FL-TAC: ENHANCED FINE-TUNING IN FEDERATED LEARNING VIA LOW-RANK, TASK-SPECIFIC ADAPTER CLUSTERING

Siqi Ping[1]*, **Yuzhu Mao**[1]*, **Yang Liu**[2,3], **Xiao-Ping Zhang**[1,5] & **Wenbo Ding**[1,3,4]*
[1]Tsinghua-Berkeley Shenzhen Institute,
Tsinghua Shenzhen International Graduate School, Tsinghua University
[2] Institute for AI Industry Research (AIR), Tsinghua University
[3] Shanghai AI Lab, Shanghai, China
[4] RISC-V International Open Source Laboratory, Shenzhen, China
[5] Department of Electrical, Computer and Biomedical Engineering, Ryerson University
psq22@mails.tsinghua.edu.cn, myz20@tsinghua.org.cn
liuy03@air.tsinghua.edu.cn, xzhang@ee.ryerson.ca
ding.wenbo@sz.tsinghua.edu.cn

ABSTRACT

Although large-scale pre-trained models hold great potential for adapting to downstream tasks through fine-tuning, the performance of such fine-tuned models is often limited by the difficulty of collecting sufficient high-quality, task-specific data. Federated Learning (FL) offers a promising solution by enabling fine-tuning across large-scale clients with a variety of task data, but it is bottlenecked by significant communication overhead due to the pre-trained models' extensive size. This paper addresses the high communication cost for fine-tuning large pre-trained models within FL frameworks through *low-rank fine-tuning*. Specifically, we train a low-rank adapter for each individual task on the client side, followed by server-side clustering for similar group of adapters to achieve task-specific aggregation. Extensive experiments on various language and vision tasks, such as GLUE and CIFAR-10/100, reveal the evolution of task-specific adapters throughout the FL training process and verify the effectiveness of the proposed low-rank **t**ask-specific **a**dapter **c**lustering (TAC) method.

1 INTRODUCTION

Large-scale pre-trained models, such as Large Language Models (LLMs) trained on extensive data, demonstrate superior performance in natural language processing and remarkable adaptability to various downstream tasks (Brown et al., 2020; Ouyang et al., 2022; Touvron et al., 2023; Zhang et al., 2022; Dosovitskiy et al., 2020; Brohan et al., 2023). However, fine-tuning these models for specific tasks necessitates high-quality, task-relevant data, posing challenges for edge users with diverse, yet limited, private datasets (Zhang et al., 2023). In this context, Federated Learning (FL) offers a solution by enabling collaborative model training across devices with distributed data, achieving comparable results to centralized training with abundant data (McMahan et al., 2017).

Despite its advantages, FL faces significant communication overhead due to frequent exchanges of models or gradients between distributed clients and central servers (Lim et al., 2020; Zhao et al., 2023). This issue becomes particularly critical when fine-tuning LLMs that have a vast number of parameters (Yi et al., 2023). Consequently, Parameter-Efficient Fine-Tuning (PEFT) methods, such as Low-Rank Adaptation (LoRA), are essential in FL environments to reduce communication costs (Houlsby et al., 2019; Lester et al., 2021; Li & Liang, 2021; Zaken et al., 2022; Fu et al., 2023; Hu et al., 2021). For example, LoRA leverages the inherent low-dimensional structure of over-parameterized models by training only a low-rank adapter with fewer parameters during fine-tuning (Hu et al., 2021). Nevertheless, existing research has indicated a trade-off: A lower number

*Siqi Ping and Yuzhu Mao contribute equally. Wenbo Ding is the corresponding author.

of trainable parameters can lead to increased approximation errors and reduced model performance for downstream tasks, particularly when the training target is complex, for instance, when there is a significant optimization error between the target model and the model before fine-tuning (Zeng & Lee, 2023).

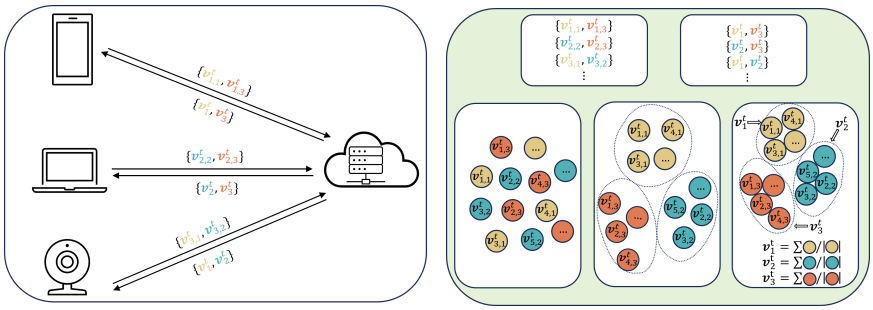
Addressing the challenges of low-rank fine-tuning in FL environments catering to diverse tasks, this paper investigates:

How to save communication costs through low-rank fine-tuning
while still enabling effective task adaptation for each client?

A natural solution to this problem is to customize the low-rank adapter for each task on the client side and aggregate the adapters for the same task on the server side. To this aim, this paper examines the evolution and interrelationships of low-rank adapters from distributed clients for different tasks. The findings are employed to enhance FL aggregation for multi-task objectives. Specifically, the proposed FL process with low-rank **t**ask-specific **a**dapter **c**lustering (FL-TAC) allows each client to train and upload a unique, extremely low-rank adapter for each task. The server subsequently performs a clustering step to differentiate adapters for different tasks and aggregates the adapters with the same task target.

The contributions of this paper are threefold: 1) The proposed FL-TAC algorithm enables an efficient and effective FL framework, which is capable of accommodating various target tasks from image to text classification and generation, and outperforms single-adapter baselines in both performance and communication efficiency. 2) The successful clustering of task-specific adapters provides insights into the evolution of trainable parameters during the FL training process, demonstrating how this property can be utilized for effective generalization across multiple downstream tasks in a distributed setting. 3) Extensive experiments on text datasets such as the GLUE benchmark and image datasets like CIFAR-10/100 demonstrate high clustering accuracy of FL-TAC algorithm and its general applicability to a broad range of downstream tasks within a unified framework.

The rest of this paper is organized as follows: Section 2 briefly outlines previous work related to this paper. In Section 3, the proposed algorithm is described in detail. The experimental results are provided in Section 4, and the concluding remarks are summarized in Section 5.



(a) Server-Client adapter exchange. (b) Server clustering and aggregation.

Figure 1: FL-TAC framework.

2 Related Work

2.1 FL for Fine-tuning Large-scale Pre-trained Models

The fine-tuning of large-scale pre-trained models in FL scenarios has been recently studied, leading to several significant contributions particularly relevant to this work. A line of research, represented by Federated Instruction Tuning (FedIT), has made substantial progress in fine-tuning LLMs in FL scenarios (Zhang et al., 2023). FedIT offers a generalizable experimental framework that utilizes distributed instruction data sources to enhance the final fine-tuned model's performance. This work has laid essential groundwork in the field, demonstrating the potential of FL in the context of fine-tuning LLMs. Following this, the FedLoRA approach has enabled efficient FL fine-tuning and com-

munication through the PEFT methods (Yi et al., 2023). PEFT methods finetune pre-trained models with reduced trainable parameters, and can be broadly categorized into addition-based methods, specification-based methods, and reparameterization-based methods as follows (Zhou et al., 2023):

Addition-based Method. The core idea of addition-based methods is to integrate extra trainable modules into the pre-trained model. Two representatives of addition-based methods are adapter-based tuning (Houlsby et al., 2019) and prompt-based tuning (Lester et al., 2021; Li & Liang, 2021). The former introduces small, task-specific neural networks as adapters between the layers of the pre-trained model, while the latter incorporates additional context by wrapping the original input. In this way, the number of trainable parameters during fine-tuning is limited to the size of the added adapters or input context.

Specification-based Method. This type of PEFT methods freezes most of the pre-trained model parameters and updates only a selected subset of parameters during fine-tuning. For instance, BitFit (Zaken et al., 2022), a sparse finetuning method, only adjusts the bias terms in neural networks, and the Second-order Approximation Method (SAM) identifies the subset of parameters to be adjusted by solving an optimization problem (Fu et al., 2023).

Reparameterization-based Method. This significant line of research is inspired by previous observations that the performance yielded by a low-dimensional matrix can match that of a fine-tuned deep layer in large-scale pre-trained models (Aghajanyan et al., 2021; Karimi Mahabadi et al., 2021). In this context, LoRA is a widely-used reparameterization-based algorithm that takes advantage of the existence of a low intrinsic rank to reduce trainable parameters (Hu et al., 2021).

In this paper, we adopt the well-recognized LoRA method, similar to FedLoRA, for its wide empirical success and robust theoretical guarantee (Zeng & Lee, 2023). It should be noted that the current analysis and experimental results can be easily extended to other PEFT methods by substituting the LoRA adapters with the training outcomes derived from other PEFT techniques.

While previous works mark significant advances in the field of efficient fine-tuning through FL, the challenges arising from task diversity remain underexplored and inadequately addressed. Considering the motivation of FL, which is to utilize distributed data sources, it is natural for each unique data source, such as an edge device or user, to have diverse local tasks to handle. Consequently, there remains a need for an FL framework that enables parameter-efficient training and aggregation among clients with different optimization targets to approximate.

2.2 FL for Multi-Task Learning

Multi-task learning involves the simultaneous training on multiple related tasks to leverage their commonalities or differences for improved performance over all tasks. Methods to achieve multi-task learning in FL scenarios include federated transfer learning (Liu et al., 2020; Gao et al., 2019) and clustered FL (Ghosh et al., 2020; Sattler et al., 2020; Briggs et al., 2020). Among these approaches, federated transfer learning aims to transfer information learned from one task to another, while clustered FL focuses on classifying distributed clients into multiple clusters based on certain task representations prior to the aggregation step. This pre-aggregation clustering improves multi-task performance by enhancing the information sharing within clients handling the same task. Existing studies in clustered FL have explored various forms of task representation, including models, gradients, logits, and latent representations (Kulkarni et al., 2020; Mansour et al., 2020; Mao et al., 2023; Zhu et al., 2021).

Given its ability to classify clients according to the underlying tasks, clustered FL aligns more closely with the objective of this study, which is to customize multiple task-specific low-rank local adapters for each client within the FL framework and to perform aggregation within the same task. Therefore, clustered FL is employed in this paper to enhance performance for each downstream task involved in an FL training process. Moreover, the recent emergence of PEFT methods driven by LLMs fine-tuning has yet to be incorporated in previous clustered FL studies, leaving a gap in the current literature for a clustered FL framework for multi-task learning that accommodates task representations provided by PEFT methods.

Based on the above analysis, this study aims to address, to some extent, the limitations and gaps identified in the aforementioned works. The algorithm introduced in Section 3 offers a novel ap-

proach to managing multiple tasks per client during the efficient fine-tuning of large pre-trained models within FL frameworks.

3 METHOD

3.1 OVERVIEW OF FINE-TUNING THROUGH FL

Consider an FL scenario involving a central server and m clients, collectively addressing a total of N downstream tasks. Each client i in the client set $\mathcal{M} = \{1, 2, \ldots, m\}$ is responsible for a subset of these tasks, with a total of $n_i \leq N$ local tasks to handle. Thus, the local dataset for client i can be denoted as $\mathcal{D}_i = \{\mathcal{D}_i^j | j \in \mathcal{N}_i\}$, where each $\mathcal{D}_i^j$ represents a downstream task dataset of client i, and $\mathcal{N}_i$ denotes the indices of client i's local tasks within the global set of tasks.

In centralized learning scenarios, each client i fine-tunes a pre-trained model using its local dataset $\mathcal{D}_i$. However, FL leverages the collective dataset $\mathcal{D}_\mathcal{M} = \bigcup_{i=1}^m \mathcal{D}_i$ as a comprehensive, global dataset to provide richer information. This is achieved by optimizing the global function defined as:

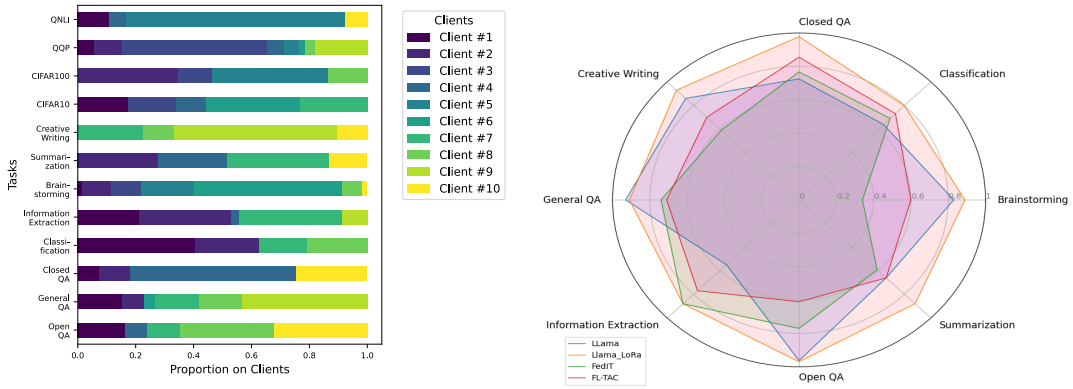
$$\begin{aligned} \min_{\boldsymbol{\theta}} \mathcal{F}(\boldsymbol{\theta}) &= \sum_{i=1}^{m} \frac{|\mathcal{D}_i|}{|\mathcal{D}_\mathcal{M}|} \mathcal{F}_i(\boldsymbol{\theta}), \\ \mathcal{F}_i(\boldsymbol{\theta}) &:= \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_i} \left[\mathcal{L}(\boldsymbol{\theta}; (\boldsymbol{x}, y))\right]. \end{aligned} \tag{1}$$

Here, $(\boldsymbol{x}, y) \sim \mathcal{D}_i$ denotes data-label pairs sampled from client i's local dataset $\mathcal{D}_i$. $|\mathcal{D}_i|$ and $|\mathcal{D}_\mathcal{M}| := \sum_{i\in\mathcal{M}} |\mathcal{D}_i|$ represent the sizes of the local and global datasets, respectively. The function $\mathcal{F}_i(\cdot)$ represents the expected local objective concerning client i's dataset, a specific loss function $\mathcal{L}(\cdot)$, and the global model parameters $\boldsymbol{\theta}$. The global objective $\mathcal{F}(\cdot)$ is a weighted average of these local objectives, with weights proportional to the size of each client's dataset. The optimal global model $\boldsymbol{\theta}^*$ that minimizes this function is expected to perform well on the global dataset $\mathcal{D}$.

In practice, during the t-th communication round, the server first selects a subset of clients $\mathcal{M}_t$ with total number of m_t clients, either randomly or based on specific criteria, and broadcasts the current global model, denoted as $\boldsymbol{\theta}^{t-1}$, to these clients. Each chosen client i in $\mathcal{M}_t$ then updates this model $\boldsymbol{\theta}^{t-1}$ through τ local training steps to obtain its local model $\boldsymbol{\theta}_i^t$. This is typically done using the update rule $\boldsymbol{\theta}_{i,k}^t = \boldsymbol{\theta}_{i,k-1}^t - \eta \nabla \boldsymbol{\theta}_{i,k-1}^t \mathcal{F}_i(\boldsymbol{\theta}_{i,k-1}^t)$ for $k = 1, \ldots, \tau$, where η is the learning rate and $\boldsymbol{\theta}_{i,0}^t = \boldsymbol{\theta}^{t-1}$. Post local training, clients send their updated local models $\boldsymbol{\theta}_i^t = \boldsymbol{\theta}_{i,\tau}^t$ back to the server, which then aggregates them into the updated global model $\boldsymbol{\theta}^t = \sum_{i\in\mathcal{M}_t} \frac{|\mathcal{D}_i|}{|\mathcal{D}_{\mathcal{M}_t}|} \boldsymbol{\theta}_i^t$. This cycle repeats until T rounds are completed, resulting in the final model $\boldsymbol{\theta}^T$ that approximates the optimal $\boldsymbol{\theta}^*$.

However, for initial global models like LLMs that have a significant scale, fine-tuning through the FL process becomes infeasible due to the heavy communication overhead from transmitting large-scale models. As a solution, the LoRA technique can be incorporated to reduce the number of transmitted parameters (Hu et al., 2021). Specifically, for an original weight matrix $\boldsymbol{W} \in \mathbb{R}^{d\times k}$ in a pre-trained model and its corresponding target weight matrix $\boldsymbol{W'} \in \mathbb{R}^{d\times k}$ after fine-tuning, LoRA approximates the fine-tuned output $\boldsymbol{W'z}$ for an input $\boldsymbol{z}$ using a low-rank decomposition: $\boldsymbol{W'z} = \boldsymbol{Wz} + \boldsymbol{BAz}$, where $\boldsymbol{B} \in \mathbb{R}^{d\times r}$, $\boldsymbol{A} \in \mathbb{R}^{r\times k}$, and the rank $r \ll \min(d; k)$. This allows the original weight matrix $\boldsymbol{W}$ to remain unchanged during fine-tuning, with only the smaller matrices (also termed as adapters) $\boldsymbol{A}$ and $\boldsymbol{B}$ being adjustable.

By incorporating the LoRA technique, the large-scale pre-trained model, denoted as $\boldsymbol{\theta}$, only needs to be transmitted from the central server to the clients at the beginning of the first communication round. In subsequent rounds (i.e., the t-th round), the server first sends a global adapter $\boldsymbol{v}^{t-1}$, and each client i updates $\boldsymbol{v}^{t-1}$ through efficient local fine-tuning to obtain the local adapter $\boldsymbol{v}_i^t = [\boldsymbol{A}_i^0, \boldsymbol{B}_i^0, \ldots, \boldsymbol{A}_i^L, \boldsymbol{B}_i^L]$ for a total of L weight matrices in the pre-trained model $\boldsymbol{\theta} = [\boldsymbol{W}^0, \ldots, \boldsymbol{W}^L]$. Global aggregation is subsequently performed using the local adapters provided by the clients, resulting in $\boldsymbol{v}^t = \sum_{i\in\mathcal{M}_t} \frac{|\mathcal{D}_i|}{|\mathcal{D}_{\mathcal{M}_t}|} \boldsymbol{v}_i^t$, and the new global model can be approximated as $\boldsymbol{\theta}^t = \boldsymbol{\theta}^{t-1} + \boldsymbol{v}^t$.



(a) Proportion of task data among clients.

(b) Scores for each task in Databricks-Dolly-15k.

Figure 2: Visualization of data distribution across clients and performance evaluation on Databricks-Dolly-15k multitask dataset.

3.2 Core Concepts and Approach

In the general process of fine-tuning large pre-trained models through FL, each client i typically leverages a single LoRA adapter for downstream task adaptation, despite having multiple local tasks. Achieving empirical success is feasible due to the exceptional emergent abilities provided by recent pre-trained models like GPT and Llama (Brown et al., 2020; Touvron et al., 2023). However, recent theoretical research indicates that the expressive power of fine-tuned models can be significantly restricted by the aggressive reduction of LoRA rank, such as 16 or even 8, as often seen in literature (Zeng & Lee, 2023). This presents an additional challenge to fine-tuning via FL, where such aggressive use of the LoRA method is considered necessary.

In the context of fine-tuning large models through FL, the key of this work lies in training a low-rank task-specific adapter for each individual task at the client side. The motivation arises from the observation that when the LoRA rank falls below a certain threshold, the approximation error begins to significantly increase. For tasks with higher complexity (measured by the approximation error before fine-tuning), reducing the LoRA rank below this threshold results in a more rapid increase in the approximation error, as shown in previous study (Zeng & Lee, 2023).

Intuitively, using a single adapter for multiple tasks results in a higher approximation error before fine-tuning and thus a more rapid and significant increase in the approximation error caused by LoRA rank reduction. Conversely, using a unique adapter for each individual task leads to a slower and more gradual increase in the approximation error. To validate this intuition, a toy simulation is conducted in this study. In this simulation, data points in simulated task 1 are randomly initialized scalars with labels created by applying a noisy sinusoidal transformation to the data points and adding Gaussian noise. Data points in simulated task 2 are generated similarly, except for using a noisy sinusoidal transformation with a phase shift of 0.5 and Gaussian noise of higher variance. A Fully Connected Neural Network (FNN) with an adaptable hidden layer size is employed to simulate the LoRA adapter, train the model, and compute the generalization error (measured by Mean Squared Error (MSE), also referred to as approximation error to maintain consistency with previous work) after 1000 epochs. Figure 3 shows preliminary results indicating that training an FNN for each simulated task consistently yields a lower average approximation error compared to training an FNN for both simulated tasks, particularly when the LoRA rank (simulated by the hidden layer size of

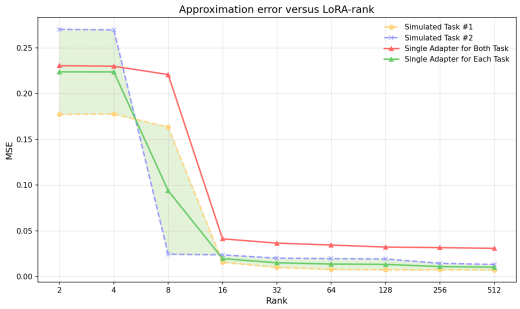


Figure 3: Approximation error (measured by MSE) versus LoRA-rank.

the FNN) is low (e.g., below 16). This observation is particularly relevant when considering the fine-tuning/communication costs closely associated with the LoRA rank. For instance, when the LoRA rank r is below 8, training two adapters with rank lower than $r/2$ yields a lower average approximation error compared to training one adapter with rank r for both tasks.

Based on this finding, we propose training task-specific adapters for each client, aiming for an overall strategy that reduces the total number of trainable parameters while maintaining or reducing the average approximation error across all tasks. With this revised approach to fine-tuning via FL, each client i must transmit multiple adapters to the central server in each communication round. To facilitate aggregation over adapters without explicit task labels from clients, a *clustering step* is introduced on the server side to group adapters corresponding to the same task. This clustering step is essential to facilitate knowledge exchange and aggregation among clients handling the same task in the proposed FL fine-tuning process, thereby enhancing overall performance for that task.

3.3 The FL-TAC Algorithm

Building upon the introduced concepts and approaches, the proposed FL-TAC method enables each client i to fine-tune n_i distinct local adapters for its n_i local tasks. The final goal is to acquire N distinct global adapters for the total N downstream tasks in the system through FL iterations between local fine-tuning and global aggregation. In this way, the global objective is rewritten as:

$$\begin{aligned}\min_{\{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_N\}} \hat{\mathcal{F}}(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_N) &= \sum_{i=1}^{m} \frac{|\mathcal{D}_i|}{|\mathcal{D}_{\mathcal{M}}|} \hat{\mathcal{F}}_i\left(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_N\right), \\ \hat{\mathcal{F}}_i\left(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_N\right) &:= \sum_{j\in\mathcal{N}_i} \hat{\mathcal{F}}_{i,j}(\boldsymbol{v}_j, D_i^j) \\ &:= \sum_{j\in\mathcal{N}_i} \mathbb{E}_{(\boldsymbol{x},y)\sim D_i^j}\left[\hat{\mathcal{L}}(\boldsymbol{\theta};\boldsymbol{v}_j;(\boldsymbol{x},y))\right], \end{aligned} \tag{2}$$

where $\hat{\mathcal{F}}_i(\cdot)$ denotes client i's local objective, constructed by the task-specific objective $\hat{\mathcal{F}}_{i,j}(\cdot)$ for each task $j \in \mathcal{N}_i$, and $\hat{\mathcal{L}}$ refers to specific loss functions. It's important to note that $\hat{\mathcal{F}}_{i,j}(\cdot)$ is influenced only by the task-specific trainable adapter $\boldsymbol{v}_j$ and the underlying task data distribution D_i^j, as the pre-trained model $\boldsymbol{\theta}$ remains frozen during fine-tuning.

Algorithm 1 FL-TAC

Inputs: Pre-trained model $\boldsymbol{\theta}$; Client set $\mathcal{M}$; Local datasets D_i^j for $i \in \mathcal{M}, j \in \mathcal{N}_i$.
Parameters: Number of communication rounds T; Number of local steps τ; Number of downstream tasks N.

Server randomly initializes a global adapter $\boldsymbol{v}^0$.
Server broadcasts the initial global adapter $\boldsymbol{v}^0$ to $\mathcal{M}$.
for client i in $\mathcal{M}$ in parallel **do**
  Initializes local task-specific adapters $\boldsymbol{v}_{i,j}^0 = \boldsymbol{v}^0$ for each local task $j \in \mathcal{N}_i$.
end for
for $t = 1$ to T **do**
  Server selects subset of clients $\mathcal{M}_t$ for round t.
  for client i in $\mathcal{M}_t$ in parallel **do**
    Performs *FL-TAC Local Fine-tuning*($\boldsymbol{\theta}, \tau, \{\boldsymbol{v}_{i,j}^{t-1} | j \in \mathcal{N}_i\}$).
    Sends updated local task-specific adapters $\boldsymbol{v}_{i,j}^t$for each local task $j \in \mathcal{N}_i$ to the server.
  end for
  Server performs *FL-TAC Global Clustering and Aggregation*($\{\boldsymbol{v}_{i,j}^t | i \in \mathcal{M}_t, j \in \mathcal{N}_i\}$).
  for each client i in $\mathcal{M}_t$ **do**
    Server sends back updated local task-specific adapters $\{\boldsymbol{v}_{i,j}^t | j \in \mathcal{N}_i\}$.
  end for
end for
return Global task-specific adapters $\boldsymbol{v}_j^T$ for each task $j \in \{1,\ldots,N\}$.

The FL-TAC algorithm (Algorithm 1) begins with the server initializing a global adapter $\boldsymbol{v}^0$ and broadcasting it to all clients in the set $\mathcal{M}$. Each client i initializes local task-specific adapters $\boldsymbol{v}_{i,j}^0$

Algorithm 2 FL-TAC Local Fine-tuning

Inputs:Pre-trained model $\boldsymbol{\theta}$; Number of local steps τ; Local task-specific adapters $\{\boldsymbol{v}_{i,j}^{t-1} | j \in \mathcal{N}_i\}$.

Parameters: Local dataset D_i^j for $j \in \mathcal{N}_i$; Learning rate η.

for each task $j \in \mathcal{N}_i$ **do**
  Initiates $\boldsymbol{v}_{i,j,0}^t = \boldsymbol{v}_{i,j}^{t-1}$.
  for $k = 1$ to τ **do**
    Updates task-specific adapter: $\boldsymbol{v}_{i,j,k}^t = \boldsymbol{v}_{i,j,k-1}^t - \eta \nabla \boldsymbol{v}_{i,j,k-1}^t \hat{\mathcal{F}}_{i,j}(\boldsymbol{v}_{i,j,k-1}^t, D_i^j)$.
  end for
  Updates task-specific adapter: $\boldsymbol{v}_{i,j}^t = \boldsymbol{v}_{i,j,\tau}^t$.
end for
return Updated local task-specific adapters $\boldsymbol{v}_{i,j}^t$ for each task $j \in \mathcal{N}_i$.

Algorithm 3 FL-TAC Global Clustering and Aggregation

Inputs: The received clients' task-specific adapters $\{\boldsymbol{v}_{i,j}^t | i \in \mathcal{M}_t, j \in \mathcal{N}_i\}$.

Parameters: Number of downstream tasks N.

Initializes $\mathcal{V} = \{\boldsymbol{v}_{i,j}^t | i \in \mathcal{M}_t, j \in \mathcal{N}_i\}$
Performs K-means algorithm to split $\mathcal{V}$ into N clusters $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_N$.
for $n = 1$ to N **do**
  for all $\boldsymbol{v}_{i,j}^t \in \mathcal{V}_n$ **do**
    $\boldsymbol{v}_j^t = \sum_{\boldsymbol{v}_{i,j}^t \in \mathcal{V}_n} \boldsymbol{v}_{i,j}^t / |\mathcal{V}_n|$.
    $\boldsymbol{v}_{i,j}^t = \boldsymbol{v}_j^t$.
  end for
end for
return Global task-specific adapters $\boldsymbol{v}_j^t$ for each task $j \in \{1, \ldots, N\}$,
  Aggregated local task-specific adapters $\{\boldsymbol{v}_{i,j}^t | j \in \mathcal{N}_i\}$ for each client $i \in \mathcal{M}_t$.

for their tasks $j \in \mathcal{N}_i$. In each communication round t, a subset of clients $\mathcal{M}_t$ is selected to perform local fine-tuning as described in Algorithm 2. More specifically, each selected client i executes τ local training steps with the pre-trained model $\boldsymbol{\theta}$ and local task-specific adapters $\boldsymbol{v}_{i,j}^{t-1}$ for $j \in \mathcal{N}_i$. Each adapter $\boldsymbol{v}_{i,j}^{t-1}$ is updated iteratively using the local dataset D_i^j and a learning rate η, resulting in updated local task-specific adapters $\boldsymbol{v}_{i,j}^t$ for each local task j.

After the local fine-tuning, each client $i \in \mathcal{M}_t$ sends its local adapters to the server, which performs global clustering and aggregation as presented in Algorithm 3. This process involves the server initializing a collection $\mathcal{V}$ of the received adapters $\boldsymbol{v}_{i,j}^t$ for each client $i \in \mathcal{M}_t$ and their local tasks $j \in \mathcal{N}_i$. The K-means algorithm is then applied to cluster $\mathcal{V}$ into N clusters $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_N$. Adapters within the same cluster n are aggregated to form $\boldsymbol{v}_n^t$ as a new global task-specific adapter for the n-th task involved in the system, and each local adapter in the cluster is updated to this new aggregated global adapter. Therefore, at the end of the t-th communication round, the server obtains the global task-specific adapters $\boldsymbol{v}_n^t$ for each task n and returns the updated local task-specific adapters corresponding to local tasks $j \in \mathcal{N}_i$ to each client $i \in \mathcal{M}_t$. This process is iterative over T communication rounds, ultimately yielding global task-specific adapters $\boldsymbol{v}_n^T$ as approximated optimal $\boldsymbol{v}_n^*$ for each of the N tasks within the system. Figure 1 shows the process of FL-TAC algorithm with three example clients, each of which tackles two local tasks.

4 EXPERIMENT

In this section, experiment settings and results for evaluating the performance of the proposed FL-TAC algorithm are presented in details.

4.1 EXPERIMENT SETUP

Datasets. The evaluation is conducted across various datasets, including Databricks-Dolly-15k for text generation, several GLUE datasets for text classification, and CIFAR-10 and CIFAR-100 for image classification (Wang et al., 2019). Databricks-Dolly-15k encompasses eight tasks, including Brainstorming, Summarization, Classification, Creative Writing, Information Extraction, Closed Question Answering (QA), Open QA, and General QA. The GLUE datasets considered in this study are Stanford Sentiment Treebank Binary Classification (SST2), Microsoft Research Paraphrase Corpus (MRPC), Quora Question Pairs (QQP), Question-answering Natural Language Inference (QNLI), and Recognizing Textual Entailment (RTE).

Base Models. The tasks of text generation, text classification, and image classification are handled by distinct pretrained models. Specifically, the Llama-7B model (Touvron et al., 2023) is utilized for text generation, the Bert model (Devlin et al., 2018) for text classification, and the Vision Transformer (ViT) pretrained on the ImageNet-21k dataset (Dosovitskiy et al., 2020) is employed for image classification.

Data Distribution. For simulating the real-world scenario where each client possesses unique data distribution, Dirichlet Partition, following the work of Hsu et al. (2020), is taken into consideration. For each downstream task, the training data is allocated to clients according to the Dirichlet distribution $Dir(\alpha)$. Note that if a client's value from the Dirichlet distribution is below the 0.01 threshold, the task's data will not be allocated to that client. The total number of local clients is 10 and α is set to 0.5 for all downstream tasks. Detailed data distribution can be found in Figure 2(a).

Baselines. For fair and comprehensive evaluation, a range of standard baselines from centralized learning (CL) and FL are selected for comparison. Specifically, **Llama-7B** and **Llama-7B-LoRA** are CL models, with Llama-7B representing a well-recognized open-source LLM and Llama-7B-LoRA is its variant fine-tuned with LoRA rank 1 on the same dataset as our method. In terms of the FL method, the baseline **FedIT** (Zhang et al., 2023) is considered, which involves fine-tuning Llama-7B with LoRA rank 16 through the FL system.

Table 1: Evaluation on Databricks-Dolly-15k dataset

Training Setting	Method	Brain-storming	Classi-fication	Closed QA	Creative Writing	General QA	Information Extraction	Open QA	Summari-zation	Average	Trainable Parameters
CL	LLama-7B	0.830	0.640	0.725	0.860	0.930	0.550	0.963	0.661	0.770	6,607,884,288
	Llama-7B-LoRA	0.890	0.799	0.978	0.930	0.910	0.880	0.970	0.880	0.905	4,259,840
FL	FedIT	0.340	0.691	0.768	0.590	0.740	0.880	0.770	0.592	0.671	8,519,680
	FL-TAC	**0.600**	**0.731**	**0.855**	**0.700**	**0.710**	**0.770**	**0.610**	**0.661**	**0.705**	**4,259,840**

4.2 EXPERIMENT RESULTS

This subsection presents and analyzes the experiment results obtained under the above experiment settings.

4.2.1 PERFORMANCE ANALYSIS FOR DOWNSTREAM TASKS

Table 1 demonstrates the performance of various methods on the Databricks-Dolly-15k dataset across multiple tasks. The evaluation follows previous works by having the GPT-4 model score the answers provided by different methods (Zhang et al., 2023). The proposed FL-TAC algorithm surpasses the FedIT baseline in most tasks, except for information extraction, general QA, and open QA. In the information extraction task, it is noteworthy that the performance of the Llama-7B model, centralized fine-tuned with LoRA, is comparatively lower than in other tasks. This observation suggests a potential need for incorporating task-specific adapters and knowledge exchange across tasks for certain cases. For the other two QA tasks, such performance could be attributed to the exceptional language generation ability of the base Llama model, which exhibits good responses to these two types of tasks even without fine-tuning. Figure 2(b) provides a clear visualization of the performance of these methods on the Databricks-Dolly-15k dataset, presenting a radar chart that highlights their respective capabilities.

Table 2: Evaluation on GLUE & Image datasets

Training Setting	Method	GLUE					Image		Trainable Parameters (Bert)	Trainable Parameters (ViT)
		QNLI	QQP	SST-2	RTE	MRPC	CIFAR10	CIFAR100		
CL	Bert	0.901	0.875	0.923	0.657	0.840	-	-	108,311,810	-
	Bert-LoRA	0.792	0.813	0.884	0.552	0.685	-	-	192,010	-
	ViT	-	-	-	-	-	0.987	0.920	-	86,862,568
	ViT-LoRA	-	-	-	-	-	0.950	0.899	-	36,864
FL	FedIT	0.670	0.510	0.652	0.524	0.520	0.935	0.854	614,432	294,912
	FL-TAC	**0.787**	**0.807**	**0.874**	**0.532**	**0.673**	**0.946**	**0.884**	**192,010**	**36,864**

Furthermore, Table 2 displays the accuracy achieved by the aforementioned methods on the GLUE dataset and image datasets. The proposed FL-TAC algorithm also demonstrates superior performance against the FedIT baseline in terms of both accuracy and communication efficiency.

4.2.2 Performance of Clustering at the Central Server

The visualization of clustering results using UMAP (McInnes et al., 2018) dimension reduction after the K-means clustering step is presented in Figure 4. As the training epochs advance, the categorization becomes more distinct and the distances between each cluster increase. This progression indicates the successful clustering of task-specific adapters through the training process of FL-TAC.

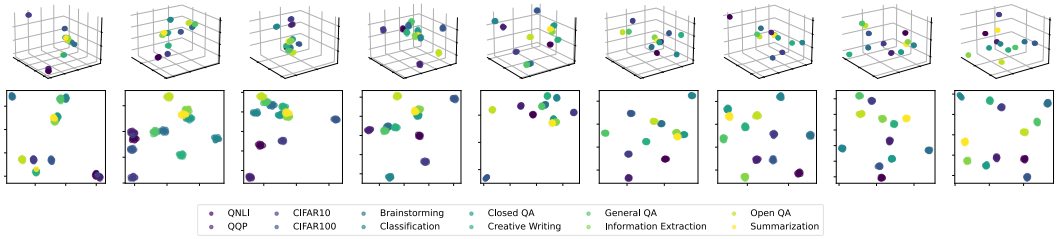


Figure 4: Visualization of clustering results from epoch 1 to epoch 9.

4.2.3 Analysis of Communication Cost and Local Training Cost

The trainable parameters, as shown in Table 1 and Table 2, display a comparison of the quantity of training parameters required by each method. Notably, the LoRA technique drastically decreases the number of training parameters required for fine-tuning. Also, when compared with FedIT, the training parameter transmitted by FL-TAC is lower, which is a result of our strategic approach of training task-specific adapters with a lower LoRA rank.

5 Conclusion

In conclusion, this work enhances fine-tuning large pre-trained models within FL by reducing communication cost and improving task adaptation performance with the presence of various downstream tasks. The introduced FL-TAC method employs low-rank task-specific adapters for each client and performs clustering and aggregation of similar adapters on the server. The extensive experiments of fine-tuning via the proposed FL-TAC method demonstrate improved communication efficiency and enhanced task adaptation performance compared to the single-adapter baseline, contributing to a unified, scalable FL framework for pre-trained model fine-tuning with both efficient communication and effective multi-task generalization. Future research directions involve investigating unresolved issues, such as the correlation between the required LoRA rank and the capabilities of base models.

6 Acknowledgement

This work was supported by Shenzhen Key Laboratory of Ubiquitous Data Enabling (ZDSYS20220527171406015), the National Key R&D Program of China under Grant No.2022ZD0160504 and Meituan.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *International Joint Conference on Natural Language Processing*, pp. 7319–7328, 2021.

Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *International Joint Conference on Neural Networks*, pp. 1–9, 2020.

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pp. 287–318, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. On the effectiveness of parameter-efficient fine-tuning. In *AAAI Conference on Artificial Intelligence*, pp. 12799–12807, 2023.

Dashan Gao, Yang Liu, Anbu Huang, Ce Ju, Han Yu, and Qiang Yang. Privacy-preserving heterogeneous federated transfer learning. In *IEEE international Conference on Big Data*, pp. 2552–2559, 2019.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597, 2020.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pp. 2790–2799, 2019.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution. In *European Conference on Computer Vision*, pp. 76–92, 2020.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035, 2021.

Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. Survey of personalization techniques for federated learning. In *Conference on Smart Trends in Systems, Security and Sustainability*, pp. 794–797, 2020.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *International Joint Conference on Natural Language Processing*, pp. 4582–4597, 2021.

Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063, 2020.

Yang Liu, Yan Kang, Chaoping Xing, Tianjian Chen, and Qiang Yang. A secure federated transfer learning framework. *IEEE Intelligent Systems*, 35(4):70–82, 2020.

Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

Yuzhu Mao, Zihao Zhao, Meilin Yang, Le Liang, Yang Liu, Wenbo Ding, Tian Lan, and Xiao-Ping Zhang. Safari: Sparsity-enabled federated learning with limited and unreliable communications. *IEEE Transactions on Mobile Computing*, 2023.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3710–3722, 2020.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

Liping Yi, Han Yu, Gang Wang, and Xiaoguang Liu. Fedlora: Model-heterogeneous personalized federated learning with lora tuning. *arXiv preprint arXiv:2310.13283*, 2023.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Annual Meeting of the Association for Computational Linguistics*, pp. 1–9, 2022.

Yuchen Zeng and Kangwook Lee. The expressive power of low-rank adaptation. In *Optimization for Machine Learning*, 2023.

Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Guoyin Wang, and Yiran Chen. Towards building the federated gpt: Federated instruction tuning. *arXiv preprint arXiv:2305.05644*, 2023.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Zihao Zhao, Yuzhu Mao, Yang Liu, Linqi Song, Ye Ouyang, Xinlei Chen, and Wenbo Ding. Towards efficient communications in federated learning: A contemporary survey. *Journal of the Franklin Institute*, 2023.

You Zhou, Xiujing Lin, Xiang Zhang, Maolin Wang, Gangwei Jiang, Huakang Lu, Yupeng Wu, Kai Zhang, Zhe Yang, Kehang Wang, et al. On the opportunities of green computing: A survey. *arXiv preprint arXiv:2311.00447*, 2023.

Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International Conference on Machine Learning*, pp. 12878–12889, 2021.