \frac{\eta\tau}{q} c_\delta^2 \epsilon^2 \\
&= -\frac{\eta\tau}{2} \|\nabla f(X^{(k)})\|_F^2 + \frac{\eta \widetilde{L}^2}{q} \sum_{s=0}^{\tau-1} \mathbb{E}_k\big[\|B_{k,s}\|_F^2\big] + \frac{\eta\tau}{q} c_\delta^2 \epsilon^2.
\end{aligned} \tag{56}$$

Applying Lemma 8 to each index $s = 0, \dots, \tau - 1$ and summing,

$$\sum_{s=0}^{\tau-1} \mathbb{E}_k\big[\|B_{k,s}\|_F^2\big] \;\le\; \tau\Big(C_1 \eta^2 q\, \Lambda_q(\tau) \|\nabla f(X^{(k)})\|_F^2 + C_2 \eta^2 \frac{\tau}{b} \sigma^2 + C_3 \eta^2 \big(c_\delta^2 \Lambda_q(\tau) + 6 \widetilde{L}^2 \tau \big) \epsilon^2 \Big). \tag{57}$$

Substituting (57) into (56), we obtain the Step 1 result

$$\begin{aligned}
\mathbb{E}_k\big[\langle \nabla g_k(0),\, B_{k,\tau} \rangle\big] &\le -\frac{\eta\tau}{2} \|\nabla f(X^{(k)})\|_F^2 + C_1 \widetilde{L}^2 \tau \eta^3 \Lambda_q(\tau) \|\nabla f(X^{(k)})\|_F^2 \\
&\quad + C_2 \widetilde{L}^2 \frac{\tau^2 \eta^3}{qb} \sigma^2 + \left(C_3 \widetilde{L}^2 \frac{\tau \eta^3}{q} \big(c_\delta^2 \Lambda_q(\tau) + 6 \widetilde{L}^2 \tau \big) + c_\delta^2 \frac{\eta\tau}{q} \right) \epsilon^2.
\end{aligned} \tag{58}$$

Step 2: the quadratic smoothness remainder. Applying Lemma 8 with $s = \tau$,

$$\mathbb{E}_k\left[\frac{\widetilde{L}}{2} \|B_{k,\tau}\|_F^2 \right] \le \frac{\widetilde{L}}{2} \Big(C_1 \eta^2 q\, \Lambda_q(\tau) \|\nabla f(X^{(k)})\|_F^2 + C_2 \eta^2 \frac{\tau}{b} \sigma^2 + C_3 \eta^2 \big(c_\delta^2 \Lambda_q(\tau) + 6 \widetilde{L}^2 \tau \big) \epsilon^2 \Big). \tag{59}$$

Combining Steps 1 and 2. Adding (58) and (59) via (50), the Grönwall condition (38) gives $32\widetilde{L}^2\eta^2\Lambda_q(\tau) \le 1$, and in particular $\widetilde{L}\eta\tau/q = \sqrt{\tau/q}\,\big(\widetilde{L}\eta\sqrt{\Lambda_q(\tau)}\big)/\sqrt{1+\tau/q} \le 1/\sqrt{32}$. Each cubic-in-$\eta$ correction in (58) equals the corresponding $\widetilde{L}\eta^2$ term of (59) multiplied by $2\widetilde{L}\eta\tau/q \le 1/(2\sqrt{2})$, and is therefore dominated by it. Moreover, since $c_\delta^2 = q\widetilde{L}^2/4$ (Lemma 5), $6\widetilde{L}^2\tau = 24\,c_\delta^2\,\frac{\tau}{q} \le 24\,c_\delta^2\,\Lambda_q(\tau)$. Consequently, there exist absolute constants $\bar{C}_1, \bar{C}_2, \bar{C}_3 > 0$ such that

$$\mathbb{E}_k\big[f(X^{(k+1)}) - f(X^{(k)})\big] \le -\frac{\eta\tau}{2}\|\nabla f(X^{(k)})\|_F^2 + \bar{C}_1\widetilde{L}\eta^2 q\,\Lambda_q(\tau)\|\nabla f(X^{(k)})\|_F^2 + \bar{C}_2\widetilde{L}\,\frac{\eta^2\tau}{b}\sigma^2 + \left(c_\delta^2\frac{\eta\tau}{q} + \bar{C}_3\,c_\delta^2\,\widetilde{L}\eta^2\Lambda_q(\tau)\right)\epsilon^2. \tag{60}$$

Since $q\,\Lambda_q(\tau) = \tau(1+\frac{\tau}{q})$, the second term on the right-hand side equals $\bar{C}_1\widetilde{L}\eta(1+\frac{\tau}{q})\cdot\eta\tau\|\nabla f(X^{(k)})\|_F^2$. Identifying with (48),

$$c_1 := \tfrac{1}{2}, \qquad c_2 := \bar{C}_1, \qquad c_3 := \bar{C}_2, \qquad c_4 := \bar{C}_3\,c_\delta^2$$

yields (48). □

E.6 Recovery of the full gradient in expectation

We now relate the subspace gradient $\nabla g_k(0) = U_k^\top \nabla f(X^{(k)})V_k$ to the full gradient $\nabla f(X^{(k)})$.

Lemma 10 (Coverage of sphere-normalized subspaces)**.** *Under Assumption 2, for every deterministic matrix $G \in \mathbb{R}^{d_1\times d_2}$,*

$$\mathbb{E}_{U_k,V_k}\|U_k^\top G V_k\|_F^2 = r_1 r_2\|G\|_F^2. \tag{61}$$

Proof. By Lemma 3, each column satisfies $\mathbb{E}[u_{k,a}u_{k,a}^\top] = I_{d_1}$ and $\mathbb{E}[v_{k,b}v_{k,b}^\top] = I_{d_2}$, and U_k and V_k are independent; the computation below uses only these facts and requires no independence across columns. Therefore,

$$\begin{aligned}\mathbb{E}_{U_k,V_k}\|U_k^\top G V_k\|_F^2 &= \sum_{a=1}^{r_1}\sum_{b=1}^{r_2}\mathbb{E}\big[(u_{k,a}^\top G v_{k,b})^2\big]\\ &= \sum_{a=1}^{r_1}\sum_{b=1}^{r_2}\mathbb{E}\big[u_{k,a}^\top G\,\mathbb{E}[v_{k,b}v_{k,b}^\top]\,G^\top u_{k,a}\big]\\ &= \sum_{a=1}^{r_1}\sum_{b=1}^{r_2}\mathbb{E}\big[u_{k,a}^\top G G^\top u_{k,a}\big] = r_1 r_2\|G\|_F^2.\end{aligned}$$

□

Lemma 11 (Projected gradient recovers the full gradient in expectation)**.** *Under Assumption 2,*

$$\mathbb{E}_{U_k,V_k}\|\nabla g_k(0)\|_F^2 = \mathbb{E}_{U_k,V_k}\|U_k^\top\nabla f(X^{(k)})V_k\|_F^2 = r_1 r_2\|\nabla f(X^{(k)})\|_F^2. \tag{62}$$

Proof. The equality is just Lemma 1 evaluated at $B = 0$. Since U_k and V_k are sampled independently of $X^{(k)}$, Lemma 10 applies conditionally on $X^{(k)}$ with $G = \nabla f(X^{(k)})$. □

E.7 Global convergence theorem

We are now ready to derive the global convergence result.

Proposition 2 (Convergence of the subspace coordinate-wise gradient estimator)**.** *Under Assumptions 1–3, let $q := r_1 r_2$, $T := K\tau$, $\widetilde{L} := L\rho_U^2\rho_V^2$ and $\Lambda_q(\tau) := \frac{\tau}{q}(1+\frac{\tau}{q})$. If $\eta \le \big(\widetilde{L}\sqrt{32\,\Lambda_q(\tau)}\big)^{-1}$ and $c_2\eta\widetilde{L}(1+\tau/q) \le c_1/2$, then*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla f(X_{k\tau})\|_F^2 \;\le\; \underbrace{\frac{2\Delta_0}{c_1\eta T}}_{\text{optimization}} + \underbrace{\frac{2c_3\widetilde{L}\eta\sigma^2}{c_1 b}}_{\text{stochastic noise}} + \underbrace{\frac{2c_4}{c_1 q}\left[1+\widetilde{L}\eta\left(1+\frac{\tau}{q}\right)\right]\epsilon^2}_{\text{finite difference bias}},$$

where $\Delta_0 := f(X_0) - f^\star$, $c_1, c_2, c_3 > 0$ are universal constants, and $c_4 > 0$ is a problem-dependent constant that may depend on the subspace dimension q and the effective smoothness parameter $\widetilde{L}$ (specifically, $c_4 = O(c_\delta^2)$ with $c_\delta = \sqrt{q}\,\widetilde{L}/2$).

Remark 3 (Constants and scaling in Proposition 2). The quantities appearing in the bound of Proposition 2 are

$$q = r_1 r_2, \qquad \Lambda_q(\tau) = \frac{\tau}{q}\left(1 + \frac{\tau}{q}\right), \qquad \widetilde{L} = L\rho_U^2\rho_V^2.$$

Under the conditioned sphere-normalized sampling of Assumption 2,

$$\rho_U^2\rho_V^2 = 16\, d_1 d_2, \qquad \widetilde{L} = 16\, L d_1 d_2,$$

so that $\widetilde{L}$ is independent of the subspace dimension q, and the stochastic-noise coefficient of Proposition 2 is the dimension-free minibatch variance σ^2 itself. The finite-difference bias constant is

$$c_\delta = \frac{\sqrt{q}\,\widetilde{L}}{2}, \qquad c_\delta^2 = O(q\widetilde{L}^2) \;= O\big(qL^2 d_1^2 d_2^2\big).$$

The descent constants satisfy $c_1 = 1/2$, $c_2, c_3 = O(1)$, and $c_4 = O(c_\delta^2)$. Hence Proposition 2 can be read as

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\|\nabla f(X_{k\tau})\|_F^2 \le O\left(\frac{\Delta_0}{\eta T} + \frac{\widetilde{L}\eta\sigma^2}{b} + \frac{c_\delta^2}{q}\left[1 + \widetilde{L}\eta\left(1 + \frac{\tau}{q}\right)\right]\epsilon^2\right).$$

Equivalently, since $c_\delta^2/q = O(\widetilde{L}^2)$, the finite-difference term is $O(\widetilde{L}^2[1 + \widetilde{L}\eta(1 + \tau/q)]\epsilon^2)$. The stepsize condition is

$$\eta = O\left(\min\left\{\frac{1}{\widetilde{L}\sqrt{\Lambda_q(\tau)}}, \frac{1}{\widetilde{L}(1 + \tau/q)}\right\}\right).$$

Proof. By the tower property,

$$\mathbb{E}\big[f(X^{(k+1)}) - f(X^{(k)})\big] = \mathbb{E}\Big[\mathbb{E}\big[f(X^{(k+1)}) - f(X^{(k)}) \;\big|\; X^{(k)}\big]\Big].$$

The first stepsize condition of Proposition 2, $\eta \le 1/(\widetilde{L}\sqrt{32\,\Lambda_q(\tau)})$, ensures that the one-period descent lemma can be applied. By Lemma 9,

$$\begin{aligned}\mathbb{E}\big[f(X^{(k+1)}) - f(X^{(k)})\big] \le &-\left(c_1 - c_2\eta\widetilde{L}\left(1 + \frac{\tau}{q}\right)\right)\eta\tau\,\mathbb{E}\|\nabla f(X^{(k)})\|_F^2 \\ &+ c_3\widetilde{L}\,\frac{\eta^2\tau}{b}\,\sigma^2 + c_4\left(\frac{\eta\tau}{q} + \widetilde{L}\eta^2\Lambda_q(\tau)\right)\epsilon^2. \end{aligned} \tag{63}$$

By the second stepsize condition of Proposition 2, $c_2\eta\widetilde{L}(1 + \tau/q) \le c_1/2$,

$$c_1 - c_2\eta\widetilde{L}\left(1 + \frac{\tau}{q}\right) \;\ge\; \frac{c_1}{2},$$

so that

$$\mathbb{E}\big[f(X^{(k+1)}) - f(X^{(k)})\big] \le -\frac{c_1}{2}\,\eta\tau\,\mathbb{E}\|\nabla f(X^{(k)})\|_F^2 + c_3\widetilde{L}\,\frac{\eta^2\tau}{b}\,\sigma^2 + c_4\left(\frac{\eta\tau}{q} + \widetilde{L}\eta^2\Lambda_q(\tau)\right)\epsilon^2. \tag{64}$$

Summing (64) over $k = 0, 1, \ldots, K-1$ and telescoping,

$$\frac{c_1}{2}\,\eta\tau\sum_{k=0}^{K-1}\mathbb{E}\|\nabla f(X^{(k)})\|_F^2 \le \Delta_0 + Kc_3\widetilde{L}\,\frac{\eta^2\tau}{b}\,\sigma^2 + Kc_4\left(\frac{\eta\tau}{q} + \widetilde{L}\eta^2\Lambda_q(\tau)\right)\epsilon^2. \tag{65}$$

Dividing both sides by $K(c_1/2)\eta\tau$ gives

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla f(X^{(k)})\|_F^2 \le \frac{2\Delta_0}{c_1\eta\tau K} + \frac{2c_3}{c_1}\,\frac{\widetilde{L}\eta}{b}\,\sigma^2 + \frac{2c_4}{c_1 q}\left[1 + \frac{q\widetilde{L}\eta\Lambda_q(\tau)}{\tau}\right]\epsilon^2. \tag{66}$$

Since $\frac{q\Lambda_q(\tau)}{\tau} = 1 + \frac{\tau}{q}$, $X^{(k)} = X_{k\tau}$, and $T = K\tau$, equation (66) is exactly the bound stated in Proposition 2. □

F SeedFlood as Data-Parallel Subspace Zeroth-Order Optimization

We now extend the single-client analysis to the idealized decentralized SeedFlood setting. The role of flooding is separated from the optimization analysis: we assume that, at each inner step, flooding disseminates all client-side zeroth-order estimators exactly before the model update is applied. Under this exact-flooding idealization, the optimization dynamics are the same as centralized data-parallel zeroth-order optimization over the shared low-rank subspace with n independent client estimators.

F.1 Decentralized setup and exact flooding

Let n clients have local objectives

$$f_i(X) := \mathbb{E}_{\xi\sim\mathcal{D}_i}[F_i(X;\xi)], \qquad f(X) := \frac{1}{n}\sum_{i=1}^{n} f_i(X).$$

For the data-parallel reduction, all clients use the same period-wise subspace U_k, V_k, generated for instance from a shared random seed, as in Assumption 2. For each period k, define

$$g_{i,k}(B) := f_i(X^{(k)} + U_k B V_k^\top), \qquad g_k(B) := \frac{1}{n}\sum_{i=1}^{n} g_{i,k}(B).$$

Thus $g_k(B) = f(X^{(k)} + U_k B V_k^\top)$, so the global subspace objective is exactly the average of the local subspace objectives.

Assumption 4 (Per-client smoothness, bounded noise, and heterogeneity)**.** f is bounded below by $f^\star$, For each client i, every realization $F_i(\cdot;\xi)$ is L-smooth (hence f_i is L-smooth), stochastic gradients are unbiased with variance $\le \sigma^2/b$, and $\frac{1}{n}\sum_i \|\nabla f_i(X) - \nabla f(X)\|_F^2 \le \zeta^2$ for all X. The global objective $f = \frac{1}{n}\sum_i f_i$ is bounded below by $f^\star$.

Equivalently, for each client i we assume that every realization $F_i(\cdot;\xi)$ is L-smooth,

$$\|\nabla F_i(X;\xi) - \nabla F_i(Y;\xi)\|_F \le L\|X - Y\|_F, \qquad \forall X, Y, \forall \xi, \tag{67}$$

so that f_i is L-smooth by Jensen's inequality, and mini-batch stochastic gradients are unbiased with bounded variance,

$$\mathbb{E}_{\xi\sim\mathcal{D}_i}[\nabla F_i(X;\xi)] = \nabla f_i(X), \qquad \mathbb{E}_{\xi\sim\mathcal{D}_i}\|\nabla F_i(X;\xi) - \nabla f_i(X)\|_F^2 \le \frac{\sigma^2}{b},$$

and that the local gradients have bounded dissimilarity,

$$\frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(X) - \nabla f(X)\|_F^2 \le \zeta^2, \qquad \forall X. \tag{68}$$

By Lemma 1 applied to each local objective and the operator norm bound in Assumption 2, Assumption 4 implies the coefficient-space heterogeneity bound

$$\frac{1}{n}\sum_{i=1}^{n} \|\nabla g_{i,k}(B) - \nabla g_k(B)\|_F^2 \le \widetilde{\zeta}^2, \qquad \widetilde{\zeta}^2 := \rho_U^2 \rho_V^2 \zeta^2. \tag{69}$$

At inner step s, client i samples an independent coordinate

$$J_{i,k,s} \sim \mathrm{Unif}\{1,\dots,q\}, \qquad C_{i,k,s} = E_{J_{i,k,s}},$$

and computes the local coefficient

$$a_{i,k,s} := \frac{G_{i,k}(B_{k,s} + \epsilon C_{i,k,s}; \xi_{i,k,s}) - G_{i,k}(B_{k,s} - \epsilon C_{i,k,s}; \xi_{i,k,s})}{2\epsilon},$$

where

$$G_{i,k}(B;\xi) := F_i(X^{(k)} + U_k B V_k^\top; \xi).$$

The local coefficient-space estimator is

$$\widehat{\nabla} g_{i,k}(B_{k,s}; \xi_{i,k,s}, J_{i,k,s}) := a_{i,k,s} E_{J_{i,k,s}}.$$

Exact flooding averages these estimators:

$$\widehat{\nabla}_{\mathrm{SF}} g_k(B_{k,s}) := \frac{1}{n} \sum_{i=1}^{n} \widehat{\nabla} g_{i,k}(B_{k,s}; \xi_{i,k,s}, J_{i,k,s}). \tag{70}$$

The common SeedFlood iterate then satisfies

$$B_{k,s+1} = B_{k,s} - \eta \, \widehat{\nabla}_{\mathrm{SF}} g_k(B_{k,s}), \qquad X_{k,s} = X^{(k)} + U_k B_{k,s} V_k^\top. \tag{71}$$

Proposition 3 (Exact flooding yields data-parallel subspace zeroth-order updates)**.** *Suppose all clients start inner step s from the same model $X_{k,s}$, and flooding returns the exact average (70) before the update. Then all clients remain synchronized, and the common model follows (71). Hence the decentralized optimization dynamics are identical to centralized data-parallel zeroth-order optimization over the shared low-rank subspace.*

Proof. Each local update direction has the form $U_k \widehat{\nabla} g_{i,k}(B_{k,s}; \xi_{i,k,s}, J_{i,k,s}) V_k^\top$. Exact flooding replaces the local direction by their average. Therefore the full-space update is

$$X_{k,s+1} = X_{k,s} - \eta U_k \left(\frac{1}{n} \sum_{i=1}^{n} \widehat{\nabla} g_{i,k}(B_{k,s}; \xi_{i,k,s}, J_{i,k,s}) \right) V_k^\top,$$

which is exactly (71) under the representation $X_{k,s} = X^{(k)} + U_k B_{k,s} V_k^\top$. □

F.2 Averaged estimator moments

For the exact-flooding analysis, the effective inner-loop drift scale is

$$\Lambda_{q,n}(\tau) := \frac{\tau^2}{q^2} + \frac{\tau}{nq}. \tag{72}$$

The first term is the deterministic drift caused by repeatedly moving in the mean direction $\nabla g_k / q$ inside a fixed subspace. The second term comes from the averaged coordinate-sampling fluctuation and is reduced by n.

Lemma 12 (Mean and second moment under exact flooding)**.** *Under Assumptions 2 and 4, and the coordinate finite-difference construction above, the averaged SeedFlood estimator satisfies*

$$\mathbb{E}\Big[\widehat{\nabla}_{\mathrm{SF}} g_k(B) \mid X^{(k)}, U_k, V_k, B \Big] = \frac{1}{q} \left(\nabla g_k(B) + \delta_k(B) \right), \tag{73}$$

where $\|\delta_k(B)\|_F \le c_\delta \epsilon$. Moreover, let $\mathcal{F}_{k,s} := \sigma\big(X^{(k)}, U_k, V_k, \{\xi_{i,k,\ell}, J_{i,k,\ell}\}_{i\in[n],\, \ell<s}\big)$; if B is $\mathcal{F}_{k,s}$-measurable and $(\xi_{i,k,s}, J_{i,k,s})_{i\in[n]}$ is the fresh per-client randomness of step s, then

$$\begin{aligned} \mathbb{E}\Big[\|\widehat{\nabla}_{\mathrm{SF}} g_k(B)\|_F^2 \mid X^{(k)} \Big] \le{}& \left(\frac{2}{q^2} + \frac{2}{nq} \right) \mathbb{E}\big[\|\nabla g_k(B)\|_F^2 \mid X^{(k)} \big] + \frac{2}{nq} \widetilde{\zeta}^2 \\ &+ 2c_\delta^2 \left(\frac{1}{q^2} + \frac{1}{nq} \right) \epsilon^2 + \frac{3\sigma^2}{nb} + \frac{12\widetilde{L}^2}{nq} \mathbb{E}\big[\|B\|_F^2 \mid X^{(k)} \big] + \frac{12\widetilde{L}^2}{n} \epsilon^2. \end{aligned} \tag{74}$$

Proof. Write

$$h_i := \nabla g_{i,k}(B), \qquad h := \nabla g_k(B) = \frac{1}{n} \sum_{i=1}^{n} h_i,$$

and let

$$Z_i := \widehat{\nabla} g_{i,k}(B; \xi_i, J_i), \qquad Z := \widehat{\nabla}_{\mathrm{SF}} g_k(B) = \frac{1}{n} \sum_{i=1}^{n} Z_i.$$

For each client, the standard CGE finite-difference property gives

$$\mu_i := \mathbb{E}[Z_i \mid X^{(k)}, U_k, V_k, B] = \frac{1}{q}(h_i + \delta_{i,k}(B)), \qquad \|\delta_{i,k}(B)\|_F \le c_\delta \epsilon.$$

Averaging over clients yields (73) with $\delta_k(B) = \frac{1}{n}\sum_i \delta_{i,k}(B)$.

For the second moment, decompose the average estimator into its $\mathcal{F}_{k,s}$-conditional mean and centered fluctuations:

$$\bar{\mu} := \mathbb{E}[Z \mid \mathcal{F}_{k,s}] = \frac{1}{n}\sum_{i=1}^n \mu_i = \frac{1}{q}(h+\delta_k(B)), \qquad \mathbb{E}[\|Z\|_F^2 \mid \mathcal{F}_{k,s}] = \|\bar{\mu}\|_F^2 + \mathbb{E}[\|Z-\bar{\mu}\|_F^2 \mid \mathcal{F}_{k,s}].$$

The mean term satisfies, pointwise on $\mathcal{F}_{k,s}$,

$$\|\bar{\mu}\|_F^2 = \frac{1}{q^2}\|h + \delta_k(B)\|_F^2 \le \frac{2}{q^2}\|h\|_F^2 + \frac{2c_\delta^2}{q^2}\epsilon^2. \tag{75}$$

Because the clients use independent fresh mini-batches and coordinates, the centered variables $Z_i - \mu_i$ are conditionally independent given $\mathcal{F}_{k,s}$ and have zero mean, so the cross terms vanish:

$$\mathbb{E}[\|Z - \bar{\mu}\|_F^2 \mid \mathcal{F}_{k,s}] = \frac{1}{n^2}\sum_{i=1}^n \mathbb{E}[\|Z_i - \mu_i\|_F^2 \mid \mathcal{F}_{k,s}] \le \frac{1}{n^2}\sum_{i=1}^n \mathbb{E}[\|Z_i\|_F^2 \mid \mathcal{F}_{k,s}]. \tag{76}$$

By Assumption 4, each pair $(g_{i,k}, G_{i,k})$ satisfies the hypotheses of Lemma 6 and Lemma 7: every realization $F_i(\cdot;\xi)$ is L-smooth (67), mini-batch gradients are unbiased with variance at most σ^2/b, and the anchor $\Delta_0^{(i)} := \nabla F_i(X^{(k)};\xi_{i,k,s}) - \nabla f_i(X^{(k)})$ is independent of (U_k, V_k). Hence, taking $\mathbb{E}[\cdot \mid X^{(k)}]$ of (76) via the tower property and applying (35) verbatim to each client,

$$\mathbb{E}[\|Z - \bar{\mu}\|_F^2 \mid X^{(k)}] \le \frac{2}{nq}\left(\frac{1}{n}\sum_{i=1}^n \mathbb{E}\big[\|h_i\|_F^2 \mid X^{(k)}\big]\right) + \frac{2c_\delta^2}{nq}\epsilon^2 + \frac{3\sigma^2}{nb} + \frac{12\widetilde{L}^2}{nq}\,\mathbb{E}\big[\|B\|_F^2 \mid X^{(k)}\big] + \frac{12\widetilde{L}^2}{n}\epsilon^2. \tag{77}$$

Finally, since $h = \frac{1}{n}\sum_i h_i$, pointwise

$$\frac{1}{n}\sum_{i=1}^n \|h_i\|_F^2 = \|h\|_F^2 + \frac{1}{n}\sum_{i=1}^n \|h_i - h\|_F^2 \le \|h\|_F^2 + \widetilde{\zeta}^2,$$

where the last step uses (69). Taking $\mathbb{E}[\cdot \mid X^{(k)}]$ of (75) and combining with (77) proves (74). □

F.3 SeedFlood convergence

Theorem 1 (Convergence of SeedFlood under exact flooding)**.** *Under Assumptions 2, 3, and 4 with exact flooding, let* $\Lambda_{q,n}(\tau) := \tau^2/q^2 + \tau/(nq)$. *If* $\eta \le \big(\widetilde{L}\sqrt{32\,\Lambda_{q,n}(\tau)}\big)^{-1}$ *and* $c_2\eta\widetilde{L}(\tau/q + 1/n) \le c_1/2$, *then*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\nabla f(X_{k\tau})\|_F^2 \;\le\; \frac{2\Delta_0}{c_1\eta T} + a_\sigma\widetilde{L}\eta\frac{\sigma^2}{nb} + a_\zeta\widetilde{L}\eta\frac{\widetilde{\zeta}^2}{qn} + a_\epsilon\frac{c_\delta^2}{q}\left[1 + \widetilde{L}\eta\left(\frac{\tau}{q} + \frac{1}{n}\right)\right]\epsilon^2,$$

where $\widetilde{\zeta}^2 := \rho_U^2\rho_V^2\zeta^2$, $c_\delta := \sqrt{q}\,\widetilde{L}/2$, *and* $a_\sigma, a_\zeta, a_\epsilon > 0$ *are universal constants.*

Remark 4 (Constants and scaling in Theorem 1). The constants $q, \widetilde{L}$, and c_δ are the same as in Proposition 2. The effective SeedFlood drift scale is

$$\Lambda_{q,n}(\tau) = \frac{\tau^2}{q^2} + \frac{\tau}{nq}.$$

The additional heterogeneity constant is

$$\widetilde{\zeta}^2 := \rho_U^2\rho_V^2\zeta^2,$$

so under the conditioned sphere-normalized sampling of Assumption 2,

$$\widetilde{\zeta}^2 = 16\, d_1 d_2\, \zeta^2.$$

In particular, the heterogeneity contribution $\widetilde{\zeta}^2/(qn) = 16\, d_1 d_2 \zeta^2/(qn)$ in Theorem 1 decreases with the subspace dimension q. Thus Theorem 1 can be written in big-O form as

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\|\nabla f(X_{k\tau})\|_F^2$$
$$\leq O\left(\frac{\Delta_0}{\eta T} + \widetilde{L}\eta\left(\frac{\sigma^2}{nb} + \frac{\widetilde{\zeta}^2}{qn}\right) + \frac{c_\delta^2}{q}\left[1 + \widetilde{L}\eta\left(\frac{\tau}{q} + \frac{1}{n}\right)\right]\epsilon^2\right).$$

The admissible stepsize conditions now improve with n, because averaging independent client estimators also reduces the coordinate-sampling fluctuation. Equivalently,

$$\eta = O\left(\min\left\{\frac{1}{\widetilde{L}\sqrt{\tau^2/q^2 + \tau/(nq)}}, \frac{1}{\widetilde{L}(\tau/q + 1/n)}\right\}\right).$$

Thus increasing n can reduce the first optimization-transient term $\Delta_0/(\eta T)$ indirectly by allowing a larger stable stepsize. This improvement saturates at the deterministic fixed-subspace drift scale $\eta = O(q/(\widetilde{L}\tau))$. In addition, larger n directly reduces the stochastic variance and client-heterogeneity terms by $1/n$. The deterministic finite-difference bias itself is not reduced by n, although its drift amplification is affected through the larger stable stepsize range.

Proof. The proof repeats the single-client proof of Proposition 2 with $\widehat{\nabla} g_k$ replaced by the averaged estimator $\widehat{\nabla}_{\mathrm{SF}} g_k$, all conditional expectations taken with respect to $X^{(k)}$ alone, and Lemma 11 applied inside the drift and descent arguments exactly as in Lemmas 8 and 9. The conditional mean (73) is identical to the single-client mean with g_k now equal to the global subspace objective, so the descent term becomes $-\eta\tau\|\nabla f(X^{(k)})\|_F^2$ up to the same corrections as in the single-client case.

For the stochastic part we use, as in the single-client analysis, the orthogonal decomposition $\mathbb{E}[\|Z\|_F^2 \mid \mathcal{F}_{k,s}] = \|\bar{\mu}\|_F^2 + \mathbb{E}[\|Z - \bar{\mu}\|_F^2 \mid \mathcal{F}_{k,s}]$ established in the proof of Lemma 12, rather than the uncentered bound (74), so that the deterministic mean is not conflated with the zero-mean fluctuation. The centered fluctuations across inner steps form a martingale-difference sequence with respect to $\{\mathcal{F}_{k,j}\}$, and the orthogonality survives the coarser conditioning on $X^{(k)}$ exactly as in Lemma 8. The mean component (75) contributes the deterministic signal/drift terms; the anchored centered-variance bound (77) contributes the dimension-free noise floor $3\sigma^2/(nb)$, the heterogeneity term $\widetilde{\zeta}^2/(nq)$, and a drift correction $12\widetilde{L}^2\,\mathbb{E}\|B\|_F^2/(nq)$ that enters the Grönwall recursion at the same order in (q, n) as the signal terms, changing only absolute constants; the induced Grönwall coefficient satisfies $\beta\tau \leq 32\,\eta^2\widetilde{L}^2\Lambda_{q,n}(\tau)$, which is the origin of the stepsize condition $\eta \leq 1/(\widetilde{L}\sqrt{32\,\Lambda_{q,n}(\tau)})$. The remaining $\widetilde{L}^2\epsilon^2/n$ correction is absorbed into the finite-difference bracket via $c_\delta^2 = q\widetilde{L}^2/4$, as in the proof of Lemma 9. Propagating these two components through the arguments of Lemmas 8 and 9 gives, for universal numerical constants $\bar{a}_\sigma, \bar{a}_\zeta, \bar{a}_\epsilon > 0$, the one-period bound

$$\mathbb{E}[f(X^{(k+1)}) - f(X^{(k)}) \mid X^{(k)}] \leq -\left(c_1 - c_2\eta\widetilde{L}\left(\frac{\tau}{q} + \frac{1}{n}\right)\right)\eta\tau\,\|\nabla f(X^{(k)})\|_F^2$$
$$+\bar{a}_\sigma\widetilde{L}\eta^2\tau\,\frac{\sigma^2}{nb} + \bar{a}_\zeta\widetilde{L}\eta^2\tau\,\frac{\widetilde{\zeta}^2}{qn} + \bar{a}_\epsilon c_\delta^2\left(\frac{\eta\tau}{q} + \widetilde{L}\eta^2\Lambda_{q,n}(\tau)\right)\epsilon^2.$$

The drift amplification factor is the SeedFlood drift scale $\Lambda_{q,n}(\tau) = \tau^2/q^2 + \tau/(nq)$: the τ^2/q^2 term comes from the deterministic mean component (75) (which is not reduced by averaging over clients), while the $\tau/(nq)$ term comes from the centered variance (77) (whose client-averaged contribution is reduced by $1/n$).

The stated result follows by summing the one-period bound over $k = 0, \ldots, K-1$, using $f(X) \geq f^\star$, and dividing by $K(c_1/2)\eta\tau$, with

$$a_\sigma := \frac{2\bar{a}_\sigma}{c_1}, \qquad a_\zeta := \frac{2\bar{a}_\zeta}{c_1}, \qquad a_\epsilon := \frac{2\bar{a}_\epsilon}{c_1}.$$

□

G EXPERIMENT SETUP

G.1 DATASET

Due to computational constraints, we evaluate our method on a subset of SuperGLUE tasks [53]. For experiments with OPT-1.3B using 16 clients, we consider BoolQ [8], MultiRC [24], ReCoRD [60], RTE [9, 4, 17], and WiC[36], and additionally include SST-2 [42].

For experiments with OPT-125M and larger client counts {16, 32, 64, 128 }, we restrict evaluation to SST-2 [42], BoolQ [8], and RTE [9, 4, 17] to ensure feasibility.

For each task, we randomly sample 1,024 training examples and distribute them uniformly across clients. We use a fixed global validation set of 500 examples and a test set of 1,000 examples.

We use the same prompts as Malladi et al. [34] for all tasks, and apply them consistently across both zeroth-order and first-order methods.

G.2 HYPERPARAMETERS

We report the hyperparameter settings used in Sections 5.2 and 5.3. Following prior work [34, 58], we do not apply momentum and weight decay to local SGD updates. To ensure feasibility under limited computational resources, all baseline methods are configured to exchange communication every five local iterations. While increasing the number of local iterations can reduce communication overhead, overly large local steps are known to degrade performance; therefore, we adopt the smallest local iteration count permitted by our computational budget. All experiments use a constant learning rate.

For full-parameter tuning of ChocoSGD, we initialize surrogate model parameters with pretrained weights, which yields substantial performance improvement.

For SeedFlood, we fix the subspace rank and refresh period without additional search. Specifically, experiments in Section 5.2 use a rank of 32 with a refresh period of 1000, while those in Section 5.3 use a rank of 64 and a refresh period of 5000. Since the total number of training iterations is 5000, this effectively corresponds to using a fixed subspace throughout training. For zeroth-order methods, the perturbation scale is set following Malladi et al. [34]. All other training-related settings not explicitly mentioned follow the configuration of [34].

G.3 LORA CONFIGURATION

For all LoRA-based experiments, we adopt a fixed LoRA configuration. Specifically, we set the LoRA rank to 8 and the scaling factor α to 16. LoRA adapters are applied to the query and value projection layers (`q_proj` and `v_proj`) of the transformer. We use a LoRA dropout rate of 0.05 across all experiments.

FFA-LoRA [46] was originally proposed for server-based federated learning, where freezing the randomly initialized A matrices removes the aggregation error caused by averaging the products of two trainable low-rank factors. We adapt it to our decentralized setting as follows: all clients initialize the A matrices identically and keep them frozen throughout training, so that only the B matrices are trained and communicated. All other configurations (rank, scaling factor, target modules, dropout) follow the LoRA setup above.

G.4 ITERATION-BUDGET EXPERIMENT SETUP

Validation loss is evaluated every 5 iterations, and a run stops once it reaches the target loss. Each run is allocated a maximum of 10,000 iterations, with early stopping if the validation loss does not improve for 1,000 consecutive iterations. The learning rate is searched over a finer grid than in the previous experiments. Specifically, we consider seven values evenly spaced on a logarithmic scale from 10^{-2} to 10^{-4} for the LoRA baselines and from 10^{-4} to 10^{-6} for SeedFlood. All other experimental settings are identical to those in Section 5.3.

Table 6: Hyperparameter setup for experiment Section 5.2 and Section 5.3 Validation loss is evaluated every one-tenth of the total training iterations, and the model achieving the best validation loss is selected for evaluation on a held-out test set.

Experiment	Hyperparameter	Value
DSGD	batch size	8
	learning rate	{ 1e-4, 1e-5, 1e-6 }
	local iteration	5
DSGD (LoRA)	batch size	8
	learning rate	{ 1e-2, 1e-3, 1e-4 }
	local iteration	5
ChocoSGD	batch size	8
	learning rate	{ 1e-4, 1e-5, 1e-6 }
	local iteration	5
	consensus step size	1
	compression	Top-K 99% sparsification
ChocoSGD (LoRA)	batch size	8
	learning rate	{ 1e-2, 1e-3, 1e-4 }
	local iteration	5
	consensus step size	1
	compression	Top-K 99% sparsification
DZSGD	batch size	16
	learning rate	{ 1e-5, 1e-6 }
	local iteration	5
	perturbation scale ϵ	1e-3
DZSGD (LoRA)	batch size	16
	learning rate	{ 1e-3, 1e-4, 1e-5 }
	local iteration	5
	perturbation scale ϵ	1e-3
SeedFlood (Section 5.2)	batch size	16
	learning rate	{ 1e-5, 1e-6 }
	flooding steps k	same as Network Diameter
	perturbation scale ϵ	1e-3
	subspace rank r	32
	subspace refresh period τ	1000
SeedFlood (Section 5.3)	batch size	16
	learning rate	{ 1e-5, 1e-6 }
	flooding steps k	same as Network Diameter
	perturbation scale ϵ	1e-3
	subspace rank r	64
	subspace refresh period τ	5000 (no change)

G.5 Compute resource

All experiments were conducted on GPU servers equipped with either NVIDIA A100-80GB GPUs or NVIDIA RTX A6000 GPUs. To simulate decentralized training while keeping all client model instances resident on GPUs, we used either two A100-80GB GPUs or four A6000 GPUs per run, placing multiple client models on each GPU. For the main experiments in Sections 5.2 and 5.3, the compute cost per run ranged from approximately 8 GPU-hours on A6000 GPUs for lighter LoRA-based first-order baselines to approximately 240 GPU-hours for the most expensive full-parameter zeroth-order baselines. This range mainly reflects differences in model size, full-parameter versus LoRA tuning, first-order versus zeroth-order iteration budgets, and the cost of simulating multiple decentralized clients. It should therefore be interpreted as a reproducibility-oriented resource estimate rather than as a controlled wall-clock efficiency comparison between methods

Table 7: Generalized performance comparison over a 16-client network among baseline decentralized methods and SeedFlood. SST2–ReCoRD are evaluated on OPT-1.3B and SQuAD/DROP on Qwen3-1.7B. FO and ZO denote first-order and zeroth-order optimization, and the -FFA suffix denotes FFA-LoRA fine-tuning. Cost is the total transmitted volume per edge over training, reported separately for each model. Δ DSGD is the average relative performance difference across all tasks with respect to DSGD (FFT) under the same network topology, and Rel. ZS is the average performance relative to zero-shot. Best results among ZO methods are in bold.

Ring Network

		OPT-1.3B						Qwen3-1.7B		Cost			
Type	Method	SST2	RTE	BoolQ	WiC	MultiRC	ReCoRD	SQuAD	DROP	OPT	Qwen	Δ DSGD	Rel. ZS
-	ZeroShot	53.56	53.43	45.50	56.90	45.40	70.50	57.57	41.11	–	–	-24.29%	100.00%
FO	DSGD	93.69	71.84	66.10	62.33	68.50	71.50	88.59	48.86	526.3GB	681.4GB	0.00%	136.15%
	DSGD-LoRA	93.58	70.76	65.70	61.44	68.80	71.20	86.93	49.06	629.1MB	635.8MB	-0.64%	135.30%
	DSGD-FFA	92.78	70.04	67.10	60.19	66.50	71.80	86.27	48.31	314.6MB	272.5MB	-1.46%	134.16%
	ChocoSGD	93.35	67.87	68.60	61.60	67.40	72.05	87.45	47.10	15.79GB	17.03GB	-1.13%	134.68%
	Choco-LoRA	83.95	59.21	63.10	59.88	55.50	72.70	79.49	47.39	18.8MB	18.96MB	-8.38%	123.77%
	Choco-FFA	91.74	60.29	65.80	58.31	60.30	72.10	84.79	45.39	9.4MB	8.12MB	-5.95%	128.00%
ZO	DZSGD	91.40	61.01	64.40	**59.56**	59.70	70.50	86.80	47.11	5.26TB	6.88TB	-5.55%	128.49%
	DZSGD-LoRA	91.17	54.51	64.40	57.99	59.95	71.00	83.18	45.22	6.29GB	6.42GB	-7.89%	125.37%
	DZSGD-FFA	90.71	**65.70**	65.10	59.40	62.10	70.20	82.29	44.21	3.14GB	2.75GB	-5.72%	128.33%
	SeedFlood	**92.78**	63.18	**67.20**	57.99	**63.70**	**71.90**	**87.09**	**48.63**	**400KB**	**400KB**	**-3.37%**	**131.62%**

Meshgrid Network

		OPT-1.3B						Qwen3-1.7B		Cost			
Type	Method	SST2	RTE	BoolQ	WiC	MultiRC	ReCoRD	SQuAD	DROP	OPT	Qwen	Δ DSGD	Rel. ZS
-	ZeroShot	53.56	53.43	45.50	56.90	45.40	70.50	57.57	41.11	–	–	-25.21%	100.00%
FO	DSGD	94.15	73.64	69.80	62.54	69.80	70.70	88.64	49.53	526.3GB	681.4GB	0.00%	138.18%
	DSGD-LoRA	93.92	71.84	67.10	61.60	68.00	71.10	87.74	50.30	629.1MB	635.8MB	-1.19%	136.37%
	DSGD-FFA	93.12	71.12	71.00	57.37	67.50	71.70	86.11	48.42	314.6MB	272.5MB	-2.25%	135.20%
	ChocoSGD	92.09	70.76	68.00	61.76	68.10	71.60	88.30	47.92	15.79GB	17.03GB	-1.84%	135.48%
	Choco-LoRA	87.62	59.93	63.10	57.84	55.40	72.90	83.09	45.04	18.8MB	18.96MB	-9.44%	124.42%
	Choco-FFA	91.74	61.73	65.30	60.03	57.00	72.40	84.83	45.67	9.4MB	8.12MB	-7.15%	127.82%
ZO	DZSGD	91.28	**66.07**	65.00	**59.25**	61.40	71.60	**87.19**	45.94	5.26TB	6.88TB	-5.64%	130.13%
	DZSGD-LoRA	90.14	54.87	64.70	58.46	56.30	71.20	84.53	44.53	6.29GB	6.42GB	-9.62%	124.51%
	DZSGD-FFA	91.17	64.98	63.30	58.93	**64.20**	70.00	83.11	44.70	3.14GB	2.75GB	-6.88%	128.54%
	SeedFlood	**92.89**	63.90	**66.70**	57.99	61.90	**71.90**	86.07	**48.36**	**400KB**	**400KB**	**-5.15%**	**130.88%**

H SUPPLEMENTARY TASK-LEVEL RESULTS

Tables 7,8 provide the numerical values underlying the results presented in Section 5.2 and Section 5.3.

Table 8: Three-run averaged results for the ring-network scalability experiment in Section 5.3. SST2, RTE, and BoolQ use OPT-125m; SQuAD uses Qwen3-0.6B. Average relative performance is computed with respect to the corresponding reference performance used in the main paper.

Task / Metric	Method	16	32	64	128
SST2	DSGD-LoRA	86.07	85.35	73.47	55.19
	DSGD-FFA	86.58	85.55	85.32	84.44
	Choco-LoRA	69.53	55.04	55.16	53.59
	Choco-FFA	85.21	84.63	83.68	82.38
	SeedFlood	84.57	85.20	84.93	84.82
RTE	DSGD-LoRA	58.36	54.99	51.96	53.67
	DSGD-FFA	59.21	56.32	54.63	54.39
	Choco-LoRA	53.91	53.67	53.79	53.55
	Choco-FFA	55.72	55.96	53.91	53.31
	SeedFlood	54.15	55.11	56.91	60.04
BoolQ	DSGD-LoRA	62.77	60.56	50.37	44.93
	DSGD-FFA	62.50	61.97	60.73	59.40
	Choco-LoRA	59.63	52.96	52.20	47.77
	Choco-FFA	61.07	59.63	58.60	57.77
	SeedFlood	60.70	61.80	60.80	62.56
SQuAD	DSGD-LoRA	83.42	78.70	72.87	63.65
	DSGD-FFA	83.66	82.45	80.75	79.82
	Choco-LoRA	74.82	73.82	63.05	59.26
	Choco-FFA	81.95	79.98	78.45	77.12
	SeedFlood	82.92	82.24	84.41	84.19