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) require fine-tuning to improve their responsiveness to natural language instructions. Federated learning (FL) offers a way to perform fine-tuning 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 heights possible with full-parameter tuning. However, the communication overhead associated with full-parameter tuning is prohibitively high for both servers and clients. This work introduces FedKSeed, a novel approach that employs zeroth-order optimization (ZOO) with a set of random seeds. It enables federated full-parameter tuning of billion-sized LLMs directly on devices. Our method significantly reduces transmission requirements between the server and clients to just a few scalar gradients and random seeds, amounting to only a few thousand bytes. Building on this, we develop a strategy to assess the significance of ZOO perturbations for FL, allowing for probability-differentiated seed sampling. This prioritizes perturbations that have a greater impact on model accuracy. Experiments across six scenarios with different LLMs, datasets and data partitions demonstrate that our approach outperforms existing federated LLM fine-tuning methods in terms of both communication efficiency and new task generalization.

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 datasets contribute to LLM tuning (Wang et al., 2022; Wei et al., 2022), the vast quantities of data continuously generated at end devices present an untapped opportunity for further exploitation. Federated learning (FL) (McMahan et al., 2017) offers a way to collaboratively tune LLMs with the distributed data and protected privacy, and has been explored by recent parameter-efficient fine-tuning (PEFT) based works (Zhang et al., 2023a; Babakniya et al., 2023; Zhang et al., 2023c; Che et al., 2023). We observe that PEFT is not a one-size-fits-all solution for tuning LLMs, as it may not always achieve the same level of accuracy as full-parameter tuning (Chen et al., 2022; Pu et al., 2023; Sun et al., 2023). This is particularly evident in FL, where the statistically heterogeneous data distributed across clients can diminish the effectiveness of PEFT (Babakniya et al., 2023; Zhang et al., 2023c). Considering full-parameter tuning's potential for higher accuracy, exploring its feasibility to LLMs in an FL context is promising for both research and development.

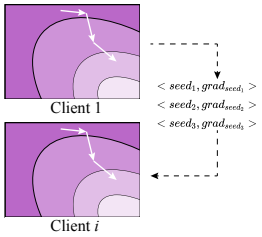
However, full-parameter tuning of billion-sized LLMs on devices with FL is impractical given current technical so-

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_{peft}, M_{full} and M_{infer} are peak memory usage for PEFT with BP, full-parameter tuning with BP and inference, 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.

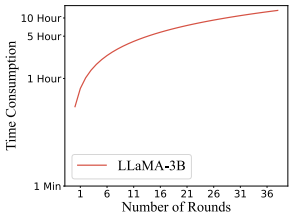
Approach	Acc.↑	Commu.↓	Mem.↓	Comput.↓
PEFT with BP	⋆	$\mathcal{O}(\nu d)$	$\mathcal{O}(\xi d)$	$\mathcal{O}(d)$
Full-param. with BP	⋆⋆	$\mathcal{O}(d)$	$\mathcal{O}(\Xi d)$	$\mathcal{O}(d)$
Full-param. with ZOO	⋆⋆	$\mathcal{O}(d)$	$\mathcal{O}(d)$	$\mathcal{O}(d)$
infinite seed-pool in uplink	⋆⋆	$\mathcal{O}(d)$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

* 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>.

Preprint.



(a) A step of ZOO update can be replicated by a scalar gradient and a random seed to generate a perturbation.



(b) As number of total steps increases, the time required to compute the latest model through update replication increases rapidly.

lutions. As shown in Table 1, backpropagation (BP) and most BP-free methods such as zeroth-order optimization (ZOO) incur communication costs that are proportional to the model sizes. These costs are substantial 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 consumes over 20GB of memory when the length of the context token is set to 400 (Malladi et al., 2023).

Although some recent ZOO-based methods reduce communication costs based on universal random seeds (Zelikman et al., 2023; Feng et al., 2023; Maritan et al., 2023), they compromise other performance factors, making these methods still impractical to use for LLMs. Specifically, as illustrated in Figure 1(a), when two clients utilize the same random number generator, they can produce identical perturbations using the same seed. Consequently, each update step can be summarized with just two values: a seed and the corresponding scalar gradient (since the product of the scalar gradient and the perturbation yields the vector gradient). As outlined in Table 1, current methods exploit this property to either (1) optimize the uplink communication for clients, at the cost of excessive downlink communication to distribute the latest model to clients in each round (Feng et al., 2023; Maritan et al., 2023), or (2) optimize bi-link communication for clients while requiring each client to replicate all update steps from others, leading to a computation cost that increases indefinitely with the number of rounds (Zelikman et al., 2023), as shown in Figure 1(b).

To achieve the best of both worlds, i.e., avoiding the massive communication overhead associated with transmitting full model parameters while limiting the ever-increasing computational cost of updating to the latest model, this work introduces a novel federated full-parameter tuning approach based on ZOO with K seeds, which we denote as FedKSeed. FedKSeed utilizes a theoretically informed paradigm of seed reuse, demonstrating that ZOO-based FL can be effectively implemented with a finite set of seeds to generate perturbations. As a result, this approach enables full-parameter tuning of LLMs in FL with a communication cost of less than 18 kilobytes per round and the same memory usage as that required for inference. Building on FedKSeed, we further introduce a strategy to assess the significance of each perturbation, assigning varied sampling probabilities to different seed candidates. By refining the seed pool in this manner, we can expedite the process of syncing with the latest model state, thereby further enhancing both computational efficiency and model accuracy.

The contributions of this work are summarized as follows:

- We propose a novel ZOO-based FL approach for LLM tuning, FedKSeed, which transmits only K seeds and their 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 perturbations in ZOO, and propose a simple yet effective strategy that selects seeds with non-uniform probabilities. This reduces the number of seeds needed by FedKSeed, thereby accelerating the synchronization with the latest model and improving accuracy.
- Experiments on 6 scenarios with different 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 score over the best-performing baseline and reduces communication costs by a factor of more than a thousand. Our codes are publicly available at `https://github.com/alibaba/FederatedScope/tree/FedKSeed`.

2. Related Work

Federated Fine-Tuning for LLMs. With the popularity of LLMs, there are some researchers have started to explore fine-tuning LLMs with PEFT techniques based on FL. Zhang et al. (2023c) provide investigations and a benchmark for PEFT techniques in FL and reveal that PEFT techniques can help to provide privacy protection. Among existing PEFT techniques, LoRA (Hu et al., 2022) is usually preferable. For example, Zhang et al. (2023a) proposes a federated instruction tuning approach based on LoRA, and Jiang et al. (2023) design a low-parameter FL approach based on LoRA for text sentiment classification. Babakniya et al. (2023) experimentally demonstrate that when facing FL with non-IID data, LoRA is not as good as full-parameter tuning in terms of accuracy on classification tasks, and thus a strategic initialization based on SVD decomposition for LoRA weights is proposed to salvage the performance of LoRA in FL.

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) provide plat-

forms to the real-world deployment of federated LLM tuning. The computational bottlenecks have been thoroughly investigated by Woisetschläger et al. (2023).

Federated Learning with Zeroth-Order Optimization. There are some researches focusing on leveraging ZOO to solve problems where gradients are difficult to obtain, such as black-box model optimization (Li & Chen, 2021). Fang et al. (2022) analyze the convergence of ZOO-based FL, and the analysis is extended to provide the generalization error bound by Chen et al. (2023e). Shu et al. (2023) propose to boost the query efficiency of ZOO in FL with optimization trajectory. 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 1(b), or (3) the reliance on BP which consumes a substantial amount of memory (Rahimi et al., 2023).

Difference between FedKSeed and Related Works. We notice a recent work FwdLLM (Xu et al., 2023) conducts FL based on both PEFT and ZOO, but with the goal and leveraged techniques different to FedKSeed. FwdLLM utilizes post-training quantization to reduce the memory cost of LLM fine-tuning, while we mainly focus on communication cost and enable full-parameter tuning of LLMs with FL. For example, FwdLLM requires a total of several hundred GB of communication cost to tune an LLM with only about 300 million parameters, since only uplink communication at the client-side is optimized.

In a nutshell, these existing works mainly focus on tuning partial parameters of LLMs with FL. In contrast, we focus on enabling full-parameter tuning of LLMs with FL, which can obtain higher accuracy as shown in Table 2. Moreover, thanks to the removal of model parameter transmission and BP processes, FedKSeed significantly reduces the communication and memory cost compared to existing baselines tailored for federated LLM tuning, as summarized in Table 3. We provide more detailed technical comparisons between FedKSeed and existing works in Appendix A.

3. Problem Formulation

Consider an FL system with N clients to collaboratively solve the federated optimization problem defined as

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

where $\mathcal{L}_i$ denotes the loss evaluated at the shared global model $\mathbf{w}\in\mathbb{R}^d$ with a data instance $\mathbf{x}$ sampled from local dataset $\mathcal{D}_i$ possessed by the i-th client, c_i is the weight of the i-th client that follows $\sum_{i=1}^{N} c_i = 1$. For mini-batch gradient descent, $\mathbf{x}$ can be replaced with a batch of data. Here we utilize $\mathbf{x}$ because we set the batch size to 1 following Malladi et al. (2023) to avoid too much memory cost. To solve Equation 1, BP-based FL requires each client i to perform several steps of gradient descent algorithms to train the local model $\mathbf{w}_i$, as

$$\mathbf{w}_{i,t+1} = \mathbf{w}_{i,t} - \nabla_{\mathbf{w}_{i,t}}\mathcal{L}_i(\mathbf{w}_{i,t};\mathbf{x}), \tag{2}$$

where $\mathbf{w}_{i,t}$ is the local model of client i at local step t. Here, we refer to $\mathbf{g}_{i,t} \triangleq \nabla_{\mathbf{w}_{i,t}}\mathcal{L}_i(\mathbf{w}_{i,t};\mathbf{x}_i)$ as the gradient used by client i to update the local model at step t.

The main difference between ZOO-based FL and BP-based FL lies in how they obtain the gradient of local models during local training. The gradient cannot be directly calculated as Equation 2 without BP, as an alternative, ZOO-based FL estimates the gradient by two forward propagations. In this work, we adhere to the ZOO paradigm with a two-point gradient estimator proposed by Malladi et al. (2023), as

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

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. For clarity, we term $\widehat{g} \triangleq \frac{\mathcal{L}(\mathbf{w}+\epsilon\mathbf{z};\mathbf{x})-\mathcal{L}(\mathbf{w}-\epsilon\mathbf{z};\mathbf{x})}{2\epsilon}$ as *scalar gradient*, where $\widehat{\mathbf{g}} = \widehat{g}\cdot\mathbf{z}$. When $\widehat{\mathbf{g}}_{i,t}$ is estimated, client i updates its local model as Equation 2. After several steps of local training with gradient estimation, clients submit their local models to the server for further model aggregation and redistribution, which is the same as that in vanilla FL (McMahan et al., 2017).

4. The proposed FedKSeed

4.1. Overview

FedKSeed is designed for federated full-parameter tuning of billion-sized LLMs with the following goals: (1) to avoid the massive communication overhead for transmitting full-model parameters, and (2) to avoid the tremendous memory consumption caused by BP with full-model parameters. Thus, we design FedKSeed based on ZOO, and propose a theoretically-informed paradigm that enables seed reuse. In FedKSeed, scalar gradients corresponding to the same seeds are accumulated, in order to limit the ever-increasing computational cost of clients to catch up to the latest model.

Figure 1 outlines FedKSeed, where the underlined components and processes are only required by an enhanced version of it, i.e., FedKSeed-Pro, which will be described

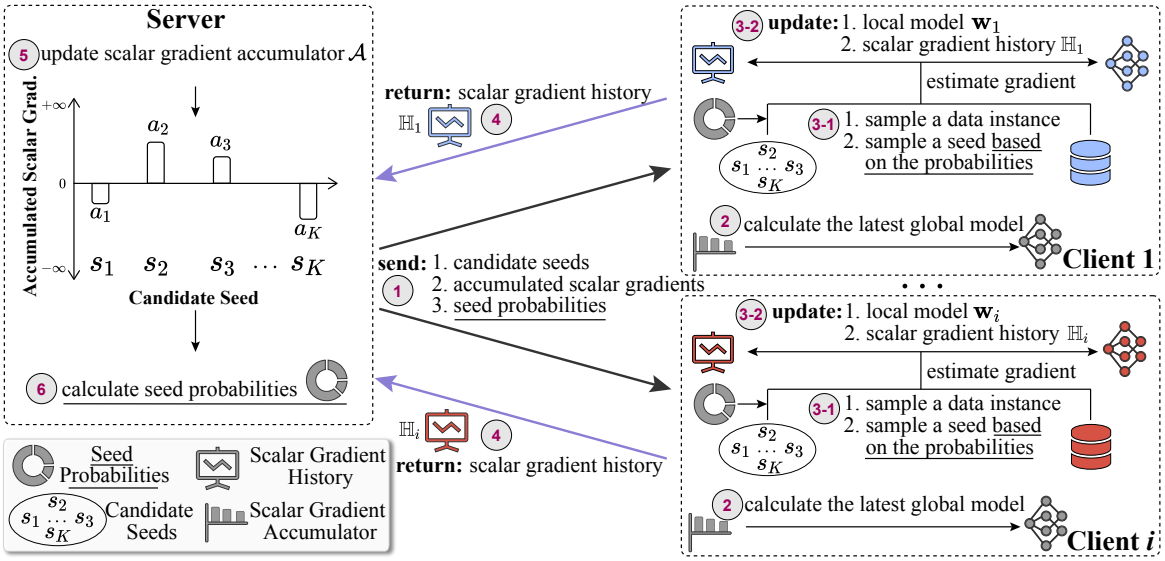


Figure 1. Overview of our approach, where the serial numbers indicate FL processes in each communication round. Components with the same colors share the same value. The underlined components and processes are only required by an enhanced version, FedKSeed-Pro.

later in Section 4.3. The server maintains (1) K candidate seeds $\mathbb{S}$ and (2) a scalar gradient accumulator $\mathcal{A}$ that records the summation of received scalar gradients for each candidate seed. Note that there are no model parameters maintained on the server, and we assume that each client has been equipped with a pre-trained LLM $\mathbf{w}^0$.

At the start of each round, the server sends $\mathbb{S}$ and $\mathcal{A}$ to the clients active in this round (process ①). Then, each client i calculates the latest global model based on the gradient accumulator (process ②) and regards it as the local model $\mathbf{w}_i$ Process ③ is a loop of local training, where in each step, the client randomly samples a seed s_j from $\mathbb{S}$ and a data instance, then calculates the scalar gradient $\widehat{g}_j$. Next, $\mathbf{w}_i$ is updated based on $\widehat{g}_j$ and s_j, and $\langle s_j, \widehat{g}_j \rangle$ is staged to the scalar gradient history $\mathbb{H}_i$. After several steps of local update, each client sends the scalar gradient history to the server (process ④). The server updates the gradient accumulator based on all received scalar gradient histories (process ⑤). The above processes are summarized in Algorithm 1.

4.2. Federated Full-Parameter Tuning by Limited Seeds

In this section, we present how to enable federated full tuning through a limited number of candidate seeds.

Recall Equation (3), if all clients utilize the same pseudo number generator, the perturbation can be encoded into one integer, i.e., a random seed. Thus, t steps of update can be replicated with the scalar gradient history $\mathbb{H} = \{\langle s_j, \widehat{g}_j \rangle\}^t$ containing t pairs of seeds and corresponding scalar gradients. From the above, an intuitive solution to alleviate model parameter transmitting is to have the server track the scalar gradient history of all clients (Zelikman et al., 2023). Assuming m clients participate in FL in each round, and each client conducts τ steps of update during local training averagely. After r rounds, a client has to perform $\tau \cdot r \cdot m$ steps of model updating to get the latest global model with $\mathbf{w}^0$ as the initial point. From Figure 1(b), when $m = 50$, $\tau = 200$, and $r = 30$, such an operation takes over 10 hours, which is unaffordable.

Limiting the number of seeds. Actually, if seeds are reused, the update steps for getting the latest model can be grouped and then merged. If we select only K candidate seeds and accumulate the scalar gradients corresponding to the same seed, *each client needs to perform at most K iterations to get the latest model.*

Let $\mathbb{S} = \{s_1, s_2, \ldots, s_K\}$ denote a set of K candidate seeds randomly selected by the server at the start of federated fine-tuning. The server maintains a scalar gradient accumulator $\mathcal{A} = \{a_1, \ldots, a_K\} \in \mathbb{R}^K$, where $a_j = \sum_{\widehat{g} \in \mathcal{G}_j} \widehat{g}$ accumulates all the scalar gradient $\mathcal{G}_j = \{\widehat{g}_{j,1}, \ldots\}$ that have been obtained on perturbation $\mathbf{z}_j$. Each client i gets the latest global model $\mathbf{w}$ by downloading $\mathcal{A}$ from the server and conducting

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

Then, the latest global model $\mathbf{w}$ is initialized 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 $\widehat{g}_j$

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

Algorithm 1 FedKSeed. The underlined components and processes are only needed by FedKSeed-Pro.

Input: N, K, $\mathbf{w}^0$, η, $\{c_1, \ldots, c_N\}$.

Server executes:

initialize K candidate seeds $\mathbb{S}$, scalar gradient accumulator $\mathcal{A}$, and their probabilities $\mathbf{p}$.
for each round $r = 0, 1, \ldots$ **do**
  for each client $i \in$ activate clients $\mathbb{C}$ **in parallel do**
    $\mathbb{H}_i \leftarrow$ ClientTraining($\mathbb{S}$, $\mathcal{A}$, $\mathbf{p}$, i) \\ ① in Figure 1
    for $\langle s_j, \widehat{g}_j \rangle \in \mathbb{H}_i$ **do**
      $a_j = a_j + c_i \cdot \widehat{g}_j$ \\ ⑤ in Figure 1
    end for
  end for
  compute the seed importance as Equation (13), then the probability $\mathbf{p}$ as Equation (14) \\ ⑥ in Figure 1
end for

ClientTraining($\mathbb{S}$, $\mathcal{A}$, $\mathbf{p}$, i):

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

UpdateModel($\mathbf{w}$, s, $\widehat{g}$):

sample perturbation $\mathbf{z} \in \mathbb{R}^d$ based on random seed s
return $\mathbf{w} - \eta \cdot \widehat{g} \cdot \mathbf{z}$

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

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

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

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

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

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 Analysis

The rationale behind FedKSeed is that the diversity of perturbations is not relevant to its convergence. Therefore, we can sample the seeds used for generating perturbations from a limited range, i.e., predefined K candidates.

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

$$\min_{r \in \{1,2,\ldots,T\}} \mathbb{E} \left\| \nabla \mathcal{L}(\mathbf{w}^r) \right\|^2 \leq \mathcal{O}\left(\sqrt{\frac{d}{\tau N T}} \right), \tag{8}$$

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

The above-mentioned four assumptions made by Fang et al. (2022) are left in Appendix C.

Proof. Consider a one-point estimator that calculates $\widehat{\mathbf{g}}$ during the local training of the i-th client by

$$\widehat{\mathbf{g}}_{i,t} = \frac{1}{b_1 b_2} \sum_{b=1}^{b_1} \sum_{j=1}^{b_2} \frac{\mathbf{z}_{b,j} \left[\mathcal{L}(\mathbf{w}_{i,t} + \epsilon \mathbf{z}_{b,j}; \mathbf{x}_b) - \mathcal{L}(\mathbf{w}_{i,t}; \mathbf{x}_b) \right]}{\epsilon}, \tag{9}$$

where b_1 is the batch size, b_2 is the number of perturbations to estimate the gradient of $\mathbf{w}_i$ for each data instance, and $\mathbf{z}$ is i.i.d. random perturbation with $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ distribution.

Lemma 1. *(Convergence of ZOO-based FL with one-point estimator (Fang et al., 2022).) With the same assumptions relied upon by Theorem 1, ZOO-based FL with one-point estimator satisfies*

$$\min_{r \in \{1,2,\ldots,T\}} \mathbb{E} \left\| \nabla \mathcal{L}(\mathbf{w}^r) \right\|^2 \leq \mathcal{O}\left(\sqrt{\frac{d}{\tau N T b_1 b_2}} \right). \tag{10}$$

The one-point estimator in Equation (9) is equivalent to the two-point estimator proposed by Malladi et al. (2023) in terms of convergence, since the gradient estimated by the two-point estimator is equivalent to the average of the gradients estimated by one-point estimator with two opposing perturbations, as

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

When b_1 and b_2 are both 1, we can have the convergence of FedKSeed in Theorem 1 based on Lemma 1. □

In FedKSeed, we adopt the memory-efficient two-point gradient estimator proposed by Malladi et al. (2023), since a two-point gradient estimator has a lower variance compared

to the one-point paradigm (Liu et al., 2018). From Theorem 1, we have a conclusion that *the diversity of perturbations has no negative impact on the convergence*. Thus, we could enable the reuse of random seeds.

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

Let $\mathbb{G} = \left[\sum_{\widehat{g}\in\mathcal{G}_1} \widehat{g}, \ldots, \sum_{\widehat{g}\in\mathcal{G}_K} \widehat{g}\right]$, where $\sum_{\widehat{g}\in\mathcal{G}_j} \widehat{g}$ means the summation of all the scalar gradients corresponding to the perturbation generated with s_j. The FL 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

$$\min_{\mathbb{G}} \left\|\mathbf{w}^0 - \eta \cdot [\mathbf{z}_1, \ldots, \mathbf{z}_k]\mathbb{G}^\top - \mathbf{w}^*\right\|. \tag{11}$$

Note that this definition is merely to facilitate understanding FL from another perspective, and it cannot be directly solved since $\mathbf{w}^*$ is unknown. From Equation (11), FL processes can be regarded 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 not sufficient 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 higher difficulty in finding the optimal solution. High-dimensional vectors sampled from a Gaussian distribution are typically orthogonal. Considering the dimension d of $\mathbf{w}$ is 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 K is, the better Equation (11) could be solved. Taking an extreme example, if $K = 1$, Equation (11) may be fundamentally unoptimizable. Based on these analyses, $\overrightarrow{K}$ theoretically exists so that Principle 1 holds. We note that 1,024 can be used as an empirically estimated value of it, as shown in our experiments (Figure 2).

Principle 2. *(Seed Excessiveness.) There exists a threshold $\overleftarrow{K}$ such that given the total number of local training steps fixed, when $K \geq \overleftarrow{K}$, the accuracy of the model decreases with the increase of K.*

Recall Equation (9) and Lemma 1, although the subscript of $\mathbf{z}$ is the combination of $\{b, j\}$, the random generation of $\mathbf{z}$ is actually independent to data instance $\mathbf{x}_b$, i.e., each $\mathbf{z}_{b,j}$ is independently randomly generated with $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. Without introducing system error, Equation (9) can be rewrite as

$$\widehat{\mathbf{g}} = \frac{1}{b_2 b_1} \sum_{j=1}^{b_2} \sum_{b=1}^{b_1} \frac{\mathbf{z}_j[\mathcal{L}(\mathbf{w}_i + \epsilon\mathbf{z}_j; \mathbf{x}_b) - \mathcal{L}(\mathbf{w}_i; \mathbf{x}_b)]}{\epsilon}. \tag{12}$$

In this new formulation, the FL process can be regarded as: for each perturbation $\mathbf{z}$, determining the step size for the model to advance in the direction of $\mathbf{z}$. Under the condition that the total number of update steps is fixed at τ, each seed is averaged to be sampled for τ/K times. 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 number of seeds 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, $\overleftarrow{K}$ theoretically exists so that Principle 2 holds. From Figure 2, an empirically estimated value of it is 4096.

Based on Principle 1 and 2, we can conclude that it is better to set K in the range of $[\overleftarrow{K}, \overrightarrow{K}]$. Unfortunately, it is hard to theoretically bridge the relations between a specific LLM and precise values of $\overleftarrow{K}$ and $\overrightarrow{K}$. In Section 5.2, we experimentally demonstrate that for models with 1B or 3B parameters, K can be set to several thousand so that FedKSeed and FedKSeed-Pro can perform well.

4.3. Sampling Seeds with Non-uniform Probabilities

This section discusses how to enhance FedKSeed with non-uniform probabilities for seed sampling, in order to further reduce K and boost the model accuracy, as experimentally demonstrated in Table 2. We term FedKSeed with the non-uniform seed **pro**babilities as FedKSeed-Pro.

Recall Equation (11), assuming an ideal perturbation $\mathbf{z}^*$ that there exists a scalar κ such that $\mathbf{z}^* = \kappa(\mathbf{w}^0 - \mathbf{w}^*)$. With several iterations along $\mathbf{z}^*$, Equation (11) can be optimized to be very close to zero. Assuming another perturbation $\mathbf{z}_{\text{bad}}$ that $\mathbf{z}_{\text{bad}} \cdot (\mathbf{w}^0 - \mathbf{w}^*) = 0$, the steps iterated along $\mathbf{z}_{\text{bad}}$ contribute minimally to optimize Equation (11). Thus, different perturbations contribute differently to the convergence. For a perturbation $\mathbf{z}_j$ that is more important to convergence, it is better to determine the step size ($\sum_{\widehat{g}\in\mathcal{G}_j} \widehat{g}$) taken in the direction of perturbation $\mathbf{z}_j$ more precisely with more data instances, i.e., larger $|\mathcal{G}_j|$.

The scalar gradient, as the directional derivative of $\mathcal{L}$ along $\mathbf{z}$, is influenced on one hand by the model and the data instances, and on the other hand by the similarity between true gradient $\mathbf{g}$ and $\mathbf{z}$. Considering the model and data instances are equivalent in expectation for all perturbations, the average amplitude of scalar gradient ψ_j can characterize the importance of $\mathbf{z}_j$, as

$$\psi_j = \frac{1}{|\mathcal{G}_j|} \sum_{\widehat{g}\in\mathcal{G}_j} |\widehat{g}|. \tag{13}$$

To avoid too large probability differences between these seeds, $\Psi = \{\psi_1, \ldots, \psi_K\}$ undergoes min-max normalization. Thus, the server computes the probabilities p_j of can-

didate seed s_j based on Ψ, as

$$p_j = \frac{\exp(\psi_j)}{\sum_{k=1}^{K} \exp(\psi_k)}. \tag{14}$$

The calculated probabilities are sent to participants in each round to guide the sampling of seeds during local training. We experimentally demonstrate in Section 5.2 that when significant seeds are sampled with higher probabilities, we can reduce the number of seeds required without decreasing the model accuracy, in fact, it even increases the accuracy.

5. Experiments

In this section, we aim to empirically answer the following questions: (i) Section 5.2: Can FedKSeed and FedKSeed-Pro achieve better accuracy than existing methods for federated LLM tuning? (ii) Section 5.3: What is the communication and memory overhead incurred by FedKSeed and FedKSeed-Pro? (iii) Section 5.4: How do the hyperparameters of our approach impact its performance?

5.1. Experimental Setup

5.1.1. Baselines

To evaluate the effectiveness of FedKSeed and FedKSeed-Pro, we compare them with 4 competitive approaches tailored for federated LLM fine-tuning, including: (1) FedP-Tuning: An LLM fine-tuning baseline used in Kuang et al. (2023) that is tailored for FL, where P-Tuning (Liu et al., 2023) is leveraged as the PEFT technique, with SGD as the optimizer; (2) FedPrompt: Another LLM fine-tuning baseline used in Kuang et al. (2023), where Prompt Tuning (Lester et al., 2021) is leveraged as the PEFT technique, with SGD as the optimizer; (3) FedIT: A federated instruction tuning approach proposed by Zhang et al. (2023a), where LoRA is adopted as the PEFT technique, with Adam (Kingma & Ba, 2015) as the optimizer; and (4) FedIT-SGD: A variation of FedIT that replaces Adam with SGD.

5.1.2. Datasets & Evaluation Metrics

We conduct experiments on two datasets to evaluate these approaches, i.e., Natural Instructions (NI) (Mishra et al., 2022; Wang et al., 2022) and Dolly-15K (Conover et al., 2023). NI provides a large collection of tasks and their natural language instructions. We utilize the dataset versioned by `v2.8` and adopt its `default` split, where there are 756 tasks for training and 119 tasks for testing, each with a task definition. Dolly-15K provides 15,015 data instances within 8 tasks. The prompt template is directly adopted from Alpaca (Taori et al., 2023). The maximum token length is set to 1,024 while ignoring excessively long data instances.

Following the methodology of Dettmers et al. (2023), we utilize Rouge-L (Lin, 2004) for the evaluation of LLMs that have been tuned using these approaches. Note that the trend of Rouge-L on classification tasks has been proved to correlate well with that of accuracy (Wang et al., 2022). To examine the performance of these approaches across models of different sizes, we take DataJuicer-1.3B (Chen et al., 2023c) and LLaMA-3B (Touvron et al., 2023) as the foundation models.

5.1.3. Federated Learning Settings

Considering the NI dataset is very large, we randomly sample 20% and 2% data instances for each training task and each test task, respectively. Each training task with no less than 20 data instances is treated as a unique client, forming an FL system with 738 clients, while the test tasks are kept on the server for evaluation. Through this, we build a non-IID scenario with feature skew (Tan et al., 2022). For Dolly-15K, we set the last task as the test task and the others as training tasks. Data instances in training tasks are split to 200 clients via Dirichlet distribution with both $\alpha = 0.5$ and $\alpha = 5.0$ to build non-IID scenarios with varying degrees of label distribution skew (Chen et al., 2023d).

For all evaluated approaches, we randomly sample 5% of the clients to participate in the aggregation process for each round. The total number of communication rounds is set as 40 for NI and 60 for Dolly-15K, based on empirical evidence indicating that all approaches can converge within these rounds. The aggregation weight of each participating client is proportional to the number of its local data instances.

5.1.4. Implementations

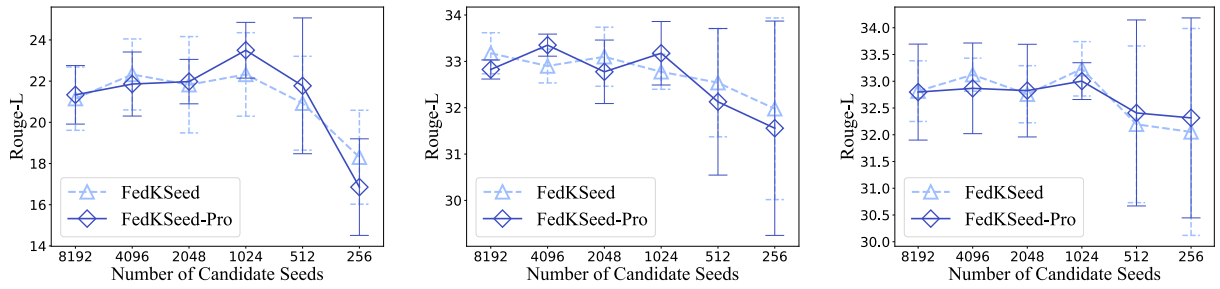
We implement these approaches by PyTorch `2.0.1` (Paszke et al., 2019) with `PEFT` (Mangrulkar et al., 2022) and `Transformers` (Wolf et al., 2020). Experiments with DataJuicer-1.3B and LLaMA-3B are conducted with an NVIDIA RTX 3090 GPU and an NVIDIA A100 GPU, respectively, with models loaded in 16-bit floating numbers.

Following Kuang et al. (2023), all approaches conduct local training with batch size at 1 to alleviate memory consumption. BP-based baselines conduct local training for one epoch, and FedKSeed and FedKSeed-Pro conduct local training for 200 steps, so that all clients perform an equal number of steps of random search. Following Kuang et al. (2023), BP-based approaches conduct local training with learning rate η of $3e^{-4}$, the number of virtual tokens in FedPTuning is set to 20, and the `rank` and `alpha` of LoRA for FedIT and FedIT-SGD are set to 8 and 16, respectively (Zhang et al., 2023a). Following Malladi et al. (2023), η and ϵ of FedKSeed and FedKSeed-Pro are set to $3e^{-7}$ and $5e^{-4}$, respectively, unless stated otherwise.

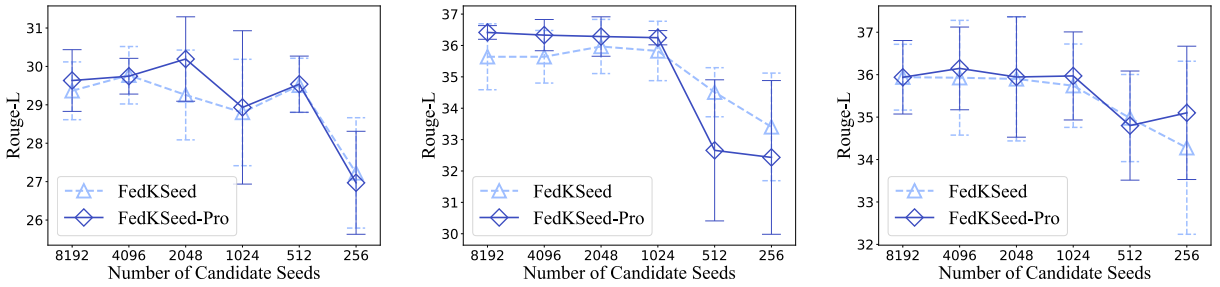
To have a fair comparison, unless stated otherwise, we uni-

Table 2. Rouge-L (%) comparisons on widely-adopted LLMs and datasets. Each cell presents the average Rouge-L at the last round of four runs with different random seeds. Bold and underlined numbers indicate the best and second-best average Rouge-L, 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
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	<u>22.33 ± 1.72</u>	<u>29.77 ± 0.75</u>	<u>32.90 ± 0.37</u>	<u>35.64 ± 0.83</u>	**33.12 ± 0.31**	<u>35.93 ± 1.35</u>
FedKSeed-Pro	**23.50 ± 1.35**	**30.19 ± 1.10**	**33.18 ± 0.68**	**36.29 ± 0.63**	<u>33.00 ± 0.34</u>	**35.95 ± 1.41**



(a) DataJuicer-1B Instruct (c) DataJuicer-1B Dolly-15K ($\alpha = 0.5$) (e) LLaMA-1B Dolly-15K ($\alpha = 5.0$)



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

Figure 2. Performance of FedKSeed and FedKSeed-Pro with different number of candidate seeds.

formly set $K = 4096$ on these datasets for FedKSeed, and we uniformly set $K = 1024$ for DataJuicer-1.3B and $K = 2048$ for LLaMA-3B on these datasets for FedKSeed-Pro. Candidate seeds are integers uniformly randomly sampled from $[0, 1e^{11})$. Note that from Figure 2, these settings are not tailored for these LLMs and datasets, and thus the performances of FedKSeed and FedKSeed-Pro reported in Table 2 are not their best values in corresponding scenarios.

5.2. Comparisons on Accuracy Performance

Overall results. Table 2 provides the Rouge-L scores of these approaches. It can be observed that FedKSeed and FedKSeed-Pro achieve the top two performances across all six scenarios. In particular, on Dolly-15K ($\alpha = 0.5$) with LLaMA-3B, FedKSeed-Pro outperforms the best 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 shown in Table 3. Furthermore, we observe that the gains achieved by FedKSeed-Pro over the best baseline, FedIT, are generally larger with LLaMA-3B compared to DataJuicer-1B. This is because, under 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 FedKSeed and FedKSeed-Pro.

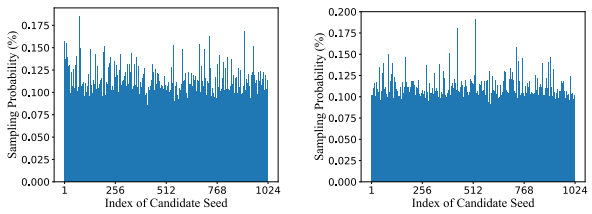
Effect of K. To validate the assertions made in Principle 1 and Principle 2, and to understand the relationship between the number of perturbation seeds (K) and the accuracy of FedKSeed and FedKSeed-Pro, we examine their performance with varying K values across six scenarios, as

Table 3. Numbers of trainable parameters, communication cost (per round), GPU memory cost and training time (per step) of approaches.

Approach	DataJuicer-1B				LLaMA-3B			
	Trainable Para.	Commun.	Memory	Time (s)	Trainable Para.	Commun.	Memory	Time (s)
FedPTuning	12.63 M	96.36 MB	11.89 GB	0.0582	30.79 M	234.94 MB	16.31 GB	0.0690
FedPrompt	40.96 K	320.0 KB	11.82 GB	0.0589	64.00 K	500.0 KB	19.01 GB	0.0683
FedIT-SGD	1.57 M	12.00 MB	12.41 GB	0.0746	2.66 M	20.31 MB	18.17 GB	0.0841
FedIT	1.57 M	12.00 MB	12.43 GB	0.0756	2.66 M	20.31 MB	18.33 GB	0.0848
FedKSeed	1.35 B	17,988 Bytes	3.52 GB	0.1479	3.43 B	17,988 Bytes	7.78 GB	0.3451
FedKSeed-Pro	1.35 B	9,796 Bytes	3.52 GB	0.1461	3.43 B	17,988 Bytes	7.78 GB	0.3448

depicted in Figure 2.

We observe that when the value of K exceeds the recommended range specified in Section 5.1.4, the accuracy does not improve and may occasionally decline. This is because the total number of optimization steps is constant, with more seeds, the likelihood that each seed processes a sufficient number of data instances to effectively determine its step size is reduced. Conversely, with too few seeds, the performance of both FedKSeed and FedKSeed-Pro deteriorates due to the limited expressiveness resulting from an insufficient number of perturbations. To conclude, the value of K should be balanced as discussed in Section 4.2: 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.



(a) Natural Instructions (b) Dolly-15K ($\alpha = 0.5$)

Figure 3. Probabilities of candidate seeds calculated by FedKSeed-Pro after the last round with LLaMA-3B.

Effect of the seed sampling. Compared to FedKSeed, the proposed approach with seed sampling, FedKSeed-Pro, gains superior performance in five out of six scenarios and comparable results on the Dolly-15K when α is set to 5.0. This highlights the effectiveness of the non-uniform seed sampling strategy proposed in Section 4.3. To demonstrate the variability among seed importance, we present the calculated seed probabilities in two scenarios in Figure 3. It can be observed that when non-uniform sampling is applied, the probabilities of each seed being sampled exhibit differences of several multiples. Combining the experimental results presented in Table 2, where FedKSeed-Pro achieves higher accuracy, 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.

5.3. Comparisons on Overheads

Table 3 presents the numbers of trainable parameters and overheads of these approaches with different LLMs on the NI dataset, where the memory consumption indicates the peak GPU memory cost across the entire running of these approaches. The comparison demonstrates that both FedKSeed and FedKSeed-Pro can achieve the highest number of trainable parameters with the least communication and memory overheads at the same time. It is worth noting that since the K seeds used in our approach are randomly selected, they can be encoded into one single seed, which results in a minimal transmission cost of just 4 Bytes. The communication efficiency obtained by FedKSeed and FedKSeed-Pro is attributed to the removal of transmitting all trainable parameters, and the memory efficiency is attributed to the removal of BP and the adoption of the in-place ZOO (Malladi et al., 2023). 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 the constraints on memory and communication. This is because the development of computing power has outpaced that of memory and communication resources. Therefore, FedKSeed and FedKSeed-Pro can be effectively applied to tune full LLMs on end devices with limited communication and memory resources.

Besides, as shown in Figure 4, FedKSeed-Pro outperforms FedKSeed not only in accuracy but also in the efficiency of synchronizing the latest model. This is because FedKSeed-Pro achieves higher accuracy with only 1,024 seeds, while FedKSeed requires a larger number of seeds, which leads to a longer computation time for calculating the latest model.

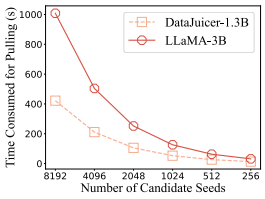


Figure 4. Time consumed to calculate the latest model.

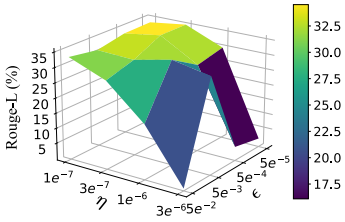


Figure 5. FedKSeed-Pro on Dolly (α=0.5) with various η and ϵ.

5.4. Hyper-parameter Sensitivity

Given that in-place Zeroth-Order Optimization (ZOO) (Malladi et al., 2023) serves as the foundational technique for local training in our approaches, we examine the impact of key hyperparameters in ZOO optimization—specifically, the learning rate (η) and the perturbation scale (ϵ)—on the performance of FedKSeed-Pro. This examination is illustrated in Figure 5 using an example from the Dolly-15K dataset ($\alpha = 0.5$) with the LLaMA-3B model. As observed in Figure 5, both η and ϵ should not be set excessively high. The perturbation scale ϵ determines the magnitude of perturbations applied during gradient estimation; theoretically, a smaller ϵ can lead to a more accurate gradient approximation. However, setting ϵ too low may result in numerical underflow, especially when using half-precision floating-point numbers. Regarding the learning rate η, an overly large value can result in update steps that are too aggressive, potentially causing the model to deviate significantly from the optimal point or even to diverge.

6. Conclusion

Existing federated fine-tuning approaches for LLMs usually rely on PEFT techniques. Considering PEFT still falls short in some 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 communication costs lower than 18 kilobytes in each round. Based on FedKSeed, motivated by the fact that the scalar gradient of a perturbation is the directional derivative of the true gradient, we propose a strategy for quantifying the importance of seeds and granting differentiated probabilities to them, which successfully reduces the number of required seeds, thus speeding up the obtaining of the latest model while achieving higher accuracy compared to FedKSeed. Extensive experiments conducted on two widely adopted instruction-tuning datasets demonstrate the superior performance of our proposed approaches surpasses baselines tailored for federated LLM tuning with respect to accuracy, communication cost and memory footprint.

Our work also raises some new potential research directions, such as building decentralized federated tuning techniques as the communication overhead is a more critical factor to be considered in this context.

References

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.

Che, T., Liu, J., Zhou, Y., Ren, J., Zhou, J., Sheng, V. S., Dai, H., and Dou, D. Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization. *arXiv preprint arXiv:2310.15080*, 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., 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, 2023c.

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, 2023d.

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*, 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*, 2023e.

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. *arXiv preprint arXiv:2305.14314*, 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 *International Conference on Learning Representations, ICLR*, 2022.

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

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, pp. 1–15, 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. *arXiv preprint arXiv:2104.08691*, 2021.

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.

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. *CoRR*, abs/2305.17333, 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. *arXiv preprint arXiv:2309.17174*, 2023.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Mishra, S., Khashabi, D., Baral, C., and Hajishirzi, H. Cross-task generalization via natural language crowdsourcing instructions. In *ACL*, 2022.

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. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2023.

Qin, Z., Yan, X., Zhou, M., Zhao, P., and Deng, S. Blockdfl: A blockchain-based fully decentralized federated learning framework, 2023.

Rahimi, M. M., Bhatti, H. I., Park, Y., Kousar, H., and Moon, J. EvoFed: Leveraging evolutionary strategies for communication-efficient federated learning. *arXiv preprint arXiv:2311.07485*, 2023.

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.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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

Xu, M., Wu, Y., Cai, D., Li, X., and Wang, S. Federated fine-tuning of billion-sized language models across mobile devices. *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. *arXiv preprint arXiv:2306.10015*, 2023.

Zhang, J., Vahidian, S., Kuo, M., Li, C., Zhang, R., Wang, G., and Chen, Y. Towards building the federated GPT: Federated instruction tuning. *arXiv preprint arXiv:2305.05644*, 2023a.

Zhang, K., Tao, G., Xu, Q., Cheng, S., An, S., Liu, Y., Feng, S., Shen, G., Chen, P., Ma, S., and Zhang, X. FLIP: A provable defense framework for backdoor mitigation in federated learning. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023b.

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, 2023c.

A. Detailed Technical Comparisons

In this section, we provide detailed technical distinctions of our approach from existing ZOO-based FL and discuss why these approaches are not suitable for full-parameter tuning of LLMs with FL.

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., 2023). 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., 2023) communication cost.

FwdLLM (Xu et al., 2023), BAFFLE (Feng et al., 2023) and FedZeN (Maritan et al., 2023) do not limit the number of candidate seeds for communication compression. Thus, 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 LLM in each round, thus, it prevents 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).

For the approach achieves bidirectional $\mathcal{O}(1)$ communication cost proposed by Zelikman et al. (2023), it optimizes communication efficiency to the utmost. However, there is no such thing as a free lunch. As we have discussed in Figure 1(b), since Zelikman et al. (2023) sample seeds from almost infinite space, each client must replicate the update steps performed by all other clients to obtain the latest model. Therefore, as the rounds of FL continue, the overhead of calculating the latest model from $\mathbf{w}^0$ also grows indefinitely, quickly reaching a level that is unsustainable for end devices. Since the fine-tuning of large models typically requires running many steps on 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 in a total of only 16,000 steps, while on a complex dataset, e.g., Natural Instructions (Wang et al., 2022), such several steps of updates are not sufficient to the convergence of LLMs.

One recent work, EvoFed (Rahimi et al., 2023), also achieves bidirectional $\mathcal{O}(1)$ communication cost. However, EvoFed is fundamentally different from FedKSeed, and it is not designed for LLM fine-tuning: (1) During local training, EvoFed first conducts the BP process to get 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, where the weight coefficient of each population is determined by the l_2 norm of the pair-wise differences between it and the true gradient. Thus, EvoFed still relies on the BP process to get the true gradient, which limits its applicability to billion-sized LLMs. As reported by Malladi et al. (2023), fine-tuning of full LLMs requires tremendous memory overhead, e.g., full-parameter tuning of an LLM with 2.7B parameters with the average number of tokens set to 400 causes 55GB memory consumption. Such a level of memory consumption is unaffordable to a single graphics card such as NVIDIA V100 (32GB), not to mention an end device. (2) The calculation of l_2 norm of the pair-wise differences between a perturbation and the true gradient also consumes tremendous computation resources for billion-sized LLMs, not to mention that such an operation needs to be conducted for K times in each round. (3) As mentioned in Equation (12) presented by Rahimi et al. (2023), when facing the scenario of partial client participation, a client that has not participated in FL for ϱ rounds has to perform $\varrho \cdot K$ 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. (2023), 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.

According to the above technical comparisons, FedKSeed is the first approach that enables the possibility of federated full-parameter tuning of billion-sized LLMs on devices, which lowers the communication cost to a constant below 18 kilobytes and the memory consumption to inference level.

B. Broader Impact

In this section, we provide discussions on broader benefits brought by FedKSeed and FedKSeed-Pro to existing techniques and the potential probability of extending existing FL techniques to LLM fine-tuning based on the proposed approaches.

B.1. Alleviating the Burden of Aggregation

Assuming there are m participated 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). 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. Thus, the financial burden of FL organizers is greatly alleviated with FedKSeed and FedKSeed-Pro.

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

Due to the transmission delay and unstable connections caused by long-distance transmission, many organizations prefer to build decentralized FL by allowing some participants to perform aggregation (Qin et al., 2023). However, it undoubtedly makes communication costs of FL even more severe, since each client may need to transmit its model parameters to more than one receiver. FedKSeed and FedKSeed-Pro can significantly reduce communication costs, thus bringing the possibility of fine-tuning LLMs with decentralized FL.

B.3. Alleviating the Burden of Saving Checkpoints

The trajectory of LLM fine-tuning does not always proceed in the desired direction. Sometimes, fine-tuning might get stuck in a local optimum or a worse region, or even be attacked by malicious participants through techniques such as backdoor attack (Zhang et al., 2023b). Thus, multiple snapshots of LLMs often need to be stored for rolling back when it is necessary. It incurs a huge storage overhead. However, with FedKSeed and FedKSeed-Pro, only the snapshots of accumulated scalar gradients $\mathbf{a}$ need to be stored for rolling back, each of which is an array with $2K$ scalars. It significantly reduces the storage consumption.

C. Detailed Assumptions

We detail the assumptions made by Fang et al. (2022) which are necessary conditions for deriving the convergence of ZOO-based FL with one-point estimator claimed in Lemma 1, with variable substitution for ease of comprehending in the context of our work.

Assumption 1. *(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 present Assumption 2 and Assumption 3, 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 $f_i(\mathbf{w}) \triangleq \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_i}\left[\mathcal{L}_i(\mathbf{w};\mathbf{x})\right]$.

Assumption 2. *(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 3. *(Boundary of the Second-Order Gradient Moment.) The second-order moment of stochastic gradient $\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w},\mathbf{x})$ satisfies*

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

where $c_g \geq 1$.

Assumption 4. *(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*

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

where c_h is a positive constant.