

FedKRSO: Communication and Memory Efficient Federated Fine-Tuning of Large Language Models

Guohao Yang*, Tongle Wu*, Yuanxiong Guo*, Ying Sun*, and Yanmin Gong*

*University of Texas at San Antonio, San Antonio, TX 78249

*Penn State University, University Park, PA 16802

Emails: guohao.yang@utsa.edu, tfw5381@psu.edu, yuanxiong.guo@utsa.edu, ybs5190@psu.edu, yanmin.gong@utsa.edu

Abstract—Fine-tuning is essential to adapt general-purpose large language models (LLMs) to domain-specific tasks. As a privacy-preserving framework to leverage decentralized data for collaborative model training, Federated Learning (FL) is gaining popularity in LLM fine-tuning, but remains challenging due to the high cost of transmitting full model parameters and computing full gradients on resource-constrained clients. While Parameter-Efficient Fine-Tuning (PEFT) methods are widely used in FL to reduce communication and memory costs, they often sacrifice model performance compared to FFT. This paper proposes FedKRSO (Federated K -Seed Random Subspace Optimization), a novel method that enables communication and memory efficient FFT of LLMs in federated settings. In FedKRSO, clients update the model within a shared set of random low-dimension subspaces generated by the server to save memory usage. Furthermore, instead of transmitting full model parameters in each FL round, clients send only the model update accumulators along the subspaces to the server, enabling efficient global model aggregation and dissemination. By using these strategies, FedKRSO can substantially reduce communication and memory overhead while overcoming the performance limitations of PEFT, closely approximating the performance of federated FFT. The convergence properties of FedKRSO are analyzed rigorously under general FL settings. Extensive experiments on the GLUE benchmark across diverse FL scenarios demonstrate that FedKRSO achieves both superior performance and low communication and memory overhead, paving the way towards on federated LLM fine-tuning at the resource-constrained edge.

Index Terms—Federated learning, large language models, fine-tuning, and resource efficiency.

I. INTRODUCTION

Large Language Models (LLMs) have demonstrated impressive capabilities in a wide range of real-world applications, including question answering, text summarization, and dialogue generation [1]. Despite their strong generalization capabilities, task-specific fine-tuning is often necessary to further enhance their responsiveness and alignment with downstream objectives. As task-specific data are typically privacy-sensitive and distributed across multiple clients, Federated Learning (FL) [2] that enables collaborative model training without centralizing data has received increasing attention

G. Yang and T. Wu contributed equally to this work. Y. Sun and Y. Gong are the co-corresponding authors. The work of G. Yang, Y. Guo, and Y. Gong was supported in part by the U.S. National Science Foundation under Grants CNS-2047761, CNS-2106761, and CNS-2318683. The work of T. Wu and Y. Sun was supported in part by a Seed Grant award from the Institute for Computational and Data Sciences (ICDS) at the Pennsylvania State University (PSU). This content is solely the responsibility of the authors and does not necessarily represent the views of the ICDS.

for LLM fine-tuning. However, applying FL to LLM fine-tuning faces significant challenges. In particular, the massive number of parameters in modern LLMs makes full-parameter fine-tuning (FFT) memory, computation, and communication-intensive, rendering it impractical for deployment in resource-constrained edge environments [3]–[6].

To address the above challenge, recent studies have Parameter-Efficient Fine-Tuning (PEFT) techniques within the FL framework [7]–[14]. In particular, as the most commonly used PEFT method, Low-Rank Adaption (LoRA) [15] adapts only a small subset of model parameters. While effective in reducing memory footprint, LoRA often suffers from a performance drop compared to FFT, especially in the presence of heterogeneous data distributions in FL [14]. It has been observed that the limited capacity of low-rank updates in LoRA makes it difficult to fully capture the global model dynamics, leading to slower convergence or sub-optimal accuracy [16], [17].

To achieve strong performance under federated settings in resource-constrained edge environments, we propose a new federated fine-tuning method that integrates subspace optimization with a finite seed-based random projection mechanism. Our approach named **FedKRSO** (**F**ederated K -**S**econd **R**andom **S**ubspace **O**ptimization) maintains the resource efficiency benefits of PEFT while more closely approaching the model performance of FFT. Specifically, FedKRSO includes two key components: First, during local training in a FL round, each client compresses the full gradients by projecting them onto a shared finite set of random subspaces and then uses the compressed gradients to optimize the model. These subspaces are generated by the server to ensure that the compression is unbiased so that the local updates take the same form as FFT in expectation. By using compressed gradients instead of full gradients, the memory usage at each client is greatly reduced. Second, after local training in a FL round, each client merges the local model updates along each subspace and uploads the total model updates to the server. The server then aggregates the model updates from all clients subspace by subspace and send them back to the clients. This strategy enables clients to reconstruct the global model used in the next round, while reducing the communication cost for both uplinks and downlinks. Overall, FedKRSO can achieve communication and memory efficiency similar to PEFT-based approaches, while maintaining model performance comparable to FFT.

In summary, our key contributions are as follows:

- We introduce FedKRSO, a novel method that employs

random subspace optimization with a finite set of random seeds, to achieve communication and memory efficient federated fine-tuning of LLMs at the edge, while maintaining high model performance.

- We perform rigorous theoretical analyses of FedKRSO and establish its convergence rates under non-convex loss and heterogeneous data distributions in federated settings.
- We conduct extensive experiments across multiple models, datasets and data heterogeneity settings to demonstrate the advantages of FedKRSO over state-of-the-art baselines in terms of both model performance and communication and memory efficiency.

II. RELATED WORK

1) Memory-Efficient Centralized Optimization Methods for LLMs: PEFT methods are essential for adapting large pre-trained models under resource constraints. Early techniques like Adapter [18] and Prefix-Tuning [19] enable task-specific adaptation with small parameter updates. LoRA has emerged as the most widely adopted method. These methods either update or add a relatively small number of tunable parameters to reduce the memory and computation cost. As a trade-off, such a restriction may prevent the algorithms from finding models with high accuracy that lie outside the searchable space. It has been noted in several recent works that they may not reach a performance comparable to FFT [16], [17], [20].

Zeroth-order (ZO) optimization achieves memory efficiency by estimating gradients using finite differences of function values [21]. Various ZO algorithms have been developed by adapting first-order (FO) methods, including ZO-SGD [22] and ZO-Adam [23]. Although conceptually simple, these approaches often suffer from high variance and slow convergence, particularly due to the high dimensionality of the underlying models [24].

Algorithms more related to our local training procedure are subspace methods, such as [25]–[29], which aim to bridge the gap between PEFT and FFT by approximating its updates. These methods either compress the full gradient or optimize the model restricted to randomly generated low-dimensional subspaces. Although doing so reduces the memory for storing the optimizer states, they still require computing and storing the full gradient once in a while. This will result in significant peak memory usage and limit their applicability to edge devices with memory constraints, whereas our method overcomes this limitation. We refer the readers to Remark 3 and 4 for more details.

2) Federated Fine-Tuning for LLMs: Building on the memory-efficient centralized methods introduced earlier, various federated fine-tuning methods of LLMs have been proposed, which can be roughly divided into three categories.

One line of work integrates LoRA into FL to reduce communication cost [7], [9]–[14], and can be categorized by whether they assume homogeneous or heterogeneous LoRA ranks among clients. In the homogeneous setting, all clients use the same LoRA rank during local model training. For example, Fedit [10] and its variants [12], [14] integrate

LoRA-based local fine-tuning of LLMs directly into standard FL procedures. In the heterogeneous setting, methods such as FLoRA [11] and FlexLoRA [13] assign different LoRA ranks to clients to handle system heterogeneity. FedSA-LoRA [30] shares only one of the LoRA low-rank matrices with the server for aggregation, while keeping the other matrix local to each client to handle heterogeneity.

Another line of work applied ZO during local model training in FL to achieve memory efficiency [31]. For example, FwdLLM [32] eliminates the need for gradient computation, enabling scalable training on mobile devices. Similarly, FedKSeed [33] proposes a seed-reuse technique to reduce per-round communication costs. These methods share similar limitations to their centralized counterparts.

There are recent works [34], [35] that adopt the idea of subspace compression. In Ferret [35], clients perform FFT for the local training and project the model changes for each communication round to a finite set of random directions, making it sufficient to send the projected coordinates to the server. SEPARATE [34] also computes the full gradients locally and applies gradient compression with error-feedback for the communication step. The variable introduced to compensate for the compression error is of the same size as the full gradient. These methods focus on improving communication efficiency, but put less emphasis on memory costs.

III. PRELIMINARIES

A. Federated Fine-Tuning

Consider an FL system for fine-tuning an LLM that consists of a server and N clients (e.g., edge devices). Each client $n \in [N]$ has a local private dataset \mathcal{D}_n . Federated fine-tuning aims at collaboratively tuning model W with the pre-trained weight $W^0 \in \mathbb{R}^{d_m \times d_n}$ at initialization, which can be formulated as

$$\min_W \left\{ F(W) := \frac{1}{N} \sum_{n=1}^N F_n(W) \right\}, \quad (1)$$

where $F(W)$ is the global objective function, and $F_n(W) := \mathbb{E}_{\xi_n \sim \mathcal{D}_n} [f_n(W; \xi_n)]$ is the local objective function of client n with ξ_n being a datapoint sampled from distribution \mathcal{D}_n and $f_n(W; \xi_n)$ being the loss of model W evaluated on ξ_n , respectively.

The classical FL algorithm, FedAvg [2], solves (1) via multiple rounds of local training followed by global model aggregation. Specifically, in each round t , client $n \in [N]$ performs J stochastic gradient descent (SGD) steps given by

$$W_n^{t,j+1} = W_n^{t,j} - \eta \cdot \hat{\nabla} f_n(W_n^{t,j}), \quad j = 0, \dots, J-1, \quad (2)$$

where $W_n^{t,j}$ is the local model of client n at local step j , and $\hat{\nabla} f_n(W_n^{t,j}) := \nabla f_n(W_n^{t,j}, \xi_n^{t,j})$ is a mini-batch stochastic gradient of F_n evaluated at $W_n^{t,j}$, and η is a learning rate. After local training, the server averages all local models sent from the clients and broadcasts it, which gives the following initialization for the next round:

$$W_n^{t+1,0} = \frac{1}{N} \sum_{n'=1}^N W_{n'}^{t,J}. \quad (3)$$

However, a major challenge in applying the above federated FFT to LLMs lies in the extremely high communication, computation, and memory overheads due to the massive parameter size of LLMs. For example, fine-tuning a Llama 7B model using the BF16 numerical format requires at least 58 GB of memory [26]. In addition to the substantial GPU memory required to store and update the full set of model parameters, federated FFT also incurs significant communication overhead, as the entire model (14 GB for trainable parameters) must be transmitted between clients and the central server in each communication round. As a result, FFT in federated settings is both memory and communication-intensive, rendering it impractical for real-world deployment in resource-constrained edge environments.

To address these bottlenecks, recent methods attempt to reduce the memory and communication overheads by training and transmitting only some lightweight adapters, particularly LoRA [15]. However, while LoRA-based approaches are more memory and communication-efficient, empirical results show that they typically fall short of the performance achieved by FFT [16], [17]. Therefore, LoRA-based federated fine-tuning methods like FedIT [10], though efficient, are not yet on par with federated FFT in terms of model performance.

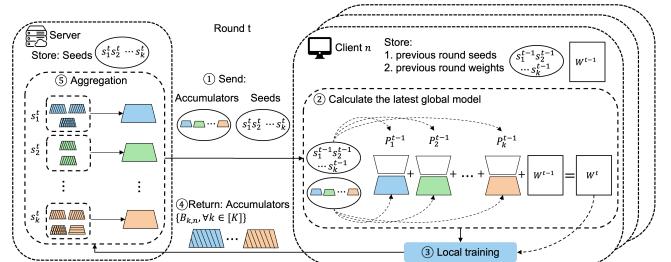
IV. DESIGN OF FEDKRSO

To address the challenges of federated fine-tuning, this section proposes a novel method named FedKRSO, which is designed with two primary objectives: (1) to approach the performance of federated FFT; and (2) to maintain or lower the computation, communication, and memory overhead compared to federated LoRA fine-tuning. We first give an overview of the method in Sec. IV-A and then elaborate on the design details in Sec. IV-B and Sec. IV-C. The communication and memory efficiency of FedKRSO is analyzed in Sec. IV-D.

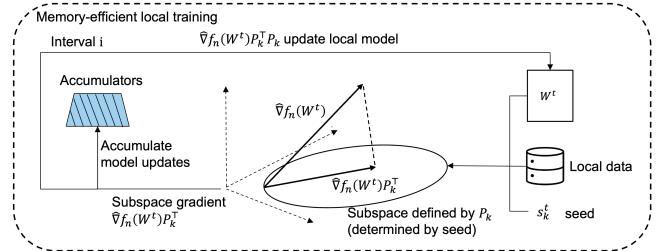
A. Overview of FedKRSO

In FedKRSO, the server maintains a pool of K random seeds for subspace generation, denoted as $\mathcal{S} := \{s_1, s_2, \dots, s_K\}$, along with a corresponding set of K model update accumulators, each of size $d_m \times r$, denoted as $\mathcal{B} := \{B_1, B_2, \dots, B_K\}$. Both \mathcal{S} and \mathcal{B} are updated in each FL round. Each client, on the other hand, is initialized with a copy of the base model W^0 . As depicted in Fig. 1, for each FL round t , FedKRSO proceeds as follows:

- 1) **Seed and accumulator broadcast.** The server sends the set of random seeds \mathcal{S}^t and accumulators of model update \mathcal{B}^t to all clients (Line 5 in Algorithm 1).
- 2) **Model reconstruction.** Each client uses \mathcal{B}^t and \mathcal{S}^{t-1} received in the previous round to reconstruct the latest global model W^t , which is then used to initialize its local model (Line 8 in Algorithm 1).
- 3) **Local training along random subspaces.** Local training is divided into I intervals, each consisting of J local iterations. At the beginning of each interval, client n randomly samples a seed $s_k \in \mathcal{S}^t$, generates the corresponding random matrix P_k , and performs J steps



(a) The overall workflow of FedKRSO. The color of the accumulators represents different seeds, while the patterns within the shapes indicate different clients (Algorithm 1).



(b) The memory-efficient local training process, which includes I intervals, each consisting of J local iterations. The reconstructed latest global model W^t is used as the initialization for local training (Algorithm 2).

Fig. 1. Overview of FedKRSO at FL round t .

of local updates on the current model W using the compressed gradient G_B in the subspace defined by P_k . (Lines 8-14 in Algorithm 2). Meanwhile, the compressed gradient is accumulated into the model update accumulator $B_{k,n}$ corresponding to seed s_k (Line 15 in Algorithm 2). At the end of local training, we will reset the local model weights to their initial values (Line 18 in Algorithm 2) in order to reconstruct the latest global model in Step 2.

- 4) **Local update uploading.** Once finishing the local training stage, each client n sends at most K accumulators $\mathcal{B}_n^{t+1} := \{B_{k,n}^{t+1}, \forall k \in [K]\}$ back to the server.
- 5) **Server-side model aggregation.** After receiving the updates \mathcal{B}_n^{t+1} from all clients, the server computes the global accumulator set \mathcal{B}^{t+1} by averaging the local ones $B_{k,n}^{t+1}$ from all clients seed by seed (Line 12 in Algorithm 1).

The pseudo code for the overall procedure is shown in Algorithms 1 and 2.

B. Memory-Efficient Local Training

In this section, we elaborate on the local training step presented in Algorithm 2. We start with $I = 1$, which means that each client n randomly selects only one seed from the pool \mathcal{S} . We simplify the notations here by dropping the dependency of the local variables, such as P_k , on the client index n and round index t without introducing ambiguity. Recall that in each FL round t , the local update of FFT is given by (2).

Our goal is to approximate the update (2) but with a memory cost comparable to LoRA, which crucially relies on computing the stochastic gradient in a memory-efficient manner. To this end, the proposed algorithm leverages

Algorithm 1 FedKRSO Algorithm

- 1: **Input:** Number of clients N , communication rounds T , pre-trained model weight matrix $W^0 \in \mathbb{R}^{d_m \times d_n}$, rank r , number of intervals I , interval length J .
- 2: **Initialize:** global model update accumulators $B_k^0 = 0$ and $P_k^{-1} = 0 \forall k \in [K]$.
- 3: **for** round $t = 0$ to $T - 1$ **do**
- 4: Server selects a set of K random seeds \mathcal{S}^t
- 5: Server sends $\mathcal{B}^t := \{B_k^t\}_{k=1}^K$ and \mathcal{S}^t to all clients
- 6: **for** client $n = 1$ to N in parallel **do**
- 7: Specify $\mathcal{P}^t := \{P_k^{t-1}\}_{k \in [K]}$ according to \mathcal{S}^{t-1}
- 8: Reconstruct the global model to initialize local training as
- 9:
$$W^t = W^{t-1} + \sum_{k=1}^K B_k^t P_k^{t-1}$$
- 10:
$$\mathcal{B}_n^{t+1} = \text{LocalTraining}(W^t, \mathcal{S}^t, n)$$
- 11: Upload \mathcal{B}_n^{t+1} to the server
- 12: **end for**
- 13: Server aggregates:
- 14:
$$B_k^{t+1} = \frac{1}{N} \sum_{n=1}^N B_{k,n}^{t+1}, \forall k \in [K]$$
- 15: **end for**
- 16: **Output:** W^T

Algorithm 2 LocalTraining(W, \mathcal{S}, n)

- 1: **Initialize:** local model update accumulators set $\mathcal{B}_n = \{B_{1,n}, B_{2,n}, \dots, B_{K,n}\}$ with $B_{k,n} = 0, \forall k \in [K]$, number of intervals I , interval length J , learning rate η , rank r , decay rates β_1, β_2 , coefficient ϵ , fist-order moment $M \leftarrow 0$, second-order moment $V \leftarrow 0$.
- 2: **for** interval $i = 1$ to I **do**
- 3: Uniformly sample a seed $s_k \in \mathcal{S}$
- 4: Specify $P_k \sim \mathcal{N}(0, (1/r) \mathbb{I}_{r \times d_n})$ using seed s_k , $P_k \in \mathbb{R}^{r \times d_n}$
- 5: $M, V \leftarrow 0$
- 6: **for** iteration $j = 1$ to J **do**
- 7: Sample a mini-batch from local dataset \mathcal{D}_n
- 8: Compute $\hat{\nabla} f_n(W) P_k^\top$ according to Eq. (6) and let $G_B \leftarrow \hat{\nabla} f_n(W) P_k^\top$
- 9: $M \leftarrow \beta_1 \cdot M + (1 - \beta_1) \cdot G_B$
- 10: $V \leftarrow \beta_2 \cdot V + (1 - \beta_2) \cdot G_B^2$
- 11: $M \leftarrow M / (1 - \beta_1)$
- 12: $V \leftarrow V / (1 - \beta_2)$
- 13: $G_B \leftarrow M / (\sqrt{V} + \epsilon)$
- 14: $W \leftarrow W - \eta G_B P_k$
- 15: $B_{k,n} \leftarrow B_{k,n} - \eta G_B$
- 16: **end for**
- 17: **end for**
- 18: Reset initial model: $W \leftarrow W - \sum_{k=1}^K B_{k,n} P_k$.
- 19: **Output:** $\{B_{k,n}, \forall k \in [K]\}$

the random sketching technique for gradient compression, combined with a carefully designed mechanism that enables computation of the compressed gradient *without explicitly evaluating the full gradient*.

Gradient compression. Let W be the model to be updated by client n and $\hat{\nabla} f_n(W)$ be the corresponding stochastic gradient. To reduce the size of the gradient, we compress it using a

linear sketch given by

$$G_B = \hat{\nabla} f_n(W) P^\top, \quad (4)$$

where $P \in \mathbb{R}^{r \times d_n}$ with $r \ll \min\{d_m, d_n\}$ is sampled randomly from a distribution satisfying $\mathbb{E}[P^\top P] = I_{d_n}$, independent from $\hat{\nabla} f_n(W)$. It is then not hard to verify that $\mathbb{E}[G_B P] = \mathbb{E}[\hat{\nabla} f_n(W) P^\top P] = \mathbb{E}[\hat{\nabla} f_n(W)]$ suggesting that we can reconstruct the full gradient in expectation via computing $G_B P$ and updating W as

$$W \leftarrow W - \eta G_B P. \quad (5)$$

Remark 1: Note that to implement (5), it suffices to store the local model W and the two factors G_B and P , which are much smaller than the full gradient. The update of W can be done in-place instead of first computing the product $G_B P$ and then storing it. Therefore, the total memory cost of this step is $d_m \times d_n + (d_m + d_n) \times r$.

Compute the compressed gradient G_B . Notably, naively implementing the gradient compressor (4) requires evaluating $\hat{\nabla} f_n(W)$, making the cost same as FFT. In what follows, we address the challenge and introduce a method for computing G_B with a cost comparable to LoRA via low-rank parameterization.

We consider the loss function $F_n(W + BP)$, where *constants* W and P are, respectively, the local model and the matrix for compression used in (4), and B is the variable. Applying the chain rule, the gradient of F_n with respect to B is given by:

$$\nabla_B F_n(W + BP) = \nabla F_n(W + BP) P^\top. \quad (6)$$

Comparing (6) to (4), we can see that G_B can be computed by evaluating $\hat{\nabla}_B f_n(W + BP)$ on the same mini-batch of data as (4), with W set to be the current model and $B = 0$.

Remark 2: LoRA reparameterizes W as $W = BA$ to save the memory cost. The model is updated by solving $\min_{A,B} F_n(W^0 + BA)$, which requires computing the partial gradients of F_n with respect to A and B for every iteration. Comparing (6) to LoRA reveals that our method reduces the gradient evaluation cost by half per iteration.

Repeating the iteration (5) for J times gives the basic SGD version of Algorithm 2 for $I = 1$, whose convergence proof will be provided in Sec. V.

Change projection direction and momentum acceleration. We proceed to introduce two extensions of the algorithm, which give the complete description of Algorithm 2.

In the basic version, we use a fixed P to compress the gradient. The approach can be extended to using multiple P s. Specifically, for each round t , the server generates a set of seeds \mathcal{S}^t to determine these matrices (Line 4-5 in Algorithm 1). Then in local training, for each $i \in [I]$, each client n samples a seed from \mathcal{S} independently uniformly at random, indexed by k , to generate P_k (Line 4 in Algorithm 2). Furthermore, to accelerate convergence, we introduce state variables M and V updated according to Line 9 to 13 in Algorithm 2. The update mimics Adam [36] but accumulates

the first and second momentum of the compressed gradient G_B , making their storage cost $d_m \times r$ rather than $d_m \times d_n$. The update of local model W is then given by Line 14 in Algorithm 2 using the preconditioned gradient. Once the client finishes J times local updates, it resamples P_k and resets M and V to zero.

The following two remarks compare our proposed local updates to two recent fine-tuning algorithms, both of which apply *only to the centralized setting*.

Remark 3 (Comparison to Flora [25]): The idea of gradient compression is also used in Flora. The key difference is that Flora still requires computing the full gradient once in a while, whereas our method directly computes the compressed gradient via (6). The significant reduction of the peak memory cost makes our algorithm suitable for implementation on resource-constrained edge devices for federated learning.

Remark 4 (Comparison to RSO [27]): In view of (4) and (6), we can interpret the local update (5) as minimizing F_n along a subspace of weights parameterized by $W = BP$. The algorithm shares similarities with RSO but has key differences. RSO minimizes Problem (1) subspace by subspace via the iterates: $B_k \approx \arg \min_B F_n(W_k + BP_k) + \frac{1}{2\eta^k} \|B\|_F^2$ and $W_{k+1} = B_k P_k$. To compute B_k , RSO requires finding the exact minimizer up to an ϵ -accuracy. This requires η^k to be chosen small enough to make the objective convex, which introduces a hyperparameter whose value is often difficult to estimate in practice. Another difference is that RSO is a double-loop algorithm with computing B_k being the inner loop. The convergence accuracy of RSO is determined by ϵ , which means that achieving high accuracy requires a large number of iterations for the inner loop, and hence a high computational cost. In contrast, ours is essentially a single-loop algorithm since J can be an arbitrary fixed constant (notably, J can be set to 1).

We finish this section by explaining the function of model update accumulators $B_{k,n}$ along with their update given by Line 15. The model update in Line 14 implies that for each $i \in [I]$, model change is equal to the accumulated gradients in the interval. Since matrix P_k is fixed over the J iterations, it suffices to use $B_{k,n}$ to accumulate the compressed gradients to reproduce the model change as $B_{k,n}P_k$ in this interval. The set \mathcal{B}_n^{t+1} thus fully encodes the model change of each client n , and is sent to the server after one round of local updates. This step will be useful for implementing the server-side update described subsequently.

C. Communication-Efficient Model Aggregation and Dissemination

Section IV-B presented a memory-efficient approximation of the local updates in federated fine-tuning given by (2). Next, we describe how to perform the model aggregation and dissemination step (3) without communicating the full model weights using the K -seed mechanism.

Since the local models of all clients are synchronized to be W^t at the beginning of round t , the model aggregation (3)

can be rewritten as¹ $W^{t+1} = W^t + (1/N) \sum_{n=1}^N \Delta W_n^t$, where $\Delta W_n^t := W_n^{t,J} - W^t$. With \mathcal{B}_n^{t+1} sent by each client n and the random seeds \mathcal{S}^t , the server can thereby compute the model change as

$$\frac{1}{N} \sum_{n=1}^N \Delta W_n^t = \frac{1}{N} \sum_{n=1}^N \left(\sum_{k=1}^K B_{k,n}^{t+1} P_k^t \right), \quad (7)$$

where $B_{k,n}^{t+1}$ denotes the value of the accumulator $B_{k,n}$ at the end of round t and P_k^t is the random matrix generated by seed s_k^t . By exchanging the order of the summation, we find that the server can first aggregate $B_{k,n}^{t+1}$ for each seed $k \in [K]$ over the clients using a global accumulator B_k^{t+1} as specified by Line 12 in Algorithm 1. The global model update after round t is therefore

$$W^{t+1} = W^t + \sum_{k=1}^K B_k^{t+1} P_k^t. \quad (8)$$

Since the clients already have the information on \mathcal{S}^t , it suffices to broadcast the set of global accumulators $\mathcal{B}^{t+1} := \{B_k^{t+1}\}_{k=1}^K$ for each client to implement (8), as given in Line 5-8 in Algorithm 1.

Lastly, we mention that since the clients modify the model weights W in-place, the initial model W^t will be erased at the end of local training. Therefore, we need to recover W^t to implement (8) as specified in Line 18 in Algorithm 2.

D. Communication Overhead and Memory Footprint

TABLE I
MEMORY AND PER-ROUND COMMUNICATION COST COMPARISON IN TERMS OF NUMBER OF PARAMETERS. HERE $P = d_m \times d_n$, $L = (d_m + d_n) \times r$, $Q = d_m \times r$.

		FedIT	FFA-LoRA	FedFFT	FedKRSO
Mem.	Weights	$P + L$	$P + L$	P	$P + L$
	Gradients	L	Q	P	Q
	Opt. states	$2L$	$2Q$	$2P$	$2Q$
Comm.	Uplink	L	Q	P	IQ
	Downlink	L	Q	P	$KQ + K$

The theoretical communication and memory analysis for the different algorithms was conducted on a single linear layer with weight matrix $W_l \in \mathbb{R}^{d_m \times d_n}$. We first analyze the communication cost of FedKRSO. Since each client at most selects I different P_k matrices and uploads the corresponding $B_{k,n}$ in each round, the uplink communication cost for each client is bounded by $I \times d_m \times r$. On the other hand, the server broadcasts \mathcal{B}^t and \mathcal{S}^t , which gives communication overhead of $K \times d_m \times r + K$ parameters.

As for the memory footprint, the cost on the client side for storing the weights is $d_m \times d_n + (d_m + d_n) \times r$ as given in Remark 1, and that for the compressed gradients and optimizer states is $3 \times d_m \times r$ ². A comparison with existing methods is provided in Table I.

¹Note that $W_n^{t,0} = W^t$ for all n and t .

²Here we have not accounted for the cost of activations, which depends on the model architecture and the batch size.

V. CONVERGENCE ANALYSIS

In this section, we establish the convergence guarantee of FedKRSO in Algorithm 1 under the SGD setting with $I = 1$ in Algorithm 2. That is, at each communication round t , every client $n \in [N]$ draws a single seed $k(n, t) \sim \text{Unif}\{1, 2, \dots, K\}$, and then performs J steps of SGD using a random matrix denoted as $P_{k(n),t}^t$. The generalization to cases $I > 1$ and with momentum is more involved and is left as future work.

To simplify notation, we omit the dependency of $k(n, t)$ on t , and denote it as $k(n)$ without affecting the analysis. Let $B_{k(n),n}^{t,j}$, $j = 0, 1, \dots, J$, $n \in [N]$ denote the model update accumulators of client n in its j -th local SGD step of round t , using seed $k(n)$. Then the intermediate local model is given by $W_n^{t,j} = W^t + B_{k(n),n}^{t,j} P_{k(n)}^t$. We make the following standard assumptions on the loss functions.

Assumption 1 (L-smoothness): Each local loss function F_n , $n \in [N]$, is differentiable and has L -Lipschitz continuous gradient. That is, $\forall n \in [N], \forall W_1, W_2 \in \mathbb{R}^{d_m \times d_n}$,

$$\|\nabla F_n(W_1) - \nabla F_n(W_2)\|_F \leq L \|W_1 - W_2\|_F. \quad (9)$$

Assumption 2 (Bounded noise variance): Each client $n \in [N]$ has access to unbiased stochastic gradients $\hat{\nabla} f_n(W)$ satisfying $\mathbb{E}_{\xi_n \sim \mathcal{D}_n} [\hat{\nabla} f_n(W)] = \nabla F_n(W)$ with bounded variance

$$\mathbb{E}_{\xi_n \sim \mathcal{D}_n} \|\hat{\nabla} f_n(W) - \nabla F_n(W)\|_F^2 \leq \sigma^2, \quad \forall W \in \mathbb{R}^{d_m \times d_n}. \quad (10)$$

We consider a Non-IID setting where the data heterogeneity of the clients is characterized by the following assumption.

Assumption 3 (Bounded gradient heterogeneity): We assume that there exists constant ς such that $\forall W \in \mathbb{R}^{d_m \times d_n}$,

$$\frac{1}{N} \sum_{n=1}^N \|\nabla F_n(W)\|_F^2 \leq \varsigma^2 + \|\nabla F(W)\|_F^2. \quad (11)$$

We further impose the following assumption on random matrices $\{P_k^t\}$, which is also used in existing works [27], [37].

Assumption 4: The random matrices $\{P_k^t\}$, $\forall k \in [K], t = 1, \dots, T$, are independent and each is drawn from a distribution such that $P_k^t (P_k^t)^\top = d_n/r I_r$ and $\mathbb{E}[(P_k^t)^\top P_k^t] = I_{d_n}$.

The assumption can be satisfied with high probability if the elements of P_k^t are i.i.d. drawn from a centered Gaussian distribution and $\min\{d_m, d_n\} \gg r$.

With the assumptions above, we first bound the decrease of the loss function in one communication round.

Lemma 1: If Problem (1) satisfies Assumption 1 and 3, then the iterates generated by FedKRSO under Assumption 2 and 4 with learning rate $\eta \leq \frac{r}{16Ld_nJ}$ satisfy

$$\begin{aligned} \mathbb{E}[F(W^{t+1})] &\leq \mathbb{E}[F(W^t)] + \frac{2Ld_nJ^2\eta^2}{r} \varsigma^2 + \frac{\eta d_n^2 L^2}{r^2} Q^t \\ &\quad - \frac{3J\eta}{8} \mathbb{E}[\|\nabla F(W^t)\|_F^2] + \frac{LJ\eta^2 d_n^2 \sigma^2}{Nr^2}, \end{aligned} \quad (12)$$

where $Q^t := \frac{1}{N} \sum_{n=1}^N \sum_{j=0}^{J-1} \mathbb{E} \left[\left\| B_{k(n),n}^{t,j} P_{k(n)}^t \right\|_F^2 \right]$.

Proof: From Algorithm 1, there is

$$W^{t+1} = W^t - \eta \frac{\sum_{n=1}^N B_{k(n),n}^{t,J} P_{k(n)}^t}{N}. \quad (13)$$

Then, based on the descent lemma under L -smoothness Assumption (1), there is

$$\begin{aligned} \mathbb{E}[F(W^{t+1})] &\leq \mathbb{E} \left[F(W^t) + \frac{L}{2} \left\| \frac{\sum_{n=1}^N B_{k(n),n}^{t,J} P_{k(n)}^t}{N} \right\|_F^2 \right] \\ &\quad + \mathbb{E} \left[\left\langle \nabla F(W^t), \frac{\sum_{n=1}^N B_{k(n),n}^{t,J} P_{k(n)}^t}{N} \right\rangle \right] \\ &= \mathbb{E} \left[F(W^t) + \frac{L\eta^2}{2} \left\| \frac{\sum_{n,j} \nabla \hat{f}_n(W_n^{t,j}) (P_{k(n)}^t)^\top P_{k(n)}^t}{N} \right\|_F^2 \right] \\ &\quad - \eta \mathbb{E} \left[\left\langle \nabla F(W^t), \frac{\sum_{n,j} \hat{\nabla} f_n(W_n^{t,j}) (P_{k(n)}^t)^\top P_{k(n)}^t}{N} \right\rangle \right] \\ &\stackrel{(a)}{\leq} \mathbb{E}[F(W^t)] - \frac{J\eta}{2} \mathbb{E}[\|\nabla F(W^t)\|_F^2] + \frac{LJ\eta^2 d_n^2 \sigma^2}{Nr^2} + \frac{\eta}{2N} \\ &\quad \cdot \sum_{n=1}^N \sum_{j=0}^{J-1} \mathbb{E} \left[\left\| (\nabla F_n(W_n^{t,j}) - \nabla F_n(W^t)) (P_{k(n)}^t)^\top P_{k(n)}^t \right\|_F^2 \right] \\ &\quad + \frac{2L\eta^2}{N^2} \left(\mathbb{E} \left[\left\| \sum_{n=1}^N \sum_{j=0}^{J-1} \nabla F_n(W^t) (P_{k(n)}^t)^\top P_{k(n)}^t \right\|_F^2 \right] \right. \\ &\quad \left. + \left\| \sum_{n,j} (\nabla F_n(W_n^{t,j}) - \nabla F_n(W^t)) (P_{k(n)}^t)^\top P_{k(n)}^t \right\|_F^2 \right) \\ &\stackrel{(b)}{\leq} \mathbb{E}[F(W^t)] - \frac{J\eta}{2} \mathbb{E}[\|\nabla F(W^t)\|_F^2] \\ &\quad + 2LJ^2\eta^2 \mathbb{E} \left[\left\| \sum_{n=1}^N \sum_{j=0}^{J-1} \frac{\nabla F_n(W^t) (P_{k(n)}^t)^\top P_{k(n)}^t}{NJ} \right\|_F^2 \right] \\ &\quad + \left(\frac{\eta d_n^2 L^2}{2r^2} + \frac{2Jd_n^2 \eta^2 L^3}{r^2} \right) Q^t + \frac{LJ\eta^2 d_n^2 \sigma^2}{Nr^2} \\ &\stackrel{(c)}{\leq} \mathbb{E}[F(W^t)] - \frac{J\eta}{2} \mathbb{E}[\|\nabla F(W^t)\|_F^2] + \frac{LJ\eta^2 d_n^2 \sigma^2}{Nr^2} \\ &\quad + \frac{2LJ\eta^2}{N} \sum_{n=1}^N \sum_{j=0}^{J-1} \mathbb{E} \left[\left\| \nabla F_n(W^t) (P_{k(n)}^t)^\top P_{k(n)}^t \right\|_F^2 \right] \\ &\quad + \left(\frac{\eta d_n^2 L^2}{2r^2} + \frac{2Jd_n^2 \eta^2 L^3}{r^2} \right) Q^t \\ &\stackrel{(d)}{=} \mathbb{E}[F(W^t)] + \left(\frac{\eta d_n^2 L^2}{2Nr^2} + \frac{2Jd_n^2 \eta^2 L^3}{Nr^2} \right) Q^t + \frac{LJ\eta^2 d_n^2 \sigma^2}{Nr^2} \\ &\quad + \frac{2LJ^2\eta^2 d}{Nr} \sum_{n=1}^N \mathbb{E}[\|\nabla F_n(W^t)\|_F^2] - \frac{J\eta}{2} \mathbb{E}[\|\nabla F(W^t)\|_F^2] \\ &\leq \mathbb{E}[F(W^t)] + \left(-\frac{J\eta}{2} + \frac{2Ld_nJ^2\eta^2}{r} \right) \mathbb{E}[\|\nabla F(W^t)\|_F^2] \end{aligned}$$

$$+ \frac{\eta d_n^2 L^2 + 4Jd_n^2 \eta^2 L^3}{2Nr^2} Q^t + \frac{2Ld_n J^2 \eta^2 \zeta^2}{r} + \frac{LJ\eta^2 d_n^2 \sigma^2}{Nr^2}, \quad (14)$$

where (a) is due to Assumption 2, $\|a+b\|^2 \leq 2(\|a\|^2 + \|b\|^2)$ and Assumption 4, which has $\|P_{k(n)}^t\|_2 \leq \sqrt{\frac{d_n}{r}}$; (b) and (c) are based on (9) and Jensen's inequality, respectively; and (d) is because $P_{k(n)}^t$ is independent of W^t and thus $\mathbb{E}[\|\nabla F_n(W^t)(P_{k(n)}^t)^\top P_{k(n)}^t\|_F^2] = \mathbb{E}[\text{Tr}((P_{k(n)}^t)^\top P_{k(n)}^t (\nabla F_n(W^t))^\top \nabla F_n(W^t)(P_{k(n)}^t)^\top P_{k(n)}^t)] = \frac{d_n}{r} \|\nabla F_n(W^t)\|_F^2$ by Assumption 4. The last inequality and (12) is due to the learning rate condition $\eta \leq \frac{r}{16Ld_n}$. ■

Our next lemma bounds Q^t in (12).

Lemma 2: In the same setting of Lemma 1, FedKRSO has

$$\begin{aligned} Q^t &\leq \frac{8J^3\eta^2 d_n}{r} \zeta^2 + \frac{8J^3\eta^2 d_n}{r} \mathbb{E}[\|\nabla F(W^t)\|_F^2] \\ &\quad + \frac{4J^2 d_n^2 \eta^2 \sigma^2}{r^2} \end{aligned} \quad (15)$$

Proof: Based on Line 15 in Algorithm 2:

$$B_{k(n),n}^{t,j+1} = B_{k(n),n}^{t,j} - \eta \hat{\nabla} f_n(W_n^{t,j}) (P_{k(n)}^{t,k(n)})^\top. \quad (16)$$

By Assumption 2, there is

$$\begin{aligned} \mathbb{E}[\|B_{k(n),n}^{t,j+1} P_{k(n)}^t\|_F^2] &= \frac{d^2 \eta^2 \sigma^2}{r^2} \\ &\quad + \mathbb{E}\left[\left\| \left(B_{k(n),n}^{t,j} - \eta \nabla F_n(W_n^{t,j}) (P_{k(n)}^t)^\top \right) P_{k(n)}^t \right\|_F^2\right] \\ &\stackrel{(e)}{\leq} \left(1 + \frac{1}{J}\right) E\left[\left\| B_{k(n),n}^{t,j} P_{k(n)}^t \right\|_F^2\right] + \frac{d_n^2 \eta^2 \sigma^2}{r^2} + 4J\eta^2 \\ &\quad \cdot \mathbb{E}\left[\left\| (\nabla F_n(W_n^{t,j}) - \nabla F_n(W^t)) (P_{k(n)}^t)^\top P_{k(n)}^t \right\|_F^2\right] \\ &\quad + 4J\eta^2 \mathbb{E}\left[\left\| \nabla F_n(W^t) (P_{k(n)}^t)^\top P_{k(n)}^t \right\|_F^2\right] \\ &\leq \left(1 + \frac{1}{J} + \frac{4J\eta^2 d_n^2 L^2}{r^2}\right) \mathbb{E}\left[\left\| B_{k(n),n}^{t,j} P_{k(n)}^t \right\|_F^2\right] \\ &\quad + \frac{4J\eta^2 d_n}{r} \mathbb{E}[\|\nabla F_n(W^t)\|_F^2] + \frac{d_n^2 \eta^2 \sigma^2}{r^2} \\ &\leq \left(1 + \frac{2}{J}\right) \mathbb{E}\left[\left\| B_{k(n),n}^{t,j} P_{k(n)}^t \right\|_F^2\right] + \frac{d_n^2 \eta^2 \sigma^2}{r^2} \\ &\quad + \frac{4J\eta^2 d_n}{r} \mathbb{E}[\|\nabla F_n(W^t)\|_F^2], \end{aligned} \quad (17)$$

where (e) uses the Young's inequality and the last inequality is due to $\eta \leq \frac{r}{16Ld_n}$.

Unrolling above inequality for $j = 0, \dots, J-1$ yields

$$\mathbb{E}[\|B_{k(n),n}^{t,j} P_{k(n)}^t\|_F^2] \leq \frac{8J^2 \eta^2 d_n}{r} \mathbb{E}[\|\nabla F_n(W^t)\|_F^2] + \frac{4Jd_n^2 \eta^2 \sigma^2}{r^2}, \quad (18)$$

where we use fact that $\sum_{p=0}^{J-1} (1 + \frac{2}{J})^p \leq 4J$. Summing up over all clients and local updates gives

$$Q^t \leq \frac{8J^3 \eta^2 d_n}{r} \mathbb{E}\left[\frac{1}{N} \sum_{n=1}^N \|\nabla F_n(W^t)\|_F^2\right] + \frac{4J^2 d_n^2 \eta^2 \sigma^2}{r^2}. \quad (19)$$

Applying Assumption 3 to the above inequality concludes the proof. ■

Combining the above two lemmas, we have the following convergence result. Denote the minimum of the objective in Problem (1) as F^* and the initial optimality gap as $\Delta = F(W^0) - F^*$.

Theorem 1: Let Assumption 1-4 hold. Consider the FedKRSO given by Algorithm 1 under SGD setting without momentum acceleration, and with interval number $I = 1$. Then, with the learning rate chosen such that $\eta \leq \mathcal{O}(\frac{r^2}{JLd_n^2})$, the iterates generated by the algorithm satisfy

$$\begin{aligned} \frac{1}{T+1} \sum_{t=0}^T \mathbb{E}[\|\nabla F(W^t)\|_F^2] &\leq \frac{64d_n^2 L \Delta}{r^2 T} + 14\sqrt{\frac{Ld_n \Delta}{rT}} \\ &\quad + 24 \left(\frac{d_n^2 L \Delta \sigma}{r^2 T J^{\frac{1}{2}}} \right)^{\frac{2}{3}} + 8\sqrt{\frac{d_n^2 L \Delta \sigma^2}{r^2 N JT}}. \end{aligned} \quad (20)$$

Proof: Substituting (19) into (12) gives

$$\begin{aligned} \mathbb{E}[F(W^{t+1})] &\leq \mathbb{E}[F(W^t)] + \frac{2Ld_n J^2 \eta^2}{r} \zeta^2 + \frac{LJ\eta^2 d_n^2 \sigma^2}{Nr^2} \\ &\quad + \left(-\frac{J\eta}{2} + \frac{2Ld_n J^2 \eta^2}{r} \right) \mathbb{E}[\|\nabla F(W^t)\|_F^2] \\ &\quad + \left(\frac{\eta d_n^2 L^2}{2r^2} + \frac{2Jd_n^2 \eta^2 L^3}{r^2} \right) \left(\frac{8J^3 \eta^2 d_n}{r} \mathbb{E}[\|\nabla F(W^t)\|_F^2] \right. \\ &\quad \left. + \frac{8J^3 \eta^2 d_n}{r} \zeta^2 + \frac{4J^2 d_n^2 \eta^2 \sigma^2}{r^2} \right) \\ &\leq \mathbb{E}[F(W^t)] - \frac{J\eta}{4} \mathbb{E}[\|\nabla F(W^t)\|_F^2] \\ &\quad + \left(\frac{3Ld_n J^2 \zeta^2}{r} + \frac{LJd_n^2 \sigma^2}{Nr^2} \right) \eta^2 + \frac{4J^2 L^2 d_n^4 \sigma^2}{r^4} \eta^3, \end{aligned} \quad (21)$$

where the last inequality is due to the condition on the learning rate. Taking telescoping cancellation over t gives

$$\begin{aligned} \frac{1}{T+1} \sum_{t=0}^T \mathbb{E}[\|\nabla F(W^t)\|_F^2] &\leq \frac{4(F(W^0) - F^*)}{J\eta T} \\ &\quad + \left(\frac{12Ld_n J \zeta^2}{r} + \frac{4Ld_n^2 \sigma^2}{Nr^2} \right) \eta + \frac{16JL^2 d_n^4 \sigma^2}{r^4} \eta^2. \end{aligned} \quad (22)$$

The theorem follows by applying [38, Lemma 15]. ■

VI. EVALUATION

In this section, we evaluate FedKRSO comprehensively from multiple perspectives, including model performance, memory usage, and communication cost. Specifically, we first compare the model performance of FedKRSO with several baselines under different levels of data heterogeneity. Next, we then analyze its efficiency in terms of GPU memory footprint and communication overhead. Finally, we conduct an ablation study to examine the impacts of interval length J and total seed number K on the model performance in FedKRSO.

A. Experimental Setup

Datasets and models. We evaluate the performance of FedKRSO on GLUE [39]: a benchmark for evaluating the performance of NLP models on eight tasks, namely, CoLA (grammatical acceptability), SST-2 (sentiment polarity), MRPC (paraphrase detection), QQP (duplicate-question detection), STS-B (semantic-similarity regression), MNLI (three-way natural-language inference), QNLI (question-answer entailment), and RTE (binary natural-language inference). We conducted the FL experiments with $N = 10$ clients. For each task, we consider both IID and non-IID settings, where the non-IID data distributions across clients are generated using a Dirichlet distribution following [40]. We use two pre-trained language models as base models: RoBERTa-base (125M) and RoBERTa-large (350M) [41].

Baselines. We consider the following three baselines for performance comparison:

- **FedFFT** [2]: FedAvg with FFT for local training, which serves as the upper bound for model performance but is not feasible in resource-constrained edge environments.
- **Fedit** [10]: FedAvg with LoRA for local training, which is a common baseline in federated fine-tuning.
- **FFA-LoRA** [9]: FedIT with partially frozen LoRA modules, which serves as the state-of-the-art baseline in federated LoRA fine-tuning and offers improved memory and communication efficiency.

System settings. In all methods, clients perform a total of 100 local iterations in each FL round using the AdamW optimizer with a cosine decay scheduler. For all LoRA-based methods, we set the rank to 4. For our method, we set $K = 10$ and search for the optimal interval length J within the set of $\{10, 20, 50, 100\}$.

The default value for batch size in all experiments is 16, except for CoLA, which uses 32. The total number of FL rounds was 30. Each experiment is run with three random seeds, and we report the average results. All experiments are conducted on a Linux server equipped with four NVIDIA Quadro RTX 8000 GPUs using PyTorch.

B. Experimental Results

1) *Fine-Tuning Model Performance*: GLUE uses task-specific evaluation metrics: CoLA is evaluated with the Matthews correlation coefficient (MCC), STS-B with Pearson correlation (Corr), and all other tasks with accuracy (Acc). Table II reports the performance of all methods under these metrics on both IID and non-IID (Dirichlet distribution with $\alpha = 0.5$ and $\alpha = 0.25$) settings, using RoBERTa-base and RoBERTa-large, respectively.

From the results, we have the following observations. FedFFT consistently achieves the highest overall performance, with a noticeable margin over LoRA-based baselines. This gap further increases under non-IID settings, increasing from 1.1 (IID) to 2.65 ($\alpha = 0.5$) on RoBERTa-base, highlighting the needs for FFT, consistent with observations in [14].

Second, our proposed method, FedKRSO, outperforms both FedIT and FFA-LoRA across all setting and better approaches

FedFFT. Also, under non-IID scenarios, all methods exhibit performance degradation, which is consistent with prior findings [9], [14], [30]. However, compared to FedIT, FedKRSO experiences a smaller performance drop, achieving better performance over FedIT by 0.65 and 1.17 on RoBERTa-base under $\alpha = 0.5$ and $\alpha = 0.25$, respectively. As data heterogeneity increases, the larger performance gap between FedKRSO and FedIT underscores the effectiveness of FedKRSO in highly heterogeneous data settings.

Third, RoBERTa-large experiences less performance degradation than RoBERTa-base when fine-tuned with LoRA under data heterogeneity. For instance, under IID settings, the performance gap between FedIT and FedFFT is only 0.63 on RoBERTa-large, compared to 1.07 on RoBERTa-base. This trend aligns with the findings in [20], suggesting that larger models are more robust to PEFT methods due to greater parameter redundancy.

2) *Memory and Communication Efficiency*: Table III summarizes the GPU memory usage and communication overhead of different approaches. The reported communication overhead includes both downlink and uplink costs, while GPU memory usage refers to the peak GPU memory footprint at each client during training, as measured by `nvidia-smi`.

From the table, we can observe that FedKRSO maintains the least peak GPU memory footprint, requiring 2.65 GB and 5.00 GB for RoBERTa-base and RoBERTa-large models, respectively. This memory efficiency primarily stemmed from FedKRSO’s subspace optimization strategy as analyzed in Section IV-B. Also, FedKRSO demonstrated significantly lower communication costs compared to FedFFT for RoBERTa-base and RoBERTa-large. Notably, while FedKRSO incurred slightly higher communication overhead than FedIT, the gap between them was relatively small. In contrast, FedFFT required over $300\times$ higher communication cost than FedKRSO for RoBERTa-base. A similar trend holds for RoBERTa-large. These results suggest that FedKRSO achieves a strong balance between performance and communication efficiency, offering a substantial improvement over FedFFT while maintaining communication costs in the same order of magnitude as LoRA-based baselines.

3) *Ablation Study*: In this part, we conduct an ablation study to investigate the impacts of two key hyperparameters on the model performance in FedKRSO: interval length J and seed number K . These studies aim to provide a deeper understanding of how each hyperparameter influences the effectiveness and robustness of FedKRSO.

Effect of interval length. To investigate the effect of the local training interval on the performance of FedKRSO, we conducted an ablation study by varying the interval length $J \in \{10, 20, 50, 100\}$ under different data heterogeneity settings. The results are shown in Fig 2a.

We observe that FedKRSO is robust to the choice of the interval length across all settings, with only minor variations in the average model performance. The interval length introduces a trade-off between subspace exploration and momentum preservation. A smaller interval enabled more frequent sub-

TABLE II
PERFORMANCE COMPARISON ON GLUE UNDER IID AND NON-IID (DIRICHLET DISTRIBUTION WITH CONCENTRATION PARAMETER α) SETTINGS USING PRE-TRAINED (A) ROBERTA-BASE AND (B) ROBERTA-LARGE. BEST AND SECOND-BEST RESULTS ARE IN **BOLD** AND UNDERLINE, RESPECTIVELY.

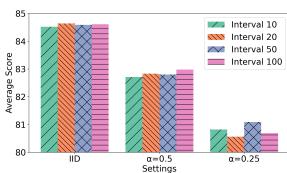
Setting	Task	FedIT	FFA-LoRA	FedFFT	FedKRSO	Setting	Task	FedIT	FFA-LoRA	FedFFT	FedKRSO
IID	CoLA	59.98	59.26	<u>60.08</u>	60.28	IID	CoLA	66.44	62.90	66.93	<u>66.60</u>
	MRPC	88.15	86.36	89.95	88.97		MRPC	<u>90.93</u>	88.24	91.18	90.69
	SST-2	<u>94.30</u>	93.04	94.53	94.04		SST-2	95.64	95.15	96.41	95.91
	STS-B	90.66	90.01	91.11	<u>90.89</u>		STS-B	92.32	91.38	92.26	<u>92.27</u>
	QNLI	<u>90.05</u>	86.97	91.45	89.55		QNLI	93.26	91.36	94.28	<u>93.96</u>
	QQP	85.16	82.96	87.22	<u>85.62</u>		QQP	87.08	85.37	88.28	<u>87.33</u>
	MNLI-mm	83.43	79.79	84.88	<u>83.86</u>		MNLI-mm	88.51	86.73	89.25	<u>88.71</u>
$\alpha = 0.5$	Average	84.53	82.63	85.60	84.74		Average	87.74	85.87	88.37	<u>87.92</u>
	CoLA	56.05	55.99	59.76	<u>59.32</u>	$\alpha = 0.5$	CoLA	<u>66.77</u>	59.23	66.48	66.90
	MRPC	84.64	80.23	87.91	<u>86.44</u>		MRPC	85.78	79.49	89.05	88.48
	SST-2	<u>93.12</u>	92.24	93.27	92.58		SST-2	95.07	94.57	95.72	<u>95.15</u>
	STS-B	88.94	88.44	<u>88.83</u>	88.83		STS-B	91.27	89.93	90.95	<u>91.02</u>
	QNLI	<u>89.12</u>	86.11	90.38	87.80		QNLI	<u>93.28</u>	91.20	93.86	92.59
	QQP	84.47	82.30	86.43	<u>85.06</u>		QQP	86.34	84.25	87.65	86.84
$\alpha = 0.25$	MNLI-mm	81.03	76.81	83.24	<u>81.87</u>		MNLI-mm	<u>87.14</u>	84.84	87.99	86.63
	Average	82.48	80.30	84.26	<u>83.13</u>		Average	86.52	83.36	87.39	<u>86.80</u>
	CoLA	56.83	56.34	59.86	<u>59.45</u>	$\alpha = 0.25$	CoLA	64.94	56.60	68.03	<u>67.10</u>
	MRPC	77.45	72.63	85.54	<u>82.68</u>		MRPC	77.29	73.69	86.60	<u>85.62</u>
	SST-2	<u>91.90</u>	89.68	92.24	91.40		SST-2	<u>94.95</u>	94.15	95.03	<u>95.03</u>
	STS-B	88.70	87.59	89.10	<u>89.02</u>		STS-B	<u>90.80</u>	88.38	90.27	90.88
	QNLI	<u>87.03</u>	82.71	88.36	85.75		QNLI	90.61	86.29	92.52	<u>90.80</u>
	QQP	83.09	80.79	84.89	<u>83.51</u>		QQP	84.93	81.57	86.67	<u>85.61</u>
	MNLI-mm	77.67	70.60	81.21	<u>79.03</u>		MNLI-mm	85.03	81.20	86.58	<u>85.35</u>
	Average	80.38	77.19	83.03	<u>81.55</u>		Average	84.08	80.27	86.53	<u>85.77</u>

(a) RoBERTa-base

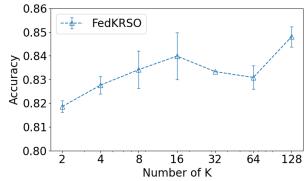
(b) RoBERTa-large

TABLE III
PER-ROUND COMMUNICATION OVERHEAD AND GPU MEMORY USAGE OF DIFFERENT APPROACHES, MEASURED IN GB.

Approach	RoBERTa-base		RoBERTa-large	
	Comm.	Mem.	Comm.	Mem.
FedIT	0.00958	2.87	0.0215	5.62
FFA-LoRA	0.00688	2.70	0.0144	5.05
FedKRSO	0.02913	2.65	0.0738	5.00
FedFFT	0.938	4.31	2.67	9.43



(a) Average scores across GLUE tasks with different interval lengths.



(b) Accuracy on the MRPC task with different K .

Fig. 2. Ablation study of FedKRSO on RoBERTa-base.

space changes, potentially enhancing parameter space exploration and improving performance in certain tasks, particularly under non-IID settings. For example, when $\alpha = 0.25$, an interval of 10 achieved second best performance. Conversely, larger interval length (e.g., 50 or 100) reduced the frequency of subspace switching, which helps retain accumulated momentum and leads to more stable optimization dynamics. For example, when $\alpha = 0.5$, an interval of 100 achieved best performance.

Effect of seed number. To understand the relationship between the number of random projection matrices (i.e., seeds

K) and the accuracy of FedKRSO, we examine its performance under varying K , as shown in Figure 2b. We observe that the accuracy improves as K increases, suggesting that a larger number of subspaces allows the model to explore more diverse directions. When K is too small, the performance of FedKRSO degrades due to limited expressiveness stemming from an insufficient number of subspaces. It is important to note that although settings such as $K = 128$ yield better accuracy, they also incur higher communication costs, as more model updates need to be accumulated. Therefore, the choice of K should strike a balance: it should be large enough to ensure model expressiveness, but not so large as to impose excessive communication overhead.

VII. CONCLUSION

This paper proposes FedKRSO, a novel framework for federated fine-tuning of LLMs under resource-constrained edge environments. By using random subspace optimization with a finite set of random seeds, FedKRSO can enhance communication and memory efficiency while maintaining high model performance. Convergence properties of FedKRSO are provided, and experimental results demonstrate the advantages of FedKRSO compared to existing methods. In the future, we will extend FedKRSO to multimodal LLMs and evaluate its effectiveness on other benchmarks.

REFERENCES

- [1] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, “A survey of large language models,” *arXiv preprint arXiv:2303.18223*, vol. 1, no. 2, 2023.
- [2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.
- [3] Z. Zhang, Z. Gao, Y. Guo, and Y. Gong, “Scalable and low-latency federated learning with cooperative mobile edge networking,” *IEEE Transactions on Mobile Computing*, vol. 23, no. 1, pp. 812–822, 2022.
- [4] Y. Guo, Y. Sun, R. Hu, and Y. Gong, “Hybrid local SGD for federated learning with heterogeneous communications,” in *International Conference on Learning Representations*, 2022.
- [5] Z. Zhang, Z. Gao, Y. Guo, and Y. Gong, “Heterogeneity-aware cooperative federated edge learning with adaptive computation and communication compression,” *IEEE Transactions on Mobile Computing*, vol. 24, no. 3, pp. 2073–2084, 2024.
- [6] Z. Gao, Z. Zhang, Y. Zhang, Y. Gong, and Y. Guo, “Heterogeneity-aware resource allocation and topology design for hierarchical federated edge learning,” *IEEE Internet of Things Journal*, vol. 12, no. 19, pp. 39 803–39 816, 2025.
- [7] Z. Gao, Z. Zhang, Y. Guo, and Y. Gong, “Federated adaptive fine-tuning of large language models with heterogeneous quantization and LoRA,” in *IEEE International Conference on Computer Communications*, 2025.
- [8] S.-Y. Yeh, Y.-G. Hsieh, Z. Gao, B. B. Yang, G. Oh, and Y. Gong, “Navigating text-to-image customization: From LyCORIS fine-tuning to model evaluation,” in *International Conference on Learning Representations*, 2023.
- [9] Y. Sun, Z. Li, Y. Li, and B. Ding, “Improving LoRA in privacy-preserving federated learning,” in *International Conference on Learning Representations*, 2024.
- [10] J. Zhang, S. Vahidian, M. Kuo, C. Li, R. Zhang, T. Yu, G. Wang, and Y. Chen, “Towards building the FederatedGPT: Federated instruction tuning,” in *IEEE ICASSP*, 2024, pp. 6915–6919.
- [11] Z. Wang, Z. Shen, Y. He, G. Sun, H. Wang, L. Lyu, and A. Li, “FLoRA: Federated fine-tuning large language models with heterogeneous low-rank adaptations,” in *38th Conference on Neural Information Processing Systems*, 2024.
- [12] Y. Yan, C. Feng, W. Zuo, R. S. M. Goh, Y. Liu, and L. Zhu, “Federated residual low-rank adaptation of large language models,” in *International Conference on Learning Representations*, 2025.
- [13] J. Bai, D. Chen, B. Qian, L. Yao, and Y. Li, “Federated fine-tuning of large language models under heterogeneous tasks and client resources,” in *38th Conference on Neural Information Processing Systems*, 2024.
- [14] S. Babakniya, A. Elkordy, Y. Ezzeldin, Q. Liu, K.-B. Song, M. EL-Khamy, and S. Avestimehr, “SLoRA: Federated parameter efficient fine-tuning of language models,” in *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*, 2023.
- [15] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “LoRA: Low-rank adaptation of large language models,” in *International Conference on Learning Representations*, 2022.
- [16] G. Chen, F. Liu, Z. Meng, and S. Liang, “Revisiting parameter-efficient tuning: Are we really there yet?” in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022.
- [17] G. Pu, A. Jain, J. Yin, and R. Kaplan, “Empirical analysis of the strengths and weaknesses of PEFT techniques for LLMs,” in *International Conference on Learning Representations 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2023.
- [18] N. Houldsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, “Parameter-efficient transfer learning for NLP,” in *International Conference on Machine Learning*, 2019.
- [19] X. L. Li and P. Liang, “Prefix-tuning: Optimizing continuous prompts for generation,” 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)*, 2021, pp. 4582–4597.
- [20] V. Lialin, V. Deshpande, and A. Rumshisky, “Scaling down to scale up: A guide to parameter-efficient fine-tuning,” *arXiv preprint arXiv:2303.15647*, 2023.
- [21] S. Malladi, T. Gao, E. Nichani, A. Damian, J. D. Lee, D. Chen, and S. Arora, “Fine-tuning language models with just forward passes,” in *37th Conference on Neural Information Processing Systems*, 2023.
- [22] S. Liu, P.-Y. Chen, X. Chen, and M. Hong, “signSGD via zeroth-order oracle,” in *International Conference on Learning Representations*, 2019.
- [23] X. Chen, S. Liu, K. Xu, X. Li, X. Lin, M. Hong, and D. Cox, “ZO-AdaMM: Zeroth-order adaptive momentum method for black-box optimization,” in *33rd Conference on Neural Information Processing Systems*, 2019.
- [24] Y. Zhang, P. Li, J. Hong, J. Li, Y. Zhang, W. Zheng, P.-Y. Chen, J. D. Lee, W. Yin, M. Hong *et al.*, “Revisiting zeroth-order optimization for memory-efficient LLM fine-tuning: A benchmark,” in *International Conference on Machine Learning*, 2024.
- [25] Y. Hao, Y. Cao, and L. Mou, “FLORA: Low-rank adapters are secretly gradient compressors,” in *International Conference on Machine Learning*, 2024.
- [26] J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, and Y. Tian, “GaLore: Memory-efficient LLM training by gradient low-rank projection,” in *International Conference on Machine Learning*, 2024.
- [27] Y. Chen, Y. Zhang, Y. Liu, K. Yuan, and Z. Wen, “A memory efficient randomized subspace optimization method for training large language models,” in *International Conference on Machine Learning*, 2025.
- [28] X.-H. Liu, Y. Du, J. Wang, and Y. Yu, “On the optimization landscape of low rank adaptation methods for large language models,” in *International Conference on Learning Representations*, 2025.
- [29] Y. He, P. Li, Y. Hu, C. Chen, and K. Yuan, “Subspace optimization for large language models with convergence guarantees,” in *International Conference on Machine Learning*, 2025.
- [30] P. Guo, S. Zeng, Y. Wang, H. Fan, F. Wang, and L. Qu, “Selective aggregation for low-rank adaptation in federated learning,” in *International Conference on Learning Representations*, 2025.
- [31] W. Fang, Z. Yu, Y. Jiang, Y. Shi, C. N. Jones, and Y. Zhou, “Communication-efficient stochastic zeroth-order optimization for federated learning,” *IEEE Transactions on Signal Processing*, vol. 70, pp. 5058–5073, 2022.
- [32] M. Xu, D. Cai, Y. Wu, X. Li, and S. Wang, “FwdLLM: Efficient federated finetuning of large language models with perturbed inferences,” in *Proceedings of the 2024 USENIX Conference on Usenix Annual Technical Conference*, 2024.
- [33] Z. Qin, D. Chen, B. Qian, B. Ding, Y. Li, and S. Deng, “Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes,” in *International Conference on Machine Learning*, 2024.
- [34] H. Zhao, X. Xie, C. Fang, and Z. Lin, “SEPARATE: A simple low-rank projection for gradient compression in modern large-scale model training process,” in *International Conference on Learning Representations*, 2025.
- [35] Y. Shu, W. Hu, S. Ng, B. K. H. Low, and F. R. Yu, “Ferret: Federated full-parameter tuning at scale for large language models,” in *International Conference on Machine Learning*, 2025.
- [36] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations*, 2015.
- [37] Y. Chen, Y. Zhang, L. Cao, K. Yuan, and Z. Wen, “Enhancing zeroth-order fine-tuning for language models with low-rank structures,” in *International Conference on Learning Representations*, 2025.
- [38] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich, “A unified theory of decentralized SGD with changing topology and local updates,” in *International Conference on Machine Learning*, 2020.
- [39] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “GLUE: A multi-task benchmark and analysis platform for natural language understanding,” in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2018.
- [40] H. Hsu, H. Qi, and M. Brown, “Measuring the effects of non-identical data distribution for federated visual classification,” in *International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with NeurIPS 2019*, 2019.
- [41] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “RoBERTa: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.