MIRA: A Method of Federated Multi-Task Learning for Large Language Models

Ahmed Elbakary, *Graduate Student Member, IEEE*, Chaouki Ben Issaid, *Member, IEEE*, Tamer ElBatt, Karim Seddik, *Senior Member, IEEE*, and Mehdi Bennis, *Fellow, IEEE*

***Abstract*—In this letter, we introduce a method for fine-tuning Large Language Models (LLMs), inspired by Multi-Task learning in a federated manner. Our approach leverages the structure of each client's model and enables a learning scheme that considers other clients' tasks and data distribution. To mitigate the extensive computational and communication overhead often associated with LLMs, we utilize a parameter-efficient fine-tuning method, specifically Low-Rank Adaptation (LoRA), to reduce the number of trainable parameters. Experimental results, with different datasets and models, demonstrate the proposed method's effectiveness compared to existing frameworks for federated fine-tuning of LLMs in terms of global and local performances. The proposed scheme outperforms existing baselines by achieving lower local loss for each client, while maintaining comparable global performance.**

***Index Terms*—Large language models, federated learning, multi-task learning, deep neural networks.**

I. Introduction

PRE-TRAINED Large Language Models (LLMs) are proven to be few-shot learners [1], which means that they can perform a diverse set of tasks without retraining on those specific tasks. This property is attributed mainly to two factors: the wide range of their training data and the scale of the architecture. While in-context learning [2] might be enough to adapt the LLM to the task at hand, this might not be possible without further model fine-tuning. The problem arises when adapting these models to a domain-specific area where the data comes from a different distribution than the training data itself. This requires further fine-tuning of the models so they can perform at a reasonable level. Gathering such domain-specific data might be impractical because of the associated cost of data collection in addition to privacy concerns that might arise.

Federated learning (FL) [3] is one technique that can be leveraged to solve such a problem without revealing local data. The idea is to enable different clients to train a model jointly to solve a specific problem by leveraging local data for local training and sharing only the model's parameters, instead of the local data. Traditional FL algorithms aim to train a global model that can be deployed across all participating clients, regardless of the underlying distribution of their data. Multiple frameworks already exist to leverage FL for fine-tuning LLMs. For example, Zhang et al. [4] proposed the usage of different Parameter-efficient fine-tuning (PEFT) methods like prompt tuning to make the fine-tuning process of LLMs communication efficient. The idea of prompt tuning is to add a set of virtual tokens, often called a soft prompt, to the original prompt, i.e., the hard prompt. Then, it tries to optimize the soft prompt to generate the desired output. In [5], the authors made use of the fact that most FL settings operate with edge devices designed for inference not training. This characteristic leads to solving the problem using a zeroth-order method without the need for backpropagation (BP).

The main problem with the existing federated fine-tuning approaches for LLMs is the fact that they all learn a single global model based on model averaging, which might not be the optimal solution for highly heterogeneous settings. Moreover, a single global model might not be the optimal solution when the model is expected to perform well across different tasks where the data distribution differs significantly. For example, different medical entities around the world might train a chatbot system to interact with their patients. A single global model may fail to capture the nuanced medical terminologies, treatment protocols, or diagnostic criteria unique to each region or institution, thus reducing the model's effectiveness in real-world clinical settings. Federated Multi-Task learning (FMTL) [6] provides a promising framework that can be leveraged to solve the issue of heterogeneous clients. Several approaches have been proposed to adapt MTL to model FL problems. For example, in [7], the authors aligned the locally learned model and the global one using regularization, improving both model personalization and fairness among clients. The existence of a hidden relationship between clients' models was assumed in [8], and the Laplacian matrix was leveraged to capture this relationship.

The second issue is concerned with the computation and communication costs associated with LLMs. An LLM is, at its core, a very deep neural network, with millions of weights. Most state-of-the-art techniques for training neural networks utilize gradient-based methods like stochastic gradient descent (SGD). An SGD step requires computing the gradient of the loss function with respect to (w.r.t) the model's parameters using BP. However, BP suffers from one major bottleneck, which is the memory footprint needed to compute the gradient. Along with storing models' parameters, gradient-based methods require the activation of individual neurons in the network

Received 7 November 2024; revised 8 January 2025; accepted 31 January 2025. Date of publication 7 February 2025; date of current version 14 October 2025. This work was supported by the European Union through the Project 6G-INTENSE under Grant 101139266. The associate editor coordinating the review of this article and approving it for publication was J. Wang. *(Corresponding author: Ahmed Elbakary.)*

Ahmed Elbakary, Chaouki Ben Issaid, and Mehdi Bennis are with the Centre for Wireless Communications, University of Oulu, 90014 Oulu, Finland (e-mail: ahmed.elbakary@oulu.fi; chaouki.benissaid@oulu.fi; mehdi.bennis@oulu.fi).

Tamer ElBatt is with the Department of Computer Science and Engineering, American University in Cairo, New Cairo City 11835, Egypt, and also with the Department of Electronics and Communications Engineering, Cairo University, Giza 12613, Egypt (e-mail: tamer.elbatt@aucegypt.edu).

Karim Seddik is with the Department of Electronics and Communications Engineering, American University in Cairo, New Cairo City 11835, Egypt (e-mail: kseddik@aucegypt.edu).

Digital Object Identifier 10.1109/LNET.2025.3539810

to be stored, enabling the optimizer to compute a backward pass, which is needed to compute the gradient and the SGD update. This comes with almost a minimal cost in the case of shallow or deep networks but not very deep networks like LLMs. In fact, in the case of a billion-sized network, like an LLM, storing both the model's parameters and the activation values would imply that tens of gigabytes are needed for such a model. Since those kinds of models are usually trained using graphics processing units (GPUs), the cost might be quite high for even a very limited training time and might take days or weeks unless a very powerful GPU cluster is available. PEFT methods solve this problem by adding a small set of parameters to the pre-trained model and only training the newly added parameters. Low-Rank Adaptation (LoRA) [9] is one technique that enables such decomposition, where the original weight matrix is decomposed into two sub-matrices. The two matrices are then initialized with Gaussian and zero initialization, respectively. During training, the original weight matrix is frozen, while LoRA's weights are the only trainable parameters to be fine-tuned. This fine-tuning technique ensures that the stored activation values and the optimizer's state are minimal in comparison to that of the original model. We leverage both the FMTL paradigm and PEFT and apply them to federate the process of finetuning LLMs. The contributions of this letter are multi-fold:

- We propose MIRA, an FMTL-based approach for fine-tuning LLMs in a federated manner to address the issue of data heterogeneity among clients and enable each client to learn a custom model tailored to its data distribution.
- We leverage a PEFT mechanism, specifically LoRA, to significantly reduce the computational and communication overhead associated with LLMs, ensuring efficient and scalable fine-tuning.
- Extensive evaluation is carried out using two different LLMs: Data-juicer [10] and GPT2-large [1], and two datasets: Natural Instructions (NI) [11] and Dolly15-k [12], to show the effectiveness of the proposed scheme.

II. System Model and Problem Formulation

We assume a network of K clients, with the set of all clients defined as $\mathcal{N} = \{1, \ldots, K\}$ and $\mathcal{N}_k = \mathcal{N} \setminus \{k\}$ being the set of all neighboring clients for client k. Clients communicate only with the parameter server (PS), but not with each other directly. Each client trains a separate model $\boldsymbol{w}_k \in \mathbb{R}^d$, according to its local data $(\boldsymbol{X}_k, \boldsymbol{y}_k)$, where $\boldsymbol{X}_k$ represents the feature matrix, the instruction to the LLM in our case, and $\boldsymbol{y}_k$ represents the label vectors, the expected output from the LLM. Along with that, clients have their own tasks t_k and objective function $f_k(\boldsymbol{w}_k) : \mathbb{R}^d \to \mathbb{R}, \forall k \in \mathcal{N}$. The setting is depicted in Figure 1. We model the interactions among clients as a connected graph $\mathcal{G}$, with an adjacency matrix $\boldsymbol{M}$ that quantifies the similarity between tasks of different clients. The entries of the matrix $\boldsymbol{M}$ represent how strong the relationship is between two clients, i.e., the degree of similarity between the two tasks.

Let $\boldsymbol{W} = [\boldsymbol{w}_1^T, \ldots, \boldsymbol{w}_K^T] \in \mathbb{R}^{dK}$ be a concatenation of all clients' models and let $\boldsymbol{D} = \operatorname{diag}[\delta_1, \delta_2, \ldots, \delta_K] \in \mathbb{R}^{K \times K}$ be a diagonal matrix that represents the summation of all

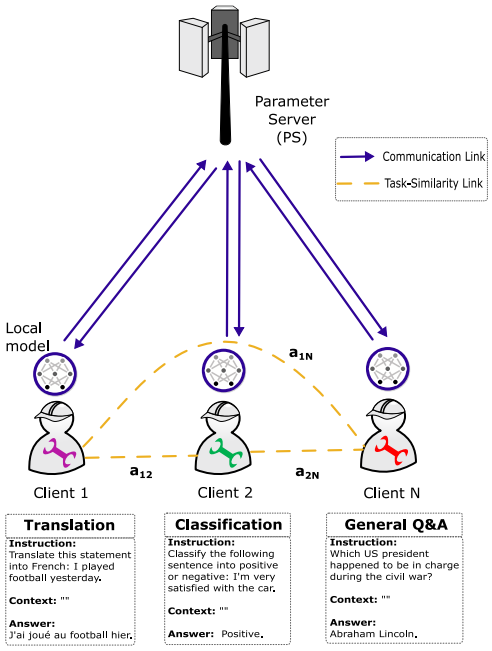


Fig. 1. A schematic illustration of the FMTL framework for fine-tuning LLMs. Each client is equipped with the same pre-trained model and a different task.

neighboring clients connections where $\delta_k = \sum_{\ell \in \mathcal{N}_k} a_{k\ell}$. The Laplacian matrix of the graph is defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{M}$, and its extended version as $\boldsymbol{\mathcal{L}} = \boldsymbol{L} \otimes \boldsymbol{I}_d$, where $\otimes$ denotes the Kronecker product of two matrices and $\boldsymbol{I}_d$ is the identity matrix. The FMTL problem aims to minimize the overall objective function, which comprises the global loss and a regularization term enforcing task similarity as follows

$$\min_{\boldsymbol{W}} J(\boldsymbol{W}) = F(\boldsymbol{W}) + \lambda R(\boldsymbol{W}), \tag{1}$$

where λ is a regularization hyperparameter which controls how much weight a client gives to other clients' models. In other words, if $\lambda = 0$, the problem is converted into a traditional FL problem with full local training and no collaboration between clients is taken into account. On the other hand if $\lambda > 0$, we encourage similar tasks/clients to align their models close to each other. The global loss $F(\boldsymbol{W})$ is defined as the sum of local losses across all clients

$$F(\boldsymbol{W}) = \sum_{k=1}^{K} f_k(\boldsymbol{w}_k). \tag{2}$$

The Laplacian regularization term $\mathcal{R}(\boldsymbol{W})$ captures the similarity between clients' task models and is defined as

$$\mathcal{R}(\boldsymbol{W}) = \boldsymbol{W}^T \boldsymbol{\mathcal{L}} \boldsymbol{W} = \frac{1}{2} \sum_{k=1}^{N} \sum_{\ell \in \mathcal{N}_k} a_{k\ell} \|\boldsymbol{w}_k - \boldsymbol{w}_\ell\|^2. \tag{3}$$

In this formulation, $f_k(\boldsymbol{w}_k)$ represents the local loss function at client k, computed by evaluating a sample $\boldsymbol{x}_k^{(i)}$ using the model $\boldsymbol{w}_k$. The norm $\|\cdot\|$ denotes the Euclidean norm, and $a_{k\ell}$ quantifies the similarity between tasks of clients k and ℓ. In other words, with higher values of $a_{k,\ell}$, it indicates stronger relationships. The objective function $J(\boldsymbol{W})$ consists of two

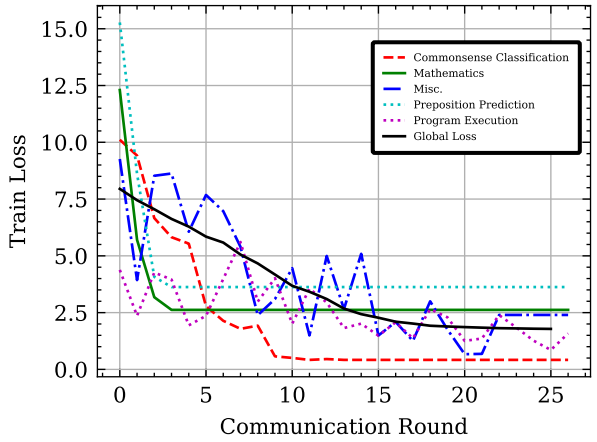


Fig. 2. Global model versus per-task model performance.

components: the global loss, which is optimized locally by each client, and the regularization term, which is handled by the server. To minimize $J(\boldsymbol{W})$, we perform both local and global updates iteratively. At each client k, we perform R local optimization steps to update $\boldsymbol{w}_k$. After completing the local updates, the server minimizes the regularization term $\mathcal{R}(\boldsymbol{W})$ by adjusting the clients' models based on their similarities. Specifically, the server updates each client's model as follows

$$\boldsymbol{w}_k^{(t+1)} = \boldsymbol{w}_{k,R}^{(t)} - \eta\lambda \sum_{\ell \in \mathcal{N}_k} a_{k\ell}\Big(\boldsymbol{w}_{k,R}^{(t)} - \boldsymbol{w}_{\ell,R}^{(t)}\Big), \forall k \in \mathcal{N}, \tag{4}$$

where $\boldsymbol{w}_{k,R}^{(t)}$ is the locally updated model of client k after R steps at iteration t, and η is the server's learning rate.

To motivate the FMTL setting, we conduct a preliminary experiment with the DataJuicer LLM [10] on the NI dataset [11]. We measure the performance of the global learned model on different tasks by reporting the average loss per task using one of the latest proposed methods, FedKSeed [13] against learning a fully local model. As we can see from Figure 2, the performance of the model varies across tasks. Most importantly, the global model's performance is worse than the local ones in certain tasks with a high variance in the case where the size of the local dataset is small. This can be attributed to the fact that learning a single global model, as in the traditional FL approach, is sub-optimal in the case of different tasks or data distributions across clients. On the other hand, learning a fully local model is near optimal in case sufficient compute and data are available, which is not the case for most edge devices. To address the limitation of the traditional FL approach, we hypothesize that learning a separate model per client can solve this problem, enabling clients to handle their local tasks better while still taking into account the structure and the similarity of other clients' tasks.

III. Proposed Algorithm

In this section, we propose MIRA, a novel algorithm that enables fine-tuning LLMs in an FMTL manner with efficiency in terms of *computation* and *communication* costs. Our approach leverages LoRA as a PEFT method, which addresses the memory footprint issue by reducing the number of trainable parameters in the model. We assume that each client is equipped with a pre-trained model $\boldsymbol{W}^0 \in \mathbb{R}^{d\times v}$. This weight matrix is then decomposed into two smaller matrices such that

$$\boldsymbol{W}^0 + \Delta\boldsymbol{W} = \boldsymbol{W}^0 + \boldsymbol{B}\boldsymbol{A}, \tag{5}$$

where $\Delta\boldsymbol{W}$ represents the accumulated gradient updates, $B \in \mathbb{R}^{d\times r}$ and $\boldsymbol{A} \in \mathbb{R}^{r\times v}$, where r being the rank of both $\boldsymbol{A}$, and $\boldsymbol{B}$. Instead of optimizing the original weights $\boldsymbol{W}^0$, we only update $\boldsymbol{A}$ and $\boldsymbol{B}$. LoRA hypothesizes that projecting LLMs into lower-dimensional spaces preserves performance, allowing us to optimize these smaller matrices efficiently. In the rest of this letter, we refer to $\boldsymbol{B}$ and $\boldsymbol{A}$ as $\Delta\boldsymbol{W}$.

At communication round t, the server uniformly selects a subset of clients $S^{(t)}$. Each client, $k \in S^{(t)}$, performs R local rounds of updates on the lower-dimensional matrix, $\Delta\boldsymbol{W}_k$, such that

$$\Delta\boldsymbol{W}_{k,R}^{(t)} = \textbf{InstructionTuning}\Big(\Delta\boldsymbol{W}_{k,r}^{(t)}\Big), r \in \{1,\dots,R\}, \tag{6}$$

where **InstructionTuning**$(\cdot)$ refers to the process of adapting the weight matrix to the new dataset by computing a forward pass, and a backward pass, and then updating the model's weights. By leveraging LoRA, our forward pass becomes

$$\boldsymbol{h} = \boldsymbol{W}^0\boldsymbol{x} + \boldsymbol{B}\boldsymbol{A}\boldsymbol{x}, \tag{7}$$

where $\boldsymbol{h}$ is the output of the forward pass, and $\boldsymbol{x}$ is a sample drawn from the dataset which is of the form of instructions $\boldsymbol{X}_k$ and the expected response from the model $\boldsymbol{y}_k$. After the forward pass, a backward pass is calculated with BP, computing the gradient w.r.t the LoRA's weights. Finally, we update the model weights using an optimizer step, yielding a locally updated model. This process continues for R local rounds. The final locally updated model $\Delta\boldsymbol{W}_{k,R}^{(t)}$ is sent to the server. The server then performs a *regularization update* to align clients with similar tasks, encouraging their models to be closer to each other. The server updates each client's matrix, $\forall k \in S^{(t)}$, as follows

$$\Delta\boldsymbol{W}_k^{(t+1)} = \Delta\boldsymbol{W}_{k,R}^{(t)} - \eta\lambda \sum_{\ell \in \mathcal{N}_k} a_{k\ell}\Big(\Delta\boldsymbol{W}_{k,R}^{(t)} - \Delta\boldsymbol{W}_{\ell,R}^{(t)}\Big). \tag{8}$$

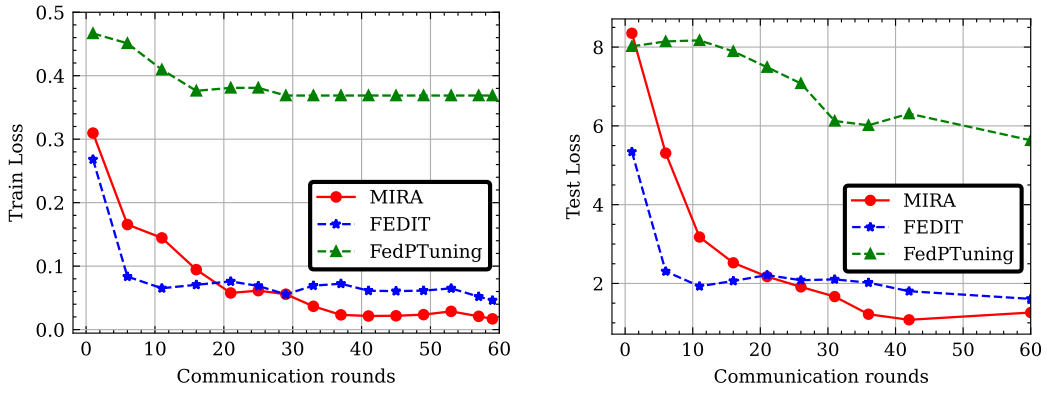
It is worth noting that the performance of the traditional aggregation scheme that averages the model's parameters might not be suitable due to the heterogeneity of the tasks. The update for non-sampled clients is given by

$$\Delta\boldsymbol{W}_k^{(t+1)} = \Delta\boldsymbol{W}_k^{(t)}, \forall k \notin S^{(t)}. \tag{9}$$

The process continues for T communication rounds until convergence as depicted in Algorithm 1.

IV. Experiments

In this section, we discuss the details of our experiments and compare the performance of our approach, MIRA, to baseline schemes of existing federated fine-tuning methods



(a) (b)

Fig. 3. Performance comparison of the proposed method and the baselines on Data-Juicer using the Natural Instruction dataset.

Algorithm 1: MIRA

Parameters: η, $(T, f, R, \mathcal{N}, \boldsymbol{W}^{(0)})$
Initialize: $\boldsymbol{B}$, $\boldsymbol{A}$
for communication round $t = 1$ to T **do**
  Sample a subset $S^{(t)}$ to initiate local training
  for each client $k \in S^{(t)}$ **do**
    if $t > 1$ **then**
      Get latest model $\Delta \boldsymbol{W}_k^{(t)}$ from the server.
      $\Delta \boldsymbol{W}_{k,0}^{(t)} = \Delta \boldsymbol{W}_k^{(t)}$
    end if
    for each local round $r = 1$ to R **do**
      $\Delta \boldsymbol{W}_{k,r+1}^{(t)} =$**InstructionTuning**$(\Delta \boldsymbol{W}_{k,r}^{(t)})$
    end for
    Send $\Delta \boldsymbol{W}_{k,R}^{(t)}$ to the server.
  end for
  On Server
  for each client $k \in S^{(t)}$ **do**
    $\Delta \boldsymbol{W}_k^{(t+1)} = \Delta \boldsymbol{W}_{k,R}^{(t)} - \eta\lambda \sum_{\ell \in \mathcal{N}_k} a_{k\ell} \left(\Delta \boldsymbol{W}_{k,R}^{(t)} - \Delta \boldsymbol{W}_{\ell,R}^{(t)} \right)$
  end for
end for

for LLMs. Afterwards, we compare the computational and communication overhead attributed to MIRA, in comparison to the baseline schemes. Finally, we analyze the effect of FMTL on the overall per-task performance.

A. Experimental Setup

We conduct our experiments using two models, Data-Juicer [10], a 1.3 billion parameters Llama-based model and GPT-2-large [1], which has around 774 million parameters. We utilize two datasets, namely NI [11] and Dolly-15k [12]. The number of communication rounds for all experiments is set to 60. We follow the same pre-processing steps described by [13] and only sample around 20% of the training set and 2% of the test set. For Dolly-15k and Natural Instruction, we consider a scenario with 80 clients, where every client has a unique local task t. At each communication round, we sample 10% of all clients to participate in the round. The metrics we utilize to assess the performance are the average train and test losses and the Rouge-L [14], a metric that is widely adopted to measure the quality of LLM's output. Rouge-L measures the longest matching sequence of words. The batch size is set to 8 to guard against any memory limitations. LoRA has two important hyperparameters, α and r. As mentioned in [9], we do not need to optimize α since it is functionally equivalent to optimizing the learning rate when working with the Adam optimizer. We vary the rank r with values $\{4, 8, 16\}$ and choose the value that gives the best performance, 16 in our case. We set the adjacency matrix M's values randomly. All algorithms are tested and run on an A100 GPU equipped with a total of 40 GB VRAM. We compare our method to two other baseline schemes that are tailored specifically for federated fine-tuning of LLMs, namely FedIT [15] and FedPTuning [16]. The best hyperparameters for all three methods are selected.

B. Performance Comparison

In Fig. 3, we compare the performance of our proposed scheme, MIRA, against the baselines. For Natural Instruction with Data-Juicer, it is observed that MIRA outperforms the two baselines. In fact, it takes around 20 communication rounds for MIRA to outperform FedIT, the closest baseline. The Rouge-L scores of the three methods are reported in Table I for the Dolly-15k dataset. We can see that MIRA significantly outperforms FedPTuning leading to an improvement in the Rouge-L scores by 6.1% and 14.3%, for the Data-Juicer and GPT2-Large models, respectively. On the other hand, MIRA remains highly competitive with FedIT achieving comparable Rouge-L scores.

C. Memory and Communication Costs

We study the computation and communication costs of our approach and the baselines. For communication cost, we quantify the number of communicated bits from the clients to the server and vice versa. We denote the upload and download link communication costs as U and D, respectively. Then, the total communication cost is $U + D$. The memory cost is estimated as a function of the original model size C_w, the

TABLE I
ROUGE-L VALUES OF THE DIFFERENT ALGORITHMS ON DOLLY-15K DATASET

Model	Algorithm	Rouge-L(%)
Data-Juicer	MIRA	**20.8**
	FedIT	19.1
	FedPTuning	14.7
GPT2-large	MIRA	26.6
	FedIT	**28.4**
	FedPTuning	12.3

TABLE II
COMMUNICATION AND MEMORY COSTS OF THE DIFFERENT ALGORITHMS

Model	Algorithm	Communication cost	Memory Cost
Data-Juicer	MIRA	12 MB	2.60 GB
	FedIT	12 MB	2.60 GB
	FedPTuning	2.22 MB	2.57 GB
GPT2-large	MIRA	7.74 MB	1.51 GB
	FedIT	7.74 MB	1.51 GB
	FedPTuning	0.71 MB	1.48 GB

TABLE III
LOCAL TEST LOSS PER CLIENT FOR SOME SELECTED SUBSET OF CLIENTS ON DATA-JUICER USING THE NATURAL INSTRUCTION DATASET

Client Task	Algorithm	Test Loss
Question Answering	MIRA	2.7
	FedIT	**1.8**
	FedPTuning	9.55
Program Execution	MIRA	**0.23**
	FedIT	0.37
	FedPTuning	0.87
Speaker Identification	MIRA	**1.87**
	FedIT	7.01
	FedPTuning	11.98
Explanation	MIRA	**1.99**
	FedIT	2.55
	FedPTuning	3.08

size of the trainable parameters $C_{\Delta w}$ and any other overhead that results from saving the optimizer's states C_O. The overall memory cost is then $C_w + C_{\Delta w} + C_O$. We report the cost of one round per client. As we can see in Table II, MIRA exhibits the same communication and memory costs as FedIT, given that the same rank is used by both of them to obtain the PEFT model. On the other hand, FedPTuning is lighter compared to them, consuming less communication and memory costs.

D. Per-Task Performance Analysis

Finally, we analyze the effect of FMTL on the overall per-task performance to understand how well MIRA adapts to individual tasks compared to the baselines. We randomly select four clients/tasks and report the local test loss. As observed in Table III, MIRA achieves a lower average loss per client/task in three out of the four clients, which indicates the effectiveness of FMTL and its adaptability to task-specific requirements in contrast to the model averaging scheme of FedIT and FedPTuning.

V. CONCLUSION

This letter presented MIRA, a novel approach to federated fine-tuning of LLMs that integrates multi-task learning principles with parameter-efficient techniques to tackle the unique challenges of federated settings. By leveraging LoRA, MIRA achieves a balance between computational efficiency and communication overhead, making it practical for real-world deployments. The key innovation lies in MIRA's ability to harmonize client-specific personalization with global coherence by aligning models based on task similarities and data distribution. Casting the problem into a fully decentralized setting, eliminating the need for a parameter server, is a potential area for future work that could emulate real-world applications such as mobile networks.

REFERENCES

[1] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, *Language Models Are Unsupervised Multitask Learners*, OpenAI, San Francisco, CA, USA, 2019.

[2] S. Min et al., "Rethinking the role of demonstrations: What makes in-context learning work?" in *Proc. EMNLP*, 2022, pp. 11048–11064.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Stat.*, 2017, pp. 1273–1282.

[4] Z. Zhang et al., "Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models," in *Proc. Annu. Meeting Assoc. Comput. Linguist.*, 2023, pp. 9963–9977.

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

[6] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[7] T. Li, S. Hu, A. Beirami, and V. Smith, "DITTO: Fair and robust federated learning through personalization," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 6357–6368.

[8] C. T. Dinh, T. T. Vu, N. H. Tran, M. N. Dao, and H. Zhang, "A new look and convergence rate of federated multitask learning with Laplacian regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 6, pp. 8075–8085, May 2024.

[9] E. J. Hu et al., "LoRa: Low-rank adaptation of large language models," 2021, *arXiv:2106.09685*.

[10] D. Chen et al., "Data-juicer: A one-stop data processing system for large language models," 2023, *arXiv:2309.02033*.

[11] Y. Wang et al., "Super-naturalinstructions: Generalization via declarative instructions on 1600+ NLP tasks," 2022, *arXiv:2204.07705*.

[12] M. Conover et al., "Free dolly: Introducing the world's first truly open instruction-tuned LLM." 2023. [Online]. Available: https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm

[13] 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," 2023, *arXiv:2312.06353*.

[14] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Proc. Text Summarization Branches Out*, 2004, pp. 74–81.

[15] J. Zhang et al., "Towards building the federatedGPT: Federated instruction tuning," in *Proc. IEEE Int. Conf. Acoust. SpeechSignal Process. (ICASSP)*, 2024, pp. 6915–6919.

[16] W. Kuang et al., "FederatedScope-LLM: A comprehensive package for fine-tuning large language models in federated learning," 2023, *arXiv:2309.00363*.