Ferret: Federated Full-Parameter Tuning at Scale for Large Language Models

Yao Shu [*1] **Wenyang Hu** [*2 3] **See-Kiong Ng** [3] **Bryan Kian Hsiang Low** [3] **Fei Yu** [4]

Abstract

Large Language Models (LLMs) have become indispensable in numerous real-world applications. However, fine-tuning these models at scale, especially in federated settings where data privacy and communication efficiency are critical, presents significant challenges. Existing approaches often resort to parameter-efficient fine-tuning (PEFT) to mitigate communication overhead, but this typically comes at the cost of model accuracy. To this end, we propose *federated full-parameter tuning at scale for LLMs* (Ferret), **the first first-order method with shared randomness** to enable scalable full-parameter tuning of LLMs across decentralized data sources while maintaining competitive model accuracy. Ferret accomplishes this through three aspects: **(I)** it employs widely used first-order methods for efficient local updates; **(II)** it projects these updates into a low-dimensional space to considerably reduce communication overhead; and **(III)** it reconstructs local updates from this low-dimensional space with shared randomness to facilitate effective full-parameter global aggregation, ensuring fast convergence and competitive final performance. Our rigorous theoretical analyses and insights along with extensive experiments, show that Ferret significantly enhances the scalability of existing federated full-parameter tuning approaches by achieving high computational efficiency, reduced communication overhead, and fast convergence, all while maintaining competitive model accuracy. Our implementation is available at https://github.com/allen4747/Ferret.

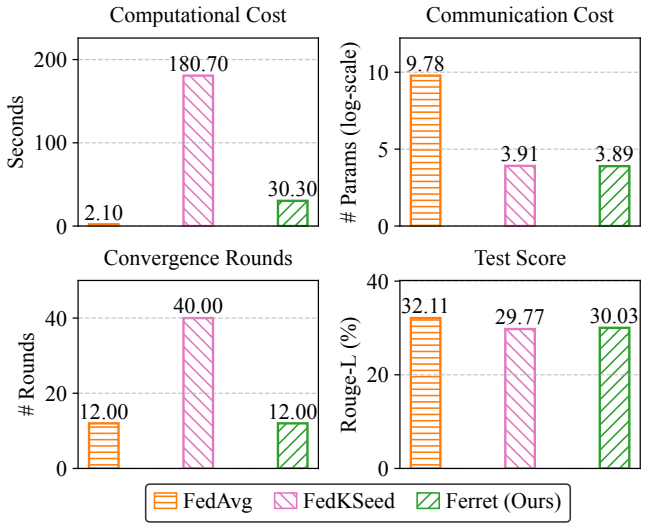


Figure 1: Performance comparison of various federated full-parameter tuning algorithms on Natural Instructions dataset with LLaMA-3B. Our Ferret shows significantly improved scalability, with a 6.0× reduction in computational cost and 3.3× fewer convergence rounds than FedKSeed, alongside a $10^6\times$ reduction in communication overhead than FedAvg, while achieving comparable test score.

1. Introduction

Recently, Large Language Models (LLMs) have become indispensable tools across a wide range of real-world applications, from natural language processing tasks like translation (Xu et al., 2024) and summarization (Van Veen et al., 2024) to more complex tasks such as code generation (Liu et al., 2024) and decision-making systems (Shao et al., 2023). The immense scale and versatility of LLMs make them highly valuable in practice, but they also introduce significant challenges, particularly when they are fine-tuned in federated settings. Federated Learning (FL) offers a decentralized approach to fine-tuning LLMs while retaining data on local clients to ensure privacy. However, while this approach effectively addresses privacy concerns, it also results in **prohibitive communication overhead** when the model parameters of LLMs scale to billions.

One of the straightforward strategies to mitigate the prohibitive communication costs in the federated tuning of

[*]Equal contribution [1]Hong Kong University of Science and Technology (Guangzhou) [2]SAP [3]National University of Singapore [4]Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ). Correspondence to: Yao Shu <yaoshu@hkust-gz.edu.cn>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

LLMs is parameter-efficient fine-tuning (PEFT). PEFT methods (Hu et al., 2022; Wei et al., 2024) focus on fine-tuning only a subset of model parameters, which is able to significantly reduce the communication overhead between clients and a central server (Che et al., 2023; Zhang et al., 2023; Kuang et al., 2024; Zhang et al., 2024b). Despite the effectiveness in reducing bandwidth usage, this type of approach often compromises **model accuracy** (Pu et al., 2023), as fine-tuning a subset of model parameters may fail to fully capture the nuances of local data distributions. Thus, recent efforts, e.g., FedKSeed (Qin et al., 2024), have been devoted to utilizing zeroth-order optimization (ZOO) (Nesterov & Spokoiny, 2017; Berahas et al., 2022) in federated full-parameter tuning of LLMs, aiming to maintain competitive model accuracy while reducing the communication overhead by transmitting only thousands of scalar gradients per round between clients and a central server. Unfortunately, this approach often suffers from its **poor scalability**, including **increased computational cost** per round and **a larger number of communication rounds** required for convergence, compared to FL methods that use first-order optimization (FOO), e.g., FedAvg (McMahan et al., 2017).

Therefore, we develop *federated full-parameter tuning at scale for LLMs* (Ferret), **the first first-order FL approach with shared randomness** to enable scalable federated full-parameter tuning of LLMs with *compelling computational efficiency*, *reduced communication overhead*, and *fast convergence speed*, while maintaining *competitive model accuracy*, as shown in Fig. 1. Ferret achieves this through three aspects: First, it uses widely applied first-order methods to perform efficient local updates on each client, which usually requires fewer iterations to achieve the same local update process compared to existing ZOO-based FL. Next, Ferret projects these updates into a low-dimensional space, resulting in a significantly reduced communication cost compared to existing FOO-based FL. Finally, Ferret reconstructs local updates from the low-dimensional space with shared randomness for effective full-parameter global aggregation, ensuring fast convergence and competitive model accuracy compared to existing ZOO-based FL. We further complement Ferret with rigorous theoretical analyses and insights, showing the theoretical advantages of Ferret over other baselines and guiding the best practices for its implementation. Finally, through extensive experiments, we verify that Ferret significantly outperforms existing methods with superior scalability and competitive model accuracy, making it a desirable solution for deploying LLMs in large-scale federated environments.

To summarize, our contributions in this work include:

- We novelly propose Ferret, to the best of our knowledge, **the first first-order FL approach with shared randomness**, which *significantly enhances the scalability* of federated full-parameter tuning of LLMs while maintaining *competitive model accuracy*.
- We present **rigorous theoretical analyses and insights** to support the effectiveness of our Ferret, demonstrating its *theoretical advantages* over other baselines and *guiding its best practices*.
- Through **extensive experiments**, we demonstrate that Ferret consistently improves over existing methods in practice, offering both *superior scalability* and *competitive model accuracy*.

2. Problem Setup

In this paper, we consider the federated full-parameter tuning of an LLM using decentralized data $\{\mathcal{D}_i\}_{i=1}^N$ on N local clients while preserving data privacy, i.e., without sharing raw data. Specifically, given a loss function $\ell(\cdot;\cdot)$, we aim to minimize a global objective $\mathcal{L}(\mathbf{w})$ defined as the average loss across $\{\mathcal{D}_i\}_{i=1}^N$ over the model parameters $\mathbf{w} \in \mathbb{R}^d$ of an LLM. That is,

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \triangleq \frac{1}{N} \sum_{i\in[N]} \mathcal{L}^{(i)}(\mathbf{w}) \quad \text{where} \quad \mathcal{L}^{(i)}(\mathbf{w}) \triangleq \mathbb{E}_{\mathbf{x}^{(i)}\in\mathcal{D}_i}\left[\ell(\mathbf{w};\mathbf{x}^{(i)})\right]. \tag{1}$$

Following the practice in federated learning (FL), (1) can be solved through multiple rounds of local training and global aggregation. In each communication round, each client i independently updates its local model parameters by minimizing its local objective $\mathcal{L}^{(i)}(\mathbf{w})$ based on its local data $\mathcal{D}_i$. After local training, the clients transmit their updated local model parameters to a central server, where they are aggregated to form an updated global model. This updated global model is then redistributed to all clients, and the process is repeated over rounds.

The main challenge in LLM federated full-parameter tuning is to ensure the **computational efficiency and the convergence speed** of the global model while **reducing the communication overheads**, particularly given that the parameter size d of LLMs often reaches billions. While existing first-order FL (McMahan et al., 2017; Li et al., 2020; Karimireddy et al., 2020) can ensure compelling computational efficiency and convergence speed by applying first-order updates, they typically incur $\mathcal{O}(d)$ communication overheads due to the need to transmit the entire set of model parameters between clients and the central server. This type of methods hence is impractical for LLM federated full-parameter tuning due to the enormous size of LLMs. In contrast, although zeroth-order FL (Qin et al., 2024) can reduce these communication costs by transmitting only several scalar gradients from their finite difference-based gradient estimation with shared randomness, they often incur more computational cost to achieve the same local update progress and a larger

number of communication rounds to converge compared with first-order FL. These naturally raise the question:

Can we combine the strengths of these different types of methods to achieve scalable federated full-parameter tuning of LLMs with high computational efficiency, reduced communication overhead, and fast convergence?

3. The Ferret Algorithm

To answer this question, we introduce Ferret, *federated full-parameter tuning at scale for LLMs*, in Algo. 1. We present an overview of Ferret algorithm in Sec.3.1, followed by a detailed explanation of its key techniques in Sec. 3.2.

3.1. Overview of Ferret

To achieve scalable LLM federated full-parameter tuning, our Ferret algorithm combines the strengths of both first-order FL, which offers efficient computation and fast convergence, and zeroth-order FL, which reduces communication overhead. Specifically, Ferret *(a)* follows first-order FL to apply first-order optimization methods for local updates on clients, ensuring both computational efficiency and fast convergence, and *(b)* draws inspiration from zeroth-order FL by projecting updates into a low-dimensional space using random bases that can be regenerated using shared randomness among clients for the reconstruction of these updates, thereby reducing communication overhead.

Our Ferret algorithm operates by repeating the following three sequential steps over many communication rounds, denoted by $r \in [R]$, where R is the total number of rounds. For simplicity, we omit the subscript r from the seeds, random bases, and projected coordinates in our notation.

Step ①: Global Aggregation (Line 3-6 in Algo. 1). At the beginning of the first round ($r = 1$), each client initializes its local model parameters using the pre-trained model parameters $\mathbf{w}_0$, i.e., $\mathbf{w}_1 \leftarrow \mathbf{w}_0$. For subsequent rounds ($r > 1$), each client $j \in [N]$ receives the random seeds $s^{(i)}$ and the corresponding K projected coordinates $\{\gamma_k^{(i)}\}_{k=1}^K$ of every client $i \in [N]$ from the previous round. These random seeds (i.e., shared randomness) are then used to generate d-dimensional random bases $\{\mathbf{v}_k^{(i)}\}_{k=1}^K$ for each client i. [1] These random bases, along with the corresponding projected coordinates $\{\gamma_k^{(i)}\}_{k=1}^K$, are applied to reconstruct local updates as $\widetilde{\Delta}_{r-1}^{(i)}$ in every client i. The global model is then updated by aggregating these local contributions as follows:

$$\mathbf{w}_{r-1} \leftarrow \mathbf{w}_{r-2} - \frac{1}{N} \sum_{i\in[N]} \widetilde{\Delta}_{r-1}^{(i)}, \text{with } \widetilde{\Delta}_{r-1}^{(i)} \triangleq \sum_{k\in[K]} \gamma_k^{(i)} \mathbf{v}_k^{(i)} . \tag{2}$$

Step ②: Local Updates (Line 7-9 in Algo. 1). After **Step ①**, each client j will perform T-iteration first-order optimization on its local loss function by using the randomly sampled data for its local updates. Formally, if stochastic gradient descent with a local learning rate η is used, the update rule for client $j \in [N]$ at iteration $t \in [T]$ of round $r \in [R]$ can then be represented as below with $\mathbf{w}_{r,0} \leftarrow \mathbf{w}_r$:

$$\mathbf{w}_{r,t}^{(j)} \leftarrow \mathbf{w}_{r,t-1}^{(j)} - \eta \nabla \ell \left(\mathbf{w}_{r,t-1}^{(j)}; \mathbf{x}_{t-1}^{(j)} \right) . \tag{3}$$

Different from the zeroth-order update in (Qin et al., 2024) that requires many local update iterations, the first-order update in (3) enables each client to efficiently and effectively adapt the global model $\mathbf{w}_r$ to its specific data using a small T, thereby enhancing both the computational efficiency of this local update. Here, (3) can be implemented using any gradient method variant, e.g., Adam (Kingma & Ba, 2014).

Step ③: Projected Updates (Line 10-12 in Algo. 1). After completing the local updates above, each client j randomly chooses a single new seed $s^{(j)}$ to generate K new random bases $\{\mathbf{v}_k^{(j)}\}_{k=1}^K$ and employ these K new random bases to project the local update $\Delta_r^{(j)}$ into a K-dimensional coordinates $\{\gamma_k^{(j)}\}_{k=1}^K$ based on the techniques in Sec. 3.2. Seed $s^{(j)}$ and projected coordinates $\{\gamma_k^{(j)}\}_{k=1}^K$ are then shared with other clients to facilitate the next round of global aggregation. By sharing only one single random seed and K projected coordinates among N clients where random bases $\{\mathbf{v}_k^{(j)}\}_{k=1}^K$ can be regenerated for global aggregation as shown in **Step ①** above, the communication overhead in LLM full-parameter tuning is therefore considerably reduced compared with first-order methods (e.g., FedAvg) especially when $T \ll d$. The communication of seed $s^{(j)}$ can be mitigated if the same seed is used across all rounds $r \in [R]$, which can further reduce communication overhead.

3.2. Update Projection and Reconstruction

As mentioned before, we aim to project the local updates into K-dimensional coordinates ($K \ll d$) to substantially reduce the communication overhead in LLM full-parameter tuning. To accomplish this, let $\Delta \in \mathbb{R}^d$ denote any local update, and let $\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_K] \in \mathbb{R}^{d\times K}$ represent the K random bases generated by any random seed s, we solve the following convex minimization problem to determine the K-dimensional projected coordinates $\boldsymbol{\gamma} = [\gamma_1\ \gamma_2\ \cdots\ \gamma_K]^\top$:

$$\boldsymbol{\gamma} \triangleq \arg\min_{\mathbf{y}} \|\mathbf{V}\mathbf{y} - \Delta\| . \tag{4}$$

As $\mathbf{V}$ is singular with $K \ll d$, the close-form of $\boldsymbol{\gamma}$ and its corresponding reconstruction $\widetilde{\Delta}$ will be

$$\boldsymbol{\gamma} = (\mathbf{V}^\top\mathbf{V})^{-1}\mathbf{V}^\top\Delta, \quad \widetilde{\Delta} = \mathbf{V}(\mathbf{V}^\top\mathbf{V})^{-1}\mathbf{V}^\top\Delta . \tag{5}$$

[1] As in (Qin et al., 2024), we can get K random seeds from a single seed $s^{(i)}$ and use them to generate K random bases independently. So, one seed is sufficient for each client.

Algorithm 1 Ferret

Input: $\mathbf{w}_0$, N, R, T, K, η

for each round $r \in [R]$ **do**
  for each client $j \in [N]$ in parallel **do**
    if $r > 1$ **then** // Step ①: Global Aggregation
      Receive $\{s^{(i)}\}_{i=1}^{N}$ and $\{\gamma_k^{(i)}\}_{i=1,k=1}^{N,K}$
      Generate bases $\{\mathbf{v}_k^{(i)}\}_{i=1,k=1}^{N,K}$ using $\{s^{(i)}\}_{i=1}^{N}$
      $\mathbf{w}_{r-1} \leftarrow \mathbf{w}_{r-2} - \sum_{i\in[N]} \left(\sum_{k=1}^{K} \gamma_k^{(i)} \mathbf{v}_k^{(i)}\right) / N$
    $\mathbf{w}_{r,0} \leftarrow \mathbf{w}_r$
    for $t \in [T]$ **do** // Step ②: Local Updates
      $\mathbf{w}_{r,t}^{(j)} \leftarrow \mathbf{w}_{r,t-1}^{(j)} - \eta \nabla \ell(\mathbf{w}_{r,t-1}^{(j)}; \mathbf{x}_{r,t-1}^{(j)})$
    // Step ③: Projected Updates
    Randomly set $s^{(j)}$ and generate bases $\{\mathbf{v}_k^{(j)}\}_{k=1}^{K}$
    $\Delta_r^{(j)} \leftarrow \mathbf{w}_{r-1}^{(j)} - \mathbf{w}_r^{(j)}$, compute $\{\gamma_k^{(j)}\}_{k=1}^{K}$ with (6)
    Send $s^{(j)}$ and $\{\gamma_k^{(j)}\}_{k=1}^{K}$ to the central server

Choice of Random Bases V. Particularly, if $\mathbf{V}$ is a rectangular matrix with ones on its main diagonal, meaning that each $\mathbf{v}_k$ is a standard basis vector, (5) simplifies to $\boldsymbol{\gamma} = \mathbf{V}^\top \Delta$, which then corresponds to a block-wise dimension selection for local update projection and reconstruction. However, this approach significantly reduces the number of parameters updated per round as $K \ll d$, potentially hindering the overall tuning performance. We thus propose to sample each element in $\mathbf{v}_k$ ($k \in [K]$) independently from a normal distribution with bounded 2-norm, i.e., $\|\mathbf{v}_k\| \leq 1$, aiming to realize and stabilize full-parameter tuning of LLMs for competitive overall performance. To achieve this, we can sample from a truncated normal distribution: $\mathrm{v} \sim \mathcal{N}(0,1)$ with $\mathrm{v} \in [-1/\sqrt{d}, 1/\sqrt{d}]$ instead. The efficacy of this bounded norm will be demonstrated in Sec. 4.1 shortly.

Reconstruction w/o Inversion. Unfortunately, (5) incurs a computational complexity of $\mathcal{O}(K^2 d + K^3)$ and storage complexity of $\mathcal{O}(Kd)$ owing to the inversion of $\mathbf{V}^\top \mathbf{V}$ in (5), which is prohibitively costly, especially when K is large and d reaches billions. Since $\mathbf{V}^\top \mathbf{V}$ is a scaled empirical covariance for the aforementioned distribution of an identity covariance matrix (Vershynin, 2012) , we propose to approximate $\mathbf{V}^\top \mathbf{V}$ with $\mathbf{I}_K$ (i.e., $K \times K$-dimensional identity matrix) and (5) as

$$\boldsymbol{\gamma} \approx (\rho K)^{-1} \mathbf{V}^\top \Delta \,. \tag{6}$$

Here, $\rho \triangleq 1 - \frac{2\psi(1/\sqrt{d})/\sqrt{d}}{2\Phi(1/\sqrt{d})-1}$, where $\psi(\frac{1}{\sqrt{d}})$ and $\Phi(\frac{1}{\sqrt{d}})$ is the probability density function (PDF) and cumulative distribution function (CDF) of the standard normal distribution evaluated at $1/\sqrt{d}$, respectively. This approximation leads to improved computational complexity of $\mathcal{O}(Kd)$ and storage complexity of $\mathcal{O}(\max\{K, d\})$, where the storage complexity is reduced due to the in-place operations on random bases $\{\mathbf{v}_k\}_{k=1}^{K}$ when computing $\{\gamma_k\}_{k=1}^{K}$ sequentially. Consequently, we can reconstruct the true update Δ approximately using $\widetilde{\Delta}$ below

$$\widetilde{\Delta} = (\rho K)^{-1} \mathbf{V} \mathbf{V}^\top \Delta \,, \tag{7}$$

whose efficacy will be theoretically justified in Sec. 4.1. Finally, our (6) and (7) simplify the update projection and reconstruction in (5) into straightforward matrix multiplications.

Block-Wise Reconstruction. The computational complexity of $\mathcal{O}(Kd)$ and storage complexity of $\mathcal{O}(\max\{K, d\})$ for our reconstruction in (7) is still prohibitively costly, particularly for LLMs with billions of parameters. To address this, we propose a block-wise reconstruction technique to reduce both computational and storage complexities. Specifically, suppose the full dimension d is divided into L blocks, each with dimension d_l such that $\sum_{l\in[L]} d_l = d$. Let Δ_l be the update for block l and K_l (with $\sum_{l\in[L]} K_l = K$) be the number of random bases allocated to this block. We propose to compute $\boldsymbol{\gamma}_l$ and reconstruct Δ_l using random bases $\mathbf{V}_l$ of dimension $d_l \times K_l$ as follows:

$$\boldsymbol{\gamma}_l = (\rho_l K)^{-1} \mathbf{V}_l^\top \Delta_l, \quad \widetilde{\Delta}_l = (\rho_l K_l)^{-1} \mathbf{V}_l \mathbf{V}_l^\top \Delta_l \,. \tag{8}$$

Here, $\rho_l \triangleq 1 - \frac{2\psi(1/\sqrt{d_l})/\sqrt{d_l}}{2\Phi(1/\sqrt{d_l})-1}$. This trick reduces the storage complexity to $\mathcal{O}(\max\{\{K_l, d_l\}_{l=1}^{L}\})$ that is straightforward to verify, and lowers the computational complexity to $\mathcal{O}(\sum_{l\in[L]} K_l d_l)$. Of note, (8) also significantly reduces the computational complexity of global aggregation compared to existing methods (Qin et al., 2024) (verified in Sec. 5). This block-wise reconstruction thus further enhances the scalability of our Ferret in the federated full-parameter tuning of LLMs. Interestingly, this block-wise strategy has also been widely applied in other fields due to it efficacy (Frantar & Alistarh, 2023; Luo et al., 2024).

4. Theoretical Analyses and Insights

We now provide theoretical analyses to substantiate the effectiveness of Ferret: *(a)* reconstruction analysis in Sec. 4.1; *(b)* convergence analysis in Sec. 4.2; and *(c)* scalability and beyond in Sec. 4.3.

4.1. Reconstruction Analysis

Theorem 1 (Unbiased Reconstruction). *Given the reconstruction in (7), we have*

$$\mathbb{E}\left[\widetilde{\Delta}\right] = \Delta \,.$$

To begin with, we demonstrate in Thm. 1 that our reconstruction in (7) is unbiased, with the proof provided in Appx. B.1.

Of note, Thm. 1 shows that *(a)* the scalar $1/(\rho K)$ is crucial for (7) to achieve an unbiased reconstruction of the ground-truth update Δ, and *(b)* our (7) avoids the bias commonly found in zeroth-order FL methods (Berahas et al., 2022), including FedZO (Fang et al., 2022) and FedKSeed (Qin et al., 2024). As a result, (7) is expected to provide a more accurate update reconstruction as shown below.

Theorem 2 (Reconstruction Error). *Given the reconstruction in (7), we have*

$$\mathbb{E}\left[\left\|\widetilde{\Delta}-\Delta\right\|\right] \leq \max\left\{2\sqrt{\frac{2\ln(2d)}{\rho K}}, \frac{2\ln(2d)}{\rho K}\right\}\|\Delta\| \ .$$

We then demonstrate the efficacy of our reconstruction in (7) by theoretically bounding the difference between the reconstructed update $\widetilde{\Delta}$ and the ground truth Δ in Thm. 2. The proof is in Appx. B.2. Of note, $1/\rho$ typically has an asymptotic rate of $\mathcal{O}(d)$, which we will verify empirically in Appx. C.5. Thm. 2 offers three critical insights of our Ferret: *(a)* Our reconstruction in (7) incurs a reconstruction error at a rate of $\widetilde{\mathcal{O}}(d/K)$ for T local update iterations when $\sqrt{d} \geq K$, which generally aligns with the results in (Vershynin, 2010). This indicates that the reconstruction error of our (7) can be linearly reduced by increasing K. *(b)* Ferret avoids additional constant error items (Berahas et al., 2022) that are caused by the biased estimation in these zeroth-order FL methods, implying that our (7) can be more accurate. We will justify this further in our Thm. 3 below. *(c)* Thanks to the independence from the iterations (i.e., T) of local updates in Thm. 2, Ferret prevents the error accumulation over the local update iterations T, which is a common issue in zeroth-order FL methods (Fang et al., 2022; Qin et al., 2024).

Theorem 3 (Connection with Zeroth-Order Method). *Define* $g_k \triangleq \frac{\ell(\mathbf{w}+\epsilon\mathbf{v}_k;\mathbf{x}^{(i)})-\ell(\mathbf{w};\mathbf{x}^{(i)})}{\epsilon}$*, in which each element* v *in* $\mathbf{v}_k$ *is sampled from* $\mathrm{v} \sim \mathcal{N}(0,1)$ *with* $\mathrm{v} \in [-1/\sqrt{d}, 1/\sqrt{d}]$*,* $\boldsymbol{g} \triangleq [g_1 \ \cdots \ g_K]^\top$*, and* $\mathbf{V} \triangleq [\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \ \mathbf{v}_K] \in \mathbb{R}^{d\times K}$*, assume* $\ell(\cdot;\cdot)$ *is* β*-smooth w.r.t its first argument, the zeroth-order reconstruction* $\mathbf{V}\boldsymbol{g}/K$ *used in (Fang et al., 2022; Qin et al., 2024) then incurs:*

$$\left\|\frac{1}{K}\mathbf{V}\boldsymbol{g}-\frac{1}{K}\mathbf{V}\mathbf{V}^\top\nabla\ell(\mathbf{w};\mathbf{x}^{(i)})\right\| \leq \frac{1}{2}\beta\epsilon \ .$$

We then show in Thm. 3 the connection between our update projection (6) and zeroth-order method used in (Fang et al., 2022; Qin et al., 2024). The proof is provided in Appx. B.3. Thm. 3 delivers three essential insights: *(a)* When $\epsilon \to 0$, the reconstruction $\mathbf{V}\boldsymbol{g}/K$ in zeroth-order method is equivalent to $\mathbf{V}\mathbf{V}^\top\nabla\ell(\mathbf{w};\mathbf{x}^{(i)})/K$ and shares a similar form of (7) when Δ is replaced by $\nabla\ell(\mathbf{w};\mathbf{x}^{(i)})$, implying that zeroth-order method in fact aims to approximate our reconstruction (7). *(b)* In practice, $\epsilon > 0$. So, zeroth-order method leads to a biased reconstruction with an additional error term of $\beta\epsilon/2$ compared to our (7), and this error will accumulate over T local iterations, implying that our (7) can indeed be more accurate as we have demonstrated above. *(c)* In addition, zeroth-order method is typically coupled with a single gradient (i.e., $\nabla\ell(\mathbf{w};\mathbf{x}^{(i)})$), whereas our (7) can be applied to any vector, making it more general. Overall, these results further verify the advantages of our (7) over the zeroth-order method used in (Fang et al., 2022; Qin et al., 2024), which we will also support empirically in Appx. C.5.

Proposition 1 (Block-Wise Reconstruction Speedup). *For block-wise reconstruction (8) of size* L*,*

$$\sum_{l\in[L]} d_l K_l < \left(\sum_{l\in[L]} d_l\right)\left(\sum_{l\in[L]} K_l\right) = dK \ .$$

We next highlight the computational advantage of our block-wise reconstruction (8) in Prop. 1. The proof is in Appx. B.4. Prop. 1 indicates that by dividing the reconstruction of d-dimensional updates into smaller blocks $\{d_l\}_{l=1}^L$, we get a reduction in overall computational complexity that is strictly less than that of the full dimension d in (7). E.g., when $d_1 = \cdots = d_L$ and $K_1 = \cdots = K_L$, we have $\sum_{l\in[L]} K_l d_l = Kd/L$, showing that our block-wise reconstruction (8) reduces the computational complexity of (7) by a factor of $1/L$. This implies that increasing the number of blocks L can further enhance the computational efficiency of our block-wise reconstruction (8).

Proposition 2 (Block-Wise Reconstruction Error). *For block-wise reconstruction (8) of size* L*, when* $\sqrt{d_l} \geq K_l$ *for any* $l \in [L]$*,*

$$\mathbb{E}\left[\left\|\widetilde{\Delta}-\Delta\right\|\right] < \widetilde{\mathcal{O}}\left(\sum_{l\in[L]}\frac{\|\Delta_l\|}{\rho_l K_l}\right) ,$$

which is minimized by choosing $K_l \propto \sqrt{\|\Delta_l\|/\rho_l}$*.*

We conclude by analyzing the error induced by our block-wise reconstruction (8) and the corresponding optimal random bases allocation in Prop. 2. The proof is provided in Appx. B.5. Prop. 2 demonstrates that reconstruction error can be minimized by adaptively allocating the number of random bases according to the gradient norm of each block. This is intuitively reasonable because a larger gradient norm typically indicates a need for more immediate model updates in practice. Hence, this insight not only provides a theoretical foundation for optimizing Ferret but also offers practical guidance. That is, by aligning the num-

ber of random bases with gradient norms, practitioners can enhance reconstruction accuracy and overall model performance. This adaptive approach ensures efficient use of computational resources, making Ferret versatile and effective across different datasets and federated learning scenarios.

4.2. Convergence Analysis

In this subsection, we present the convergence of Ferret in our Thm. 4 below when using stochastic gradient descent (SGD) for the local updates in (3). To simplify the analysis, we primarily focus on deriving theoretical results for a homogeneous setting, where $\mathcal{L}^{(i)}(\mathbf{w}) = \mathcal{L}(\mathbf{w})$ in (1). Results in the heterogeneous setting can be derived by following the same proof idea.

Theorem 4 (Convergence). *Define* $D \triangleq \mathcal{L}(\mathbf{w}_0) - \min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$. *Assume that* $\mathcal{L}(\mathbf{w})$ *is* β*-smooth and non-convex, and* $\mathbb{E}[\|\nabla\mathcal{L}^{(i)}(\mathbf{w}) - \nabla\ell(\mathbf{w};\mathbf{x})\|^2] \leq \sigma^2$ *for any* $\mathbf{x}, \mathbf{w}$, *when choosing* $\eta \leq \frac{1}{20\beta T}$ *in Algo. 1, the following holds for federated full-parameter tuning with* $\mathcal{L}^{(i)}(\mathbf{w}) = \mathcal{L}(\mathbf{w})$,

$$\min_{r\in[R]} \mathbb{E}\left[\|\nabla\mathcal{L}(\mathbf{w}_r)\|^2\right] \leq \mathcal{O}\left(\frac{D}{\eta TR} + \eta T\sigma^2\right)$$

where $[R)$ *is the half-open interval* $[0, R)$. *Especially, by choosing* $\eta = \frac{1}{20\beta T\sqrt{R}}$ *in Algo. 1, the number of communication rounds are required to be* $R = \mathcal{O}(1/\epsilon^2)$ *to achieve an* ϵ *convergence error.*

Its proof is in Appx. B.6. Particularly, when $T = 1$, Thm. 4 recovers the result of standard SGD (Ghadimi & Lan, 2013). Thm. 4 provides three essential insights: *(a)* Thanks to our improved update reconstruction (7) as justified above, Ferret avoids the additional constant terms accumulated over T local iterations, which are typically caused by the biased gradient estimation in zeroth-order FL methods (e.g., FedZO and FedKSeed) (Fang et al., 2022), thereby highlighting the superior advantage of Ferret over these zeroth-order FL methods in convergence speed. *(b)* Given a proper η, Ferret shares the same communication round complexity as SGD, at a rate of $\mathcal{O}(1/\epsilon^2)$, showing that the communication round complexity of Ferret is asymptotically comparable to that of standard SGD. *(c)* This communication rounds complexity is improved over that of zeroth-order FL methods (Fang et al., 2022) due to its independence from d and other constant factors required by these zeroth-order FL methods, further highlighting the advantage of Ferret in communication round complexity and its improved efficacy in federated full-parameter tuning over these methods.

4.3. Scalability and Beyond

With the theoretical results above, we summarize the scalability of Ferret and compare it to existing methods like zeroth-order FL (e.g., FedZO and FedKSeed) and first-order FL (e.g., FedAvg) in Tab. 1.

Computation Per Round. Of note, Ferret enjoys a computational complexity of $\mathcal{O}(\tau_1 T)$ for any client $i \in [N]$ per round, where τ_1 is the per-iteration complexity of the first-order update (including forward and backward passes) in (3), and T is the number of local iterations. This is comparable to the well-established FedAvg. In contrast, both FedZO and FedKSeed incur a complexity of $\mathcal{O}(\tau_0 K)$, with τ_0 being the per-iteration complexity of the zeroth-order update (i.e., forward pass) and K representing the number of forward passes. As first-order updates use more accurate gradients, T will be smaller than K (i.e., $T \ll K$) to attain the same local update progress. Although τ_1 can be at most twice τ_0, our Ferret is still more computationally efficient than FedZO and FedKSeed (see Sec. 5).

Communication Per Round. As only one seed and K projected coordinates $\{\gamma_k^{(i)}\}_{k=1}^K$ from a client $i \in [N]$ need to be transmitted per round in Algo. 1 with $K \ll d$, Ferret incurs a communication overhead of $\mathcal{O}(K)$, which is similar to that of FedKSeed. This is significantly more efficient than FedAvg and FedZO, which have a communication complexity of $\mathcal{O}(d)$ due to their need to transmit the entire model (or gradients). This significantly reduced communication cost therefore makes Ferret especially suitable for federated full-parameter tuning of LLMs with billions of parameters.

Rounds to Converge. As revealed in Sec.4.2, our Ferret benefits from unbiased update reconstruction in (7) (validated in Thm. 1), enabling fast convergence with a small number of communication rounds to achieve ϵ convergence error (see Thm. 4). This is significantly more efficient than zeroth-order FL methods like FedZO and FedKSeed, which require many more communication rounds to converge due to poor gradient estimation (Fang et al., 2022). FedAvg, applying the ground truth local update for its global aggregation, surely converges with the fewest rounds. Overall, Ferret remains a strong choice for federated full-parameter tuning of LLMs, even in terms of rounds to converge.

Beyond Scalability. Our Ferret also offers benefits in adaptability, generalization, and privacy. Unlike FedKSeed, which is limited to SGD, Ferret is highly adaptable, because both global aggregation (2) and local update (3) in Ferret can be implemented with any gradient method variant, e.g., the widely used AdamW (Loshchilov & Hutter, 2019) in LLM training. This adaptability thus makes it much easier to integrate Ferret into existing centralized tuning workflows for LLMs, facilitating a seamless transition to federated tuning. Besides, since Ferret enables federated tuning with full

Table 1: Comparison of scalability (computation and communication per round, and #rounds to converge) and other factors (adaptability, generalization, and privacy). Here, $d \gg K \gg T$. Symbols: ○ (fewer is better), ♡ and ◇ (more is better).

Method	Type	Scalability			Others		
		Comp.	Comm.	#Rounds	Adapt.	Gen.	Privacy
FedZO	ZOO	$\mathcal{O}(\tau_0 K)$	$\mathcal{O}(d)$	○ ○ ○	√	♡ ♡ ♡	◇ ◇
FedKSeed	ZOO	$\mathcal{O}(\tau_0 K)$	$\mathcal{O}(K)$	○ ○ ○	×	♡ ♡ ♡	◇ ◇ ◇
FedAvg	FOO	$\mathcal{O}(\tau_1 T)$	$\mathcal{O}(d)$	○	√	♡ ♡ ♡	◇
Ferret (ours)	FOO	$\mathcal{O}(\tau_1 T)$	$\mathcal{O}(K)$	○ ○	√	♡ ♡ ♡	◇ ◇ ◇

parameters, it is expected to deliver strong generalization performance as other federated full-parameter tuning methods like FedAvg, as supported in Sec. 5. Finally, by transmitting only seeds and low-dimensional projected coordinates among clients, rather than the entire model (or gradients) as in FedZO and FedAvg, Ferret ensures improved privacy for federated full-parameter tuning of LLMs.

Overall, Ferret strikes an optimal balance between computational efficiency, communication overhead, convergence speed, and other critical factors such as adaptability, generalization, and privacy. This makes it a scalable and desirable solution for federated full-parameter tuning of LLMs.

5. Experiments

In this section, we evaluate the efficacy of Ferret, following the practice in FedKSeed (Qin et al., 2024). We primarily compare Ferret with other federated full-parameter tuning baselines, including both zeroth-order methods (e.g., FedZO (Fang et al., 2022) and FedKSeed (Qin et al., 2024)) and first-order methods (e.g., FedAvg (McMahan et al., 2017)). Our evaluations use DataJuicer-1.3B (Chen et al., 2023) and LLaMA-3B (Touvron et al., 2023a) on the Natural Instructions (Wang et al., 2022) and Dolly-15K (Conover et al., 2023) datasets, as well as larger models (i.e., LLaMA2-7B and LLaMA2-13B (Touvron et al., 2023b)) on the CodeAlpaca (Chaudhary, 2023) and GSM8K (Cobbe et al., 2021).

FL Settings. In each round of federated learning, 5% of clients were randomly selected to participate. Following the same practice in FedKSeed (Qin et al., 2024), we set the total number of communication rounds to 40 for the NI dataset and 60 for Dolly-15K for all baselines. Due to the compelling efficiency of our method, we set the total number of communication rounds to 12 for the NI dataset and 20 for Dolly-15K for Ferret. However, for more complex tasks such as CodeAlpaca and GSM8K, we run all algorithms, including our Ferret, for 20 rounds to ensure a fair comparison. First-order baselines trained locally for one epoch, and FedKSeed trained for 200 steps, while our Ferret algorithm trained for 10 iterations (i.e., $T = 10$ in Algo. 1). The K value was set to 4096 for FedKSeed. All approaches perform local update with a batchsize of 1 to reduce memory consumption. For each local update iteration in Ferret, we accumulate the gradients from 4 samples.

More experimental details and ablation studies are provided in Appx. C.1 and Appx. C.5, C.6 respectively.

Table 2: Comparison of Rouge-L (%) among various algorithms. Each cell reports the mean ± std of Rouge-L scores from the final round of four runs, each with a different random seed. The highest and second-highest scores are shown in **bold** and underline, respectively.

Algorithm	Natural Instructions		Dolly-15K	
	DataJuicer-1.3B	LLaMA-3B	DataJuicer-1.3B	LLaMA-3B
FedPTuning	19.61 ± 2.71	25.41 ± 1.14	23.98 ± 3.23	30.30 ± 1.16
FedPrompt	6.04 ± 0.12	8.95 ± 2.47	32.73 ± 0.87	24.50 ± 4.78
FedIT-SGD	19.40 ± 1.83	28.14 ± 0.85	27.23 ± 0.68	29.28 ± 0.50
FedIT	22.30 ± 0.42	28.13 ± 0.50	30.80 ± 0.98	33.23 ± 1.51
FedZO	21.74 ± 1.91	29.46 ± 0.38	26.99 ± 0.17	31.67 ± 0.35
FedKSeed	22.33 ± 1.72	29.77 ± 0.75	**30.91** ± 0.29	34.56 ± 0.28
FedAvg	23.95 ± 2.76	**32.11** ± 0.70	29.67 ± 1.26	30.98 ± 1.66
Ferret (ours)	**24.99** ± 0.99	30.03 ± 0.99	30.63 ± 0.84	**34.57** ± 0.57

Table 3: More comparison of Rouge-L (%) among various algorithms. Each cell reports the mean ± std of Rouge-L scores from the final round of four runs, each with a different random seed. The highest and second-highest scores are shown in **bold** and underline, respectively.

Algorithm	CodeAlpaca		GSM8K	
	LLaMA2-7B	LLaMA2-13B	LLaMA2-7B	LLaMA2-13B
FedIT	4.66 ± 0.18	6.10 ± 0.18	30.31 ± 0.29	13.46 ± 0.34
FedZO	4.58 ± 0.26	6.19 ± 0.32	30.41 ± 0.31	13.63 ± 0.34
FedKSeed	8.33 ± 0.98	10.70 ± 0.47	28.26 ± 3.60	33.67 ± 1.15
FedAvg	**15.41** ± 0.43	**14.68** ± 0.26	**38.30** ± 0.40	**39.82** ± 0.17
Ferret (ours)	12.10 ± 0.47	11.84 ± 0.91	36.10 ± 1.18	34.50 ± 1.42

5.1. Comparison on Accuracy

We present the model accuracy of different federated tuning methods in Tab. 2 and 3. The results in Tab. 2 demonstrate that federated full-parameter tuning methods (including FedAvg, FedZO, FedKSeed, and Ferret) generally achieve better model accuracy compared to PEFT-based federated

Table 4: Comparison of per-round computational cost and communication overhead on LLaMA-3B, including (a) the computational costs of local updates, global aggregation, and overall cost per round; and (b) the per-round communication cost. The improvement achieved by our Ferret is reported in brackets using blue (compared with FedKSeed) and orange (compared with FedAvg).

Algorithm	Computational Cost (Sec.)			Communication Cost
	Local Update	Global Aggr.	Overall	(# params.)
FedZO	32.6	0.3	32.9	6.0×10^9
FedKSeed	56.9	123.8	180.7	8.2×10^3
FedAvg	1.8	0.3	2.1	6.0×10^9
Ferret (ours)	**5.6** (10.2×)	**24.7** (5.0×)	**30.3** (6.0×)	**7.8×10^3** (10^6×)

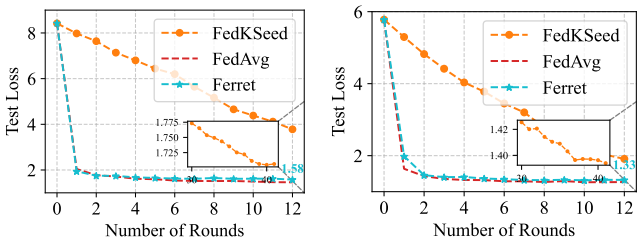
Table 5: Comparison of per-round computational cost and communication overhead on LLaMA2-7B, focusing on (a) the computational costs from local updates, global aggregation, and the overall cost per round; and (b) the per-round communication cost. The improvement achieved by our Ferret is reported in brackets using blue (compared with FedKSeed) and orange (compared with FedAvg).

Algorithm	Computational Cost (Sec.)			Communication Cost
	Local Update	Global Aggr.	Overall	(# params.)
FedZO	54.1	0.7	54.8	1.4×10^{10}
FedKSeed	117.0	510.0	627.0	8.2×10^3
FedAvg	5.8	0.7	6.5	1.4×10^{10}
Ferret (ours)	**8.9** (13.1×)	**88.3** (5.8×)	**97.2** (6.5×)	**6.4×10^3** (10^6×)

tuning methods (such as FedPTuning, FedPrompt, FedIT-SGD, and FedIT). This underscores the importance of full-parameter tuning for Large Language Models (LLMs). Importantly, the results in both tables show that our proposed method consistently delivers strong or competitive performance across four different scenarios. Specifically, on the Natural Instructions dataset, our method outperforms all others for different model sizes, with up to a 2.66% improvement over the next best method, FedKSeed. On the Dolly-15K dataset, our method maintains competitive performance. Moreover, on both the CodeAlpaca and GSM8K datasets, our method achieves noticeably improved accuracy over FedIT and other zeroth-order baselines (i.e., FedZO and FedKseed). Of note, Ferret slightly underperform FedAvg, likely due to reconstruction errors caused by our method for these complex tasks. Overall, these results have well demonstrated the ability of our method to sustain strong model accuracy in practice across various datasets and model sizes.

5.2. Comparison on Scalability

Since we focus on federated full-parameter tuning of LLMs, we primarily provide a detailed scalability comparison of this type of methods, including FedZO, FedKSeed, FedAvg, and Ferret. We evaluate their scalability performance on Natural Instructions using LLaMA-3B (see Tab. 4) and



(a) DataJuicer-1.3B (b) LLaMA-3B

Figure 2: Comparison of communication rounds required by Ferret, FedKSeed, and FedAvg for convergence on Natural Instructions with (a) DataJuicer-1.3B and (b) LLaMA-3B.

Table 6: Peak GPU memory footprint of different methods.

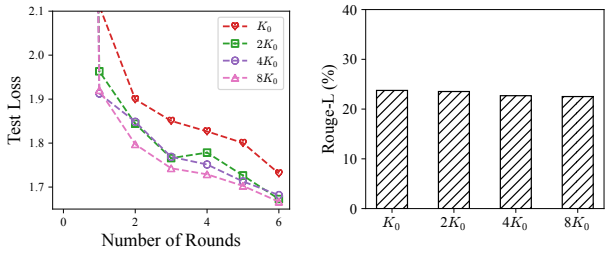
Method	DataJuicer-1.3B	LLaMA-3B
FedAvg	9.9 GB	19.1 GB
FedKSeed	3.5 GB	7.8 GB
Ferret (ours)	9.9 GB	19.1 GB

GSM8K using LLaMA2-7B (see Tab. 5), where the calculation of computational cost and communication overhead is in Appx. C.2 and more comparison on LLaMA2-13B is in Appx. C.4. The results in Tab. 4 and Tab. 5 demonstrate that compared with FedKSeed, Ferret achieves substantial reductions in computational costs: a 10.2× improvement for local updates on LLaMA-3B and 13.1× on LLaMA2-7B, a 5.0× improvement in global aggregation on LLaMA-3B and 5.8× on LLaMA2-7B, as well as a 6.5× improvement for overall computational cost per round on LLaMA-3B and 6.8× on LLaMA2-7B. These advancements stem from several key innovations: our first-order local updates, which reduce the number of required iterations; block-wise reconstruction, which optimizes global aggregation; and precise reconstruction, which significantly decreases communication round complexity. Furthermore, compared to FedAvg that does not leverage any shared randomness, Ferret exhibits an enormous reduction in overall communication costs, i.e., $10^6\times$ on LLaMA-3B and $10^7\times$ on LLaMA2-7B. This emphasizes the ability of Ferret in scaling federated full-parameter tuning.

In Fig. 2, we also compare the convergence speeds of our Ferret with other baselines (e.g., FedKSeed and FedAvg) on Natural Instructions (with DataJuicer-1.3B and LLaMA-3B). The findings show that, Ferret converges remarkably fast, requiring only two communication rounds in line with FedAvg compared to the 40 rounds needed by FedKSeed. This results in a 20× reduction in communication round complexity for both DataJuicer-1.3B and LLaMA-3B.

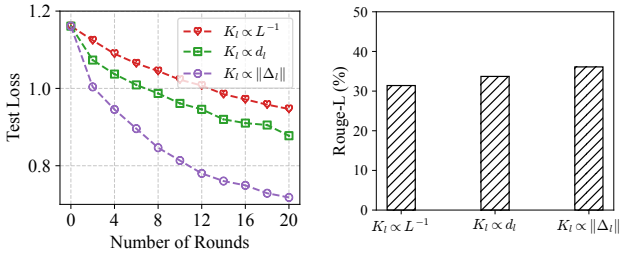
5.3. Comparison on GPU Memory Consumption

We further provide the comparison on GPU memory consumption for different methods in Tab. 6. The results demon-



(a) Convergence (b) Generalization

Figure 3: Convergence and generalization of our Ferret under varying K on Natural Instructions with DataJuicer-1.3B where $2K_0$ corresponds to the communication cost of 7.8×10^3 per round in Tab. 4.



(a) Convergence (b) Generalization

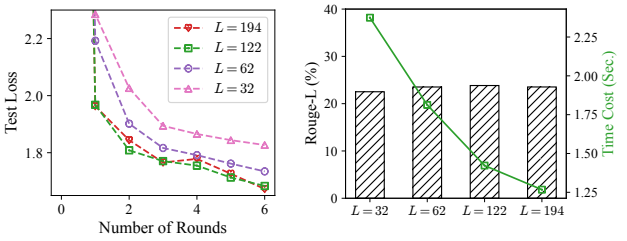
Figure 4: Convergence and generalization of Ferret under varying allocation scheme of K for our block-wise reconstruction, in which $K_l \propto \|\Delta_l\|$ corresponds to our results in the previous sections.

strate that our proposed method Ferret maintains the same peak GPU memory footprint as FedAvg, requiring 9.9 GB and 19.1 GB for DataJuicer-1.3B and LLaMA-3B models, respectively, on the Natural Instructions dataset. Although FedKSeed, as a zeroth-order method, exhibits lower memory requirements, our Ferret delivers superior converged performance (refer to Sec. 5.1) and improved scalability (refer to Sec. 5.2) without introducing additional memory overhead compared to the baseline FedAvg method.

5.4. Ablation Studies

Convergence and Generalization of Ferret under Varying K. In Fig. 3, we present the convergence and generalization of Ferret under varying K on the Natural Instructions dataset with DataJuicer-1.3B, using the same experimental setup as described in Appx. C.1. Notably, Fig. 3 shows that: *(a)* a larger number of random bases (i.e., a larger K_0) generally leads to improved convergence, while the generalization performance remains comparable; *(b)* $2K_0$ already provides compelling convergence and generalization performance, and further increasing K yields only marginal improvements in convergence; and *(c)* a slight decrease in generalization performance as K increases is likely due to the reduced regularization effect from noisy gradients.

Convergence and Generalization of Ferret under Varying Allocation of K. In Fig. 4, we present the convergence and generalization of Ferret under different allocation schemes for K in our block-wise reconstruction, using the same experimental setup described in Appx. C.1, where $K_l \propto \|\Delta_l\|$ corresponds to our results in Sec. 5. Notably, Fig. 4 shows that Ferret achieves both faster convergence and improved generalization performance by following the best practices guided by Prop. 2. These findings therefore validate the significance and correctness of our Prop. 2.



(a) Convergence (b) Generalization & Time Cost

Figure 5: Convergence, generalization, and projection time cost per round of Ferret under varying L on Natural Instructions with DataJuicer-1.3B where $L = 194$ is used in Tab. 2.

Convergence and Generalization of Ferret under Varying L. In Fig. 5, we present the convergence, generalization, and projection time cost of Ferret under varying block sizes L on the Natural Instructions dataset with DataJuicer-1.3B, using the same experimental setup as described in Appx. 20. Notably, Fig. 5 shows that increasing the number of blocks (i.e., a larger L) leads to improved convergence and reduced time cost for projection and reconstruction, while the generalization performance remains comparable. This improved convergence is likely due to the logarithmic term in the reconstruction error of our (7), as a larger number of blocks reduces the dimensionality of each block, thereby minimizing reconstruction error. In addition, the reduced time cost aligns with our analysis in Sec. 4.1 and the empirical results shown in Fig. 8(c), further highlighting the efficacy of our block-wise reconstruction method (8).

6. Conclusion

In this paper, we introduce Ferret, which offers a highly desirable solution for scalable federated full-parameter tuning of LLMs. By achieving high computational efficiency, fast convergence, and reduced communication overhead, Ferret overcomes the limitations of existing methods, striking an improved balance among these critical factors. Moreover, our rigorous theoretical analyses and extensive experiments validate Ferret as a robust and reliable approach for deploying LLMs in large-scale federated settings. While our work focuses on the performance and efficiency of Ferret, a critical area for future research is the in-depth privacy analysis of our method and exploring privacy-preserving mechanisms to ensure data security.

Acknowledgements

This research/project is supported by the National Research Foundation, Singapore under its National Large Language Models Funding Initiative (AISG Award No: AISG-NMLP-2024-001). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore. This research/project is supported by SAP and Singapore's Economic Development Board under the Industrial Postgraduate Programme.

Impact Statement

This paper presents work whose goal is to advance the field of federated learning for large language models. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Berahas, A. S., Cao, L., Choromanski, K., and Scheinberg, K. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. Found. Comput. Math., 22(2):507–560, 2022.

Bernstein, J., Wang, Y., Azizzadenesheli, K., and Anandkumar, A. SIGNSGD: compressed optimisation for non-convex problems. In Proc. ICML, 2018.

Chaudhary, S. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca, 2023.

Che, T., Liu, J., Zhou, Y., Ren, J., Zhou, J., Sheng, V. S., Dai, H., and Dou, D. Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization. In Proc. EMNLP, 2023.

Chen, D., Huang, Y., Ma, Z., Chen, H., Pan, X., Ge, C., Gao, D., Xie, Y., Liu, Z., Gao, J., et al. Data-juicer: A one-stop data processing system for large language models. arXiv:2309.02033, 2023.

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

Conover, M., Hayes, M., Mathur, A., Xie, J., Wan, J., Shah, S., Ghodsi, A., Wendell, P., Zaharia, M., and Xin, R. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023.

Dorfman, R., Vargaftik, S., Ben-Itzhak, Y., and Levy, K. Y. Docofl: Downlink compression for cross-device federated learning. In Proc. ICML, 2023.

Fang, W., Yu, Z., Jiang, Y., Shi, Y., Jones, C. N., and Zhou, Y. Communication-efficient stochastic zeroth-order optimization for federated learning. IEEE Trans. Signal Process., 70:5058–5073, 2022.

Feng, H., Pang, T., Du, C., Chen, W., Yan, S., and Lin, M. Does federated learning really need backpropagation? arXiv:2301.12195, 2023.

Frantar, E. and Alistarh, D. Sparsegpt: Massive language models can be accurately pruned in one-shot. In Proc. ICML, 2023.

Ghadimi, S. and Lan, G. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. SIAM Journal on Optimization, 23(4):2341–2368, 2013.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. In Proc. ICLR, 2022.

Hu, W., Shu, Y., Yu, Z., Wu, Z., Lin, X., Dai, Z., Ng, S.-K., and Low, B. K. H. Localized zeroth-order prompt optimization. In Proc. NeurIPS, 2024.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. SCAFFOLD: Stochastic controlled averaging for federated learning. In Proc. ICML, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Proc. ICML, 2014.

Kuang, W., Qian, B., Li, Z., Chen, D., Gao, D., Pan, X., Xie, Y., Li, Y., Ding, B., and Zhou, J. Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning. arXiv:2309.00363, 2023.

Kuang, W., Qian, B., Li, Z., Chen, D., Gao, D., Pan, X., Xie, Y., Li, Y., Ding, B., and Zhou, J. Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning. In Proc. KDD, 2024.

Lau, G. K. R., Hu, W., Diwen, L., Jizhuo, C., Ng, S.-K., and Low, B. K. H. Dipper: Diversity in prompts for producing large language model ensembles in reasoning tasks. In NeurIPS 2024 Workshop on Foundation Model Interventions, 2024.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. In Proc. EMNLP, 2021.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In Proc. ICML, 2020.

Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out, pp. 74–81. Association for Computational Linguistics, 2004.

Lin, X., Wu, Z., Dai, Z., Hu, W., Shu, Y., Ng, S.-K., Jaillet, P., and Low, B. K. H. Use your instinct: Instruction optimization using neural bandits coupled with transformers. In Proc. ICML, 2024.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, B. Deep gradient compression: Reducing the communication bandwidth for distributed training. In Proc. ICLR, 2018.

Liu, J., Xia, C. S., Wang, Y., and Zhang, L. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In Proc. NeurIPS, 2024.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In Proc. ICLR, 2019.

Luo, Q., Yu, H., and Li, X. Badam: A memory efficient full parameter optimization method for large language models. In Proc. NeurIPS, 2024.

Malladi, S., Gao, T., Nichani, E., Damian, A., Lee, J. D., Chen, D., and Arora, S. Fine-tuning language models with just forward passes. In Proc. NeurIPS, 2023.

Maritan, A., Dey, S., and Schenato, L. Fedzen: Towards superlinear zeroth-order federated learning via incremental hessian estimation. arXiv:2309.17174, 2023.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In Proc. AISTATS, 2017.

Nesterov, Y. E. and Spokoiny, V. G. Random gradient-free minimization of convex functions. Found. Comput. Math., 17(2):527–566, 2017.

Pu, G., Jain, A., Yin, J., and Kaplan, R. Empirical analysis of the strengths and weaknesses of peft techniques for llms. arXiv:2304.14999, 2023.

Qin, Z., Chen, D., Qian, B., Ding, B., Li, Y., and Deng, S. Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes. In Proc. ICML, 2024.

Rahimi, M. M., Bhatti, H. I., Park, Y., Kousar, H., Kim, D.-Y., and Moon, J. Evofed: leveraging evolutionary strategies for communication-efficient federated learning. In Proc. NeurIPS, 2024.

Rothchild, D., Panda, A., Ullah, E., Ivkin, N., Stoica, I., Braverman, V., Gonzalez, J., and Arora, R. Fetchsgd: Communication-efficient federated learning with sketching. In Proc. ICML, 2020.

Shao, Y., Li, L., Dai, J., and Qiu, X. Character-llm: A trainable agent for role-playing. In Proc. EMNLP, 2023.

Shu, Y., Lin, X., Dai, Z., and Low, B. K. H. Heterogeneous federated zeroth-order optimization using gradient surrogates. In ICML 2024 Workshop on Differentiable Almost Everything: Differentiable Relaxations, Algorithms, Operators, and Simulators, 2024.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford alpaca: an instruction-following llama model. https://github. com/tatsu-lab/stanford_alpaca, 2023.

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

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. arXiv:2307.09288, 2023b.

Tropp, J. A. et al. An introduction to matrix concentration inequalities. Foundations and Trends® in Machine Learning, 8(1-2):1–230, 2015.

Van Veen, D., Van Uden, C., Blankemeier, L., Delbrouck, J.-B., Aali, A., Bluethgen, C., Pareek, A., Polacin, M., Reis, E. P., Seehofnerová, A., et al. Adapted large language models can outperform medical experts in clinical text summarization. Nature medicine, 30(4):1134–1142, 2024.

Vershynin, R. Introduction to the non-asymptotic analysis of random matrices. arXiv:1011.3027, 2010.

Vershynin, R. How close is the sample covariance matrix to the actual covariance matrix? Journal of Theoretical Probability, 25(3):655–686, 2012.

Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., Mirzaei, A., Arunkumar, A., Ashok, A., Dhanasekaran, A. S., Naik, A., Stap, D., et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In Proc. EMNLP, 2022.

Wei, C., Shu, Y., He, Y. T., and Yu, F. R. Flexora: Flexible low rank adaptation for large language models. arXiv:2408.10774, 2024.

Xu, H., Sharaf, A., Chen, Y., Tan, W., Shen, L., Van Durme, B., Murray, K., and Kim, Y. J. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. In Proc. ICML, 2024.

Xu, M., Wu, Y., Cai, D., Li, X., and Wang, S. Federated fine-tuning of billion-sized language models across mobile devices. arXiv:2308.13894, 2023.

Zelikman, E., Huang, Q., Liang, P., Haber, N., and Goodman, N. D. Just one byte (per gradient): A note on low-bandwidth decentralized language model finetuning using shared randomness. arXiv:2306.10015, 2023.

Zhang, J., Vahidian, S., Kuo, M., Li, C., Zhang, R., Yu, T., Wang, G., and Chen, Y. Towards building the federatedgpt: Federated instruction tuning. In Proc. ICASSP, 2024a.

Zhang, J., Vahidian, S., Kuo, M., Li, C., Zhang, R., Yu, T., Wang, G., and Chen, Y. Towards building the federatedgpt: Federated instruction tuning. In Proc. ICASSP, 2024b.

Zhang, Z., Yang, Y., Dai, Y., Wang, Q., Yu, Y., Qu, L., and Xu, Z. Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In Proc. ACL, 2023.

A. Related Work

Federated Parameter-Efficient Fine-Tuning (PEFT) for LLMs. Federated learning (FL) has become a crucial approach for training large language models (LLMs) while preserving data privacy. A significant body of work in this area has focused on parameter-efficient fine-tuning (PEFT) techniques (Hu et al., 2022; Wei et al., 2024; Lester et al., 2021; Lin et al., 2024; Hu et al., 2024; Lau et al., 2024) to address the substantial communication costs associated with training massive LLMs in distributed settings. These methods, such as those explored in (Zhang et al., 2023; Kuang et al., 2024; Zhang et al., 2024a; Kuang et al., 2023), aim to reduce the number of trainable parameters, thereby decreasing communication overhead. While PEFT approaches offer a viable solution, they often compromise on model accuracy compared to full-parameter fine-tuning, particularly in challenging non-IID (non-independent and identically distributed) data scenarios common in FL (Pu et al., 2023). In contrast, our work tackles the challenge of federated full-parameter tuning of LLMs, aiming to achieve accuracy levels comparable to centralized training without the prohibitive communication costs. This gap in achieving optimal accuracy with existing federated PEFT methods underscores the need for our approach.

Federated Learning with Gradient Compression. Another line of research explores gradient compression techniques to reduce communication overhead in federated learning. Methods like top-k sparsification (Lin et al., 2018), signSGD (Bernstein et al., 2018), and sketching-based compression (Rothchild et al., 2020) aim to reduce the size of transmitted gradients. Yet, these methods often achieve limited compression ratios and may not be effective for the high-dimensional parameter spaces of LLMs, particularly during full-parameter tuning. The millions of parameters involved in full-parameter LLM tuning far exceed the compression capabilities of these approaches. Furthermore, these techniques are not specifically tailored for the unique characteristics of LLMs, potentially limiting their effectiveness. In contrast, our proposed method leverages shared randomness within first-order FL to achieve significantly higher compression rates, scaling down the communication cost by orders of magnitude. This specialization for LLMs provides a critical advantage.

Federated Learning with Shared Randomness. The use of shared randomness has emerged as a promising avenue for reducing communication costs in FL. Methods like (Qin et al., 2024; Xu et al., 2023; Maritan et al., 2023; Feng et al., 2023; Dorfman et al., 2023; Zelikman et al., 2023; Rahimi et al., 2024) demonstrate that transmitting only random seeds and scalar gradients can drastically reduce communication overhead. However, these methods typically rely on zeroth-order optimization (ZOO) for local updates on each client. ZOO methods, while communication-efficient, are computationally expensive, requiring more rounds to converge than first-order methods like FedAvg (McMahan et al., 2017) and FedProx (Li et al., 2020). This computational inefficiency hinders scalability, particularly in large-scale federated environments. In contrast, our work introduces the use of shared randomness within first-order FL, combining the communication efficiency of shared randomness with the computational efficiency of first-order optimization. To the best of our knowledge, this is the first time that shared randomness has been successfully integrated with first-order FL for LLM fine-tuning, addressing a critical gap in the field.

B. Proofs

B.1. Proof of Thm. 1

Suppose v is randomly and independently sampled from a truncated normal distribution, i.e., $\mathrm{v} \sim \mathcal{N}(0,1)$ with $\mathrm{v} \in [-1/\sqrt{d}, 1/\sqrt{d}]$, we have

$$\mathbb{E}\left[\mathrm{v}\right] = 0\,, \tag{9}$$

and also

$$\begin{aligned}
\mathbb{E}\left[\mathrm{v}^2\right] &= \left(\mathbb{E}\left[\mathrm{v}\right]\right)^2 + \mathrm{VAR}(\mathrm{v}) \\
&= \mathrm{VAR}(\mathrm{v}) \\
&= 1 - \frac{1/\sqrt{d}(\psi(1/\sqrt{d}) + \psi(-1/\sqrt{d}))}{\Phi(1/\sqrt{d}) - \Phi(-1/\sqrt{d})} - \left(\frac{\psi(1/\sqrt{d}) - \psi(-1/\sqrt{d})}{\Phi(1/\sqrt{d}) - \Phi(-1/\sqrt{d})}\right)^2 \\
&= 1 - \frac{2\psi(1/\sqrt{d})/\sqrt{d}}{2\Phi(1/\sqrt{d}) - 1}
\end{aligned} \tag{10}$$

where $\psi(\frac{1}{\sqrt{d}})$ and $\Phi(\frac{1}{\sqrt{d}})$ is the probability density function (PDF) and cumulative distribution function (CDF) of the standard normal distribution evaluated at $1/\sqrt{d}$, respectively.

According to Sec. 3.2, each element v in $\mathbf{V}$ is randomly and independently sampled from the truncated normal distribution above. We therefore have the following to conclude our proof:

$$\mathbb{E}\left[\widetilde{\Delta}\right] = \frac{1}{\rho K}\mathbb{E}\left[\mathbf{V}\mathbf{V}^\top\right]\Delta = \frac{1}{\rho K}\mathbb{E}\left[\sum_{k=1}^{K}\mathbf{v}_k\mathbf{v}_k^\top\right]\Delta = \Delta\,. \tag{11}$$

B.2. Proof of Thm. 2

To begin with, we introduce the lemma below to ease our proof.

Lemma 1 (Matrix Bernstein Inequality, Thm. 1.6.2 in (Tropp et al., 2015))**.** *Let $\mathbf{X}_1, \cdots, \mathbf{X}_K$ be independent, zero mean, and symmetry matrices of size $d \times d$, if $\|\mathbf{X}_k\| \leq C$ for any $k \in [K]$, we then have*

$$\mathbb{E}\left[\left\|\sum_{k=1}^{K}\mathbf{X}_k\right\|\right] \leq \sqrt{2\nu\ln(2d)} + \frac{1}{3}C\ln(2d) \tag{12}$$

where $\nu \triangleq \left\|\sum_{k=1}^{K}\mathbb{E}\left[\mathbf{X}_k^2\right]\right\|$.

Define $\mathbf{X}_k \triangleq \left(\mathbf{v}_k\mathbf{v}_k^\top - \rho\mathbf{I}_d\right)/K$, We have

$$\begin{aligned}
\|\mathbf{X}_k\| &\overset{(a)}{=} \frac{1}{K}\left\|\mathbf{v}_k\mathbf{v}_k^\top - \rho\mathbf{I}_d\right\| \\
&\overset{(b)}{\leq} \frac{1}{K}\left(\left\|\mathbf{v}_k\mathbf{v}_k^\top\right\| + \rho\left\|\mathbf{I}_d\right\|\right) \\
&\overset{(c)}{=} \frac{1}{K}\left(\left\|\mathbf{v}_k^\top\mathbf{v}_k\right\| + \rho\right) \\
&\overset{(d)}{\leq} \frac{2}{K}
\end{aligned} \tag{13}$$

where (b) comes from triangle inequality and (c) is due to the fact that outer product $\mathbf{v}_k\mathbf{v}_k^\top$ and inner product $\mathbf{v}_k^\top\mathbf{v}_k$ shares the same operator norm. Finally, (d) results from $\rho < 1$ and $\mathbf{v}_k^\top\mathbf{v}_k \leq 1$.

Besides, we also have

$$\begin{aligned}
\mathbb{E}\left[\mathbf{X}_k^2\right] &\overset{(a)}{=} \frac{1}{K^2}\mathbb{E}\left[\mathbf{v}_k\mathbf{v}_k^\top\mathbf{v}_k\mathbf{v}_k^\top - 2\rho\mathbf{v}_k\mathbf{v}_k^\top + \rho^2\mathbf{I}_d\right] \\
&\overset{(b)}{\preceq} \frac{1}{K^2}\mathbb{E}\left[\mathbf{v}_k\mathbf{v}_k^\top - 2\rho\mathbf{v}_k\mathbf{v}_k^\top + \rho^2\mathbf{I}_d\right] \\
&\overset{(c)}{=} \frac{1}{K^2}\left(\rho - \rho^2\right)\mathbf{I}_d \\
&\overset{(d)}{\preceq} \frac{\rho}{K^2}\mathbf{I}_d
\end{aligned} \tag{14}$$

where (b) comes from the fact that $\mathbf{v}_k^\top\mathbf{v}_k \leq 1$ and (c) is due to the fact that $\mathbb{E}\left[\mathbf{v}_k\mathbf{v}_k^\top\right] = \rho\mathbf{I}_d$.

As a result, by introducing the results above with a triangle inequality, we have

$$\left\|\sum_{k=1}^{K}\mathbb{E}\left[\mathbf{X}_k^2\right]\right\| \leq \sum_{k=1}^{K}\left\|\frac{\rho}{K^2}\mathbf{I}_d\right\| \leq \frac{\rho}{K}\,. \tag{15}$$

By introducing the results above into Lemma. 1,

$$\begin{aligned}\mathbb{E}\left[\left\|\widetilde{\Delta}-\Delta\right\|\right] &= \mathbb{E}\left[\left\|\frac{1}{\rho K}\mathbf{V}\mathbf{V}^\top\Delta-\Delta\right\|\right]\\ &\leq \mathbb{E}\left[\left\|\frac{1}{\rho K}\mathbf{V}\mathbf{V}^\top-\mathbf{I}_d\right\|\right]\|\Delta\|\\ &= \frac{1}{\rho}\mathbb{E}\left[\left\|\sum_{k=1}^{K}\mathbf{X}_k\right\|\right]\|\Delta\|\\ &\leq \sqrt{\frac{2\ln(2d)}{\rho K}}+\frac{\ln(2d)}{\rho K}\,,\end{aligned} \tag{16}$$

which finally concludes our proof.

Remark 1. Sampling from a truncated normal distribution (rather than a standard normal distribution) ensures a bounded norm, which is crucial for achieving a bounded reconstruction error by our method in Sec. 3.2.

B.3. Proof of Thm. 3

Since the loss function $\ell(\cdot;\cdot)$ is assumed to be β-smooth w.r.t its first argument, we then have

$$\ell(\mathbf{w}+\epsilon\mathbf{v}_k;\mathbf{x}^{(i)})-\ell(\mathbf{w};\mathbf{x}^{(i)})\leq\epsilon\left(\nabla\ell(\mathbf{w};\mathbf{x}^{(i)})\right)^\top\mathbf{v}_k+\frac{1}{2}\beta\epsilon^2\left\|\mathbf{v}_k\right\|^2\leq\epsilon\left(\nabla\ell(\mathbf{w};\mathbf{x}^{(i)})\right)^\top\mathbf{v}_k+\frac{1}{2}\beta\epsilon^2\,. \tag{17}$$

By dividing ϵ on both sides of the inequality above, we have

$$g_k-\mathbf{v}_k^\top\nabla\ell(\mathbf{w};\mathbf{x}^{(i)})\leq\frac{1}{2}\beta\epsilon\,. \tag{18}$$

We therefore can conclude our proof using the results below:

$$\begin{aligned}\left\|\frac{1}{K}\mathbf{V}\boldsymbol{g}-\frac{1}{K}\mathbf{V}\mathbf{V}^\top\nabla\ell(\mathbf{w};\mathbf{x}^{(i)})\right\| &\overset{(a)}{=}\left\|\frac{1}{K}\sum_{k=1}^{K}\left(\mathbf{v}_kg_k-\mathbf{v}_k\mathbf{v}_k^\top\nabla\ell(\mathbf{w};\mathbf{x}^{(i)})\right)\right\|\\ &\overset{(b)}{\leq}\frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{v}_kg_k-\mathbf{v}_k\mathbf{v}_k^\top\nabla\ell(\mathbf{w};\mathbf{x}^{(i)})\right\|\\ &\overset{(c)}{\leq}\frac{1}{K}\sum_{k=1}^{K}\left|g_k-\mathbf{v}_k^\top\nabla\ell(\mathbf{w};\mathbf{x}^{(i)})\right|\|\mathbf{v}_k\|\\ &\overset{(d)}{\leq}\frac{1}{2}\beta\epsilon\end{aligned} \tag{19}$$

where (b) comes from triangle inequality and (d) results from (18) and $\|\mathbf{v}_k\|\leq 1$.

Remark 2. When $\epsilon\to 0$, (18) indicates that $g_k=\mathbf{v}_k^\top\nabla\ell(\mathbf{w};\mathbf{x}^{(i)})$, implying that this scalar gradient in zeroth-order method, e.g., FedKSeed (Qin et al., 2024), is an approximation of directional derivative, i.e., our projected update in (6) when Δ is replaced with $\nabla\ell(\mathbf{w};\mathbf{x}^{(i)})$.

B.4. Proof of Prop. 1

Due to the fact that $d=\sum_{l\in[L]}d_l$, $K=\sum_{l\in[L]}K_l$, and $K_l>0$ for any $l\in[L]$, we have

$$dK=\left(\sum_{l\in[L]}d_l\right)\left(\sum_{l\in[L]}K_l\right)=\sum_{l\in[L]}d_l\left(\sum_{l\in[L]}K_l\right)>\sum_{l\in[L]}d_lK_l\,,$$

which therefore concludes our proof.

B.5. Proof of Prop. 2

Based on our block-wise reconstruction in (8) and Thm. 2, we have

$$\begin{aligned}\mathbb{E}\left[\left\|\widetilde{\Delta}-\Delta\right\|\right] &\overset{(a)}{=} \mathbb{E}\left[\sqrt{\sum_{l\in[L]}\left\|\widetilde{\Delta}_l-\Delta_l\right\|^2}\right] \\ &\overset{(b)}{<} \mathbb{E}\left[\sqrt{\left(\sum_{l\in[L]}\left\|\widetilde{\Delta}_l-\Delta_l\right\|\right)^2}\right] \\ &\overset{(c)}{=} \sum_{l\in[L]}\mathbb{E}\left[\left\|\widetilde{\Delta}_l-\Delta_l\right\|\right] \\ &\overset{(d)}{\leq} \sum_{l\in[L]}\left(\sqrt{\frac{2\ln(2d_l)}{\rho_l K_l}}+\frac{\ln(2d_l)}{\rho_l K_l}\right)\|\Delta_l\|\end{aligned} \tag{20}$$

where (a) is based on the definition of $\widetilde{\Delta}_l$ and Δ_l and (b) is from the fact that $\left\|\widetilde{\Delta}_l-\Delta_l\right\|>0$.

Given that $\sqrt{d_l}>K_l$ and we can then use $\widetilde{\mathcal{O}}$ to hide the logarithm term in the result above, the following then holds:

$$\mathbb{E}\left[\left\|\widetilde{\Delta}-\Delta\right\|\right] < \widetilde{\mathcal{O}}\left(\sum_{l\in[L]}\frac{\|\Delta_l\|}{\rho_l K_l}\right) . \tag{21}$$

To minimize the upper bound above w.r.t $\{K_l\}_{l=1}^L$ with $\sum_{l\in[L]}K_l=K$, we resort to KKT conditions. Specifically, define $\boldsymbol{k}\triangleq[K_1,\cdots,K_L]^\top$ and the following Lagrangian function based on $\lambda>0$:

$$F(\boldsymbol{k},\lambda)\triangleq\sum_{l\in[L]}\frac{\|\Delta_l\|}{\rho_l K_l}+\lambda\left(\sum_{l\in[L]}K_l-K\right) . \tag{22}$$

To minimize (21), for any $l\in[L]$, K_l and λ then needs to satisfy the following condition:

$$\frac{\partial F(\boldsymbol{k},\lambda)}{\partial K_l}=-\frac{\|\Delta_l\|/\rho_l}{K_l^2}+\lambda=0 . \tag{23}$$

That is,

$$\lambda=\frac{\|\Delta_1\|/\rho_1}{K_1^2}=\cdots=\frac{\|\Delta_L\|/\rho_L}{K_L^2} . \tag{24}$$

This finally leads to $K_l\propto\sqrt{\|\Delta_L\|/\rho_L}$, which consequently concludes our proof.

Remark 3. Prop. 2 provides a looser bound than Thm. 2, primarily owing to the inequality (b) in (20). Based on this looser bound, one might expect that block-wise reconstruction would incur a larger error compared to the vanilla reconstruction in (7). However, empirical results in Appx. C.5 and Appx. C.6 show that block-wise reconstruction yields comparable performance to the vanilla approach.

B.6. Proof of Thm. 4

Of note, we follow the general idea in (Shu et al., 2024) to prove the convergence of Ferret. To begin with, we introduce the following lemmas borrowed from (Shu et al., 2024):

Lemma 2. *Let $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_\tau\}$ be any τ vectors in $\mathbb{R}^d$. Then the following holds for any $a > 0$:*

$$\|\boldsymbol{u}_i\| \left\|\boldsymbol{u}_j\right\| \leq \frac{a}{2} \|\boldsymbol{u}_i\|^2 + \frac{1}{2a} \left\|\boldsymbol{u}_j\right\|^2 , \tag{25}$$

$$\left\|\boldsymbol{u}_i + \boldsymbol{u}_j\right\|^2 \leq (1+a) \|\boldsymbol{u}_i\|^2 + \left(1 + \frac{1}{a}\right) \left\|\boldsymbol{u}_j\right\|^2 , \tag{26}$$

$$\left\|\sum_{i=1}^{\tau} \boldsymbol{u}_i\right\|^2 \leq \tau \sum_{i=1}^{\tau} \|\boldsymbol{u}_i\|^2 . \tag{27}$$

Lemma 3. *For any β-smooth function f, inputs $\boldsymbol{x}, \boldsymbol{y}$ in the domain of f, the following holds for any constant $\eta > 0$:*

$$\|\boldsymbol{x} - \eta \nabla f(\boldsymbol{x}) - \boldsymbol{y} + \eta \nabla f(\boldsymbol{y})\|^2 \leq (1 + \eta\beta)^2 \|\boldsymbol{x} - \boldsymbol{y}\|^2 .$$

Let $\eta \leq 1/(T\beta)$, we can bound the discrepancy between $\mathbf{w}_{r,t}^{(i)}$ and $\mathbf{w}_r$ for any client i as below

$$\begin{aligned}
&\mathbb{E}\left[\left\|\mathbf{w}_{r,t}^{(i)} - \mathbf{w}_r\right\|^2\right] \\
&\overset{(a)}{=} \mathbb{E}\left[\left\|\mathbf{w}_{r,t-1}^{(i)} - \eta \nabla \ell(\mathbf{w}_{r,t-1}^{(i)}; \mathbf{x}_{r,t-1}^{(i)}) - \mathbf{w}_r\right\|^2\right] \\
&\overset{(b)}{=} \mathbb{E}\Big[\Big\|\mathbf{w}_{r,t-1}^{(i)} - \eta \nabla \mathcal{L}(\mathbf{w}_{r,t-1}^{(i)}) + \eta \nabla \mathcal{L}(\mathbf{w}_r) - \mathbf{w}_r + \eta \left(\nabla \mathcal{L}(\mathbf{w}_{r,t-1}^{(i)}) - \nabla \mathcal{L}^{(i)}(\mathbf{w}_{r,t-1}^{(i)})\right) \\
&\qquad\qquad + \eta \left(\nabla \mathcal{L}^{(i)}(\mathbf{w}_{r,t-1}^{(i)}) - \nabla \ell(\mathbf{w}_{r,t-1}^{(i)}; \mathbf{x}_{r,t-1}^{(i)}) - \nabla \mathcal{L}(\mathbf{w}_r)\right)\Big\|^2\Big] \\
&\overset{(c)}{\leq} \frac{T}{T-1} \mathbb{E}\left[\left\|\mathbf{w}_{r,t-1}^{(i)} - \eta \nabla \mathcal{L}(\mathbf{w}_{r,t-1}^{(i)}) + \eta \nabla \mathcal{L}(\mathbf{w}_r) - \mathbf{w}_r\right\|^2\right] \\
&\qquad\qquad + 2\eta^2 T\, \mathbb{E}\left[\left\|\nabla \mathcal{L}^{(i)}(\mathbf{w}_{r,t-1}^{(i)}) - \nabla \ell(\mathbf{w}_{r,t-1}^{(i)}; \mathbf{x}_{r,t-1}^{(i)})\right\|^2 + \|\nabla \mathcal{L}(\mathbf{w}_r)\|^2\right] \\
&\overset{(d)}{\leq} \frac{T(1+\eta\beta)^2}{T-1} \mathbb{E}\left[\left\|\mathbf{w}_{r,t-1}^{(i)} - \mathbf{w}_r\right\|^2\right] + 2\eta^2 T \sigma^2 + 2\eta^2 T^2 \|\nabla \mathcal{L}(\mathbf{w}_r)\|^2 \\
&\overset{(e)}{\leq} 24\eta^2 T^2 \sigma^2 + 24\eta^2 T^2 \|\nabla \mathcal{L}(\mathbf{w}_r)\|^2
\end{aligned} \tag{28}$$

where (a) is from the local update of $\mathbf{w}_{r,t-1}^{(i)}$ on each client i, and (c) is based on (26) in Lemma 2 with $a = 1/(T-1)$ and $\mathcal{L}(\mathbf{w}_{r,t-1}^{(i)}) = \mathcal{L}^{(i)}(\mathbf{w}_{r,t-1}^{(i)})$. Besides, (d) results from Lemma 3 and the assumption that $\mathbb{E}[\|\nabla \mathcal{L}^{(i)}(\mathbf{w}) - \nabla \ell(\mathbf{w}; \mathbf{x})\|^2] \leq \sigma^2$. Finally, (e) comes from the summation of geometric series and the fact that $\eta\beta \leq 1/T$ as well as

$$\begin{aligned}
\sum_{\tau=0}^{t-1} \left(\frac{(T+1)^2}{T(T-1)}\right)^\tau &\leq \sum_{\tau=0}^{T-1} \left(\frac{(T+1)^2}{T(T-1)}\right)^\tau \\
&= \frac{\left((T+1)^2/[T(T-1)]\right)^T - 1}{(T+1)^2/[T(T-1)] - 1} \\
&= \frac{T(T-1)}{3T+1} \left(\left(1 + \frac{3T+1}{T(T-1)}\right)^T - 1\right) \\
&< \frac{T(T-1)}{3T+1} \left(\exp\left(\frac{3T+1}{T}\right) - 1\right) \\
&< \frac{T}{3} \left(\exp\left(\frac{7}{2}\right) - 1\right) \\
&< 12T .
\end{aligned} \tag{29}$$

Besides $\mathbb{E}\left[\mathbf{V}_r\mathbf{V}_r^\top\right] = \rho\mathbf{I}_d$, one can also verify that $\mathbb{E}\left[\mathbf{V}_r\mathbf{V}_r^\top\mathbf{V}_r\mathbf{V}_r^\top\right] = \rho^2\mathbf{I}_d$, we therefore have

$$
\begin{aligned}
&\mathbb{E}\left[\left\|\mathbf{w}_{r+1}-\mathbf{w}_r\right\|^2\right] \\
=&\mathbb{E}\left[\left(\mathbf{w}_{r+1}-\mathbf{w}_r\right)^\top\left(\mathbf{w}_{r+1}-\mathbf{w}_r\right)\right] \\
=&\mathbb{E}\left[\left(\frac{\eta}{\rho KN}\right)^2\left(\sum_{i=1}^N\sum_{t=1}^T\nabla\ell(\mathbf{w}_{r,t-1}^{(i)};\mathbf{x}_{r,t-1}^{(i)})\right)^\top\mathbf{V}_r\mathbf{V}_r^\top\mathbf{V}_r\mathbf{V}_r^\top\sum_{i=1}^N\sum_{t=1}^T\nabla\ell(\mathbf{w}_{r,t-1}^{(i)};\mathbf{x}_{r,t-1}^{(i)})\right] \\
=&\left(\frac{\eta}{\rho KN}\right)^2\mathbb{E}\left[\left(\sum_{i=1}^N\sum_{t=1}^T\nabla\ell(\mathbf{w}_{r,t-1}^{(i)};\mathbf{x}_{r,t-1}^{(i)})\right)^\top\mathbb{E}\left[\mathbf{V}_r\mathbf{V}_r^\top\mathbf{V}_r\mathbf{V}_r^\top\right]\sum_{i=1}^N\sum_{t=1}^T\nabla\ell(\mathbf{w}_{r,t-1}^{(i)};\mathbf{x}_{r,t-1}^{(i)})\right] \\
=&\left(\frac{\eta}{N}\right)^2\mathbb{E}\left[\left\|\sum_{i=1}^N\sum_{t=1}^T\nabla\ell(\mathbf{w}_{r,t-1}^{(i)};\mathbf{x}_{r,t-1}^{(i)})\right\|^2\right] \\
\le&\frac{\eta^2}{N}\sum_{i=1}^N\mathbb{E}\left[\left\|\sum_{t=1}^T\nabla\ell(\mathbf{w}_{r,t-1}^{(i)};\mathbf{x}_{r,t-1}^{(i)})\right\|^2\right]
\end{aligned}
\tag{30}
$$

where the last inequality comes from the (27) in Lemma 2. Here, we omit the subscript r from the random bases $\mathbf{V}$ in our notation for simplicity.

Since $\mathbb{E}\left[\left\|\mathbf{w}_{r,t}^{(i)}-\mathbf{w}_r\right\|^2\right] = \eta^2\mathbb{E}\left[\left\|\sum_{\tau=1}^t\nabla\ell(\mathbf{w}_{r,\tau-1}^{(i)};\mathbf{x}_{r,\tau-1}^{(i)})\right\|^2\right]$, by replacing τ with T, we have

$$
\mathbb{E}\left[\left\|\mathbf{w}_{r+1}-\mathbf{w}_r\right\|^2\right] \le 24\eta^2T^2\sigma^2 + 24\eta^2T^2\left\|\nabla\mathcal{L}(\mathbf{w}_r)\right\|^2 . \tag{31}
$$

Besides, since $\mathbb{E}\left[\mathbf{w}_{r+1}-\mathbf{w}_r\right] = -\frac{\eta}{N}\sum_{i=1}^N\sum_{t=1}^T\nabla\mathcal{L}^{(i)}(\mathbf{w}_{r,t-1}^{(i)})$, we have

$$
\begin{aligned}
&\mathbb{E}\left[\nabla\mathcal{L}(\mathbf{w}_r)^\top(\mathbf{w}_{r+1}-\mathbf{w}_r)\right] \\
\overset{(a)}{=}&-\frac{\eta}{N}\mathbb{E}\left[\sum_{i=1}^N\sum_{t=1}^T\nabla\mathcal{L}(\mathbf{w}_r)^\top\nabla\mathcal{L}^{(i)}(\mathbf{w}_{r,t-1}^{(i)})\right] \\
\overset{(b)}{=}&-\frac{\eta}{N}\mathbb{E}\left[\sum_{i=1}^N\sum_{t=1}^T\nabla\mathcal{L}(\mathbf{w}_r)^\top\left(\nabla\mathcal{L}^{(i)}(\mathbf{w}_{r,t-1}^{(i)})-\nabla\mathcal{L}(\mathbf{w}_{r,t-1}^{(i)})+\nabla\mathcal{L}(\mathbf{w}_{r,t-1}^{(i)})-\nabla\mathcal{L}(\mathbf{w}_r)+\nabla\mathcal{L}(\mathbf{w}_r)\right)\right] \\
\overset{(c)}{\le}&\frac{\eta}{N}\sum_{i=1}^N\sum_{t=1}^T\left(\eta\beta T\left\|\nabla\mathcal{L}(\mathbf{w}_r)\right\|^2+\frac{\beta}{4\eta T}\mathbb{E}\left[\left\|\mathbf{w}_{r,t-1}^{(i)}-\mathbf{w}_r\right\|^2\right]\right)-\eta T\left\|\nabla\mathcal{L}(\mathbf{w}_r)\right\|^2 \\
\overset{(d)}{\le}&\left(7\eta^2T^2\beta-\eta T\right)\left\|\nabla\mathcal{L}(\mathbf{w}_r)\right\|^2+6\eta^2T^2\beta\sigma^2
\end{aligned}
\tag{32}
$$

where (c) comes from Cauchy–Schwarz inequality and the fact that $\mathcal{L}(\mathbf{w}_{r,t-1}^{(i)}) = \mathcal{L}^{(i)}(\mathbf{w}_{r,t-1}^{(i)})$. In addition, (d) results from (31).

Finally, based on the assumption that $\mathcal{L}$ is β-smooth, we naturally have

$$
\begin{aligned}
\mathbb{E}\left[\mathcal{L}(\mathbf{w}_{r+1})-\mathcal{L}(\mathbf{w}_r)\right] &\le \mathbb{E}\left[\nabla\mathcal{L}(\mathbf{w}_r)^\top\left(\mathbf{w}_{r+1}-\mathbf{w}_r\right)\right]+\frac{\beta}{2}\mathbb{E}\left[\left\|\mathbf{w}_{r+1}-\mathbf{w}_r\right\|^2\right] \\
&\le \left(19\eta^2T^2\beta-\eta T\right)\mathbb{E}\left[\left\|\nabla\mathcal{L}(\mathbf{w}_r)\right\|^2\right]+18\eta^2T^2\beta\sigma^2 .
\end{aligned}
\tag{33}
$$

By rearranging and letting $\eta \leq \frac{1}{20T\beta}$, we have

$$\mathbb{E}\left[\left\|\nabla\mathcal{L}(\mathbf{w}_r)\right\|^2\right] \leq \frac{20\,\mathbb{E}\left[\mathcal{L}(\mathbf{w}_r) - \mathcal{L}(\mathbf{w}_{r+1})\right]}{\eta T} + 360\eta T\beta\sigma^2\,, \tag{34}$$

Finally, by summarizing both sides over R rounds and scaling them with $1/R$, we have the following results to conclude our proof:

$$\min_{r\in[R)} \mathbb{E}\left[\left\|\nabla\mathcal{L}(\mathbf{w}_r)\right\|^2\right] \leq \frac{20\left(\mathcal{L}(\mathbf{w}_0) - \min_{\mathbf{w}}\mathcal{L}(\mathbf{w})\right)}{\eta T R} + 360\eta\beta T\sigma^2\,. \tag{35}$$

Remark 4. Note that the large constant in (35) arises from our bound in (28) for sufficiently large T. This bound can be improved in practice by considering a smaller T instead.

C. Experiments

C.1. Experimental Setup

Baselines. In line with the comparison in (Qin et al., 2024), we selected four practical methods for federated LLM tuning as our baselines: (1) FedPTuning (Kuang et al., 2024), (2) FedPrompt (Kuang et al., 2023), (3) FedIT (Zhang et al., 2024a), and (4) FedIT-SGD, a variant of FedIT that replaces Adam with SGD. In addition, we included four full-parameter tuning methods for comparison: (1) FedAvg (McMahan et al., 2017), (2) FedZO (Fang et al., 2022), (3) FedMeZO, a hybrid of FedAvg and MeZO (Malladi et al., 2023), and (4) FedKSeed (Qin et al., 2024).

C.1.1. Setup on the Natural Instruction and Dolly-15K Datasets

Datasets. We conducted our experiments using the Natural Instructions (NI) (Wang et al., 2022) and Dolly-15K (Conover et al., 2023) datasets, following a setup similar to (Qin et al., 2024). For the NI dataset, we allocated 738 training tasks to individual clients for local updates and reserved 119 test tasks for global evaluation, reflecting a non-IID distribution. Meanwhile, for the Dolly-15K dataset, the final task was utilized for global evaluation, while the remaining tasks were distributed among 200 clients with varying levels of label distribution skew. Rouge-L (Lin, 2004) was chosen as the evaluation metric. Given our resource constraints, we selected DataJuicer-1.3B (Chen et al., 2023) and LLaMA-3B (Touvron et al., 2023a) as the base models for our study. The corresponding HuggingFace model paths are "datajuicer/LLaMA-1B-dj-refine-150B" and "openlm-research/open_llama_3b".

Hyper-parameters. For Ferret, the local update learning rate η for each client is set to 1×10^{-4}, where the selected learning rate is searched from $[2 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-5}]$. The global aggregation learning rates on Natural Instruction and Dolly-15K are set to 10.0 and 3.0, respectively, which is search from $[10.0, 5.0, 1.0]$. For other baselines in Tab. 1 of our main paper, we reported their accuracy performances using the results from FedKSeed (Qin et al., 2024).

Prompt Template. In our experiments, the raw input data is pre-processed to follow a structured format, where we warp the input text to the Alpaca prompt template (Taori et al., 2023). The corresponding templates for the NI and Dolly-15K dataset are shown in Tab. 7 and 8.

Table 7: Prompt template for Natural Instructions.

> Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.
>
> ### Instruction: {Definition}
>
> ### Input: {input}
>
> ### Response:

C.1.2. Setup on the CodeAlpaca and GSM8K Datasets

Datasets. To further demonstrate that Ferret can also improve the capability of larger LLMs for code generation and mathematical reasoning, we conducted more experiments using the CodeAlpaca (Chaudhary, 2023) and GSM8K (Cobbe et al., 2021) datasets, following a similar federated setup. The CodeAlpaca dataset (of around 8.0k samples) is a code dataset that consists of ten programming languages, including C, C#, C++, Go, Java, PHP, Pascal, Python, Scale, and X86-64 Assemble. We exclude the X86-64 Assembly data due to limited samples in the dataset. We uniformly randomly sampled 10% instances from the original data as the hold-out test set for evaluation, and we split the remaining 10% samples into nine subsets based on the programming language category and assign each subset to one client as its local training data. For GSM8K, its official train set is split into three subsets, where each client's dataset consists of grade school math questions randomly partitioned from the original dataset, forming a IID distribution. We use the official GSM8K test split as the evaluation dataset. Rouge-L (Lin, 2004) was chosen as the evaluation metric. To demonstrate the scalability of Ferret, we

Table 8: Prompt template for Dolly-15K. If some data instances do not have the `context` attribute, we will discard the line "### Input: " in the template.

> Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.
>
> ### Instruction: {instruction}
>
> ### Input: {context}
>
> ### Response:

extended the experiments to larger models: LLaMA2-7B and LLaMA2-13B (Touvron et al., 2023a) as the base models for our study. The corresponding HuggingFace model paths are "meta-llama/Llama-2-7b-hf" and "meta-llama/Llama-2-13b-hf".

Hyperparameters. For FedZO and FedKSeed, the local update learning rate is set to 3×10^{-7} for all models. For FedAvg on both LLaMA2-7B and LLaMA2-13B, the local update learning rate η for each client is set to 3×10^{-4}, and the global aggregation learning rate is set to 1.0. For Ferret on LLaMA2-7B, the local update learning rate η is set to 3×10^{-4} and the global aggregation learning rate is set to 5.0. For Ferret on LLaMA2-13B, the local update learning rate η is set to 5×10^{-4} and the global aggregation learning rate is set to 10.0. The selected learning rate is searched from $[5 \times 10^{-4}, 3 \times 10^{-4}, 1 \times 10^{-4}]$ and the selected global aggregation learning rates is searched from $[10.0, 5.0, 1.0]$.

C.2. Calculation of Computational Cost and Communication Overhead

In this subsection, we provide the details of how the computational cost and communication cost are calculated for all methods listed in Tab. 4.

Calculation of Computational Cost. For FedZO, we follow the same hyper-parameters ($b_1 = 200, b_2 = 1$) for FedZO from FedKSeed paper (Qin et al., 2024), which employs 200 local update steps and 1 perturbation for each local update step. For calculating the computational cost of FedAvg and Ferret, we apply 10 local update steps for each client. Same as our experimental setting in Tab. 1, the batch size is set to 1 for all methods. The time cost incurred at gradient projection is also included in the Local Update.

In the global aggregation process for both FedZO and FedAvg, raw gradients from all clients are averaged and then used to update the global model. In contrast, for FedKSeed and Ferret, the projected gradients are first aggregated through averaging, then reconstructed, and finally used to update the global model.

For the overall computation cost per round, we follow the calculation below:

$$\text{Overall} = \text{Local Update} + \text{Global Aggr.}$$

Calculation of Communication Overhead. The per-round communication cost refers to the total number of parameters exchanged between a client and the central server during a single round. This includes both the raw or projected gradients that the client sends to the server and the aggregated gradients that the client receives from the server. Each parameter (or projected gradient) is encoded as 16-bit floating point numbers. In accordance with the practice in FedKSeed, we set the number of rounds R to 40 for both FedZO and FedKSeed. Given the notable convergence rate of Ferret, we set R to 12 for both Ferret and FedAvg. Although (Qin et al., 2024) employs $R = 40$ for FedAvg, we use $R = 12$ to provide a strong basis for comparison and to highlight the computational efficiency of Ferret.

C.3. More Comparison on LLaMA3-8B and Qwen2.5-7B Models

We conducted additional experiments on CodeAlpaca and GSM8K using LLaMA3-8B and Qwen2.5-7B in Tab. 9. The results demonstrate the consistent effectiveness of our Ferret, achieving near FedAvg performance with significantly reduced

Table 9: More comparison of Rouge-L (%) among various algorithms on LLaMA3-8B and Qwen2.5-7B models. Each cell reports the mean ± std of Rouge-L scores from the final round of four runs, each with a different random seed. The highest and second-highest scores are shown in **bold** and underline, respectively.

Algorithm	CodeAlpaca		GSM8K	
	LLaMA3-8B	Qwen2.5-7B	LLaMA3-8B	Qwen2.5-7B
FedZO	16.66 ± 0.50	9.14 ± 0.32	7.79 ± 1.36	23.84 ± 1.19
FedKSeed	5.73 ± 1.26	9.14 ± 0.32	7.79 ± 1.36	23.84 ± 1.19
FedAvg	**19.88** ± 0.67	**17.47** ± 0.49	**45.48** ± 0.51	**43.86** ± 0.36
Ferret (ours)	19.59 ± 0.66	14.64 ± 0.74	45.07 ± 0.78	38.28 ± 1.70

Table 10: Comparison of computational cost and communication overhead on LLaMA2-13B, focusing on (a) the computational costs from local updates, global aggregation, and the overall tuning process; and (b) the per-round and overall communication costs. The improvement achieved by our Ferret is reported in brackets using blue (compared with FedKSeed) and orange (compared with FedAvg).

Algorithm	Computational Cost (Sec.)			Communication Cost
	Local Update	Global Aggr.	Overall	(# params.)
FedZO	114.1	25.7	139.8	2.6×10^{10}
FedKSeed	188.4	666.2	854.6	8.2×10^{3}
FedAvg	24.9	25.7	50.6	2.6×10^{10}
Ferret (ours)	**19.2** (9.8×)	**169.4** (3.9×)	**188.6** (4.5×)	$\mathbf{7.6\times10^{3}}$ ($10^{6}\times$)

communication overhead across models and tasks.

C.4. More Comparison of Computational Cost and Communication Overhead

Tab. 10 compares the computational cost and communication overhead of LLaMA2-13B using the GSM8K dataset. Because of GPU memory constraints, FedZO and FedAvg have slightly higher computational costs, as gradients need to be stored on the CPU. The results show that even for large models like LLaMA2-13B, Ferret still demonstrates superior scalability. Compared to FedKSeed, Ferret reduces computational costs significantly: 9.8× for local updates, 3.9× for global aggregation, and 4.5× for overall cost per bound. Additionally, compared to FedAvg, which does not utilize shared randomness, Ferret achieves a dramatic $10^7\times$ reduction in communication costs. These results, along with the evidence in Sec. 5, further highlight the scalability of Ferret in federated full-parameter tuning.

C.5. Ablation Studies on Reconstruction

Rate of $1/\rho$ w.r.t Dimension d. In Fig. 6, we present the rate of $1/\rho$ where ρ is defined in Sec. 3.2 to verify our claim following Thm. 2. The results in Fig. 6 confirm that $1/\rho$ indeed follows a rate of $\mathcal{O}(d)$.

Comparison of Reconstruction Accuracy between Ferret and ZO Method. In Fig. 7, we present the reconstruction accuracy (measured by cosine similarity) for the $d = 10^5$-dimensional gradient of the function $F(\boldsymbol{x}) = \sum_{i=1}^{d} x_i^2$ at a randomly sampled input $\boldsymbol{x}$ with varying K by using our method in (7) and zeroth-order method with different values of ϵ. The goal is to compare the reconstruction accuracy of our (7) with that of the ZO method under varying K and ϵ. The results in Fig. 7 indicate that: *(a)* our method (7) achieves improved reconstruction accuracy compared to the ZO method, particularly the one with an optimal $\epsilon = 0.1$, which indeed aligns with the insights from our Thm. 3; *(b)* both our method (7) and the ZO method exhibit the same increasing rate in reconstruction accuracy as K increases, highlighting the connection between these two methods as implied by our Thm. 3; and *(c)* this increasing rate is generally linear, which is consistent with Thm. 2. These results therefore further verify the insights in Thm. 2 and Thm. 3, and support the advantages of our method (7) over the ZO method.

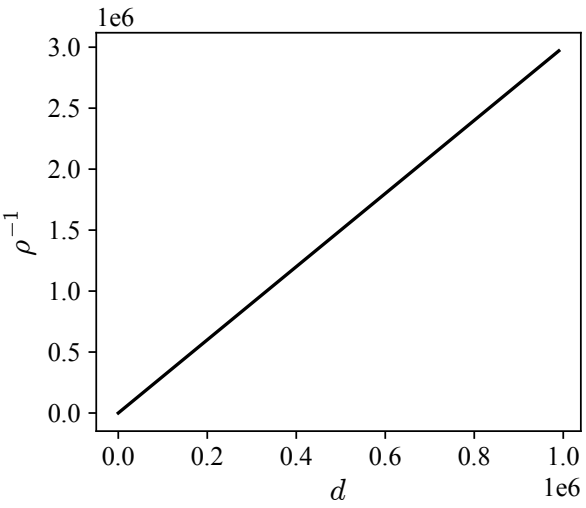


Figure 6: Rate of $1/\rho$ w.r.t. dimension d.

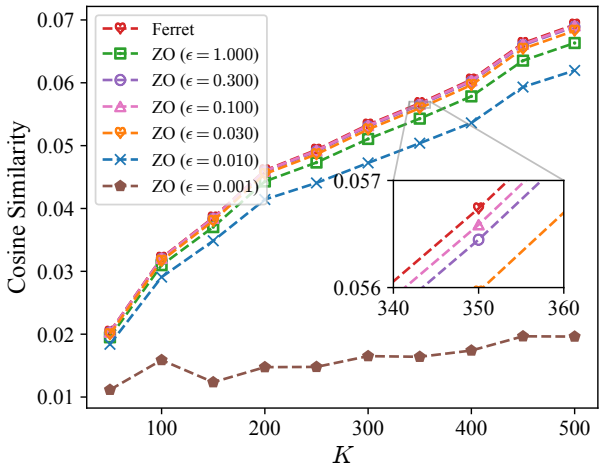
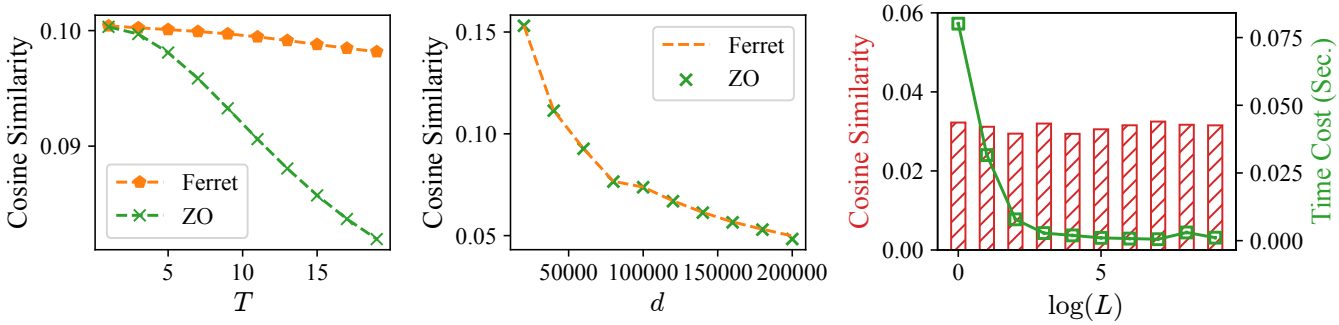


Figure 7: Reconstruction accuracy of our (8) vs. zeroth-order method under varying K and ϵ.



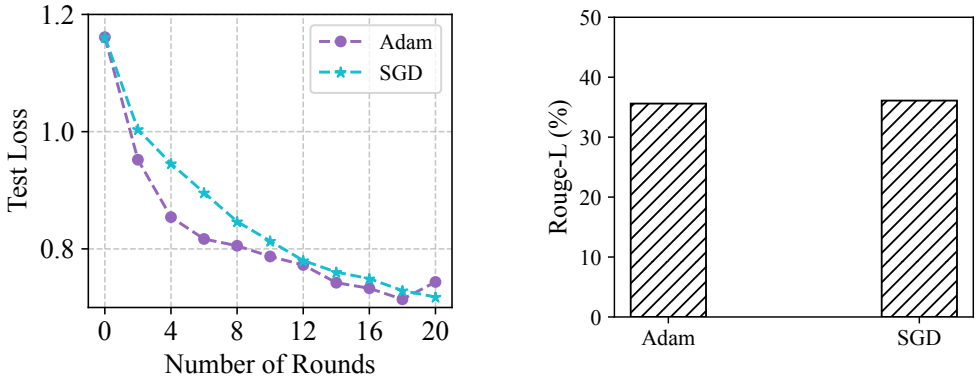
(a) Varying T (b) Varying d (c) Varying L

Figure 8: Reconstruction Accuracy (measured by cosine similarity between reconstruction and ground truth) of our (8) vs. zeroth-order method under varying T, d, and L.

Reconstruction Accuracy of Ferret under Varying T. In Fig. 8 (a), we present the reconstruction accuracy (measured by cosine similarity) of a T-iteration gradient descent update for the function $F(\boldsymbol{x}) = \sum_{i=1}^{d} \sin^2(x_i)$ with a learning rate of 0.1, $d = 5 \times 10^4$, $L = 1$, and $K = 500$, using our method in (8) and the zeroth-order (ZO) method described in Thm. 3 with $\epsilon = 0.1$. The goal is to compare the accumulated error from our (8) with that of the ZO method. Interestingly, Fig. 8 (a) shows that our method maintains consistent reconstruction accuracy as the number T of gradient descent iterations increases, whereas the ZO method experiences a noticeable decline in accuracy. This result implies that our (8) effectively avoids the accumulated error typical in zeroth-order methods, aligning with the theoretical justification provided in Sec. 4.1.

Reconstruction Accuracy of Ferret under Varying d. In Fig. 8 (b), we show the reconstruction accuracy (measured by cosine similarity) of d-dimensional gradient of $F(\boldsymbol{x}) = \sum_{i=1}^{d} \sin^2(x_i)$ at a randomly sampled input $\boldsymbol{x}$, with $L = 1$ and $K = 500$, using our method in (8) and the zeroth-order (ZO) method described in Thm. 3 with $\epsilon = 0.1$. The goal is to compare the reconstruction accuracy rate with respect to the dimension d between our (8) method and the ZO method. Interestingly, Fig. 8 (b) shows that both methods achieve the same reconstruction accuracy rate with respect to d. More importantly, when d becomes large, the accuracy rate is approximately linear, which aligns with the theoretical insights provided in Thm. 2.

Reconstruction Accuracy of Ferret under Varying L. In Fig. 8 (c), we present the reconstruction accuracy (measured by cosine similarity) and computational complexity (measured by time cost) for the $d = 5.12 \times 10^5$-dimensional gradient of



(a) Convergence (b) Generalization

Figure 9: Convergence and generalization of Ferret under varying optimizers for local updates.

function $F(\boldsymbol{x}) = \sum_{i=1}^{d} \sin^2(x_i)$ at a randomly sampled input $\boldsymbol{x}$, under varying L of the same number of dimensions and $K = 512$, using our method in (8). The goal is to study the impact of block size L on our (8). Notably, Fig. 8 (c) shows that our block-wise reconstruction (8) significantly reduces computational complexity (in line with Prop. 1), while maintaining consistent reconstruction accuracy as L increases. These results further verify the efficacy of our block-wise reconstruction (8).

C.6. More Ablation Studies on Convergence and Generalization

Convergence and Generalization of Ferret under Varying Optimizers. In Fig. 9, we present the convergence and generalization of Ferret under different optimizers for its local updates, using the same experimental setup described in Appx. C.1. Notably, Fig. 9 demonstrates that Ferret achieves faster convergence with an improved optimizer (e.g., Adam vs. SGD) while maintaining comparable generalization performance. These findings further support the adaptability of Ferret, as discussed in Sec. 4.3.