Federated Full-Parameter Tuning of Billion-Sized Language Models with Communication Cost under 18 Kilobytes

Zhen Qin [1 *] **Daoyuan Chen** [2] **Bingchen Qian** [2] **Bolin Ding** [2] **Yaliang Li** [2] **Shuiguang Deng** [1]

Abstract

Pre-trained large language models (LLMs) need fine-tuning to improve their responsiveness to natural language instructions. Federated learning offers a way to fine-tune LLMs using the abundant data on end devices without compromising data privacy. Most existing federated fine-tuning methods for LLMs rely on parameter-efficient fine-tuning techniques, which may not reach the performance height possible with full-parameter tuning. However, federated full-parameter tuning of LLMs is a non-trivial problem due to the immense communication cost. This work introduces FedKSeed that employs zeroth-order optimization with a finite set of random seeds. It significantly reduces transmission requirements between the server and clients to just a few random seeds and scalar gradients, amounting to only a few thousand bytes, making federated full-parameter tuning of billion-sized LLMs possible on devices. Building on it, we develop a strategy enabling probability-differentiated seed sampling, prioritizing perturbations with greater impact on model accuracy. Experiments across six scenarios with various LLMs, datasets and data partitions demonstrate that our approach outperforms existing federated LLM fine-tuning methods in both communication efficiency and zero-shot generalization.

Table 1. Comparing federated tuning methods w.r.t. accuracy and client-side costs, with *computation cost* referring to that incurred by obtaining the latest model, d as the model parameter count, ν as the ratio of trainable parameters in PEFT versus full-parameter tuning, τ as the average number of local steps performed by each client per round, r as the number of communication rounds, and m as the number of active clients in each round. M_{infer}, M_{peft} and M_{full} are peak memory usage for inference, PEFT with BP, and full-parameter tuning with BP, respectively. For simplicity, we denote $\xi = M_{\text{peft}}/M_{\text{infer}}$ and $\Xi = M_{\text{full}}/M_{\text{infer}}$. Generally, $\nu \ll 1 < \xi < \Xi \ll \tau r m$, and d is in billions for LLMs. FedKSeed delivers top-tier performance across these aspects simultaneously.

Approach	Acc.↑	Commu.↓	Mem.↓	Comput.↓
PEFT with BP	$\star$	$\mathcal{O}(\nu d)$	$\mathcal{O}(\xi d)$	$\mathcal{O}(d)$
Full-param. with BP	$\star\star$	$\mathcal{O}(d)$	$\mathcal{O}(\Xi d)$	$\mathcal{O}(d)$
Full-param. with ZOO	$\star\star$	$\mathcal{O}(d)$	$\mathcal{O}(d)$	$\mathcal{O}(d)$
infinite seed-pool in uplink	$\star\star$	$\mathcal{O}(d)$	$\mathcal{O}(d)$	$\mathcal{O}(d)$
infinite seed-pool in bi-link	$\star\star$	$\mathcal{O}(1)$	$\mathcal{O}(d)$	$\mathcal{O}(\tau r m d)$
FedKSeed (|seed-pool|=K)	$\star\star$	$\mathcal{O}(1)$	$\mathcal{O}(d)$	$\mathcal{O}(d)$

1. Introduction

Large language models (LLMs) exhibit outstanding performance on various natural language tasks yet require fine-tuning to enhance their task responsiveness (Chen et al., 2023a; Dong et al., 2023). While existing open datasets contribute to LLM tuning (Wang et al., 2022; Wei et al., 2022), the vast quantities of private data continuously generated at end devices present an untapped opportunity for further exploitation, especially as the reservoir of high-quality language data may become depleted in the future (Villalobos et al., 2022). Federated learning (FL) (McMahan et al., 2017; Kairouz et al., 2021) offers a privacy-protected way to collaboratively tune LLMs with distributed data, which has been explored by recent parameter-efficient fine-tuning (PEFT) based works (Zhang et al., 2024; Babakniya et al., 2023; Zhang et al., 2023; Che et al., 2023). Nonetheless, PEFT is not a universal solution for LLM tuning, as it may not consistently match the accuracy of full-parameter tuning (Chen et al., 2022; Pu et al., 2023; Sun et al., 2023), particularly in FL scenarios where the statistically heterogeneous client data diminish the effectiveness of PEFT (Babakniya et al., 2023; Zhang et al., 2023). Considering full-parameter tuning's potential for higher accuracy, exploring its feasibility to LLMs with FL is promising.

However, full-parameter tuning of billion-sized LLMs with FL on devices is impractical with current technology, as

[*] Work done as an intern at Alibaba Group. [1]College of Computer Science and Technology, Zhejiang University, Hangzhou, China [2]Alibaba Group. Correspondence to: Yaliang Li <yaliang.li@alibaba-inc.com>, Shuiguang Deng <dengsg@zju.edu.cn>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

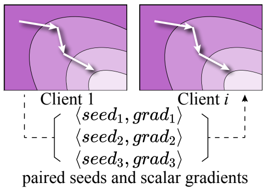

Client 1
Client i
$\langle seed_1, grad_1 \rangle$
$\langle seed_2, grad_2 \rangle$
$\langle seed_3, grad_3 \rangle$
paired seeds and scalar gradients

Figure 1. Each step of ZOO can be replicated by 1) a random seed that is used to generate a perturbation, and 2) a scalar gradient on it.

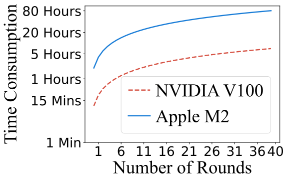


Figure 2. With more total steps, the time required to compute the latest global model by update replication grows rapidly (calculated with LLaMA-3B).

backpropagation (BP) and most BP-free methods, such as zeroth-order optimization (ZOO) (Fang et al., 2022), incur communication costs that scale with model size, as shown in Table 1. These costs become prohibitive for billion-sized LLMs. In particular, BP-based approaches also require significant memory that is not feasible for most end devices, e.g., tuning a full LLM with 1.3 billion parameters may consume over 20GB of memory (Malladi et al., 2023).

We note an interesting property in ZOO: a parameter update step of ZOO can be replicated with just two values: a seed (with an identical random number generator) and the corresponding scalar gradient (the product of the scalar gradient and the perturbation yields the vector gradient). Some recent ZOO-based FL methods (Zelikman et al., 2023; Feng et al., 2023; Maritan et al., 2023) explore this property to reduce communication cost as shown in Figure 1, however, they compromise other performance factors, making them still impractical for LLMs. As outlined in Table 1, current methods either (1) optimize the uplink communication for clients but neglect the significant downlink cost to distribute the latest global model in each round (Feng et al., 2023; Maritan et al., 2023), or (2) optimize bi-link communication but require each client to replicate all update steps from the others to synchronize the latest global model, leading to a computation cost that increases indefinitely with the number of rounds (Zelikman et al., 2023), as shown in Figure 2.

To achieve the best of both worlds, i.e., avoiding the massive *communication cost* associated with *transmitting full model parameters* while limiting the ever-increasing *computation cost* of *syncing* to the *latest global model*, this work introduces a novel federated full-parameter tuning approach for LLMs, based on ZOO with only K random seeds (denoted as FedKSeed). It employs a theoretically informed paradigm of seed reuse, implementing federated tuning with a finite set of seeds to generate perturbations, thus enabling full-parameter tuning of LLMs in FL with a communication cost of less than 18 kilobytes per round, and the memory footprint equivalent to inference requirements. Building on FedKSeed, we introduce a strategy to assess the significance of perturbations, assigning varied sampling probabilities to candidate seeds. It narrows the seed pool to expedite the syncing to the latest global model, thereby further enhancing both computational efficiency and model accuracy.

Our main contributions are summarized as follows:

- We propose a novel federated full-parameter tuning approach for LLM based on ZOO, FedKSeed, which transmits only K seeds and corresponding scalar gradients between the server and clients. To the best of our knowledge, this is the first work to make full-parameter tuning of billion-sized LLMs feasible on federated devices, with a communication cost of less than 18 kilobytes per round.
- We investigate the differentiated importance of ZOO perturbations, and propose a simple yet effective strategy that samples seeds with non-uniform probabilities. It improves accuracy while reducing the cardinality of candidate seeds needed by FedKSeed, thereby accelerating the client-side synchronization with the latest global model.
- Experiments on 6 scenarios with various LLMs, datasets and data partitions show that FedKSeed with the proposed non-uniform seed sampling attains an average relative improvement of 7.26% in Rouge-L over the best-performing practical baseline on held-out tasks and reduces communication costs by a factor of around a thousand. Our codes are publicly available at `https://github.com/alibaba/FederatedScope/tree/FedKSeed`.

2. Related Work

Federated Fine-Tuning for LLMs. There are some studies exploring fine-tuning LLMs with FL based on PEFT techniques, e.g., Zhang et al. (2023) provide benchmarks for PEFT techniques in FL. Among existing PEFT techniques, LoRA (Hu et al., 2022) is usually preferable. Zhang et al. (2024) proposes a federated instruction tuning approach based on LoRA. Jiang et al. (2023) design a low-parameter FL approach based on LoRA for text classification. Babakniya et al. (2023) experimentally demonstrate that when facing FL with non-IID data, LoRA is not as good as full-parameter tuning and propose a strategic initialization of LoRA weights based on SVD decomposition of full parameters fine-tuned with BP. There are also some works contributing to the deployment of LLM tuning with FL, e.g., FederatedScope-LLM (Kuang et al., 2023) and FATE-LLM (Fan et al., 2023). The computational bottlenecks have been thoroughly investigated by Woisetschläger et al. (2023).

Federated Learning with Zeroth-Order Optimization. There are some researches using ZOO for non-differentiable problems (Li & Chen, 2021). Shu et al. (2023) boost the query efficiency of ZOO in FL by optimization trajectory. Some researches analyze the convergence and generalization of ZOO-based FL (Fang et al., 2022; Chen et al., 2023d).

However, these approaches are only validated for small models with no more than 10 million parameters.

There are also some works leveraging random seeds to optimize communication efficiency. However, they are not suitable for full-parameter tuning of LLMs with FL due to (1) distributing the latest model parameters in each round (Xu et al., 2023; Maritan et al., 2023; Feng et al., 2023) that hinders the important download efficiency of clients (Dorfman et al., 2023), or (2) tremendous computation overhead for calculating the latest model (Zelikman et al., 2023) as in Figure 2, or (3) the reliance on BP which consumes a substantial amount of memory (Rahimi et al., 2024).

Difference from Related Works. A recent work FwdLLM (Xu et al., 2023) conducts FL based on PEFT and ZOO, but with the goal and techniques different from FedKSeed. FwdLLM uses quantization and PEFT to reduce memory cost, while we mainly focus on communication cost and enable full-parameter tuning of LLMs with FL. FedKSeed is orthogonal to quantization techniques (Xi et al., 2023; Dettmers et al., 2023). FwdLLM requires a total of several hundred GB of communication cost to tune an LLM with only about 300 million parameters, as it only optimizes client-side uplink. Besides, although some BP-based methods optimize the communication (Rahimi et al., 2024), they are not tailored and are not suitable for LLM tuning on end devices due to the tremendous memory footprint.

In a nutshell, existing works mainly focus on tuning partial LLM parameters, while our method enables full-parameter tuning of LLMs with FL, obtaining higher accuracy (Table 2). FedKSeed significantly cuts the communication and memory costs by eliminating model parameter transmission and BP, outperforming existing approaches tailored for federated LLM tuning (Table 3). Further technical comparisons between FedKSeed and existing works are in Appendix A.

3. Problem Formulation

Consider an FL system with N clients, each with a private dataset $\mathcal{D}_i$, federated fine-tuning aims at collaboratively tuning model $\mathbf{w} \in \mathbb{R}^d$ with the pre-trained weight $\mathbf{w}^0 \in \mathbb{R}^d$ at initialization, which can be formulated as

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) \triangleq \sum_{i=1}^{N} c_i \cdot \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_i}\left[\mathcal{L}(\mathbf{w};\mathbf{x})\right], \tag{1}$$

where $\mathcal{L}(\mathbf{w};\mathbf{x})$ is the loss evaluated at model $\mathbf{w}$ on a data instance $\mathbf{x}$ drawn from $\mathcal{D}_i$ and $c_i \geq 0$ is the aggregate weight with $\sum_{i=1}^{N} c_i = 1$. Here we utilize $\mathbf{x}$ since we set the batch size to 1 to lower memory cost as Malladi et al. (2023). The fundamental distinction between federated fine-tuning and vanilla FL (McMahan et al., 2017) is that it begins optimization from a pretrained weight $\mathbf{w}^0$ rather than from scratch. Equation (1) is solved in several rounds of local training and aggregation. In round r of BP-based FL (McMahan et al., 2017), each client i performs several steps of gradient descent algorithms on its local model $\mathbf{w}_i^r$ initialized by weight $\mathbf{w}^r$ downloaded from the server, as

$$\mathbf{w}_{i,t+1}^r = \mathbf{w}_{i,t}^r - \eta \cdot \mathbf{g}_{i,t}^r \tag{2}$$

where $\mathbf{w}_{i,t}^r$ is the local model of client i at local step t, η is the learning rate, and $\mathbf{g}_{i,t}^r$ represents the gradient computed as$\nabla_{\mathbf{w}_{i,t}^r}\mathcal{L}_i(\mathbf{w}_{i,t}^r;\mathbf{x}), \forall\mathbf{x} \in \mathcal{D}_i$. After local training, the server aggregates all received $\mathbf{w}_i^r$ for subsequent rounds.

The main difference between ZOO-based FL and BP-based FL lies in the obtaining of the gradient. ZOO-based FL estimates the gradient by forward propagations instead of direct calculation. Our work uses the ZOO with a two-point gradient estimator proposed by Malladi et al. (2023), as

$$\widehat{\mathbf{g}}_{i,t}^r \triangleq \frac{\mathcal{L}(\mathbf{w}_{i,t}^r+\epsilon\mathbf{z};\mathbf{x})-\mathcal{L}(\mathbf{w}_{i,t}^r-\epsilon\mathbf{z};\mathbf{x})}{2\epsilon}\mathbf{z} \approx \mathbf{z}\mathbf{z}^\top\mathbf{g}_{i,t}^r, \tag{3}$$

where $\widehat{\mathbf{g}}_{i,t}$ is the estimated gradient, $\mathbf{z} \in \mathbb{R}^d$ is a random perturbation that follows $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I}_d)$ and ϵ is the scale of perturbations. When $\widehat{\mathbf{g}}_{i,t}^r$ is estimated, client i updates its local model as Equation (2). For symbol convenience, we denote $\widehat{\mathbf{g}} = \widehat{\varrho} \cdot \mathbf{z}$ with omitted scripts, where $\widehat{\varrho} \triangleq \frac{\mathcal{L}(\mathbf{w}+\epsilon\mathbf{z};\mathbf{x})-\mathcal{L}(\mathbf{w}-\epsilon\mathbf{z};\mathbf{x})}{2\epsilon}$ is termed as ***scalar gradient***.

4. The proposed FedKSeed

4.1. Overview

FedKSeed is designed with the following goals: (1) to avoid the massive communication cost for transmitting full-model parameters, and (2) to avoid the tremendous memory footprint caused by BP. We design FedKSeed based on ZOO, and propose a theoretically-informed paradigm that enables seed reuse to limit the ever-increasing computational cost of clients to catch up to the latest global model.

Figure 3 outlines FedKSeed, where the server maintains K unique candidate seeds $\mathbb{S} \in \mathbb{Z}^K$ and a scalar gradient accumulator $\mathcal{A} \in \mathbb{R}^K$ recording the sum of received scalar gradients grouped by corresponding candidate seeds. Note that the server holds no model parameters, and each client possesses a pretrained LLM $\mathbf{w}^0$. The processes are as follows: ① At the start of each round, the server sends $\mathbb{S}$ and $\mathcal{A}$ to active clients. ② Each client i calculates the latest global model as its local model $\mathbf{w}_i$ based on $\mathcal{A}$. ③ A loop of local training, where in each step, the client samples a seed s_j from $\mathbb{S}$ and a data instance, then computes the scalar gradient $\widehat{\varrho}_j$. Next, $\mathbf{w}_i$ is updated based on $\widehat{\varrho}_j$ and s_j, and $(s_j, \widehat{\varrho}_j)$ is staged to the scalar gradient history $\mathbb{H}_i$. ④ Each client sends $\mathbb{H}_i$ to the server after several steps of local training. ⑤ The server updates $\mathcal{A}$ based on all received scalar gradient histories. We summarize the processes in Algorithm 1 in Appendix C and detail them in the subsequent sections.

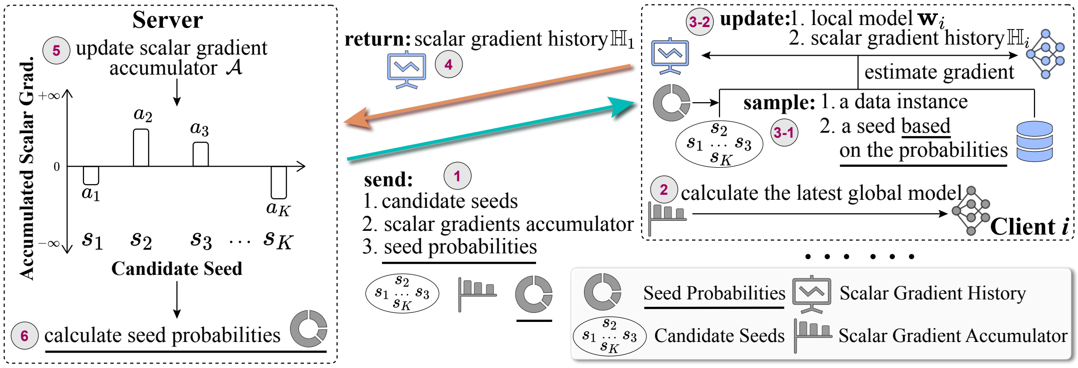

Server
5 update scalar gradient accumulator $\mathcal{A}$
Accumulated Scalar Grad.
$+\infty$
0
$-\infty$
a_1
a_2
a_3
a_K
s_1 s_2 s_3 $\cdots$ s_K
Candidate Seed
6 calculate seed probabilities
return: scalar gradient history $\mathbb{H}_1$
4
send: 1
1. candidate seeds
2. scalar gradients accumulator
3. seed probabilities
3-2 update: 1. local model $\mathbf{w}_i$
2. scalar gradient history $\mathbb{H}_i$
estimate gradient
sample: 1. a data instance
2. a seed based on the probabilities
3-1
2 calculate the latest global model
Client i
Seed Probabilities
Scalar Gradient History
Candidate Seeds
Scalar Gradient Accumulator

Figure 3. Overview of FedKSeed, where the serial numbers indicate processes in each round. Gray components share identical values among all clients. The underlined components are only required by an enhanced version of it, i.e., FedKSeed-Pro (Section 4.3).

4.2. Federated Full-Parameter Tuning by Limited Seeds

Recall Equation (3), if clients use the same pseudo number generator, a perturbation can be encoded by a random seed (one integer), so t steps of update can be replicated with a scalar gradient history $\mathbb{H} = \{(s_j, \widehat{\varrho}_j)\}^t$ containing t pairs of seeds and scalar gradients. Therefore, an intuitive solution to alleviate model transmission is to have the server track the scalar gradients of all clients (Zelikman et al., 2023). Assuming m clients participate in FL in each round, and each one conducts average τ steps of local training. After r rounds, a client has to perform τrm steps of model updating to get the latest global model from initial weight $\mathbf{w}^0$. From Figure 2, when $m = 50$, $\tau = 200$, and $r = 30$, this operation requires over 60 hours with an Apple M2, and over 5 hours even with an NVIDIA V100. Besides, just 1000 steps of updating consume 8% of a MacBook M2's battery (battery health at 92%). Thus, it is very important to reduce the computational overhead of computing the latest model.

Restrict the Cardinality of Candidate Seeds: from Infinite to K. If seeds are reused, the update steps needed to get the latest model can be merged. If we select only K candidate seeds and accumulate the scalar gradients corresponding to the same seed, *each client only needs to perform at most K iterations to get the latest global model*, unlike the solutions with infinite seeds (as outlined in Table 1).

At the start, the server randomly samples K unique candidate seeds $\mathbb{S} = \{s_1, s_2, \ldots, s_K\}$, and initializes a scalar gradient accumulator $\mathcal{A} = \{a_1, \ldots, a_K\} \in \mathbb{R}^K$, where $a_j = \sum_{\widehat{\varrho} \in \mathcal{G}_j} \widehat{\varrho}$, and $\mathcal{G}_j$ collects all scalar gradients $\widehat{\varrho}_j$ in $\mathbb{H}$ on the perturbation of s_j. Each client downloads $\mathcal{A}$ and gets the latest global model as its local model $\mathbf{w}_i$ by performing

$$\mathbf{w}_i = \mathbf{w}^0 - \eta \cdot \sum_{j=1}^{K} a_j \cdot \mathbf{z}_j. \tag{4}$$

Then, the latest global model $\mathbf{w}$ is treated as the local model $\mathbf{w}_i$. During each step of local training, the client samples a data instance $\mathbf{x}$ and a seed $s_j \in \mathbb{S}$, and calculates $\widehat{\varrho}_j$ as

$$\widehat{\varrho}_j = \frac{\mathcal{L}(\mathbf{w}_i + \epsilon \mathbf{z}_j; \mathbf{x}) - \mathcal{L}(\mathbf{w}_i - \epsilon \mathbf{z}_j; \mathbf{x})}{2\epsilon}. \tag{5}$$

Then, the local model $\mathbf{w}_i$ is updated as

$$\mathbf{w}_i \leftarrow \mathbf{w}_i - \eta \widehat{\varrho}_j \cdot \mathbf{z}_j, \tag{6}$$

and s_j and $\widehat{\varrho}_j$ are tracked in $\mathbb{H}_i = \{(s_j, \widehat{\varrho}_j), \ldots\}$. After τ steps of local training, $\mathbb{H}_i$ is sent to the server. Then, for each $(s_j, \widehat{\varrho}_j) \in \mathbb{H}_i$, the server conducts

$$a_j = a_j + c_i \cdot \widehat{\varrho}_j, \tag{7}$$

to aggregate the gradient history of client i into $\mathcal{A}$.

Considering that K is a predefined constant, this paradigm shift reduces the computation complexity of obtaining the latest global model to $\mathcal{O}(d)$, which remains constant throughout the progression of FL. Note that the server, just like any client, can obtain the latest global model using Equation (4). We refer to the above approach as FedKSeed.

4.2.1. Theoretical Support for Seed Reuse

The effectiveness of seed reuse lies in the minimal impact on convergence when the size of the seed pool (K) is not too small, and the total number of update steps is relatively large. To support this point, we analyze *the convergence similarity between FedKSeed and Federated Zeroth-order Optimization (FedZO)* (Fang et al., 2022), an FL approach based on ZOO without limited range on seed sampling, whose convergence has been proved.

Definition 1. *(Gradient estimation of FedZO). Given* $\mathbf{z}$ *as i.i.d. random perturbation with* $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ *distribution, FedZO estimates gradients in a mini-batch manner as*

$$\widehat{\mathbf{g}}^r_{i,t} = \frac{1}{b_1 b_2} \sum_{b=1}^{b_1} \sum_{j=1}^{b_2} \frac{\left[\mathcal{L}(\mathbf{w}^r_{i,t} + \epsilon \mathbf{z}_j; \mathbf{x}_b) - \mathcal{L}(\mathbf{w}^r_{i,t}; \mathbf{x}_b)\right]}{\epsilon} \mathbf{z}_j. \tag{8}$$

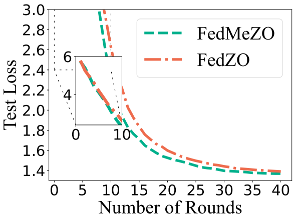


Figure 4. Full-parameter tuning convergence of LLaMA-3B on Natural Instructions by FedZO (b_1=b_2=1) and FedMeZO.

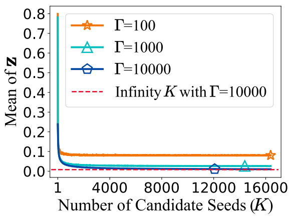


Figure 5. The mean (absolute value) of the Γ perturbations randomly sampled with the K candidate random seeds.

Theorem 1. *(Convergence of FedZO.) With the assumptions made by Fang et al. (2022) on (2) loss boundary, (3) L-smoothness of objective and loss functions, (4) the second-order gradient moment boundary and (5) local-global gradient dissimilarity boundary, FedZO satisfies*

$$\min_{r\in\{1,\dots,T\}} \mathbb{E}\|\nabla f(\mathbf{w}^r)\|^2 \le \mathcal{O}\left(\sqrt{\frac{d}{\tau m T b_1 b_2}}+\sqrt{\frac{b_1 b_2 \tau}{dmT}}\right), \tag{9}$$

where τ is the average number of local iterations within one round for each client, and T is the number of total rounds.

Assumptions 2-5 are detailed in Appendix D. Theorem 1 has been proved by Fang et al. (2022). To compare the convergence of FedKSeed and FedZO, we introduce FedMeZO by replacing BP in FedAvg with MeZO (Malladi et al., 2023), which utilizes the two-point estimator in Equation (3).

Assumption 1. *FedMeZO converges similarly or faster compared to FedZO with $b_1 = b_2 = 1$.*

The gradient estimated by the two-point estimator resembles the mean of two gradients by the one-point estimator (Equation (8)) with two opposing perturbations, as

$$\begin{aligned}\mathbf{z}[\mathcal{L}(\mathbf{w}+\epsilon\mathbf{z};\mathbf{x})-\mathcal{L}(\mathbf{w};\mathbf{x})]+(-\mathbf{z})[\mathcal{L}(\mathbf{w}-\epsilon\mathbf{z};\mathbf{x})-\mathcal{L}(\mathbf{w};\mathbf{x})]\\ =\mathbf{z}\left[\mathcal{L}(\mathbf{w}+\epsilon\mathbf{z};\mathbf{x})-\mathcal{L}(\mathbf{w}-\epsilon\mathbf{z};\mathbf{x})\right].\end{aligned}$$

Besides, the one-point estimator generally suffers from a higher estimation variance than the two-point estimator (Liu et al., 2018). Thus, FedMeZO should exhibit a convergence trend not inferior to FedZO with $b_1 = b_2 = 1$. This is also experimentally demonstrated by their test loss on Natural Instructions (Wang et al., 2022) in Figure 4 (settings aligned with Section 5.1). Note that a recent work also proves the convergence of FL based on MeZO (Ling et al., 2024).

Impact by seed restriction. Compared to FedMeZO, FedKSeed imposes additional restrictions on seed sampling. Perturbation sampling in FedKSeed can be formalized in two stages: (1) Randomly and uniformly sampling K distinct random seeds $\{s_1, s_2, \dots, s_K\}$ from an infinity space, thus obtaining K perturbations $\mathbb{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_K\}$, where $\mathbf{z}_j \in \mathbb{R}^d$ is sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. (2) Across the T rounds of FL, randomly and uniformly sampling a total of $\Gamma = m \cdot \tau \cdot T$ perturbations from $\mathbb{Z}$ with replacement. From the formal separation, the Γ perturbations are still i.i.d.

Lemma 1. *(Mean of Perturbations). Given $\mathcal{Z}' = \{Z'_1, Z'_2, \dots, Z'_K\}$ randomly sampled from $\mathcal{N}(0,1)$, Γ variables $Z_1, Z_2, \dots, Z_\Gamma$ randomly and uniformly sampled from $\mathcal{Z}'$ with replacement are i.i.d, and $\bar{S}_\Gamma = \frac{1}{\Gamma}\sum_{i=1}^{\Gamma} Z_i$ satisfy*

$$\Pr[|\bar{S}_\Gamma - 0| \ge \varepsilon] \le \frac{4}{K\varepsilon^2} + 2\exp\left(-\frac{\Gamma\varepsilon^2}{2[\max(\mathcal{Z}')-\min(\mathcal{Z}')]^2}\right). \tag{10}$$

Proof. The proof is provided in Appendix E.1.

From Lemma 1, there exists a gap ε between zero and the mean of $\mathbf{z}$ based on these seeds, which decreases with the increase of K and Γ. In our experiments, K should be in the thousands, and Γ increases by ten thousand per round on Natural Instructions. So both of them are usually large values. Next, we analyze the impact of this mean shift (ε).

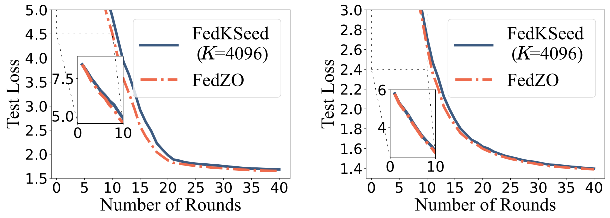
Theorem 2. *(Gradient Estimation Error). With assumption on L-smoothness of $\mathcal{L}$ (Assumption 3), for $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, the gradient $\widehat{\mathbf{g}}_1$ estimated by the two-point estimator with seed restrictions and $\widehat{\mathbf{g}}_0$ estimated by the two-point estimator without seed restrictions satisfy*

$$\|\widehat{\mathbf{g}}_1 - \widehat{\mathbf{g}}_0\| \le 2L\,\|\mathbf{z}\|\,\|\overrightarrow{\varepsilon}\| + L\,\|\overrightarrow{\varepsilon}\|^2 . \tag{11}$$

Proof. The proof is provided in Appendix E.2.

Intuitively, the estimation error $\widehat{\mathbf{g}}_1 - \widehat{\mathbf{g}}_0$ has a slight impact, because: 1) it is symmetrical in terms of the direction with the two-point gradient estimator, and 2) it is bounded by ε, which is usually very small. We demonstrate the estimation error with a toy example for a more intuitive understanding.

Empirical Observation. We generate K seeds, then sample Γ perturbations and calculate the mean (absolute value) of them. Note that the above toy example is performed with $\mathbf{z} \in \mathbb{R}^{10000}$, with K and Γ ranged in $[1, 16384]$ and $\{100, 1000, 10000\}$, respectively. With each combination of (K, Γ), we perform the example 100 times and record the average results. To have a comparison with the situation without any restriction on seed sampling, we present a dotted line to indicate the situation with $\Gamma = 10000$ and no restriction on K. The results are presented in Figure 5, and we can find that: The mean of $\mathbf{z}$ converges to zero with the increase of K, and when K is over 1024 (the least adopted value in Section 5.1), it converges to a value very similar to that without seed restriction. When fixing K, larger Γ helps to lower ε. Note that in our experiments, there are $50 * 200$ steps of updates generated in each round, thus, Γ is



(a) DataJuicer-1.3B

(b) LLaMA-3B

Figure 6. Convergence of full-parameter tuning by FedKSeed and FedZO ($b_1 = b_2 = 1$) on Natural Instructions.

significantly larger than this toy example, bringing a much smaller ε than that in Figure 5. Thus, the impact of seed restrictions is greatly limited in our settings.

Considering both: (1) the non-inferiority of the two-point estimator to the one-point estimator (Assumption 1), and (2) the minimal nature of the gradient estimation error (Theorem 2), we can conclude that *FedKSeed does not differ significantly from FedZO on convergence*. We experimentally demonstrate it by presenting the test loss on Natural Instructions (Wang et al., 2022) dataset in Figure 6 (settings aligned with Section 5.1), showing that FedKSeed and FedZO ($b_1 = b_2 = 1$) share similar convergence trends.

4.2.2. Selection of K

The selection of K can be guided by the intrinsic dimension theory (Li et al., 2018; Aghajanyan et al., 2021). Given $\mathbb{G} = \left[\sum_{\widehat{\varrho}\in\mathcal{G}_1} \widehat{\varrho}, \ldots, \sum_{\widehat{\varrho}\in\mathcal{G}_K} \widehat{\varrho}\right]^\top$, in which the j-th element is the summation of all scalar gradients corresponding to the perturbation with s_j. Equation (1) can be transformed to

$$\min_{\mathbb{G}\in\mathbb{R}^K} \sum_{i=1}^{N} c_i \cdot \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_i} \left[\mathcal{L}(\underbrace{\mathbf{w}^0 + \left[\mathbf{z}_1, \ldots, \mathbf{z}_K\right] \mathbb{G}}_{\text{The difference from Equation (1)}} ; \mathbf{x}) \right]. \quad (12)$$

Thus, FedKSeed actually performs federated tuning in a K-dimensional random subspace. Equation (12) matches the form that trains a neural network in a subspace with the dimensionality equivalent to the intrinsic dimension (Li et al., 2018), with $\left[\mathbf{z}_1, \ldots, \mathbf{z}_K\right]$ as the randomly generated projection matrix. Both $\mathbf{w}^0$ and $\left[\mathbf{z}_1, \ldots, \mathbf{z}_K\right]$ are frozen during training. Thus, we can determine K in the vicinity of the LLM's intrinsic dimension, which may approximately fall between 10^3 and 10^4 following Aghajanyan et al. (2021).

Principle 1. *(Seed Insufficiency.) There exists a threshold $\overleftarrow{K}$ such that when $K \leq \overleftarrow{K}$, the accuracy of the model decreases with the reduction of K.*

According to Li et al. (2018), when K is less than the codimension of the solution, solutions will almost not be found in the subspace, or the founded solution is low-quality, since low-dimensional subspaces do not possess sufficient complexity to embed the solution manifold. We also provide additional theoretical support for Principle 1 from the perspective of solving optimization problems in Appendix F.1. From the analyses, $\overleftarrow{K}$ should exist, and a value approximates the intrinsic dimension typically serves as a good empirical estimate of it, such as 1024, as shown in Figure 7.

Principle 2. *(Seed Excessiveness.) There exists a threshold $\overrightarrow{K}$ such that given the total number of local training steps fixed, when $K \geq \overrightarrow{K}$, there is no upward trend in the accuracy of the model with the increase of K.*

When K surpasses the intrinsic dimension, the marginal gain in accuracy becomes increasingly smaller with the increase of K, since further increasing K does not increase the ability to approximate the solution manifold. The redundancy of K affects FedKSeed similar to that reported by Li et al. (2018); Aghajanyan et al. (2021) but with slight differences. In FedKSeed, only one element of $\mathbb{G}$ is optimized in each training step. Intuitively, each element of $\mathbb{G}$ requires several steps to be accurately estimated. Given the fixed total number of update steps $\tau r m$, each element of $\mathbb{G}$ consumes $\frac{\tau r m}{K}$ steps averagely. Thus, increasing K reduces the number of training steps for each element of $\mathbb{G}$. When K has reached an ample magnitude, this increment may induce adverse effects, as shown in Figure 7. We provide additional theoretical support to Principle 2 from the perspective of batch size in Appendix F.2, and experimental support from the marginal gain on accuracy in terms of seed quantity in Appendix H.1. From these analyses, $\overrightarrow{K}$ should exist thus Principle 2 holds. From Figure 7, an empirically estimated value of it is 4096, which lies around the intrinsic dimension and tends to be slightly larger than it.

It is hard to map a specific LLM to a precise value of K due to the complex architecture of LLMs. From the analyses, we can choose K in $[\overleftarrow{K}, \overrightarrow{K}]$, i.e., an integer slightly larger than the intrinsic dimension. Section 5.2 experimentally demonstrates that for models with 1B and 3B parameters, K can be several thousand so that FedKSeed performs well.

4.3. Sampling Seeds with Non-uniform Probabilities

This section enhances FedKSeed through enabling non-uniform **pro**babilities for seed sampling to further reduce K and boost the model accuracy, termed as FedKSeed-Pro.

The gradient $\mathbf{g}$ indicates the direction of the steepest descent for loss function $\mathcal{L}$ at a given point. However, in FedKSeed, $\mathbf{g}$ is not available due to the removal of BP. The scalar gradient $\widehat{\varrho}$ can be regarded as the estimated directional derivative of $\mathcal{L}$ along $\mathbf{z}$. The similarity between different directional vectors and the gradient varies, affecting the rate of change in the objective and thus contributing differently to the descent of the loss. The scalar gradient is determined by both the model, data instance and the similarity between true gradient $\mathbf{g}$ and $\mathbf{z}$. Given the model and data instances equivalent in expectation for all perturbations, the average amplitude of scalar gradient $\psi_j = \frac{1}{|\mathcal{G}_j|}\sum_{\widehat{\varrho}\in\mathcal{G}_j}|\widehat{\varrho}|$ can quantify the importance of $\mathbf{z}_j$. To avoid excessive probability differences in $\mathbb{S}$, we perform min-max normalization on $\{\psi_1,\dots,\psi_K\}$, where ψ_j^{norm} denotes the normalized amplitude. Then, we compute the probability p_j of each seed $s_j \in \mathbb{S}$ as

$$p_j = \frac{\exp(\psi_j^{\text{norm}})}{\sum_{k=1}^{K}\exp(\psi_k^{\text{norm}})}. \tag{13}$$

The probabilities $\{p_1,\dots,p_K\}$ are updated and sent to active clients in each round to guide the seed sampling of local training. In Section 5.2, we experimentally find that when significant seeds are sampled with higher probabilities, we can reduce the cardinality of seeds without sacrificing, and sometimes enhancing, model accuracy.

5. Experiments

5.1. Experimental Setup

Baselines. We choose 4 *practical methods* tailored for federated LLM tuning as the baselines, including: (1) FedPTuning (Kuang et al., 2023) with P-Tuning (Liu et al., 2023) as the PEFT technique, trained by SGD; (2) FedPrompt (Kuang et al., 2023) with Prompt Tuning (Lester et al., 2021) as the PEFT technique, trained by SGD; (3) FedIT: a federated instruction tuning approach proposed by Zhang et al. (2024), with LoRA as the PEFT technique and Adam (Kingma & Ba, 2015) as the optimizer; and (4) FedIT-SGD: a variation of FedIT that replaces Adam with SGD.

To clarify the impact of the restriction on seed sampling in FedKSeed, we introduce 3 full-parameter tuning approaches as references, including: (1) FedAvg (McMahan et al., 2017), (2) FedZO (Fang et al., 2022), and (3) FedMeZO (synthetic of FedAvg and MeZO (Malladi et al., 2023)). Note that these three approaches incur excessive communication overheads due to the transmission of full LLM parameters in each round, thus may not be practical in real world. Their inclusion is purely for reference purposes.

Datasets & Evaluation. We use Natural Instructions (NI) (Wang et al., 2022) and Dolly-15K (Conover et al., 2023) datasets, following task held-out setups. Each of the 738 training tasks in NI is assigned to a unique client for local training while the 119 test tasks in NI are used for global evaluation, building a non-IID scenario with feature distribution skew (Tan et al., 2022). The last task of Dolly-15K is used for global evaluation while the rest are split to 200 clients via Dirichlet distribution with α={0.5, 5.0} for local training, providing label distribution skew with varying degrees (Chen et al., 2023c). Rouge-L (Lin, 2004) is used as the evaluation metric following Dettmers et al. (2023), which correlates with the trend of accuracy on classification tasks (Wang et al., 2022). Considering the limited device resources, we take DataJuicer-1.3B (Chen et al., 2024) and LLaMA-3B (Touvron et al., 2023) as the foundation models.

Implementations. We randomly sample 5% of the clients to participate in FL in each round. The total number of communication rounds is set to 40 for NI and 60 for Dolly-15K. BP-based baselines conduct local training for one epoch, and FedKSeed and FedKSeed-Pro conduct local training for 200 steps. Unless stated otherwise, we set K to 4096 for FedKSeed, 1024 for FedKSeed-Pro with DataJuicer-1.3B, and 2048 for FedKSeed-Pro with LLaMA-3B. Note that from Figure 7, these settings are not tailored for best values of K in corresponding scenarios, ensuring fair comparisons. Please refer to Appendix G for more implementation details.

5.2. Comparisons on Accuracy Performance

Comparisons with practical baselines. From Table 2, FedKSeed and FedKSeed-Pro achieve the top two performances among the practical approaches across all six scenarios. In particular, on Dolly-15K ($\alpha = 0.5$) with LLaMA-3B, FedKSeed-Pro outperforms the best practical baseline, FedIT, by 3.06%. These improvements can be attributed to the benefits of full-parameter tuning, where the number of trainable parameters is significantly larger compared to PEFT techniques as Figure 14. Further, the gains achieved by FedKSeed-Pro over the best practical baseline, FedIT, are generally larger with LLaMA-3B than DataJuicer-1.3B, since with the same LoRA configuration, the model size increase does not proportionally affect the number of trainable parameters in FedIT as much as it does in our approaches.

Comparisons with full-parameter tuning approaches. From Table 2, FedAvg, FedZO and FedMeZO do not exhibit higher accuracy than FedKSeed and FedKSeed-Pro. However, from Table 3, FedKSeed and FedKSeed-Pro have significant advantages in communication and memory costs over FedAvg, and communication advantages over FedZO and FedMeZO. Thus, the seed sampling restriction method we proposed does not introduce noticeable negative effects.

Effect of *K*. To validate Principles 1 and 2, and to understand the relationship between the number of perturbation seeds (K) and the accuracy of FedKSeed and FedKSeed-Pro, we examine their performance with varying K, as depicted in Figure 7. We observe that when the K exceeds the recommended range specified in Section 5.1, the accuracy does not improve and may occasionally decline. Because the total number of optimization steps is constant, with more seeds, the likelihood that each seed consumes sufficient data to determine its step size is reduced. Conversely, with too few seeds, the performance of both FedKSeed and

Table 2. Rouge-L (%) comparisons. Each cell presents the average Rouge-L in the last round of four runs with different random seeds. Methods with excessive communication costs that may be impractical in real world are introduced just as references and marked in gray. **Bold** and underlined numbers are the **best** and second-best values among practical approaches (middle and bottom sections), respectively.

Approach	Natural Instructions		Dolly-15K ($\alpha = 0.5$)		Dolly-15K ($\alpha = 5.0$)	
	DataJuicer-1.3B	LLaMA-3B	DataJuicer-1.3B	LLaMA-3B	DataJuicer-1.3B	LLaMA-3B
FedAvg	22.08 ± 1.52	27.88 ± 0.75	32.30 ± 1.23	34.27 ± 0.45	33.38 ± 1.43	33.95 ± 0.79
FedZO	21.74 ± 1.91	29.46 ± 0.38	32.91 ± 0.67	36.34 ± 0.39	33.28 ± 0.42	36.72 ± 0.18
FedMeZO	21.71 ± 1.26	30.18 ± 0.69	33.32 ± 0.14	35.66 ± 1.06	33.07 ± 0.47	36.21 ± 0.15
FedPTuning	19.61 ± 2.71	25.41 ± 1.14	23.98 ± 3.23	30.30 ± 1.16	25.33 ± 2.48	29.08 ± 1.33
FedPrompt	6.04 ± 0.12	8.95 ± 2.47	32.73 ± 0.87	24.50 ± 4.78	32.51 ± 1.31	23.94 ± 4.15
FedIT-SGD	19.40 ± 1.83	28.14 ± 0.85	27.23 ± 0.68	29.28 ± 0.50	27.28 ± 1.35	29.19 ± 0.89
FedIT	22.30 ± 0.42	28.13 ± 0.50	30.80 ± 0.98	33.23 ± 1.51	30.97 ± 0.43	33.68 ± 1.07
FedKSeed	22.33 ± 1.72	29.77 ± 0.75	32.90 ± 0.37	35.64 ± 0.83	**33.12** ± **0.31**	35.93 ± 1.35
FedKSeed-Pro	**23.50** ± **1.35**	**30.19** ± **1.10**	**33.18** ± **0.68**	**36.29** ± **0.63**	33.00 ± 0.34	**35.95** ± **1.41**

Table 3. Per-round communication overhead and Peak GPU memory footprint of the approaches, where "B" denotes "Bytes".

Approach	DataJuicer-1.3B		LLaMA-3B	
	Commun.	Memory	Commun.	Memory
FedAvg	5.01 GB	17.8 GB	12.76 GB	39.1 GB
FedZO	5.01 GB	3.4 GB	12.76 GB	7.6 GB
FedMeZO	5.01 GB	3.5 GB	12.76 GB	7.8 GB
FedPTuning	96.36 MB	11.9 GB	234.9 MB	16.3 GB
FedPrompt	320.0 KB	11.8 GB	500.0 KB	19.0 GB
FedIT-SGD	12.00 MB	12.4 GB	20.31 MB	18.2 GB
FedIT	12.00 MB	12.4 GB	20.31 MB	18.3 GB
FedKSeed	17,988 B	3.5 GB	17,988 B	7.8 GB
FedKSeed-Pro	9,796 B	3.5 GB	17,988 B	7.8 GB

FedKSeed-Pro deteriorates due to the limited expressiveness resulting from an insufficient number of perturbations. Thus, the value of K should be balanced as discussed in Section 4.2.2: not too high to waste computational costs, nor too low to restrict the model's expressiveness. Our experimental results indicate that for models with 1B to 3B parameters, setting K in the range of [1024, 4096] is preferable.

Effect of seed probabilities. FedKSeed-Pro gains superior performance in five out of six scenarios and comparable results on Dolly-15K ($\alpha = 5.0$) with DataJuicer-1.3B compared to FedKSeed. This highlights the effectiveness of the non-uniform seed sampling proposed in Section 4.3. FedKSeed-Pro makes the probabilities of each seed being sampled differ by several multiples, as shown in Appendix H.3. Thus, by preferentially sampling seeds with greater importance (those with larger scalar gradient magnitudes), the accuracy of FedKSeed can be further enhanced.

5.3. Comparisons on Overheads

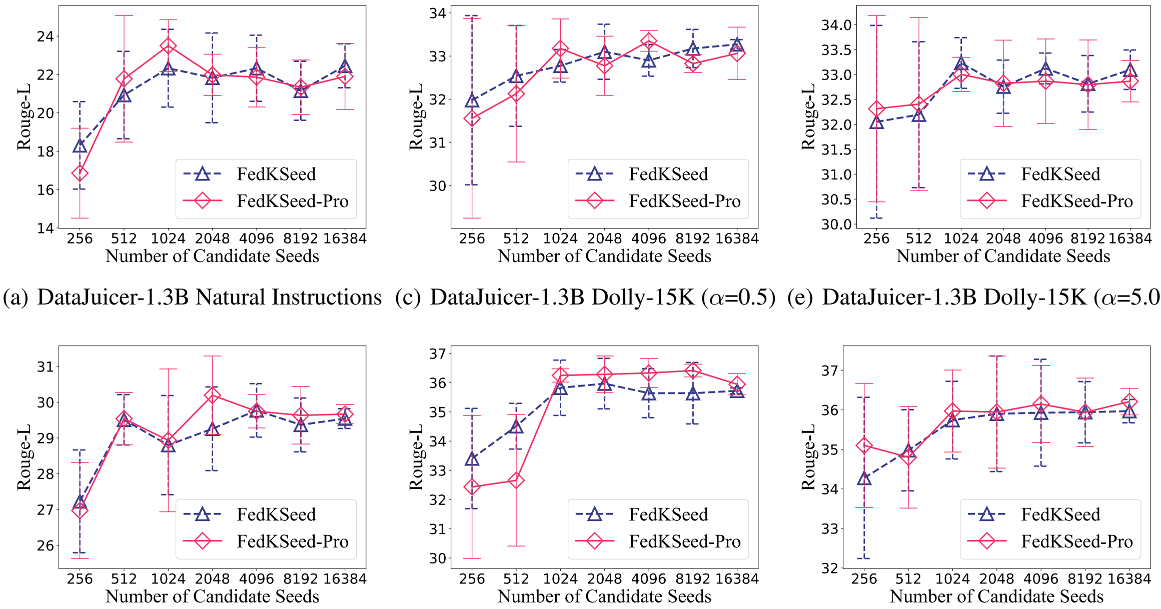
Table 3 shows that FedKSeed and FedKSeed-Pro incur the least communication and memory costs, where the K randomly selected seeds are encoded with one single seed that only occupies 4 Bytes. The time costs are illustrated in Appendix H.4. The calculation of communication costs is detailed in Appendix I. FedKSeed and FedKSeed-Pro enhance communication efficiency by removing the transmission of trainable parameters, and memory efficiency by omitting BP with the in-place ZOO (Malladi et al., 2023). Thus, they can be effectively applied to tune full LLMs on end devices with limited communication and memory budgets. Besides, FedKSeed-Pro requires fewer seeds to achieve the same accuracy compared to FedKSeed, as shown in Figure 7. It accelerates the synchronization to the latest model relative to FedKSeed, since computing the latest model takes longer with more candidate seeds, as shown in Figure 8.

5.4. Hyper-parameter Sensitivity

Given that ZOO serves as the training technique, we investigate the impact of key hyperparameters in ZOO, i.e., the learning rate (η) and the perturbation scale (ϵ), with FedKSeed-Pro for LLaMA-3B on Dolly-15K ($\alpha = 0.5$) as a case study. From Figure 9, both η and ϵ should not be excessively large. Since ϵ determines the magnitude of perturbations applied during gradient estimation, a smaller ϵ leads to a more accurate gradient approximation. However, too small ϵ may result in numerical underflow when using half-precision floating-point numbers. An overly large value of η can cause too aggressive update steps, potentially causing significant deviation from the optimum or even diverge.

5.5. Comparisons in Various Federated Scenarios

We further evaluate the two strongest baselines and our approaches under different FL settings. Figure 10(a) shows that benefiting from more valuable data, both our approaches and the baselines gain better accuracy as N increases, although not always monotonically. It further confirms the importance of federated tuning since it can leverage a broader



(a) DataJuicer-1.3B Natural Instructions (c) DataJuicer-1.3B Dolly-15K (α=0.5) (e) DataJuicer-1.3B Dolly-15K (α=5.0)

(b) LLaMA-3B Natural Instructions (d) LLaMA-3B Dolly-15K (α=0.5) (f) LLaMA-3B Dolly-15K (α=5.0)

Figure 7. Performance on Rouge-L of FedKSeed and FedKSeed-Pro with different cardinality of candidate seeds.

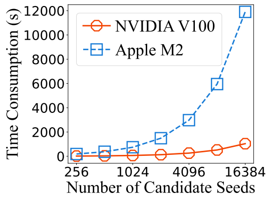


Figure 8. Time consumed to calculate the latest model (LLaMA-3B) by FedKSeed.

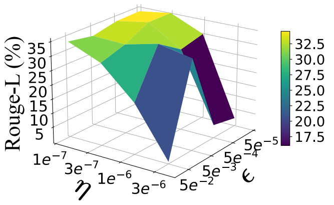
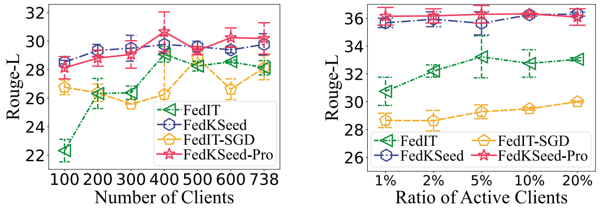


Figure 9. Rouge-L of FedKSeed-Pro on Dolly-15K ($\alpha = 0.5$) with different combinations of η and ϵ.



(a) Impact of total client count N (on Natural Instructions). (b) Impact of activate client ratio m/N (on Dolly-15K α=0.5).

Figure 10. Performances in various FL scenarios with LLaMA-3B. K=4096 in FedKSeed and 2048 in FedKSeed-Pro, as Section 5.1.

range of data sources. Figure 10(b) presents that the accuracy of these approaches increases and stabilizes as more clients participate in each round of FL. Furthermore, our approaches consistently outperform the baselines in various FL scenarios, further demonstrating their superiority.

6. Conclusion

Existing federated fine-tuning approaches for LLMs usually rely on PEFT techniques. Considering PEFT still falls short in FL scenarios compared to full-parameter tuning, we focus on enabling full-parameter tuning of billion-sized LLMs on devices with FL. To fulfill this, we design FedKSeed characterized by a theoretically-informed seed-reuse paradigm, where only a limited number of candidate seeds and corresponding scalar gradients need to be transmitted between the server and clients. It enables federated full-parameter tuning of LLMs with per-round communication costs lower than 18 kilobytes. Based on FedKSeed, inspired by the fact that the scalar gradient of a perturbation is the directional derivative of the true gradient, we propose a strategy to quantify the importance of seeds and grant differentiated sampling probabilities to them. It reduces the number of required seeds, thus speeding up the obtaining of the latest model while achieving higher accuracy than FedKSeed. Extensive experiments conducted on real-world datasets demonstrate our approaches surpass FL baselines tailored for LLM tuning on the accuracy of unseen tasks, communication cost and memory footprint at the same time.

Our work may raise some new potential research directions, such as decentralized federated fine-tuning since the communication cost is more critical in this context. More benefits brought by this work are discussed in Appendix J.

Acknowledgements

This work was supported in part by the National Science Foundation of China under Grants 62125206 and U20A20173 and in part by the Key Research Project of Zhejiang Province under Grant 2022C01145.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Aghajanyan, A., Gupta, S., and Zettlemoyer, L. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 7319–7328, 2021.

Babakniya, S., Elkordy, A. R., Ezzeldin, Y. H., Liu, Q., Song, K.-B., El-Khamy, M., and Avestimehr, S. SLoRA: Federated parameter efficient fine-tuning of language models. *arXiv preprint arXiv:2308.06522*, 2023.

Bai, J., Chen, D., Qian, B., Yao, L., and Li, Y. Federated fine-tuning of large language models under heterogeneous language tasks and client resources. *arXiv preprint arXiv:2402.11505*, 2024.

Borzunov, A., Ryabinin, M., Chumachenko, A., Baranchuk, D., Dettmers, T., Belkada, Y., Samygin, P., and Raffel, C. Distributed inference and fine-tuning of large language models over the internet. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Che, T., Liu, J., Zhou, Y., Ren, J., Zhou, J., Sheng, V., Dai, H., and Dou, D. Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7871–7888, 2023.

Chen, C., Feng, X., Zhou, J., Yin, J., and Zheng, X. Federated large language model: A position paper. *arXiv preprint arXiv:2307.08925*, 2023a.

Chen, D., Gao, D., Xie, Y., Pan, X., Li, Z., Li, Y., Ding, B., and Zhou, J. FS-REAL: Towards real-world cross-device federated learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3829–3841, 2023b.

Chen, D., Yao, L., Gao, D., Ding, B., and Li, Y. Efficient personalized federated learning via sparse model-adaptation. In *International Conference on Machine Learning, ICML*, volume 202, pp. 5234–5256, 2023c.

Chen, D., Huang, Y., Ma, Z., Chen, H., Pan, X., Ge, C., Gao, D., Xie, Y., Liu, Z., Gao, J., Li, Y., Ding, B., and Zhou, J. Data-juicer: A one-stop data processing system for large language models. In *International Conference on Management of Data*, 2024.

Chen, G., Liu, F., Meng, Z., and Liang, S. Revisiting parameter-efficient tuning: Are we really there yet? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 2612–2626, 2022.

Chen, J., Chen, H., Gu, B., and Deng, H. Fine-grained theoretical analysis of federated zeroth-order optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023d.

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. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, 2023.

Dong, X. L., Moon, S., Xu, Y. E., Malik, K., and Yu, Z. Towards next-generation intelligent assistants leveraging LLM techniques. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5792–5793, 2023.

Dorfman, R., Vargaftik, S., Ben-Itzhak, Y., and Levy, K. Y. DoCoFL: downlink compression for cross-device federated learning. In *International Conference on Machine Learning*, pp. 8356–8388. PMLR, 2023.

Fan, T., Kang, Y., Ma, G., Chen, W., Wei, W., Fan, L., and Yang, Q. FATE-LLM: A industrial grade federated learning framework for large language models. *CoRR*, abs/2310.10049, 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 Transactions on Signal Processing*, 70:5058–5073, 2022.

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

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 *The Tenth International Conference on Learning Representations, ICLR*, 2022.

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. Snapshot ensembles: Train 1, get M for free. In *International Conference on Learning Representations*, 2016.

Jiang, J., Liu, X., and Fan, C. Low-parameter federated learning with large language models. *arXiv preprint arXiv:2307.13896*, 2023.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K. A., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1-2):1–210, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.

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 preprint arXiv:2309.00363*, 2023.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 3045–3059, 2021.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations, ICLR*, 2018.

Li, Q., Diao, Y., Chen, Q., and He, B. Federated learning on non-IID data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering, ICDE*, pp. 965–978. IEEE, 2022.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems 2020, MLSys 2020*, 2020a.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations, ICLR*, 2020b.

Li, Z. and Chen, L. Communication-efficient decentralized zeroth-order method on heterogeneous data. In *International Conference on Wireless Communications and Signal Processing, WCSP*, pp. 1–6, 2021.

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

Ling, Z., Chen, D., Yao, L., Li, Y., and Shen, Y. On the convergence of zeroth-order federated tuning for large language models. *arXiv preprint arXiv:2402.05926*, 2024.

Liu, S., Kailkhura, B., Chen, P.-Y., Ting, P., Chang, S., and Amini, L. Zeroth-order stochastic variance reduction for nonconvex optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., and Tang, J. GPT understands, too. *AI Open*, 2023.

Malladi, S., Gao, T., Nichani, E., Damian, A., Lee, J. D., Chen, D., and Arora, S. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.

Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., and Bossan, B. PEFT: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

Maritan, A., Dey, S., and Schenato, L. FedZeN: Towards superlinear zeroth-order federated learning via incremental hessian estimation. *CoRR*, abs/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 *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Melas-Kyriazi, L. and Wang, F. Intrinisic gradient compression for federated learning. *arXiv preprint arXiv:2112.02656*, 2021.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

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

Qin, Z., Yan, X., Zhou, M., and Deng, S. BlockDFL: A blockchain-based fully decentralized peer-to-peer federated learning framework. In *Proceedings of the ACM on Web Conference 2024*, pp. 2914–2925, 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. *Advances in Neural Information Processing Systems*, 36, 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 *International Conference on Machine Learning*, pp. 8253–8265. PMLR, 2020.

Shu, Y., Lin, X., Dai, Z., and Low, B. K. H. Federated zeroth-order optimization using trajectory-informed surrogate gradients. *arXiv preprint arXiv:2308.04077*, 2023.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Sun, X., Ji, Y., Ma, B., and Li, X. A comparative study between full-parameter and LoRA-based fine-tuning on chinese instruction data for instruction following large language model. *arXiv preprint arXiv:2304.08109*, 2023.

Tan, A. Z., Yu, H., Cui, L., and Yang, Q. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

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., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Villalobos, P., Sevilla, J., Heim, L., Besiroglu, T., Hobbhahn, M., and Ho, A. Will we run out of data? an analysis of the limits of scaling datasets in machine learning. *arXiv preprint arXiv:2211.04325*, 2022.

Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., Mirzaei, A., Naik, A., Ashok, A., Dhanasekaran, A. S., Arunkumar, A., Stap, D., Pathak, E., Karamanolakis, G., Lai, H. G., Purohit, I., Mondal, I., Anderson, J., Kuznia, K., Doshi, K., Pal, K. K., Patel, M., Moradshahi, M., Parmar, M., Purohit, M., Varshney, N., Kaza, P. R., Verma, P., Puri, R. S., Karia, R., Doshi, S., Sampat, S. K., Mishra, S., A, S. R., Patro, S., Dixit, T., and Shen, X. Super-naturalinstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 5085–5109. Association for Computational Linguistics, 2022.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR*, 2022.

Woisetschläger, H., Isenko, A., Wang, S., Mayer, R., and Jacobsen, H.-A. Federated fine-tuning of LLMs on the very edge: The good, the bad, the ugly. *arXiv preprint arXiv:2310.03150*, 2023.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Xi, H., Li, C., Chen, J., and Zhu, J. Training transformers with 4-bit integers. *Advances in Neural Information Processing Systems*, 36:49146–49168, 2023.

Xu, M., Wu, Y., Cai, D., Li, X., and Wang, S. FwdLLM: Efficient FedLLM using forward gradient. *arXiv preprint 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. *CoRR*, abs/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 *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6915–6919. IEEE, 2024.

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 *Findings of the Association for Computational Linguistics: ACL*, pp. 9963–9977, 2023.

Appendix

We provide more discussions and experiments of this work in the appendix and organize them as follows:

- Appendix A: we discuss detailed technical distinctions of our approach to existing approaches related to our work.
- Appendix B: we provide Table 4 to list the used notations along with their corresponding meanings.
- Appendix C: we summarize the processes of FedKSeed and FedKSeed-Pro in Algorithm 1.
- Appendix D: we provide detailed assumptions relied upon Theorem 1.
- Appendix E: We provide the detailed proofs of Lemma 1 and Theorem 2.
- Appendix F: we offer additional analytical support for Principles 1 and 2 from a perspective distinct from Section 4.2.2.
- Appendix G: we provide implementation details of our approach for better reproducibility.
- Appendix H: we provide supplementary experiments, including (1) the marginal improvement on Rouge-L gained per extra seed across varied seed quantities by FedKSeed and FedKSeed-Pro in Appendix H.1; (2) the convergence illustrations in Appendix H.2; (3) illustrations of seed probabilities calculated by FedKSeed-Pro in Appendix H.3; and (4) the time consumption for local training in Appendix H.4.
- Appendix I: we provide the detailed calculation of the communication overhead of our approach and the baselines, to explain how to derive the numerical communication overheads of these approaches in Table 3.
- Appendix J: we discuss the extended benefits brought by our proposed approach in real-world applications.

A. Detailed Technical Comparisons

In this section, we compare our approach with existing approaches that have a certain relationship with our approach from technical, objective, and performance perspectives.

In recent years, there have been some researches that utilize the universal random seeds to lower the communication cost of transmitting model parameters or gradients (Xu et al., 2023; Feng et al., 2023; Zelikman et al., 2023; Rahimi et al., 2024). These approaches can respectively achieve unidirectional $\mathcal{O}(d)$ (Xu et al., 2023; Feng et al., 2023), bidirectional $\mathcal{O}(1)$ (Zelikman et al., 2023; Rahimi et al., 2024) communication cost.

FwdLLM (Xu et al., 2023), BAFFLE (Feng et al., 2023) and FedZeN (Maritan et al., 2023) do not restrict the number of candidate seeds. Therefore, they still need to distribute the latest trainable model parameters to clients in each round. If they are adopted for full-parameter tuning of LLMs with FL, clients have to consume tremendous communication resources to download the latest global LLM in each round, thus, it may prevent many clients from participating in FL since the cost and quality of a wireless connection can vary greatly between different countries (Dorfman et al., 2023).

The approach proposed by Zelikman et al. (2023) achieves bidirectional $\mathcal{O}(1)$ communication cost, optimizing the communication efficiency to the utmost. However, there is no such thing as a free lunch. As we have presented in Figure 2, Zelikman et al. (2023) sample seeds from a vast space that is almost infinite, causing each client must replicate the update steps performed by all other clients to obtain the latest model. Consequently, the overhead of calculating the latest model from $\mathbf{w}^0$ also grows indefinitely as the rounds of FL continue, quickly reaching a level that is unsustainable for end devices. Since the fine-tuning of large models typically requires many steps over a large instruction dataset, the approach proposed by Zelikman et al. (2023) is not suitable for full-parameter tuning of LLMs with FL on devices. Moreover, Zelikman et al. (2023) conduct experiments only on a small sentiment classification dataset, i.e., SST2 (Socher et al., 2013), and train a model for a total of only 16,000 steps. However, for complex datasets, such as Natural Instructions (Wang et al., 2022), this number of update steps may not be sufficient for the convergence of LLMs.

One recent work, EvoFed (Rahimi et al., 2024), also achieves bidirectional $\mathcal{O}(1)$ communication cost. However, EvoFed is fundamentally different from FedKSeed, and it is not specifically designed for fine-tuning LLMs: (1) During local training, EvoFed first conducts BP to compute the true gradient. Then it generates K noise-perturbed model populations and tries to represent the true gradient by the summation of these noise perturbations. The weight coefficient of each population is determined by the l_2 norm of its pairwise differences with the true gradient. Thus, EvoFed still relies on the BP process to get the true gradient, which is not practical for fine-tuning LLMs with billions of parameters. As reported by Malladi et al. (2023), fine-tuning full LLMs imposes a tremendous memory burden. For example, full-parameter tuning of an LLM with 2.7B parameters with an average sequence length of 400 tokens can consume up to 55GB of memory. Such a level of memory consumption exceeds the capabilities of a single graphics card such as NVIDIA V100 (32GB), not to mention an end device. (2) The calculation of l_2 norm of the pairwise differences between a perturbation and the true gradient also consumes tremendous computation resources for billion-sized LLMs. This operation must be repeated K times each round, further increasing the computational load. (3) As highlighted in Equation (12) presented by Rahimi et al. (2024), when facing the scenario of partial client participation, a client that has not participated in FL for ς rounds has to perform $\varsigma \cdot K$ model updates to calculate the latest global model, while FedKSeed and FedKSeed-Pro still only need to perform K steps. In the experiments conducted by Rahimi et al. (2024), EvoFed is evaluated on small visual datasets with small models, i.e., containing at most 2.3 million parameters, while our approach is evaluated on LLMs with at most 3.43 billion parameters.

Note that there are also some BP-based FL approaches that optimize the communication overhead by encoding the gradient into low-dimensional spaces, such as model sketch (Melas-Kyriazi & Wang, 2021; Rothchild et al., 2020). As we have discussed in Section 2, these approaches are not tailored for LLMs and are not suitable for LLM tuning on end devices due to the tremendous memory footprint, especially for full-parameter tuning of billion-sized LLMs. Besides, they are only evaluated by small models instead of billion-sized LLMs.

Based on the technical comparisons presented above, FedKSeed emerges as the first approach that enables the possibility of federated full-parameter tuning of billion-sized LLMs on devices. It achieves this by reducing the communication cost to a constant of less than 18 kilobytes and minimizing memory consumption to the level required for inference.

B. Summarization of Notations

Table 4. Notations and corresponding meanings.

Notation	Meaning
N	Number of total clients.
m	Average number of active clients in each round.
τ	Number of steps during local training in each round.
T	Number of total rounds.
r	Number of rounds that have already been elapsed.
Γ	$\Gamma = m \cdot \tau \cdot T$, denoting the total steps of updates across the T rounds of FL.
$\mathcal{D}_i$	Local dataset of client i.
$\mathbf{x}$	A data sample in $\mathcal{D}_i$, such as a Q-A pair in Natural Instructions.
c_i	The weight of aggregation for client$i$.
$\mathbf{w}_0$	Initial weights of an LLM, $\mathbf{w}_0 \in \mathbb{R}^d$.
d	The dimensionality of the LLM.
$\mathbf{w}_{i,t}^r$	Model weights of client i after the t-th step of local training in round r.
$\mathcal{L}(\mathbf{w};\mathbf{x})$	The loss evaluated at model $\mathbf{w}$ on a data instance $\mathbf{x}$.
$\mathbb{S}$	The set of distinct candidate seeds.
K	The size of $\mathbb{S}$, i.e., $\lvert\mathbb{S}\rvert$.
s_j	One random seed in $\mathbb{S}$, indexed by j.
$\mathbf{z}_j$	A randomly sampled perturbation drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, indexed by j.
$\mathbf{g}$	True gradient computed with stochastic gradient descent algorithms.
$\widehat{\mathbf{g}}_j$	The gradient estimated with zeroth-order optimization, on the perturbation $\mathbf{z}_j$.
$\widehat{\varrho}_j$	The scalar gradient corresponding to $\widehat{\mathbf{g}}_j$, where $\widehat{\mathbf{g}}_j = \widehat{\varrho}_j \cdot \mathbf{z}_j$.
$(s_j, \widehat{\varrho}_j)$	A (seed, scalar gradient) pair.
η	Learning rate during local training.
ϵ	Scale of a random perturbation.

C. The Algorithm of the Proposed FedKSeed and FedKSeed-Pro

Algorithm 1: The processes of **FedKSeed**, where the $\underline{\text{underlined}}$ components and processes are only required by the enhanced version of it that samples seeds during local training with non-uniform probabilities, i.e., $\underline{\text{FedKSeed-Pro}}$.

```
Input: $N$, $K$, $\mathbf{w}^0$, $\eta$, $\{c_1, \ldots, c_N\}$, $T$ and $\tau$.
Output: The global model $\mathbf{w}^T$ that has been fine-tuned for $T$ rounds.
Server Executes: initialize $K$ candidate seeds $\mathbb{S}$, scalar gradient accumulator $\mathcal{A}$, $\underline{\text{and their probabilities } \mathbf{p}}$.
for each round $r = 1, 2, \ldots, T$ do
    for each client $i \in$ activate clients $\mathbb{C}$ in parallel do
        $\mathbb{H}_i \leftarrow$ ClientTraining($\mathbb{S}$, $\mathcal{A}$, $\underline{\mathbf{p}}$, $i$)          // ① in Figure 3
        for $(s_j, \widehat{\varrho}_j) \in \mathbb{H}_i$ do
            $a_j = a_j + c_i \cdot \widehat{\varrho}_j$                                       // ⑤ in Figure 3
    $\underline{\text{compute the seed importance and then the probability } \mathbf{p} \text{ as Equation (13)}}$   // ⑥ in Figure 3
return the fine-tuned global model $\mathbf{w}^T$, which is calculated with $\mathbf{w}^0$ as the initial point based on $\mathcal{A}$, as Equation (4)

Function ClientTraining($\mathbb{S}$, $\mathcal{A}$, $\underline{\mathbf{p}}$, $i$):
    calculate the latest global model with $\mathbf{w}^0$ as the initial point based on $\mathcal{A}$, as Equation (4)  // ② in Figure 3
    for each local step $t = 1, 2, \ldots, \tau$ do
        sample a data instance $\mathbf{x}$ from local dataset $\mathcal{D}_i$, a seed $s_j$ from $\mathbb{S}$ $\underline{\text{based on } \mathbf{p}}$, then generate a perturbation $\mathbf{z}_j$
          with $s_j$ as the random seed                                           // ③-1 in Figure 3
        $\widehat{\varrho}_j = \frac{\mathcal{L}(\mathbf{w}+\epsilon\mathbf{z}_j;\mathbf{x})-\mathcal{L}(\mathbf{w}-\epsilon\mathbf{z}_j;\mathbf{x})}{2\epsilon}$
        $\mathbf{w}_{t+1} =$ UpdateModel($\mathbf{w}_t$, $s_j$, $\widehat{\varrho}_j$)
        stage $(s_j, \widehat{\varrho}_j)$ into $\mathbb{H}_i$                                    // ③-2 in Figure 3
    return $\mathbb{H}_i$ to the server                                           // ④-1 in Figure 3

Function UpdateModel($\mathbf{w}$, $s$, $\widehat{\varrho}$):
    sample perturbation $\mathbf{z} \in \mathbb{R}^d$ with random seed $s$
    return $\mathbf{w} - \eta \cdot \widehat{\varrho} \cdot \mathbf{z}$
```

Algorithm 1 summarizes the main processes of FedKSeed. For ease of comparison, we also include the processes and components that are only required by FedKSeed-Pro in Algorithm 1, which are underlined and freely detachable as needed.

D. Detailed Assumptions of Theorem 1

We detail the assumptions made by Fang et al. (2022) which are necessary conditions for deriving the convergence of ZOO-based FL with a one-point estimator, as claimed in Theorem 1. We have also made variable substitutions to facilitate understanding within the context of our work.

Assumption 2. *(Loss Boundary.) The global loss $f(\mathbf{w})$ defined in Equation (1) is lower bounded by f_*, thus we have*

$$f(\mathbf{w}) \geq f_* > -\infty.$$

Before presenting Assumption 3 and Assumption 4, we define the expected loss $f_i(\mathbf{w})$ of model $\mathbf{w}$ on the i-th client's local dataset $\mathcal{D}_i$ as $f_i(\mathbf{w}) \triangleq \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_i}[\mathcal{L}_i(\mathbf{w};\mathbf{x})]$.

Assumption 3. *(Objective Smoothness.) $\mathcal{L}_i(\mathbf{w};\mathbf{x})$, $f_i(\mathbf{w})$ and $f(\mathbf{w})$ are all L-smooth, i.e., for any $\mathbf{w} \in \mathbb{R}^d$ and $\mathbf{w}' \in \mathbb{R}^d$, we have*

$$\|\nabla f_i(\mathbf{w}') - \nabla f_i(\mathbf{w})\| \leq L \|\mathbf{w}' - \mathbf{w}\|, \forall i,$$

$$f(\mathbf{w}') \leq f(\mathbf{w}) + \langle \nabla f(\mathbf{w}), \mathbf{w}' - \mathbf{w} \rangle + \frac{L}{2} \|\mathbf{w}' - \mathbf{w}\|^2.$$

Assumption 4. *(Boundary of the Second-Order Gradient Moment.) The second-order moment of stochastic gradient* $\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w},\mathbf{x})$ *satisfies*

$$\mathbb{E}_{\mathbf{x}}\left\|\nabla_{\mathbf{w}}\mathcal{L}_i(\mathbf{w};\mathbf{x})\right\|^2 \le c_g\left\|\nabla f_i(\mathbf{w})\right\|^2 + \sigma_g^2, \forall \mathbf{w}\in\mathbb{R}^d,$$

where $c_g \ge 1$.

Assumption 5. *(Local-Global Gradient Dissimilarity Boundary.) The gradient dissimilarity between the local loss evaluated at each client and the global loss defined in Equation* (1) *is bounded as*

$$\left\|\nabla f(\mathbf{w}) - \nabla f_i(\mathbf{w})\right\|^2 \le c_h\left\|\nabla f(\mathbf{w})\right\|^2 + \sigma_h^2, \forall \mathbf{w}\in\mathbb{R}^d,$$

where c_h *is a positive constant.*

Assumptions 2, 3 and 4 are commonly employed in the analysis of stochastic optimizations (Fang et al., 2022), and Assumption 5 can capture the extent of statistical heterogeneity inherent in the client-side data distribution within FL settings. Similar assumptions have also been employed in existing research to analyze FL convergence in non-IID scenarios (Li et al., 2020a;b; Ling et al., 2024). Theorem 1 corresponds to Equation (12) in the work of Fang et al. (2022), with certain lower-order terms omitted for brevity.

E. Detailed Proofs

E.1. Proof of Lemma 1

Recall the two formal sampling stage of FedKSeed in Section 4.2.1, in the first sampling stage, we have K random variables $\mathcal{Z}' = Z_1', Z_2', Z_3', \dots, Z_K'$ that randomly sampled from $\mathcal{N}(0,1)$. By Chebyshev's inequality, $\forall\varepsilon > 0$, we have

$$\Pr[|\frac{1}{K}(Z_1' + Z_2' + \cdots + Z_K')| \ge \varepsilon] \le \frac{4}{K\varepsilon^2}. \tag{14}$$

In the second sampling stage, we have $Z_1, Z_2, \dots, Z_\Gamma$ that randomly and uniformly sampled from $\mathcal{Z}'$. Thus $\forall Z_i$, we have $\min(\mathcal{Z}') \le Z_i \le \max(\mathcal{Z}')$. Let $\mu = \mathbb{E}[Z_i]$, according to Hoeffding's inequality, $\forall\varepsilon' > 0$, $S_\Gamma = Z_1 + Z_2 + \ldots + Z_\Gamma$ satisfies

$$\Pr[|S_\Gamma - \mu\Gamma| \ge \varepsilon'] \le 2\exp(-\frac{2\varepsilon'^2}{\sum_{i=1}^{\Gamma}[\max(\mathcal{Z}') - \min(\mathcal{Z})]^2}). \tag{15}$$

Let $\bar{S}_\Gamma = \frac{Z_1+Z_2+\ldots+Z_\Gamma}{\Gamma}$. Thus $\forall\varepsilon > 0$, we have

$$\Pr[|\bar{S}_\Gamma - \mu| \ge \varepsilon] \le 2\exp(-\frac{2\Gamma\varepsilon^2}{[\max(\mathcal{Z}') - \min(\mathcal{Z}')]^2}). \tag{16}$$

Finally, $\forall\varepsilon > 0$, we have

$$\begin{aligned}\Pr[|\bar{S}_\Gamma - 0| \ge \varepsilon] &\le \Pr[|\mu - 0| \ge \frac{\varepsilon}{2}] + \Pr[|\bar{S}_\Gamma - \mu| \ge \frac{\varepsilon}{2}]\\ &\le \frac{4}{K\varepsilon^2} + 2\exp(-\frac{\Gamma\varepsilon^2}{2[\max(\mathcal{Z}') - \min(\mathcal{Z}')]^2}).\end{aligned} \tag{17}$$

■

E.2. Proof of Theorem 2

From Lemma 1, the perturbations follow ε-mean in FedKSeed. We denote the gradient estimated by the two-point gradient estimator by $\widehat{\mathbf{g}}_1$, calculated as

$$\widehat{\mathbf{g}}_1 = \frac{\mathbf{z} + \overrightarrow{\varepsilon}}{2\epsilon}[\mathcal{L}(\mathbf{w} + \epsilon\mathbf{z} + \epsilon\overrightarrow{\varepsilon}) - \mathcal{L}(\mathbf{w} - \epsilon\mathbf{z} - \epsilon\overrightarrow{\varepsilon})], \tag{18}$$

and the gradient estimated by the two-point gradient estimator without seed restrictions is defined as

$$\widehat{\mathbf{g}}_0 = \frac{\mathbf{z}}{2\epsilon}[\mathcal{L}(\mathbf{w} + \epsilon\mathbf{z}) - \mathcal{L}(\mathbf{w} - \epsilon\mathbf{z})]. \tag{19}$$

Thus,

$$\begin{aligned}\widehat{\mathbf{g}}_1 - \widehat{\mathbf{g}}_0 =& \frac{\mathbf{z}}{2\epsilon}[\mathcal{L}(\mathbf{w}+\epsilon\mathbf{z}+\epsilon\overrightarrow{\varepsilon}) - \mathcal{L}(\mathbf{w}+\epsilon\mathbf{z})] - \frac{\mathbf{z}}{2\epsilon}[\mathcal{L}(\mathbf{w}-\epsilon\mathbf{z}-\epsilon\overrightarrow{\varepsilon}) - \mathcal{L}(\mathbf{w}-\epsilon\mathbf{z})]+ \\ & \frac{\overrightarrow{\varepsilon}}{2\epsilon}[\mathcal{L}(\mathbf{w}+\epsilon\mathbf{z}+\epsilon\overrightarrow{\varepsilon}) - \mathcal{L}(\mathbf{w}-\epsilon\mathbf{z}-\epsilon\overrightarrow{\varepsilon})].\end{aligned} \tag{20}$$

According to the L-smoothness of objective function $\mathcal{L}$ (Assumption 3), we have

$$\left\|\mathcal{L}(\mathbf{w}+\epsilon\mathbf{z}+\epsilon\overrightarrow{\varepsilon}) - \mathcal{L}(\mathbf{w}+\epsilon\mathbf{z})\right\| \leq L\left\|\epsilon\overrightarrow{\varepsilon}\right\|, \tag{21}$$

and

$$\left\|\mathcal{L}(\mathbf{w}-\epsilon\mathbf{z}-\epsilon\overrightarrow{\varepsilon}) - \mathcal{L}(\mathbf{w}-\epsilon\mathbf{z})\right\| \leq L\left\|\epsilon\overrightarrow{\varepsilon}\right\|. \tag{22}$$

Therefore,

$$\begin{aligned}\left\|\widehat{\mathbf{g}}_1 - \widehat{\mathbf{g}}_0\right\| \leq & \left\|\frac{\mathbf{z}}{2\epsilon}\right\| L\left\|\epsilon\overrightarrow{\varepsilon}\right\| + \left\|\frac{\mathbf{z}}{2\epsilon}\right\| L\left\|\epsilon\overrightarrow{\varepsilon}\right\| + \left\|\frac{\overrightarrow{\varepsilon}}{2\epsilon}\right\| L\left\|2\epsilon\mathbf{z}+2\epsilon\overrightarrow{\varepsilon}\right\| \\ =& L\left\|\mathbf{z}\right\|\left\|\overrightarrow{\varepsilon}\right\| + L\left\|\overrightarrow{\varepsilon}\right\|\left\|\mathbf{z}+\overrightarrow{\varepsilon}\right\| \\ \leq & 2L\left\|\mathbf{z}\right\|\left\|\overrightarrow{\varepsilon}\right\| + L\left\|\overrightarrow{\varepsilon}\right\|^2.\end{aligned} \tag{23}$$

■

F. Additional Analytical Supports to Principles on the Selection of K

F.1. Additional Analytical Support to Principle 1

The federated fine-tuning process can be formally modeled as an optimization problem that seeks a model variation from $\mathbf{w}^0$ to an ideally optimal model $\mathbf{w}^*$, with the combination of K perturbations, as

$$\min_{\mathbb{G}} \quad \left\|\mathbf{w}^0 - \eta\cdot\left[\mathbf{z}_1,\ldots,\mathbf{z}_K\right]\mathbb{G} - \mathbf{w}^*\right\|. \tag{24}$$

It is important to note that this definition serves to provide an alternative perspective on FL, and it is not a formulation that can be solved outright because the ideally optimal model weights $\mathbf{w}^*$ are not known. From Equation (24), FL processes can be viewed as advancing the model towards an approximate optimal solution in an iterative manner.

With this formulation, matrix $\mathbb{Z} = [\mathbf{z}_1,\ldots,\mathbf{z}_K]$ can be regarded as the constraints of this problem. When the constraints are insufficient to uniquely determine a solution, i.e., the rank of the system is low, the solution space becomes larger and there are multiple or even infinitely many possible solutions, causing greater difficulty in finding the optimal solution. Since high-dimensional vectors sampled from a Gaussian distribution are typically orthogonal, considering the dimension d of LLM $\mathbf{w}$ is usually very high such that $d \gg K$, the rank of $\mathbb{Z} = [\mathbf{z}_1,\ldots,\mathbf{z}_K]$ is typically K. Therefore, usually the larger the value of K is, the better the optimization problem defined in Equation (24) could be finally solved. Taking an extreme example, if $K = 1$, the optimization problem defined in Equation (24) may be fundamentally unoptimizable. Thus, $\overleftarrow{K}$ theoretically exists so that Principle 1 holds.

F.2. Additional Analytical Support to Principle 2

Recall Equation (8) and Theorem 1, each $\mathbf{z}_j$ is independently randomly generated from $\mathcal{N}(\mathbf{0},\mathbf{I}_d)$. Without introducing system error, Equation (8) can be rewrite as

$$\widehat{\mathbf{g}}_{i,t}^r = \frac{1}{b_2 b_1}\sum_{j=1}^{b_2}\sum_{b=1}^{b_1}\frac{\left[\mathcal{L}(\mathbf{w}_{i,t}^r+\epsilon\mathbf{z}_j;\mathbf{x}_b) - \mathcal{L}(\mathbf{w}_{i,t}^r;\mathbf{x}_b)\right]}{\epsilon}\mathbf{z}_j. \tag{25}$$

In this new formulation, the FL process of FedKSeed and FedKSeed-Pro can be regarded as: for each perturbation $\mathbf{z}$, computing the step size that guides the model's progress in the direction of $\mathbf{z}$. This is achieved through local training on several data instances. Given that the total number of update steps is fixed at τrm, each of the K seeds is sampled on average $\frac{\tau rm}{K}$ times in FedKSeed. When there are fewer candidate seeds, more data instances are used to determine the step size on the direction of each perturbation, magnifying the batch size probabilistically. Besides, when the cardinality of candidate seeds further increases, it does not change the optimal solution area, but enlarges the optimization space and thus increases the difficulty of random searching. From the above analysis, $\overrightarrow{K}$ theoretically exists so that Principle 2 holds.

Table 5. Prompt template for Natural Instructions.

Attributes of data instances	**Prompt**

1. `Definition`
2. `input`
	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:

Table 6. Prompt templates for Dolly-15K, which vary slightly depending on whether the data instance has `context`.

Attributes of data instances	**Prompt**

Data instances with context:
1. `instruction`
2. `context`
	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:

Data instances without context: 1. `instruction`	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`} ### Response:

G. Implementation Details

For better reproducibility, in this section, we provide the detailed implementations of our approaches and the baselines. Some of the experimental settings have already been mentioned in Section 5.1 and are not reiterated here.

G.1. Datasets & Evaluation Metrics

Natural Instructions contains a large collection of tasks and their natural language instructions, and provides the splits for training and test, respectively. We utilize version `v2.8` of the dataset and adopt its `default` split, which includes 756 tasks for training and 119 tasks for testing, each with a unique task definition. Since Natural Instructions is very large, we conduct our experiments on a subset of it. Specifically, we randomly sample 20% of the data instances for each training task and 2% of the data instances for each test task. After the above subsampling, each training task with no less than 20 data instances is treated as a unique client, forming an FL system with 738 clients. The test tasks are retained on the server for evaluation purposes.

Dolly-15K provides 15,015 data instances within 8 tasks. For Dolly-15K, we reserve the last task for evaluation and use the remaining tasks for training. Experiments on Dolly-15K are conducted with 200 clients. Note that each task in Dolly-15K contains a `category` attribute with a different value. Thus, we can allocate data instances to the 200 clients via Dirichlet distribution (Li et al., 2022) with the `category` as the labels. To build non-IID scenarios with varying degrees of label distribution skew (Chen et al., 2023c), we perform data partitioning via Dirichlet distribution with $\alpha = 0.5$ and $\alpha = 5.0$, respectively, where a lower α indicates a higher degree of label distribution skew.

When feeding data instances into LLMs, we directly adopt the prompt template from Alpaca (Taori et al., 2023) following Kuang et al. (2023); Zhang et al. (2024). The utilization of the prompt template is detailed in Appendix G.2. The maximum token length is set to 1,024, and data instances exceeding this length are ignored. To reduce the impact of extraneous variables on the experiment results as much as possible, we uniformly employ greedy decoding to generate the responses during evaluation following Malladi et al. (2023); Borzunov et al. (2023).

G.2. Prompt Template

In our experiments, data instances are wrapped to prompts before processing by LLMs. We directly apply the template provided by Alpaca (Taori et al., 2023) to the datasets in our experiments. For better reproducibility, we present how we fill the fields in the template with the attributes of data instances in Tables 5 and 6.

G.3. Experimental Platforms

We implement these approaches by PyTorch (Paszke et al., 2019) `v2.0.1` with `PEFT` (Mangrulkar et al., 2022) `v0.4.0` and `Transformers` (Wolf et al., 2020) `v4.31.0`. Extensive performance evaluations on Rouge-L scores with DataJuicer-1.3B and LLaMA-3B are conducted on a platform with an NVIDIA RTX 3090 GPU and a platform with an NVIDIA A100 GPU, respectively, with the pre-trained LLMs loaded in 16-bit floating numbers. The devices used for time-related experimental results (Figure 2, Figure 8 and Figure 14) are all specified in the corresponding captions or sections. Note that from Table 3, FedKSeed and FedKSeed-Pro do not require as much memory as these platforms can provide, unlike the baselines based on BP. Therefore, this experimental setup is adopted to ensure consistency in the experimental environments among different approaches.

G.4. Implementations

Following Kuang et al. (2023) and Malladi et al. (2023), all approaches perform local training with the batch size set to 1 to reduce memory consumption. Following Kuang et al. (2023), BP-based approaches conduct local training with learning rate η of 3×10^{-4}, where the selected learning rate is searched from $[3 \times 10^{-3}, 3 \times 10^{-4}, 3 \times 10^{-5}]$. Among them, the number of virtual tokens in FedPTuning and FedPrompt are both set to 20, the type of reparameterization is set to "MLP" for FedPTuning following Kuang et al. (2023), and the `rank` and `alpha` of LoRA adapters for both FedIT and FedIT-SGD are set to 8 and 16 respectively, following Zhang et al. (2024). Following Malladi et al. (2023), η and ϵ of FedKSeed and FedKSeed-Pro are set to 3×10^{-7} and 5×10^{-4}, respectively, unless stated otherwise. The selected learning rate of FedKSeed and FedKSeed-Pro is searched from $[3 \times 10^{-5}, 3 \times 10^{-6}, 3 \times 10^{-7}, 1 \times 10^{-7}]$. The impacts of the two hyperparameters of FedKSeed and FedKSeed-Pro have been discussed in Section 5.4.

Before starting the federated tuning, the server initializes the K candidate seeds with integers uniformly and randomly sampled from [0, 10^{11}). The aggregation weights of participating clients in each round are proportional to the scale of their private training set.

G.5. Further Optimization on Updating Models

We highly thanks an anonymous reviewer for providing a more efficient implementation. The main differences between this implementation and ours lie in (1) the generation of perturbations ($\mathbf{z}$), and (2) the updating of model parameters. These differences are presented in Table 7. For the reproducibility of the Rouge-L scores in this work, we left the integration of this implementation in the future.

H. Supplementary Experiments

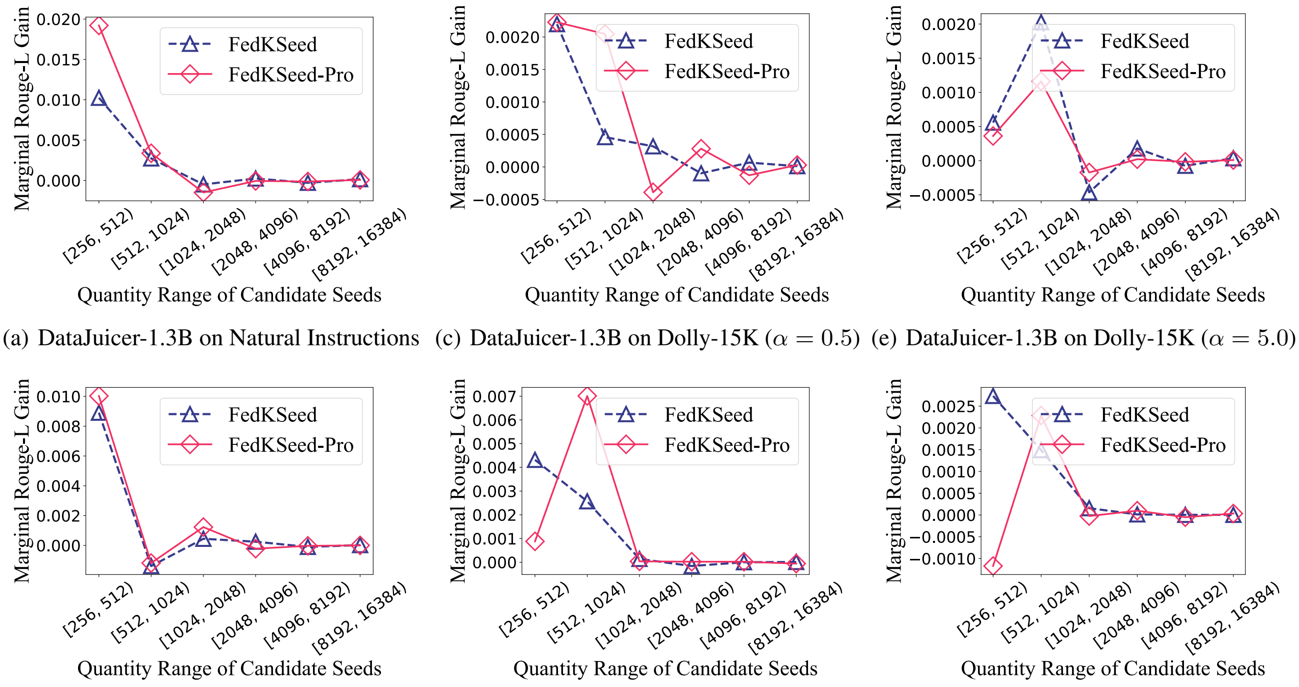
H.1. Experimental Support for Principle 2

In Section 4.2.2, we have provided Principles 1 and 2 to guide the determination of K. From Figure 7 in Section 5.2, it can be observed that once the number of seeds exceeds the threshold $\overrightarrow{K}$, which is around 4096, additional seeds do not lead to

Table 7. Main differences between our implementation and a more efficient implementation.

Functionality	Our Implementation	More Efficient Implementation
Perturbation Generation	`z = torch.normal(mean=0, std=1, size=param.size())`	1. `gen = torch.Generator()` 2. `z = torch.empty()` 3. `z.resize_(param.size())` 4. `z.normal_(mean=0, std=1, generator=gen)`
Updating Model Parameters	`param.data = param.data - (lr * scalar_grad) * z`	1. `scalar = lr * scalar_grad` 2. `param.data.add_(z, alpha=-scalar)`

improved accuracy. However, the Y-axis range in Figure 7 may obscure the visibility of the trend outlined in Principle 2. For better clarity, in Figure 11, we present the marginal improvement on Rouge-L obtained with each additional seed within different ranges of seed quantity by FedKSeed and FedKSeed-Pro in the six scenarios, respectively.



(a) DataJuicer-1.3B on Natural Instructions (c) DataJuicer-1.3B on Dolly-15K ($\alpha = 0.5$) (e) DataJuicer-1.3B on Dolly-15K ($\alpha = 5.0$)

(b) LLaMA-3B on Natural Instructions (d) LLaMA-3B on Dolly-15K ($\alpha = 0.5$) (f) LLaMA-3B on Dolly-15K ($\alpha = 5.0$)

Figure 11. Marginal improvement on Rouge-L obtained with each additional seed across different ranges of seed quantity by FedKSeed and FedKSeed-Pro, respectively.

From Figure 11, we can find that when K is lower than 1024, each additional seed yields a substantial and consistently greater than zero improvement in Rouge-L accuracy on average. In other words, within this range, reducing K would result in a noticeable decrease in Rouge-L accuracy of the global model, which can also serve as empirical evidence for Principle 1. When K falls within the range of [1024, 4096), the marginal gain in average accuracy for each additional seed is negligible in certain scenarios. When K exceeds 4096, it becomes evident that additional seeds yield negligible marginal gains in accuracy across various scenarios. Thus, we can find that $\overrightarrow{K}$ for FedKSeed should be 4096 such that when $K > \overrightarrow{K}$, there is no upward trend in the Rouge-L of the global model with the increase of K, therefore Principle 2 holds. Considering that increasing K will incur additional time costs for clients to synchronize the global model as Equation (4), we should choose a smaller K value within a reasonable range.

Based on the experimental results and analysis presented above, and in conjunction with the analytical support for Principle 2 provided in Section 4.2.2 and Appendix F.2, Principle 2 holds.

It should be noted that Figure 11 demonstrates the marginal improvements, not the actual accuracy itself. In Figure 11, the marginal gains for each seed of FedKSeed and FedKSeed-Pro are relatively similar. But as illustrated in Figure 7, FedKSeed-Pro usually requires fewer candidate seeds to achieve the same level of accuracy compared to FedKSeed. Therefore, if we are not solely pursuing the highest accuracy, FedKSeed-Pro can achieve comparable results with fewer candidate seeds. It is for this purpose that when we record the results in Table 2, we make FedKSeed-Pro to use fewer candidate seeds.

H.2. Convergence Study

We illustrate the convergence curves obtained by FedKSeed, FedKSeed-Pro and the baselines with LLaMA-3B on Natural Instructions in Figure 12, where the experimental setup is aligned with that described in Section 5.1 and Appendix G. Note that the loss values are calculated on the test set that is held by the server as described in Section 5.1.

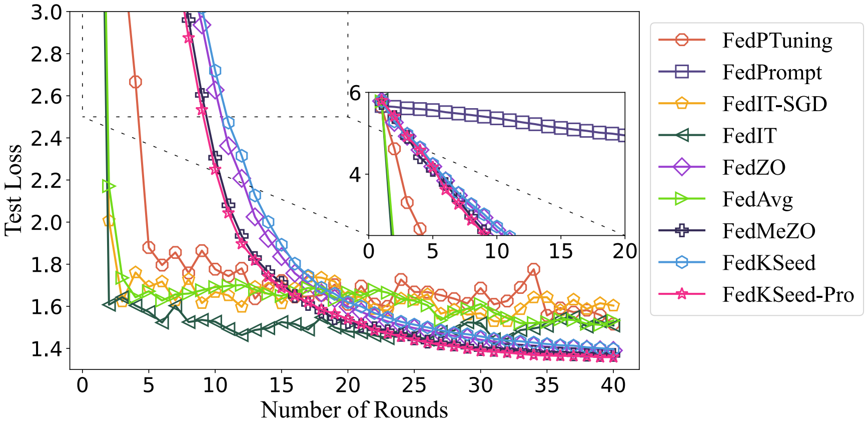


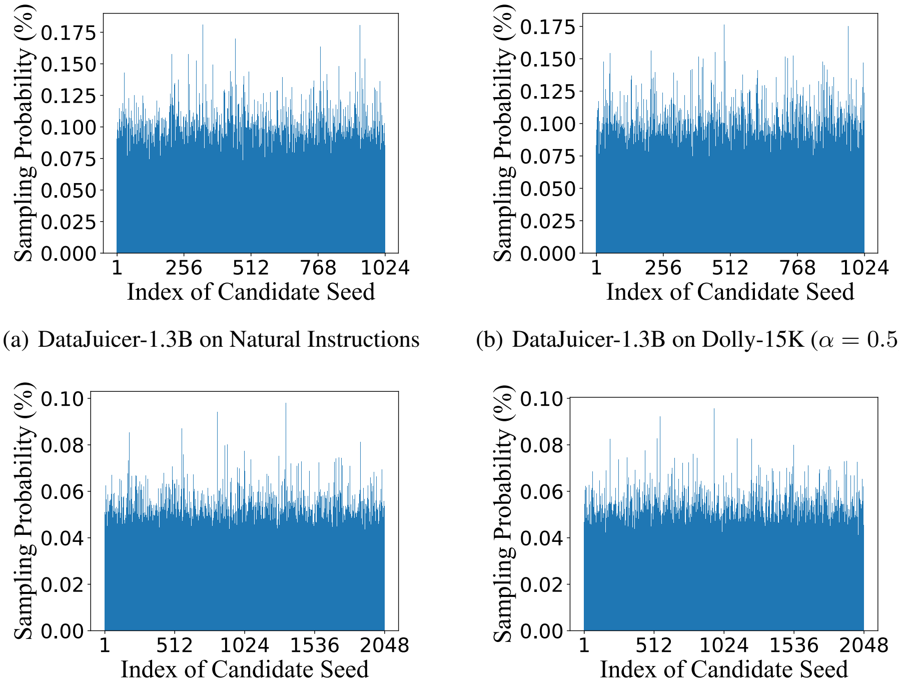
Figure 12. Convergence of the loss value on the test tasks obtained by FedKSeed, FedKSeed-Pro and the baselines with LLaMA-3B on Natural Instructions.

Different from training from sketch, the pre-trained weights offer a good initial point for fine-tuning, leading to relatively quick convergence rates for these methods. As illustrated in Figure 12, all of the approaches except for FedPrompt can converge within 40 rounds. It is worth noting that FedPrompt converges much more slowly, and as shown in Table 2, it also underperforms compared to the other baselines. Although not as swift as the backpropagation (BP)-based approaches, the convergence rates for FedKSeed and FedKSeed-Pro are considerably faster than one might expect. The reason is that fine-tuning is different from pre-training. As theoretically suggested by Malladi et al. (2023), in centralized settings, based on adequate pre-training, the convergence of ZOO usually depends on the local effective rank instead of the number of full model parameters. Therefore, it is reasonable and promising to apply ZOO to the federated full-parameter tuning of billion-sized LLMs.

H.3. Illustrations of Seed Probabilities in FedKSeed-Pro

To demonstrate the variability in seed importance, we present the seed probabilities calculated in the last round by FedKSeed and FedKSeed-Pro on Natural Instructions and Dolly-15K ($\alpha = 0.5$) with DataJuicer-1.3B and LLaMA-3B, respectively in Figure 13. As described in Section 5.1, we set K to 1024 for FedKSeed-Pro with DataJuicer-1.3B and 2048 for FedKSeed-Pro with LLaMA-3B, respectively.

It can be observed that with non-uniform sampling, there is a significant disparity in the probabilities of seeds being sampled and a relatively small subset of the candidate seeds exhibits higher sampling probabilities. Thus, we can conclude that the average amplitude corresponding to the scalar gradient of a seed is positively correlated with the importance of the corresponding perturbation to the model accuracy.



(a) DataJuicer-1.3B on Natural Instructions

(b) DataJuicer-1.3B on Dolly-15K ($\alpha = 0.5$)

(c) LLaMA-3B on Natural Instructions

(d) LLaMA-3B on Dolly-15K ($\alpha = 0.5$)

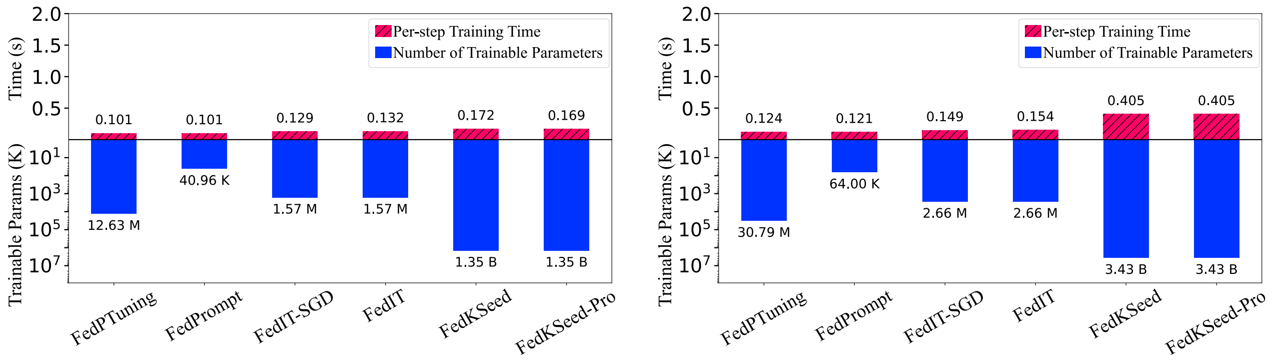
Figure 13. Probabilities of candidate seeds calculated by FedKSeed-Pro after the last round.

It is worth noting that quantifying the importance of seeds or perturbations through the amplitude of scalar gradients may be better than that through similarity evaluated by cosine or Euclidean distance, since vectors from high-dimensional Gaussian distributions tend to be orthogonal to each other, which results in the similarity distance of random perturbations with the same dimensionality as billion-sized LLMs typically being very close to 0, making it difficult to distinguish numerically. Therefore, it is challenging for distance-based methods to assign significantly different importance to various seeds or perturbations. That is the reason why we adopt the amplitude of the scalar gradient to quantify the seed importance.

H.4. Training Efficiency

To have a clear view of the training efficiency of these approaches, we present their per-step training time together with the number of trainable parameters in Figure 14. Note that to ensure the comparability of the time consumptions across different LLMs with the same approach, these time consumptions are uniformly tested on the same platform equipped with an NVIDIA V100 GPU and an Intel(R) Xeon(R) Platinum 8163 CPU, with the required libraries the same as that have been described in Appendix G.3.

As shown in Figure 14, compared to the baselines, FedKSeed and FedKSeed-Pro only incur minimal additional per-step training time overhead. This limited extra per-step computational expense brings the benefit that allows for an expansion of trainable parameters by several orders of magnitude, thus improving the accuracy of the global model, as demonstrated in Table 2. Furthermore, the communication overhead and memory footprint have also been significantly reduced by FedKSeed and FedKSeed-Pro compared to the baselines as in Table 3. While FedKSeed and FedKSeed-Pro may take more time to perform one step of local training due to the significantly larger number of trainable parameters compared to PEFT-based techniques, the constraint on time consumption is not as strict as that on memory and communication. This is because the development of computing power has generally outpaced that of memory and communication resources. Thus, the additional computational overheads of FedKSeed and FedKSeed-Pro may be worth it in comparison to the gains obtained by them in accuracy, memory footprint and communication consumption.



(a) DataJuicer-1.3B on Natural Instructions

(b) LLaMA-3B on Natural Instructions

Figure 14. Comparisons between the practical approaches in Table 2 on per-step training time (measured with an NVIDIA V100 GPU) and on the number of trainable parameters.

I. Detailed Calculation of Communication Overhead

In this section, we provide a detailed analysis to demonstrate how the client-side per-round communication overheads of FedKSeed, FedKSeed-Pro and the baselines presented in Table 3 are calculated. Note that the communication overhead referred to here includes both the downlink overhead incurred from downloading data from the server and the uplink overhead associated with uploading data to the server.

In Table 3, for the baselines, we only account for the communication overhead brought about by the transmission of model parameters, ignoring the transmission of other information, including model architecture, request headers, etc., as these costs vary under different encoding schemes. The model parameters in baselines and scalar gradients in FedKSeed and FedKSeed-Pro are encoded as 32-bit floating point numbers to prevent the loss of precision and overflow or underflow. For FedKSeed and FedKSeed-Pro, the random seeds are encoded as 32-bit integers. Note that the aforementioned encoding scheme with single-precision floating-point format is fair to all methods. For all methods, half-precision encoding can be used to uniformly reduce communication overheads presented in Table 3 by half.

Communication overheads of baselines. For these baselines, their per-round client-side communication costs are accounted as the total requirements of uploading and downloading the trainable parameters, since only this part of the parameters in the model is updated.

Communication overhead of FedKSeed. In our experiments, K is uniformly set to 4,096 for FedKSeed. Let D, U and C denote the per-round communication cost of each client for the downlink, uplink, and total, respectively. At the start of each round, each client needs to download the candidate seeds $\mathbb{S}$ and scalar gradient accumulator $\mathcal{A}$ from the server. Since the $\mathbb{S}$ can be encoded as one integer seed which only occupies 4 Bytes, D of FedKSeed can be calculated as

$$\begin{aligned} D = & \underbrace{1 \times 4 \text{ Bytes}}_{\text{one integer seed that encodes the candidate seeds in } \mathbb{S}} + \underbrace{4096 \times 4 \text{ Bytes}}_{\mathcal{A} \text{ that contains 4096 accumulated scalar gradients}} \\ = & \ 16388 \text{ Bytes}. \end{aligned}$$

After local training, each client return $\mathbb{H}_i$ that contains τ pairs of $(s_j, \widehat{\varrho}_j)$, with τ=200 as described in Section 5.1, U can be calculated as

$$\begin{aligned} U = & \underbrace{200 \times \underbrace{(2 \times 4 \text{ Bytes})}_{\text{each } (s_j, \widehat{\varrho}_j) \text{ pair}}}_{\mathbb{H}_i \text{ that contains 200 pairs when } \tau = 200} \\ = & \ 1600 \text{ Bytes}. \end{aligned}$$

Finally, we can derive the total communication overhead required by each client in each round when applying FedKSeed, i.e., $C = D + U = 17988$ Bytes.

Communication overhead of FedKSeed-Pro. As described in Section 5.1, we set K to different values, i.e., 1,024 for DataJuicer-1.3B and 2,048 for LLaMA-3B. For *FedKSeed-Pro with DataJuicer-1.3B*, D is calculated as

$$\begin{aligned} D = & \underbrace{1 \times 4 \text{ Bytes}}_{\text{one integer seed that encodes the candidate seeds in } \mathbb{S}} + \underbrace{1024 \times 4 \text{ Bytes}}_{\mathcal{A} \text{ that contains 1024 accumulated scalar gradients}} + \underbrace{1024 \times 4 \text{ Bytes}}_{\text{probabilities corresponding to the 1024 seeds}} \\ = & \ 8196 \text{ Bytes}. \end{aligned}$$

Similarly to FedKSeed, in FedKSeed-Pro, each client is also required to upload only the gradient history, such that for each client i, we have

$$\begin{aligned} U = & \underbrace{200 \times \underbrace{(2 \times 4 \text{ Bytes})}_{\text{each } (s_j, \widehat{g}_j) \text{ pair}}}_{\mathbb{H}_i \text{ that contains 200 pairs when } \tau = 200} \\ = & \ 1600 \text{ Bytes}. \end{aligned}$$

Thus, we can derive the total communication overhead required per client per round in *FedKSeed-Pro with DataJuicer-1.3B*, i.e., $C = D + U =$ 9796 Bytes.

For *FedKSeed-Pro with LLaMA-3B*, K is set to 2,048. Thus, we have

$$\begin{aligned} D = & \underbrace{1 \times 4 \text{ Bytes}}_{\text{one integer seed that encodes the candidate seeds in } \mathbb{S}} + \underbrace{2048 \times 4 \text{ Bytes}}_{\mathcal{A} \text{ that contains 2048 accumulated scalar gradients}} + \underbrace{2048 \times 4 \text{ Bytes}}_{\text{probabilities corresponding to the 2048 seeds}} \\ = & \ 16388 \text{ Bytes}. \end{aligned}$$

$$\begin{aligned} U = & \underbrace{200 \times \underbrace{(2 \times 4 \text{ Bytes})}_{\text{each } (s_j, \widehat{g}_j) \text{ pair}}}_{\mathbb{H}_i \text{ that contains 200 pairs when } \tau = 200} \\ = & \ 1600 \text{ Bytes}. \end{aligned}$$

Thus, we have the total communication cost required by *FedKSeed-Pro with LLaMA-3B* for each client per round, i.e., $C = D + U =$ 17988 Bytes.

J. Extended Benefits in Real-world Applications

In this section, we provide discussions on more benefits brought by FedKSeed and FedKSeed-Pro to existing FL systems.

J.1. Alleviating the Burden of Aggregation

Assuming there are m active clients in each round, traditional FL aggregation is conducted on the server with the computation and communication complexity of $\mathcal{O}(md)$, where d is very large when the global model possesses a huge number of parameters, and m is also large when there are many clients such as in cross-device FL (Chen et al., 2023b; Bai et al., 2024). Thus, the FL organizer usually needs to host an FL server with abundant computation and communication resources. In FedKSeed and FedKSeed-Pro, the computation and communication complexity of the server are both reduced to $\mathcal{O}(mK)$. In this case, only a few computational and communication resources are required by the server, such that even a mobile device can handle it. Consequently, the financial burden of FL organizers is greatly alleviated with FedKSeed and FedKSeed-Pro.

J.2. Enabling Possibility to Decentralized Federated Fine-Tuning of LLMs

Due to the transmission delays and unstable connections caused by long-distance transmission, many organizations opt for decentralized FL by allowing some clients to perform aggregation (Qin et al., 2024). However, it can exacerbate the communication costs of FL, as each client might be required to transmit its model parameters to multiple recipients. FedKSeed and FedKSeed-Pro can significantly reduce communication costs, thus bringing the possibility of fine-tuning LLMs with decentralized FL.

J.3. Alleviating the Burden of Saving Checkpoints

The trajectory of LLM fine-tuning does not always proceed in the desired direction. At times, fine-tuning may become trapped in a local optimum or an unfavorable region, or even be jeopardized by malicious attacks. Thus, multiple snapshots of LLMs often need to be stored during the fine-tuning of LLMs. Besides, saving checkpoints may also contribute to snapshot ensemble (Huang et al., 2016). Snapshot saving of LLMs may incur a huge storage overhead. However, with FedKSeed and FedKSeed-Pro, only the snapshots of accumulated scalar gradients need to be preserved for the potential rollback of models, each of which is an array containing $2K$ scalars. Therefore, it significantly reduces the storage consumption for model snapshots.