Communication-Efficient and Tensorized Federated Fine-Tuning of Large Language Models

Sajjad Ghiasvand[1] **Yifan Yang**[2] **Zhiyu Xue**[1]
Mahnoosh Alizadeh[1] **Zheng Zhang**[1] **Ramtin Pedarsani**[1]
Electrical and Computer Engineering Department, UC Santa Barbara[1]
Computer Science Department, UC Santa Barbara[2]
{sajjad,yifanyang,zhiyuxue,alizadeh,zhengzhang,ramtin}@ucsb.edu

Abstract

Parameter-efficient fine-tuning (PEFT) methods typically assume that Large Language Models (LLMs) are trained on data from a single device or client. However, real-world scenarios often require fine-tuning these models on private data distributed across multiple devices. Federated Learning (FL) offers an appealing solution by preserving user privacy, as sensitive data remains on local devices during training. Nonetheless, integrating parameter-efficient fine-tuning (PEFT) methods into FL introduces two main challenges: *communication overhead* and *data heterogeneity*. In this paper, we introduce FedTT and FedTT+, methods for adapting LLMs by integrating tensorized adapters into client-side models' encoder/decoder blocks. FedTT is versatile and can be applied to both cross-silo FL and large-scale cross-device FL. FedTT+, an extension of FedTT tailored for cross-silo FL, enhances robustness against data heterogeneity by adaptively freezing portions of tensor factors, further reducing the number of trainable parameters. Experiments on BERT and LLaMA family models demonstrate that our proposed methods successfully address data heterogeneity challenges and perform on par or even better than existing federated PEFT approaches while achieving up to 10× reduction in communication cost.

1 Introduction

Large Language Models (LLMs) such as GPT-4 (Achiam et al., 2023), LLaMA (Touvron et al., 2023), and BERT (Devlin et al., 2018) excel in tasks such as translation and summarization (Bommasani et al., 2021) due to the capabilities of transformer architectures (Vaswani, 2017). While fine-tuning enhances adaptability (Howard and Ruder, 2018), fully fine-tuning these massive models is computationally expensive and prone to overfitting. Parameter-efficient fine-tuning (PEFT) methods such as Adapters (Houlsby et al., 2019), Prompt-Tuning (Lester et al., 2021), and LoRA (Hu et al., 2021) address this by optimizing only a subset of parameters, reducing costs without sacrificing performance (Ding et al., 2023). However, traditional PEFT assumes centralized data, whereas real-world applications often involve private, distributed data, such as medical or legal records (Manoel et al., 2023; Shoham and Rappoport, 2023; Soltanmohammadi and Hikmet, 2024). Federated Learning (FL) emerges as a compelling solution, as it prioritizes user data privacy by ensuring sensitive information remains on individual devices during training. Instead of transmitting data to a central server, clients in FL locally update their model parameters and share only model information, such as parameters or gradients, which are then aggregated by the server into a global model (McMahan et al., 2017a).

To address the aforementioned problems, FL has been integrated into PEFT methods (Zhang et al., 2023; Fan et al., 2023; Zhao et al., 2023). These methods, however, often result in two major challenges: (i) communication and computation overhead in the FL system, and (ii) significant accuracy degradation, particularly under heterogeneous scenarios.

Communication efficiency is crucial in FL, as edge devices typically have limited storage and computational power. While some methods attempt to reduce the number of training parameters by incorporating sparsity in PEFT approaches, they either perform poorly in FL (He et al., 2022; Wu and Chen, 2022) or are computationally expensive for clients (Kuo et al., 2024), rendering them unsuitable for practical scenarios.

Data heterogeneity happens when training data is not identically and independently distributed across clients (non-i.i.d.). In such scenarios, local models on individual clients can diverge from the global model's optimal state, leading to slower conver-

gence (Hsieh et al., 2020; Li et al., 2020). This issue is particularly pronounced when training LLMs in federated environments, as data heterogeneity can severely impact model performance (Zhang et al., 2023). Although various studies have employed techniques such as gradient tracking to address this challenge in federated learning (Ghiasvand et al., 2024; Ebrahimi et al., 2024), these approaches often become inefficient in terms of communication and computational demands when applied to federated LLMs. Several works have modified LoRA to improve its efficiency in highly heterogeneous federated settings (Babakniya et al., 2023). However, these methods require communicating a large number of parameters (at least as much as LoRA).

In this work, we present a federated and tensorized framework that reduces communication load compared to other PEFT methods, while maintaining comparable or even superior accuracy, and addressing challenges such as data heterogeneity. We introduce the Federated Tensor Train (FedTT) algorithm, where tensorized adapters serve as trainable parameters embedded within the encoder/decoder blocks of models used by clients. FedTT is applicable to both cross-silo FL, where all clients participate in training, and large-scale cross-device (LSCD) FL, where only a subset of clients is selected in each round. For cross-silo FL, we propose the FedTT+ algorithm, a heterogeneity-robust method that further reduces the number of parameters by adaptively freezing portions of the tensor factors in the models. We summarize our contributions as follows:

- We study the TT decomposition of adapters to enable communication-efficient federated fine-tuning of LLMs.
- We propose FedTT, an efficient FL method to fine-tune LLMs using tensorized adapters, achieving up to $10\times$ communication reduction than other popular federated PEFT methods.
- We introduce FedTT+, an enhancement of FedTT that further reduces the number of trainable parameters and improves robustness against data heterogeneity in cross-silo FL. FedTT+ outperforms state-of-the-art (SOTA) cross-silo FL methods with fewer trainable parameters.
- We conduct extensive experiments across various settings, including data heterogeneity and differential privacy, using widely adopted LLMs such as the BERT and LLaMA-2 model families. These experiments validate the effectiveness of our proposed algorithms in reducing communication overhead while preserving high accuracy.

2 Related Work

2.1 Parameter Efficient Fine-Tuning (PEFT)

PEFT methods can be generally divided into three categories (Han et al., 2024). The first is Additive PEFT, where a small set of trainable parameters is added to the model, and only these are updated during training. Methods like Prefix-tuning (Li and Liang, 2021) and Prompt-tuning (Lester et al., 2021) fall into this category, and our approach aligns with this strategy. The second is Selective PEFT, which chooses a subset of existing model parameters for tuning, as seen in techniques like BitFit (Zaken et al., 2021) and PaFi (Liao et al., 2023). Lastly, Reparameterized PEFT introduces a low-rank parameterization of pre-trained weights for training, with methods such as LoRA (Hu et al., 2021) and DoRA (Liu et al., 2024).

2.2 PEFT in Federated Setting

(Zhang et al., 2023) tests and compares various PEFT methods such as Adapter, LoRA, Prompt Tuning, and BitFit in FL. Several works have modified LoRA to improve its efficiency in highly heterogeneous federated settings. For example, SLoRA (Babakniya et al., 2023; Yan et al., 2024) modify initialization to address data heterogeneity, whereas HetLoRA (Cho et al., 2023) and FlexLoRA (Bai et al., 2024) adaptively adjust LoRA ranks for each client to handle system heterogeneity. However, these methods still require communicating massive parameters.

Recently, (Kuo et al., 2024) introduced sparse fine-tuning to reduce the communication load in federated LoRA. Although this method lowers communication overhead and shows robustness to data heterogeneity in some tasks, it suffers from computational inefficiency due to the extensive matrix computations required during each communication round, both on the server and client sides. FFA-LoRA (Sun et al., 2024) and RoLoRA (Chen et al.) aim to improve accuracy in the presence of heterogeneity while simultaneously reducing trainable parameters. Our algorithms can reduce communication overhead while achieving comparable or better accuracy to these approaches, particularly when data heterogeneity exists.

(Kim et al., 2023) employs hypernetworks for

adapters to reduce the number of trainable parameters, and this approach can be integrated with any PEFT method, including ours. However, it introduces additional computational overhead, as each client must generate adapters from the hypernetwork (Hu et al., 2024).

2.3 Tensor-based Model Compression

Tensor compression has shown its great potential to reduce model size and enhance training efficiency. Early work by Novikov et al. (2015) employeed the TT format for network compression. Recent efforts in tensorized fine-tuning of LLMs have shown promise in achieving high performance with significantly fewer parameters compared to traditional fine-tuning methods. However, these approaches still involve training a large number of variables, especially when compared to the more recent methods like LoRA. In response, Yang et al. (2024b) developed the Low-Rank Economic Tensor-Train Adaptation, which innovates by using tensorized-layer based adapters and reshaping update matrices into smaller tensor factors. Despite demonstrating substantial reductions in trainable parameters—up to 100 times less than popular PEFT methods—the performance of these tensorized approaches in FL scenarios, particularly in the presence of data heterogeneity, remains an open question.

3 Preliminaries

3.1 Federated Fine-tuning

Federated fine-tuning is a distributed approach for collaboratively fine-tuning a global model across a central server and a network of N clients, denoted by $\mathcal{C} = \{c_1, \ldots, c_N\}$. The objective is to optimize the global trainable parameters $\boldsymbol{w}$ by minimizing the following objective function:

$$\min_{\boldsymbol{w}} \mathcal{L}(\tilde{\boldsymbol{w}}, \boldsymbol{w}) = \frac{1}{N} \sum_{i=1}^{N} \ell_i(\tilde{\boldsymbol{w}}, \boldsymbol{w}; \mathcal{D}_i),$$

where $\tilde{\boldsymbol{w}}$ is the pre-trained model parameters, which are identical and fixed for all clients, $\ell_i(\cdot)$ represents the local objective, and $\mathcal{D}_i$ represents the local data distribution for client c_i.

A well-known FL method is the FedAvg algorithm (McMahan et al., 2017a). The server selects a subset of clients $\mathcal{S} \subseteq \mathcal{C}$ in each training round. Each chosen client $c_s \in \mathcal{S}$ initializes its local model with the global model parameters from the previous round $\boldsymbol{w}^{(t)}$, and then performs local training using stochastic gradient descent on its dataset for K local updates:

$$\boldsymbol{w}_s^{(t)+k+1} \longleftarrow \boldsymbol{w}_s^{(t)+k} - \eta \nabla \ell_s(\tilde{\boldsymbol{w}}, \boldsymbol{w}_s^{(t)+k}; \mathcal{D}_s),$$

where η is the learning rate and $\boldsymbol{w}_s^{(t)+k}$ refers to the local model parameters for client c_s during communication round t and local update k. After local training is completed, clients send their updated model parameters to the server, which then aggregates these updates to form the new global model: $\boldsymbol{w}^{(t+1)} = 1/|\mathcal{S}| \sum_{s=1}^{|\mathcal{S}|} \boldsymbol{w}_s^{(t)+K}$.

Despite its advantages, FL faces two major challenges: (i) the large size of local models, which results in significant communication overhead, and (ii) data heterogeneity, which can cause local models to diverge from each other.

3.2 Tensor-Train Decomposition

In this subsection, we provide some introduction to the tensor and tensor train (TT) decomposition (Oseledets, 2011). Tensors are natural multidimensional generalizations of matrices. The tensor $\mathcal{W} \in \mathbb{R}^{k_1 \times \cdots \times k_J}$ is indexed as $\mathcal{W} = (w_{i_1 \cdots i_J})_{1 \le i_j \le k_j}$ said to have order of J and dimension $k_1, \cdots, k_J$. Given two tensors $\mathcal{W} \in \mathbb{R}^{k_1 \times \cdots \times k_J}$ and $\mathcal{V} \in \mathbb{R}^{l_1 \times \cdots \times l_M}$ with $k_s = l_t$, the multiplication between two tensors $\mathcal{C} = \mathcal{W} \times_{s,t} \mathcal{V}$ can be performed as:

$$\mathcal{C}_{(i_p)_{p \neq s}, (j_p)_{p \neq t}} = \sum_{i_s = j_t = 1}^{k_s} w_{i_1 \cdots i_s \cdots i_J} v_{j_1 \cdots j_t \cdots j_M}.$$

The TT decomposition serves as a potent alternative to traditional matrix decomposition techniques, which decompose a large tensor into a list of tensor factors (Oseledets, 2011) by TT-SVD method. As shown in Fig. 1 (a), to decompose the weight matrix into small tensors, we begin by reshaping a matrix $\boldsymbol{W} \in \mathbb{R}^{P \times Q}$ into a tensor $\mathcal{W} \in \mathbb{R}^{k_1 \times \cdots \times k_J}$. Then $\mathcal{W}$ can be parameterized compactly via a sequence of J tensor factors $\mathcal{G}_1, \ldots, \mathcal{G}_J$ as:

$$\mathcal{W} = \mathcal{G}_1 \times_{3,1} \mathcal{G}_2 \times_{3,1} \cdots \times_{3,1} \mathcal{G}_J, \qquad (1)$$

Here, each tensor factor $\mathcal{G}_j$ has the shape of $\mathcal{G}_j \in \mathbb{R}^{r_{j-1} \times k_j \times r_j}$, where $\boldsymbol{r} = (r_0, r_1, \cdots, r_J)$ is the tensor rank, and the product of dimensions $\Pi_j k_j = P \cdot Q$. The setup of tensor ranks follows the boundary conditions with $r_0 = r_J = 1$, while the other ranks $r_j, j \notin 0, J$ are chosen based on specific tasks or made adaptive (Yang et al., 2024c).

As we can see, the tensorized layer substantially reduces the parameter count for the weight matrix

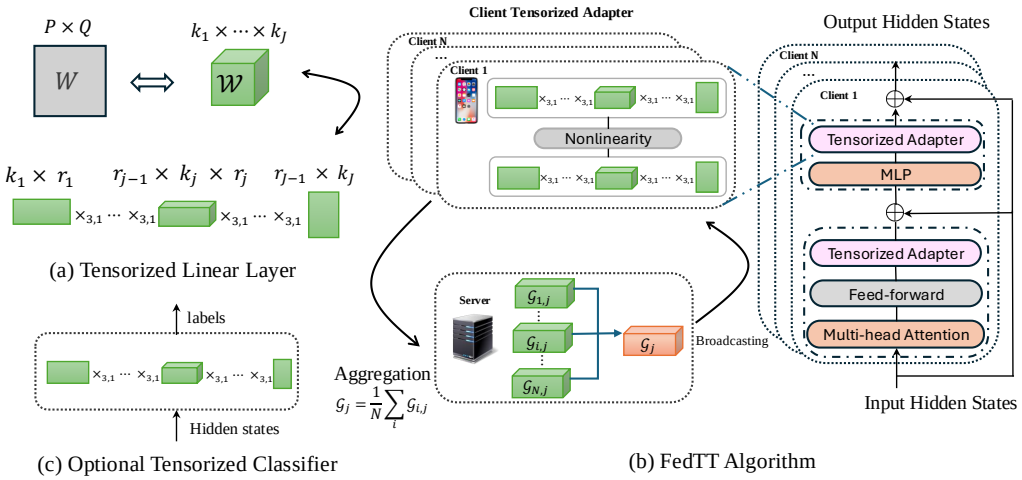

Figure 1: Illustration for the tensorized linear layer (a), FedTT algorithm (b), and the optional tensorized classifier applied for classification tasks (c). The FedTT algorithm workflow includes fine-tuning clients' tensorized adapters, aggregating tensor factors on the server, and broadcasting the updated weights back to clients.

$\boldsymbol{W}$ from $P \times Q$ to $\sum_{i=1}^{d} r_{i-1} k_i r_i$, offering a much higher compression ratio than previous PEFT work. Unlike the traditional Adapters method (Houlsby et al., 2019), which uses a bottleneck structure to reduce trainable parameters, our tensorized adapters achieve an even larger compression ratio by using two tensorized linear layers with a nonlinear activation in between. For example, with weight $\boldsymbol{W}$ with size 768×768 and bottleneck size 64, a standard Adapter incurs $2 \cdot 768 \cdot 64 \approx 98K$ trainable parameters for its weight matrices, whereas our method adds only $\sum_{i=1}^{6}(5^2 \cdot 8) = 1.2K$ parameters (assuming core dimensions $[8, 8, 8, 8, 8, 8]$ for $\mathcal{W}$ and TT rank 5). This high compression ratio enables FedTT to achieve higher performance under a similar communication cost compared with other PEFT approaches.

Instead of performing TT decomposition on the weight matrix, we directly initialize, store, and update the list of tensor factors in this work, as shown in Fig. 1 (a). During the forward pass, the tensor factors are directly contracted with the vector of activation values, and the weight matrix $\boldsymbol{W}$ does not need to be reconstructed. Since the size of the tensor factors is small, the contraction process is significantly faster than the original matrix-vector product (Yang et al., 2024c).

4 Proposed Algorithm

In this section, we introduce the proposed FedTT method, which is built on a novel parameter-efficient tensorized adapter. This adapter is integrated into clients' local models to effectively adapt LLMs. We begin with discussing the structure and setup of the tensorized adapters, followed by the introduction of FedTT and its enhanced version, FedTT+, which can be selected based on communication budget constraints. The workflow of our FedTT method is shown in Fig. 1 (b).

4.1 Tensorized Adapters

To facilitate fine-tuning in a FL framework, we incorporate tensorized adapters designed to adapt LLMs with minimal additional parameters. These adapters are underpinned by the novel tensorized linear layer (Yang et al., 2024b,a), which replaces the matrix weight in linear layers with more parameter-efficient TT format weights. This section first details the architecture of the tensorized linear layer and then elucidates the process of constructing tensorized adapters.

In Sec. 3.2, we introduced how to represent a weight matrix in the TT format, where the TT format tensor factors $\mathcal{G}_1, \ldots, \mathcal{G}_J$ are initialized, stored, and updated during the fine-tuning process. The tensorized layer is successfully integrated into matrix-based LLMs, performing tensor contraction over the list of tensor factors and reshaping them to match the traditional weight matrix $\boldsymbol{W}$. Compared to traditional matrix-based linear layers, the tensorized layers only store the smaller tensor factors, significantly reducing the number of parameters.

We now introduce the tensorized adapters, built based on the tensorized layers in Fig. 1. These adapters use a bottleneck structure similar to that in the original sequential adapter methodology (Houlsby et al., 2019), comprising two tensorized linear layers with a nonlinear layer in between. The bottleneck structure further reduces the number of trainable parameters by decreasing the channels

Algorithm 1 FedTT

for communication round $t \leftarrow 1$ to T **do**
  for clients $c_i \in \mathcal{C}$ in parallel **do**
    for local update $k \leftarrow 1$ to K **do**
      $\boldsymbol{w}_i^{(t)+k+1} \leftarrow \text{Update}(\boldsymbol{w}_i^{(t)+k}, \mathcal{D}_i)$
    end for
    Client c_i sends $\boldsymbol{w}_i^{(t)+K}$ to the server
  end for
  At server: $\boldsymbol{w}^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{w}_i^{(t)+K}$
  Server sends $\boldsymbol{w}^{(t+1)}$ to all clients in $\mathcal{C}$
end for

connected between the two tensorized layers and the nonlinearity layer, from the size of the hidden dimension to a smaller number, such as 64. As illustrated in Fig. 1 (b), the tensorized adapter is strategically placed after the attention and MLP components of an encoder/decoder block. In practical applications, we compress the classification head (linear layer) into a tensorized layer and make it trainable for sequence classification tasks, as shown in Fig. 1 (c). Instead, we retain the original language model head without compression for general language modeling tasks, as experiments show a significant reduction in performance when the language modeling head is compressed.

4.2 FedTT Method

We introduce the FedTT algorithm, as outlined in Alg. 1. In FedTT, the full model weights and architecture are initially distributed to all clients in the set $\mathcal{C} = \{c_1, \cdots, c_N\}$ at the start of the fine-tuning process. Tensorized adapters are then injected into each client's local model and designated as the trainable parameters. FedTT operates over T communication rounds, where each client performs K local updates on its trainable parameters during each round.

During the communication round t, each client $c_i \in \mathcal{C}$ updates its trainable parameters $\boldsymbol{w}_i^{(t)}$, including the tensorized classifier, tensorized adapter, and bias terms (if they exist), for K local updates. Afterward, each client sends its updated parameters, $\boldsymbol{w}_i^{(t)+K}$, to the central server. The server then aggregates the weights as $\boldsymbol{w}^{(t+1)} = 1/N \sum_{i=1}^{N} \boldsymbol{w}_i^{(t)+K}$ for the next training round and sends updated weights $\boldsymbol{w}^{(t+1)}$ back to the clients. Note that while we have primarily described cross-silo FL scenarios, the LSCD FL setting operates similarly, with the key distinction being that in each communication round, the server sends the aggregated weights to a randomly chosen subset of clients $\mathcal{S} \subseteq \mathcal{C}$, rather than to all clients in $\mathcal{C}$.

Algorithm 2 FedTT+

for communication round $t \leftarrow 1$ to T **do**
  for clients $c_i \in \mathcal{C}$ in parallel **do**
    $r \leftarrow$ (choose index r s.t. $\text{mod}(t, J-2) = r-1$)
    for local update $k \leftarrow 1$ to K **do**
      for $h \in \{1, r, J\}$:
        $\mathcal{G}_{i,h}^{(t)+k+1} \leftarrow \text{Update}(\mathcal{G}_{i,h}^{(t)+k}; \mathcal{D}_i)$
    end for
    Client c_i sends $\{\mathcal{G}_{i,1}^{(t)+K}, \mathcal{G}_{i,r}^{(t)+K}, \mathcal{G}_{i,J}^{(t)+K}\}$ to server
  end for
  At server: $\mathcal{G}_h^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{G}_{i,h}^{(t)+K}$ for $h \in \{1, r, J\}$
  Server sends $\{\mathcal{G}_1^{(t+1)}, \mathcal{G}_r^{(t+1)}, \mathcal{G}_J^{(t+1)}\}$ to all clients in $\mathcal{C}$
end for

Compared with most previous federated PEFT methods, FedTT significantly reduces communication overhead by only transferring small tensor factors. This approach leads to over $10\times$ communication reduction compared to LoRA adapters, as demonstrated in Sec. 4.1, making FedTT particularly advantageous in FL scenarios where efficient communication is critical.

4.3 FedTT+ Method

In this section, we propose an improved version of the FedTT method, named FedTT+ to further reduce the number of trainable parameters and enhance FedTT's suitability for scenarios with data heterogeneity. Before introducing FedTT+, we first explain the intuitive idea behind it.

In FedTT, the loss for back-propagation is computed on the product of tensor factors $\mathcal{G}_1, \ldots, \mathcal{G}_J$. However, under the federated setup, the aggregations in the server are performed separately on $\mathcal{G}_1, \ldots, \mathcal{G}_J$. This practice introduces additional terms in the product of the averaged $\mathcal{G}_1, \ldots, \mathcal{G}_J$, which may slow down the convergence of the algorithms. Ideally, the aggregation on the server should be performed on the product of the low-rank matrices $\mathcal{G}_1, \ldots, \mathcal{G}_J$. The left-hand side of Eq. (2) shows the parameters after aggregation with FedTT using FedAvg, while the right-hand side represents the ideal aggregation.

$$\left(\frac{1}{N}\sum_{i=1}^{N} \mathcal{G}_{i,1}\right) \times_{3,1} \ldots \times_{3,1} \left(\frac{1}{N}\sum_{i=1}^{N} \mathcal{G}_{i,J}\right) \neq \frac{1}{N}\sum_{i=1}^{N} \mathcal{G}_{i,1} \times_{3,1} \ldots \times_{3,1} \mathcal{G}_{i,J}, \tag{2}$$

where $\mathcal{G}_{i,j}$ is the tensor factor j for clientuser c_i. It is important to note that the difference between the right-hand side and the left-hand side of Eq. (2)

Table 1: Comparative analysis of various federated PEFT methods using the DeBERTa-Base model in a cross-silo FL setting with an i.i.d. data distribution and 5 clients.

Model & Method	# Param.	MRPC	SST-2	QNLI	QQP	MNLI	Avg.
DeBERTa-Base (LoRA$_{r=8}$)	$0.30M$	91.87	94.95	92.68	89.2	87.31	91.10
DeBERTa-Base (P-Tuning)	$0.30M$	82.01	90.48	82.12	84.0	80.74	83.87
DeBERTa-Base (LoRA$_{r=4}$)	$0.15M$	91.72	**94.95**	**92.66**	86.7	**86.91**	90.58
DeBERTa-Base (BitFit)	$0.10M$	91.33	94.72	91.89	88.4	86.02	90.47
DeBERTa-Base (RoLoRA$_{r=4}$)	$0.08M$	91.17	94.61	92.40	87.9	86.27	90.47
DeBERTa-Base (Prompt)	$\mathbf{0.01M}$	82.96	92.32	82.13	80.5	74.46	82.47
DeBERTa-Base (FedTT)	$0.06M$	**92.68**	94.61	92.02	**88.4**	85.99	**90.74**
DeBERTa-Base (FedTT+)	$\mathbf{0.02M}$	92.60	93.58	90.54	87.9	85.33	89.99

becomes more significant when: i) the number of clients is large, ii) the clients have non-IID data distributions, and iii) the number of local updates in each communication round is substantial.

In FedTT+, we alleviate this interference problem by freezing most of the tensor factors in each communication round and only updating a small fraction of them. In this approach, most of the tensor factors remain fixed and identical across all clients during training while only a few tensor factors are trainable in each communication round. For example, assume that we just update $\mathcal{G}_1$ and freeze $\mathcal{G}_2, \ldots, \mathcal{G}_J$ in communication t. Then, $\mathcal{G}_2, \ldots, \mathcal{G}_J$ are identical across the clients and Eq. (2) can be re-written as

$$\left(\frac{1}{N}\sum_{i=1}^{N}\mathcal{G}_{i,1}^{(t)}\right) \times_{3,1} \mathcal{G}_2^{(t-1)} \times_{3,1} \ldots \times_{3,1} \mathcal{G}_J^{(t-1)}$$
$$= \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{G}_{i,1}^{(t)} \times_{3,1} \mathcal{G}_2^{(t-1)} \times_{3,1} \ldots \times_{3,1} \mathcal{G}_J^{(t-1)}\right).$$

This modification to FedTT can improve accuracy, particularly in cases of severe data heterogeneity. The detailed algorithm is presented in Algorithm 2.

FedTT+ operates similarly to FedTT, but with a key difference: in each communication round t, an index r is selected from the set $r \in \{2, \ldots, J-1\}$ (line 3 in Alg. 2). In this process, $\{\mathcal{G}_1, \mathcal{G}_r, \mathcal{G}_J\}$ are set as trainable parameters (as the first and last tensors are always trained), while the other factors, $\{\mathcal{G}_2, \ldots, \mathcal{G}_{r-1}, \mathcal{G}_{r+1}, \ldots, \mathcal{G}_{J-1}\}$, remain frozen (line 5 and 6 in Alg. 2). As a result, clients only send their updated $\{\mathcal{G}_{i,1}, \mathcal{G}_{i,r}, \mathcal{G}_{i,J}\}$ to the server for aggregation, significantly reducing communication overhead. Note that the classification head is always trainable and present in all communication rounds.

(Sun et al., 2024; Chen et al.) have also demonstrated that freezing certain parameters improves the performance of LoRA in the presence of data heterogeneity, particularly with larger models like RoBERTa-large. In the numerical section, we compare FedTT+ with their method under the same settings and show that our approach achieves comparable or better accuracy while using even fewer trainable parameters.

5 Numerical Results

5.1 Experiments Setup

We conduct extensive experiments to evaluate the performance of the proposed algorithms across various language models. For the BERT-family models, we utiliz RoBERTa-base (Liu et al., 2019), DeBERTa-base (He et al., 2020), and RoBERTa-large (Liu et al., 2019), while for large-scale models, we employed LLaMA-2 (Touvron et al., 2023). Using these models, we compare the proposed algorithm against several PEFT methods in FL scenarios, including BitFit, LoRA, Adapter, Prefix-Tuning, and Prompt-Tuning. Additionally, we benchmark it against SOTA federated PEFT methods such as FFA-LoRA (Sun et al., 2024) and RoLoRA (Chen et al.).

We consider two main FL scenarios: the cross-silo FL scenario (Kairouz et al., 2021) and the LSCD FL scenario (Lai et al., 2022). The cross-silo scenario is suitable for networks with typically fewer than 100 clients. In this case, the server sends the updated model to all clients, meaning every client participates in training during each communication round. In contrast, LSCD FL is more appropriate for environments with thousands of clients, where only a randomly selected subset of clients is involved in each training round. For cross-silo FL, the number of clients is set to $n \in \{5, 10, 20, 50\}$, while for LSCD FL, similar to (Zhang et al., 2023), we randomly select 10 clients from a pool of 1000. All experiments are conducted using the AdamW optimizer (Loshchilov and Hutter, 2018), with a

Table 2: Comparative analysis of various federated PEFT methods using the RoBERTa-Base model. The reported accuracy for federated LoRA, Adapter, Prefix, and BitFit methods is sourced from (Zhang et al., 2023).

Model & Method	# Param.	MRPC	SST-2	QNLI	QQP	MNLI	Avg.
RoBERTa-Base (Prefix)	$3.50M$	88.1	93.7	84.6	81.8	80.4	85.7
RoBERTa-Base (P-Tuning)	$0.88M$	86.8	92.1	85.6	81.5	79.8	85.2
RoBERTa-Base (Adapter)	$0.70M$	88.5	94.0	85.9	**87.0**	**84.9**	88.1
RoBERTa-Base (IA3)	$0.65M$	88.0	93.0	**89.4**	85.4	82.7	87.7
RoBERTa-Base (LoRA)	$0.30M$	**89.8**	**94.4**	86.0	86.5	84.7	88.3
RoBERTa-Base (BitFit)	$0.10M$	88.6	92.8	80.5	84.0	80.7	85.3
RoBERTa-Base (FedTT)	$\mathbf{0.06}M$	88.9	93.8	88.9	86.2	84.2	**88.4**

similar learning rate and batch size setup across different methods. We perform the experiments on NVIDIA A6000 and V100 GPUs.

5.2 Performance on the BERT Family

We conduct experiments using the Generalized Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018), employing the complete training dataset for each task. We record the best validation results after the 100 communication round for cross-silo FL and after the 1000 communication round for LSCD FL. Initially, we compare FedTT and FedTT+ with various PEFT methods in a cross-silo FL setting, using the DeBERTa-base model and assuming independent and identically distributed (i.i.d.) data among clients. We set the number of clients to 5 and the local training epochs to 1. The results, shown in Table 1, indicate that FedTT achieves a higher average score compared to other methods with lower than $0.15M$ trainable parameters. Notably, FedTT exhibits only a 0.5% performance gap compared to LoRA with a rank of 8, which has $5\times$ more trainable parameters. FedTT+ achieves comparable accuracy to other methods while drastically reducing the number of trainable parameters. It exhibits only a 0.6% performance gap compared to LoRA with a rank of 4, while using $6\times$ fewer trainable parameters, which significantly lowers communication overhead.

We then use the RoBERTa-base model to further evaluate FedTT against other PEFT methods. For the MRPC and SST-2 tasks, we employ a cross-silo FL setup with 10 clients, while for the other tasks, we apply a LSCD FL configuration with 1000 clients, randomly selecting 10 clients per round. Our settings for the RoBERTa-base model align with those in (Zhang et al., 2023), allowing us to leverage their results of PEFT methods like LoRA, Adapter, Prefix, and BitFit under FL settings. Table 2 shows that FedTT achieves a higher

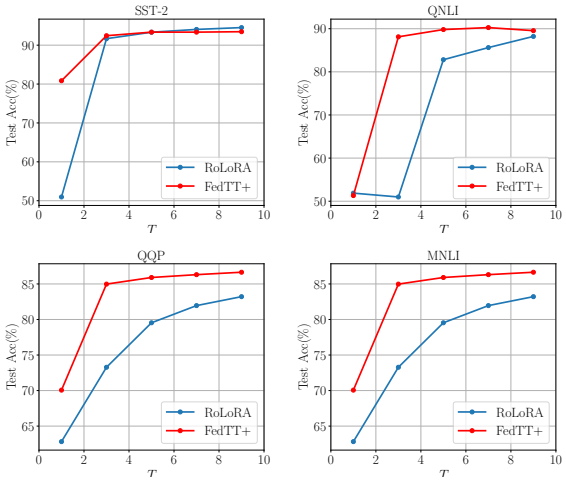

Figure 2: Test accuracies versus communication rounds for RoBERTa-Large models across different tasks in a cross-silo FL setting with 50 clients.

average score with the fewest trainable parameters among all methods. Additional details regarding Tables 1 and 2 are provided in Appendices A.3 and A.4, respectively.

5.3 Impact of Data Heterogeneity

Similar to (Chen et al.), we simulate varying levels of data heterogeneity by using different numbers of clients: 3 clients (i.i.d. data distribution), 20 clients (mild heterogeneity), and 50 clients (severe heterogeneity), with 20 local updates to further increase data heterogeneity. To ensure a comparable number of trainable parameters, we make the last 6 layers trainable for the SST-2 dataset and the last 9 layers for the other datasets in LoRA, FFA-LoRA, RoLoRA, and FedTT. In contrast, for FedTT+, all layers are made trainable. Additional details are provided in Appendix A.5. Our RoBERTa-Large model settings align with those in (Che et al., 2023), allowing us to leverage their results for LoRA, FFA-LoRA, and RoLoRA methods. As shown in Table 3, FedTT+ consistently outperforms the

Table 3: Comparison of SOTA cross-silo FL methods using RoBERTa-Large models under varying degrees of data heterogeneity. The accuracies for LoRA, FFA-LoRA, and RoLoRA methods are sourced from (Chen et al.).

Data Dist.	Model & Method	# Param.	SST-2	QNLI	QQP	MNLI	Avg.
i.i.d.	RoBERTa-Large (LoRA$_{r=2}$)	$68K$	**95.64**	92.04	85.85	86.16	89.92
	RoBERTa-Large (FFA-LoRA$_{r=2}$)	$34K$	94.91	90.11	84.06	85.48	88.64
	RoBERTa-Large (RoLoRA$_{r=2}$)	$34K$	95.60	91.62	85.66	86.16	89.76
	RoBERTa-Large (FedTT)	$51K$	94.38	93.01	88.30	87.20	90.72
	RoBERTa-Large (FedTT+)	$\mathbf{28K}$	**95.64**	**94.05**	**88.99**	**88.27**	**91.74**
mild het.	RoBERTa-Large (LoRA$_{r=2}$)	$68K$	94.27	86.91	81.22	82.07	86.12
	RoBERTa-Large (FFA-LoRA$_{r=2}$)	$34K$	93.92	89.58	80.51	82.62	86.66
	RoBERTa-Large (RoLoRA$_{r=2}$)	$34K$	94.84	90.77	85.13	85.10	88.96
	RoBERTa-Large (FedTT)	$51K$	94.15	91.38	86.25	86.53	89.58
	RoBERTa-Large (FedTT+)	$\mathbf{28K}$	**95.64**	**92.60**	**87.76**	**88.11**	**91.03**
sever het.	RoBERTa-Large (LoRA$_{r=2}$)	$68K$	93.23	82.57	58.96	76.96	77.93
	RoBERTa-Large (FFA-LoRA$_{r=2}$)	$34K$	92.32	85.15	62.79	77.78	79.51
	RoBERTa-Large (RoLoRA$_{r=2}$)	$34K$	**94.61**	89.83	85.15	85.55	88.78
	RoBERTa-Large (FedTT)	$51K$	94.38	**90.55**	85.47	85.27	88.92
	RoBERTa-Large (FedTT+)	$\mathbf{28K}$	94.50	90.17	**86.65**	**86.28**	**89.40**

Table 4: Comparison of cross-silo FL methods using RoBERTa-Large models under varying differential privacy guarantees. The accuracies for LoRA and FFA-LoRA methods are sourced from (Sun et al., 2024).

Priv. Budget	Method	# Param.	SST-2	QNLI	QQP	MNLI
$\epsilon = 6$	LoRA$_{r=8}$	$1.57M$	93.70	84.99	82.11	39.46
	FFA-LoRA$_{r=8}$	$0.79M$	93.73	87.27	83.31	78.81
	FedTT	$\mathbf{0.24M}$	**93.80**	**87.43**	**84.61**	**85.45**
$\epsilon = 3$	LoRA$_{r=8}$	$1.57M$	93.32	83.94	82.08	35.82
	FFA-LoRA$_{r=8}$	$0.79M$	93.59	86.18	83.03	77.42
	FedTT	$\mathbf{0.24M}$	**93.96**	**86.64**	**84.36**	**85.08**
$\epsilon = 1$	LoRA$_{r=8}$	$1.57M$	94.32	88.95	81.28	33.80
	FFA-LoRA$_{r=8}$	$0.79M$	94.32	90.35	82.50	75.05
	FedTT	$\mathbf{0.24M}$	**95.64**	**91.67**	**83.11**	**83.81**

other methods across varying heterogeneity settings while utilizing fewer parameters. This further demonstrates that, under a comparable number of trainable parameters and in the presence of data heterogeneity, FedTT+ outperforms FedTT, highlighting the effectiveness of adaptively freezing tensorized adapters in addressing data heterogeneity. Fig. 2 illustrates test accuracy versus the number of communication rounds in the severe heterogeneity scenario, showing that FedTT+ achieves faster convergence than RoLoRA across multiple tasks. An additional experiment examining the effect of data heterogeneity using an alternative heterogeneity measure is provided in Appendix B.

5.4 Performance on Larger Models

We use SuperGLUE tasks (Wang et al., 2019) and a generation task (SQuAD (Rajpurkar et al., 2016)) to compare FedTT and FedTT+ with LoRA. The results are shown in Table 5. We use the LLaMA2-7B model to simulate LSCD FL with 1000 clients, where 10 of them are chosen randomly in each round for training. Additional details are provided in Appendix A.6. FedTT has performance nearly identical to LoRA with a rank of 8, while achieving about $10\times$ lower communication overhead. For cross-silo FL, we utilize the LLaMA2-13B model with 10 clients. As seen in Table 5, both FedTT and FedTT+ demonstrate performance close to LoRA, with $10\times$ and $30\times$ lower communication overhead, respectively.

5.5 Communication Cost Analysis

In this section, we present the communication cost analysis for the MNLI task of Table 1 in Table 6. The analysis for all tasks of Table 1 and Table 3 can be found in Appendix C. Following (Guo et al., 2024), we compare the proposed method with baselines in terms of: (i) Up-link message size (in KB) per communication round, (ii) Number of communication rounds required to reach 95% of the prediction accuracy reported in Table 1, and (iii) Total size of transmitted messages. As demonstrated, our proposed algorithms effectively minimize both the up-link message size and the total size of transmitted messages.

5.6 Differential privacy guarantees

We compare our proposed FedTT algorithm with baseline methods under differential privacy guarantees. Definitions and theoretical results related to DP are provided in Appendix D. Following (Sun et al., 2024), we train the RoBERTa-Large model with a learning rate of $1e-3$ and three clients. We implement the DP-SGD algorithm using the Opacus (Yousefpour et al., 2021) library with privacy parameters $\delta = 1e-5$ and three different privacy budgets, $\epsilon \in \{1, 3, 6\}$. The clipping threshold is

Table 5: Comparative analysis of LoRA, FedTT, and FedTT+ using the LLaMA2-7B and LLaMA2-13B models.

Fed. Set.	Model & Method	# Param.	COPA	ReCoRD	SQuAD	Avg.
large-scale FL	LLaMA2-7B (LoRA$_{r=8}$)	$4.19M$	87	81.0	90.49	86.16
	LLaMA2-7B (FedTT)	$0.52M$	90	80.1	89.32	86.47
cross-silo FL	LLaMA2-13B (LoRA$_{r=8}$)	$6.55M$	89	83.6	91.07	87.89
	LLaMA2-13B (FedTT)	$0.64M$	89	83.5	90.06	87.52
	LLaMA2-13B (FedTT+)	$0.18M$	88	83.7	90.40	87.70

Table 6: Total transmitted messages for the MNLI task.

Methods	Total transmitted messages (KB)	Comm. overhead
LoRA	1172	×1.88
BitFit	390	×0.62
RoLoRA	624	×1.00
Prompt	1523	×2.44
FedTT	234	×0.37
FedTT+	78	×0.12

Table 7: Tensor rank analysis using the DeBERTa-Base model.

Model & Method	# Param.	SST-2	QNLI	QQP	MNLI	Avg.
FedTT$_{r=2}$	$0.03M$	93.58	90.76	86.79	85.29	89.10
FedTT$_{r=5}$	$0.06M$	93.00	91.87	87.65	85.91	89.61
FedTT$_{r=10}$	$0.17M$	93.69	92.48	88.22	87.01	90.35

chosen from $C \in \{2, 5\}$.

We follow the experimental setup from (Sun et al., 2024), enabling direct evaluation against LoRA and FFA-LoRA. The results in Table 4 demonstrates that FedTT consistently achieves higher accuracy across different privacy budgets while using fewer trainable parameters. Notably, the largest performance gap is observed in the MNLI task, a three-class classification problem.

5.7 Tensor Rank Analysis

In this section, we evaluate the performance of the proposed FedTT method with varying tensor ranks across four datasets using the DeBERTa-base model. The number of clients is set to 5, each performing one local update per round. We use a learning rate of $1e-3$ and a batch size of 32. The best validation results are reported after 20 communication rounds. As shown in Table 7, higher tensor ranks lead to better average accuracy, albeit at the cost of an increased number of trainable parameters. In our experiments, we set the rank to 5, as it offers a favorable trade-off between accuracy and model size.

6 Conclusion

This work presents a novel federated and tensorized fine-tuning framework, addressing the key challenges of communication overhead and data heterogeneity in FL. FedTT leverages tensorized adapters to significantly reduce communication costs while maintaining high performance, achieving up to a $10\times$ reduction in trainable parameters compared to existing methods. Additionally, FedTT+ enhances robustness in cross-silo FL by adaptively freezing portions of tensor factors, further optimizing parameter efficiency. Extensive experiments with models like BERT and LLaMA-2 demonstrate the effectiveness of FedTT and FedTT+ in both cross-silo and LSCD FL settings, offering a scalable and communication-efficient solution for fine-tuning LLMs in distributed environments.

Limitations

In this work, while we addressed the data heterogeneity challenge, we did not conduct experiments on another significant challenge in FL: system heterogeneity. However, our framework has the potential to tackle this challenge as well, by allowing different tensor ranks to be assigned to clients based on their computational capabilities. This presents an interesting direction for future research.

Acknowledgments

This work was supported by the National Science Foundation under Grant 2419982 and Grant 2342253. Z. Zhang and Y. Yang were supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Artificial Intelligence for Science program, under contract DE-SC0025390. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award ASCR-ERCAP0030039.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Qingfeng Liu, Kee-Bong Song, Mostafa El-Khamy, and Salman Avestimehr. 2023. SLoRA: federated parameter efficient fine-tuning of language models. *arXiv preprint arXiv:2308.06522*.

Jiamu Bai, Daoyuan Chen, Bingchen Qian, Liuyi Yao, and Yaliang Li. 2024. Federated fine-tuning of large language models under heterogeneous language tasks and client resources. *arXiv preprint arXiv:2402.11505*.

Rouzbeh Behnia, Arman Riasi, Reza Ebrahimi, Sherman SM Chow, Balaji Padmanabhan, and Thang Hoang. 2024. Efficient secure aggregation for privacy-preserving federated machine learning. In *2024 Annual Computer Security Applications Conference (ACSAC)*, pages 778–793. IEEE.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Tianshi Che, Ji Liu, Yang Zhou, Jiaxiang Ren, Jiwen Zhou, Victor S Sheng, Huaiyu Dai, and Dejing Dou. 2023. Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization. *arXiv preprint arXiv:2310.15080*.

Shuangyi Chen, Yue Ju, Hardik Dalal, Zhongwen Zhu, and Ashish J Khisti. Robust federated finetuning of foundation models via alternating minimization of LoRA. In *Workshop on Efficient Systems for Foundation Models II@ ICML2024*.

Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, Matt Barnes, and Gauri Joshi. 2023. Heterogeneous lora for federated fine-tuning of on-device foundation models. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer.

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

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 265–284. Springer.

Mohammadjavad Ebrahimi, Uday V Shanbhag, and Farzad Yousefian. 2024. Distributed gradient tracking methods with guarantees for computing a solution to stochastic mpecs. In *2024 American Control Conference (ACC)*, pages 2182–2187. IEEE.

Tao Fan, Yan Kang, Guoqiang Ma, Weijing Chen, Wenbin Wei, Lixin Fan, and Qiang Yang. 2023. Fate-llm: A industrial grade federated learning framework for large language models. *arXiv preprint arXiv:2310.10049*.

Sajjad Ghiasvand, Amirhossein Reisizadeh, Mahnoosh Alizadeh, and Ramtin Pedarsani. 2024. Robust decentralized learning with local updates and gradient tracking. *arXiv preprint arXiv:2405.00965*.

Pengxin Guo, Shuang Zeng, Yanran Wang, Huijie Fan, Feifei Wang, and Liangqiong Qu. 2024. Selective aggregation for low-rank adaptation in federated learning. *arXiv preprint arXiv:2410.01463*.

Erfan Hajihashemi and Yanning Shen. Multi-model ensemble conformal prediction in dynamic environments. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zeyu Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, et al. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Shwai He, Liang Ding, Daize Dong, Miao Zhang, and Dacheng Tao. 2022. Sparseadapter: An easy approach for improving the parameter-efficiency of adapters. *arXiv preprint arXiv:2210.04284*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. 2020. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jiahui Hu, Dan Wang, Zhibo Wang, Xiaoyi Pang, Huiyu Xu, Ju Ren, and Kui Ren. 2024. Federated large language model: Solutions, challenges and future directions. *IEEE Wireless Communications*.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210.

Yeachan Kim, Junho Kim, Wing-Lam Mok, Jun-Hyung Park, and SangKeun Lee. 2023. Client-customized adaptation for parameter-efficient federated learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1159–1172.

Kevin Kuo, Arian Raje, Kousik Rajesh, and Virginia Smith. 2024. Federated lora with sparse communication. *arXiv preprint arXiv:2406.05233*.

Fan Lai, Yinwei Dai, Sanjay Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha Madhyastha, and Mosharaf Chowdhury. 2022. Fedscale: Benchmarking model and system performance of federated learning at scale. In *International conference on machine learning*, pages 11814–11827. PMLR.

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

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Baohao Liao, Yan Meng, and Christof Monz. 2023. Parameter-efficient fine-tuning without introducing new latency. *arXiv preprint arXiv:2305.16742*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Andrea Manoel, Mirian del Carmen Hipolito Garcia, Tal Baumel, Shize Su, Jialei Chen, Robert Sim, Dan Miller, Danny Karmon, and Dimitrios Dimitriadis. 2023. Federated multilingual models for medical transcript analysis. In *Conference on Health, Inference, and Learning*, pages 147–162. PMLR.

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

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017b. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*.

Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. 2015. Tensorizing neural networks. *Advances in neural information processing systems*, 28.

Ivan V Oseledets. 2011. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Vishal Rathod, Seyedsina Nabavirazavi, Samira Zad, and Sundararaja Sitharama Iyengar. 2025. Privacy and security challenges in large language models. In

2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC), pages 00746–00752. IEEE.

Ofir Ben Shoham and Nadav Rappoport. 2023. Federated learning of medical concepts embedding using behrt. *arXiv preprint arXiv:2305.13052*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ehsan Soltanmohammadi and Neset Hikmet. 2024. Optimizing healthcare big data processing with containerized pyspark and parallel computing: A study on etl pipeline efficiency. *Journal of Data Analysis and Information Processing*, 12(4):544–565.

Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. 2024. Improving LoRA in privacy-preserving federated learning. *arXiv preprint arXiv:2403.12313*.

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

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Jiarun Wu and Qingliang Chen. 2022. Pruning adapters with lottery ticket. *Algorithms*, 15(2):63.

Nan Wu, Farhad Farokhi, David Smith, and Mohamed Ali Kaafar. 2020. The value of collaboration in convex machine learning with differential privacy. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 304–317. IEEE.

Yuxuan Yan, Shunpu Tang, Zhiguo Shi, and Qianqian Yang. 2024. FeDeRA: efficient fine-tuning of language models in federated learning leveraging weight decomposition. *arXiv preprint arXiv:2404.18848*.

Yifan Yang, Kai Zhen, Ershad Banijamal, Athanasios Mouchtaris, and Zheng Zhang. 2024a. Adazeta: Adaptive zeroth-order tensor-train adaption for memory-efficient large language models fine-tuning. *arXiv preprint arXiv:2406.18060*.

Yifan Yang, Jiajun Zhou, Ngai Wong, and Zheng Zhang. 2024b. LoRETTA: low-rank economic tensor-train adaptation for ultra-low-parameter fine-tuning of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3161–3176.

Zi Yang, Samridhi Choudhary, Xinfeng Xie, Cao Gao, Siegfried Kunzmann, and Zheng Zhang. 2024c. CoMERA: computing-and memory-efficient training via rank-adaptive tensor optimization. *arXiv preprint arXiv:2405.14377*.

Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, et al. 2021. Opacus: User-friendly differential privacy library in pytorch. *arXiv preprint arXiv:2109.12298*.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Zhuo Zhang, Yuanhang Yang, Yong Dai, Qifan Wang, Yue Yu, Lizhen Qu, and Zenglin Xu. 2023. Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In *Annual Meeting of the Association of Computational Linguistics 2023*, pages 9963–9977. Association for Computational Linguistics (ACL).

Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. 2023. Fedprompt: Communication-efficient and privacy-preserving prompt tuning in federated learning. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

A Experiment setup

A.1 Dataset Setup

We begin our experiments using the Generalized Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018), which includes a range of natural language understanding tasks. These tasks consist of perceptual analysis (SST2 (Socher et al., 2013)), similarity and paraphrase tasks (MRPC, QQP (Dagan et al., 2005)), and natural language reasoning (MNLI, QNLI (Williams et al., 2017; Rajpurkar et al., 2018)). The utilized metrics for the GLUE benchmark are summarized in Table 8. We record the best validation results after 100 communication rounds for cross-silo FL and after 1000 communication rounds for large-scale cross-device FL. For the DeBERTa-Base models on the QQP task, we randomly select 1000 samples from the validation set and report the highest accuracy achieved.

Table 8: Dataset descriptions and statistics.

Task	**# Train**	**# Dev.**	**Metric**
MRPC	3,301	367	F1 Score
SST-2	66,675	674	Accuracy
QNLI	103,695	5,463	Accuracy
QQP	360,210	40,430	Accuracy
MNLI	388,774	9,815	Accuracy

We then select two multiple-choice tasks (COPA and ReCoRD) from the SuperGLUE benchmark (Wang et al., 2019) and a question-answering generation task (SQuAD (Rajpurkar et al., 2016)). The metrics used for evaluation are summarized in Table 9.

Table 9: The utilized metrics for the SuperGLUE benchmark.

Task Name	Metric
COPA	Accuracy
ReCoRD	F1
SQuAD	F1

A.2 Additional Detail of TT-format

In this paper, we use the TT format to structure the weight matrices within the tensorized layers. To accommodate matrices of varying shapes, we design specific tensor shapes based on the hidden sizes and bottleneck configurations of different models. The tensor shapes are outlined in Table 10, with examples provided for the DeBERTa/RoBERTa-base and LLaMA-2-7b models, which have hidden sizes of 768 and 4096, respectively. For models with other hidden sizes, the appropriate tensor shape must be determined prior to training.

Table 10: The shape settings of the TT-format

Modules	Matrix Shape	Tensor Shape
Tensorized Adapters	768×64	[8, 8, 12, 8, 8]
	4096×64	[16, 16, 16, 4, 4, 4]
	64×768	[8, 8, 12, 8, 8]
	64×4096	[4, 4, 4, 16, 16, 16]
Tenosrized Classifier (Optional)	768×768	[12, 8, 8, 8, 8, 12]
	768×768	[8, 8, 8, 8, 8, 8, 8, 8]

A.3 Additional Details for Table 1

We use DeBERTa-Base models with the GLUE dataset in a cross-silo FL setup. The learning rate is selected from $[5e-3, 1e-3, 5e-4, 1e-4]$, and the batch size from $[16, 32]$ for all tasks and methods. The number of clients is set to 5, with one local update per communication round. The best validation results were recorded after 100 communication rounds. Additional details on the chosen parameters are provided in Table 11.

Table 11: The hyperparameter grids used for GLUE experiments.

Experiment	Hyperparameters	Values
LoRA	Rank	$4, 8$
RoLoRA	Rank	4
Bitfit	Bias Terms	All
Prompt	# Tokens	10
P-tuning	# Tokens	20
	Prompt Length	$[128, 768]$
FedTT	Tensor Rank	5
FedTT+	Tensor Rank	5

A.4 Additional Details for Table 2

We follow the experimental setup of (Zhang et al., 2023), using RoBERTa-Base models with the GLUE dataset in both cross-silo and large-scale cross-device FL configurations. The learning rate was selected from $[1e-2, 5e-3, 1e-3, 5e-4, 1e{-}4, 5e{-}5]$, and batch size from $[16, 32]$ across all tasks and methods. For the MRPC and SST-2 tasks, we use a cross-silo FL setup with 10 clients,

Table 12: Number of trainable parameters for results in Table 3.

Model & Method	SST-2	QNLI	QQP	MNLI	Avg.
RoBERTa-Large (LoRA$_{r=2}$)	$49K$	$74K$	$74K$	$74K$	$68K$
RoBERTa-Large (FFA-LoRA$_{r=2}$)	$\mathbf{25K}$	$37K$	$37K$	$37K$	$34K$
RoBERTa-Large (RoLoRA$_{r=2}$)	$\mathbf{25K}$	$37K$	$37K$	$37K$	$34K$
RoBERTa-Large (FedTT)	$39K$	$55K$	$55K$	$55K$	$51K$
RoBERTa-Large (FedTT+)	$28K$	$\mathbf{28K}$	$\mathbf{28K}$	$\mathbf{28K}$	$\mathbf{28K}$

while for the other tasks, we employ a large-scale cross-device FL setup with 1000 clients, selecting 10 clients per round. Accuracy results for Prefix, Adapter, LoRA, and BitFit methods are sourced from (Zhang et al., 2023). In addition to reporting the accuracy of the FedTT method, we also includ the P-Tuning method with prompt lengths $[128, 768]$, and IA3 (Liu et al., 2022) method, to further enrich the experiments.

A.5 Additional Details for Table 3

We adopt the same settings as (Chen et al.), using RoBERTa-Large models with the GLUE dataset in cross-silo FL scenarios. To simulate varying levels of data heterogeneity, we report the accuracy of our method for 3 clients (i.i.d. data distribution), 20 clients (moderate heterogeneity), and 50 clients (high heterogeneity), with 20 local updates. The learning rate is set to $1e-3$ for all tasks, with a batch size of 64 for SST-2 and 32 for other tasks. To ensure a comparable number of trainable parameters, we make the last 6 layers trainable for the SST-2 dataset and the last 9 layers for the other datasets in LoRA, FFA-LoRA, RoLoRA, and FedTT. In contrast, for FedTT+, all layers are made trainable. We provide a comparison of the number of trainable parameters for LoRA, FFA-LoRA, RoLoRA, and our proposed FedTT, and FedTT+ methods in Table 12. Accuracy results for LoRA, FFA-LoRA, and RoLoRA are sourced from (Chen et al.).

A.6 Additional Details for Table 5

Due to the large number of parameters in LLaMA-2, it is rarely used in FL scenarios, and few existing works provide results for such models. However, FedTT significantly reduces communication overhead, making it feasible to utilize these models in FL settings. We evaluate FedTT and FedTT+ against LoRA using SuperGLUE tasks (Wang et al., 2019) and a generation task (SQuAD (Rajpurkar et al., 2016)). The results are presented in Table 5.

For each task, we randomly select 1000 samples for training and 1000 for validation, reporting the best validation accuracy. We use a learning rate of $1e-4$, batch size of 2, and 3 local updates across all tasks. In large-scale cross-silo FL, we simulate training with the LLaMA2-7B model across 1000 clients, selecting 10 randomly per round. FedTT achieves performance nearly identical to LoRA with a rank of 8, while reducing communication overhead by approximately $10\times$.

For cross-silo FL, we utilize the LLaMA2-13B model with 10 clients. As shown in Table 5, both FedTT and FedTT+ achieve performance close to LoRA while reducing communication overhead by approximately $10\times$ and $30\times$, respectively.

B Additional Experiment on Data Heterogeneity

Data heterogeneity is a common challenge in most practical scenarios (Hajihashemi and Shen). In FL, local models on individual clients can diverge from the global model's optimal state, resulting in slower convergence (Hsieh et al., 2020; Li et al., 2020). This issue is particularly pronounced when training LLMs in federated environments, as data heterogeneity can severely impact model performance (Zhang et al., 2023).

As mentioned before, three primary factors contribute to data heterogeneity: (1) the number of clients, (2) the number of local updates, and (3) non-i.i.d. data distribution. In Section 5.3 of the paper, which evaluates the heterogeneity robustness of FedTT+, we considered the effects of the number of clients and local updates. Specifically, we increased the number of clients to intensify heterogeneity. In this section, we

Table 13: Comparison of cross-silo FL methods using RoBERTa-Large models under varying degrees of data heterogeneity. The accuracies for LoRA and FFA-LoRA methods are sourced from (Sun et al., 2024).

Data Dist.	Model & Method	# Param.	SST-2	QNLI	QQP	MNLI	Avg.
i.i.d.	RoBERTa-Large (LoRA$_{r=8}$)	$1.57M$	94.42	91.38	84.47	86.90	89.29
	RoBERTa-Large (FFA-LoRA$_{r=8}$)	$0.79M$	95.14	92.64	86.31	87.13	90.30
	RoBERTa-Large (FedTT+)	$\mathbf{0.03M}$	**95.41**	**94.05**	**88.15**	**88.65**	**91.56**
mild het.	RoBERTa-Large (LoRA$_{r=8}$)	$1.57M$	93.55	91.36	84.41	87.01	89.08
	RoBERTa-Large (FFA-LoRA$_{r=8}$)	$0.79M$	94.10	91.62	85.33	87.04	89.52
	RoBERTa-Large (FedTT+)	$\mathbf{0.03M}$	**95.53**	**93.54**	**87.91**	**88.45**	**91.36**
sever het.	RoBERTa-Large (LoRA$_{r=8}$)	$1.57M$	94.32	88.95	83.51	82.03	87.20
	RoBERTa-Large (FFA-LoRA$_{r=8}$)	$0.79M$	94.32	90.35	84.35	85.05	88.52
	RoBERTa-Large (FedTT+)	$\mathbf{0.03M}$	**95.64**	**91.67**	**86.66**	**87.73**	**90.42**

conduct an additional experiment to evaluate the performance of FedTT+ under varying levels of data heterogeneity. Similar to (Sun et al., 2024), we train the RoBERTa-Large model and consider three levels of data distribution for three clients:

- **i.i.d. Data Distribution**: Data is evenly distributed across all clients.
- **Mild Heterogeneity**: For binary classification tasks, data is split as $[0.15, 0.85]$, $[0.85, 0.15]$, and $[0.5, 0.5]$ among three clients. For three-class classification tasks, the splits are $[0.6, 0.2, 0.2]$, $[0.2, 0.6, 0.2]$, and $[0.2, 0.2, 0.6]$.
- **Severe Heterogeneity**: For binary classification tasks, data is split as $[0.05, 0.95]$, $[0.95, 0.05]$, and $[0.5, 0.5]$. For three-class classification tasks, the splits are $[0.9, 0.05, 0.05]$, $[0.05, 0.9, 0.05]$, and $[0.05, 0.05, 0.9]$.

To further amplify heterogeneity, we used 10 local updates. We followed the exact experimental setup described in (Sun et al., 2024), allowing us to directly compare our results with those reported for LoRA and FFA-LoRA. The results are presented in Table 13. As shown in the Table, FedTT+ consistently outperforms LoRA and FFA-LoRA across different heterogeneity settings while utilizing fewer parameters.

C Communication Cost Analysis

In this section, we provide communication cost analysis for Table 1 and Table 3. Specifically, we compare the proposed method with baselines in terms of communication cost, following (Guo et al., 2024). The analysis includes:

- Up-link message size (in KB) for each communication round.
- Number of communication rounds needed to reach the predefined target performance on the SST-2, QNLI, QQP, and MNLI datasets.
- Total transmitted messages (in KB).

The target performance is defined as 95% of the prediction accuracy reported in Table 1 and Table 3.

In a federated learning system, two key parameters influence communication efficiency: the up-link message size and the total size of transmitted messages, computed as the product of the number of communication rounds and the up-link message size. We provide detailed comparisons in Table 14 and Table 15. As demonstrated, our proposed algorithms effectively minimize both the up-link message size and the total number of transmitted messages.

D Privacy

While LLMs excel in performance due to their transformer-based architecture and vast parameter count, their ability to memorize and inadvertently reveal sensitive information from training data is a concern (Carlini et al., 2021; Rathod et al., 2025). A widely adopted framework for mitigating such privacy risks is DP (Dwork et al., 2006), which provides formal guarantees against data leakage.

In this section, we first outline key definitions of DP and introduce the DP-SGD algorithm. We then establish a theoretical privacy guarantee within the DP framework.

Definition 1. ((ϵ, δ)-Differential Privacy (Dwork et al., 2006)]) A randomized mechanism

Table 14: Communication cost analysis for Table 1.

Method	Up-link Message Size (KB)	# Communication Round				Total transmitted messages (KB)			
		SST-2	QNLI	QQP	MNLI	SST-2	QNLI	QQP	MNLI
LoRA$_{r=4}$	586	2	2	2	2	1172	1172	1172	1172
RoLoRA$_{r=4}$	312	2	2	1	2	624	624	312	624
FedTT	234	2	2	2	2	468	**468**	468	468
FedTT+	**78**	5	6	3	3	**390**	**468**	**234**	**234**

Table 15: Communication cost analysis for sever heterogeneity in Table 3.

Method	Up-link Message Size (KB)	# Communication Round				Total transmitted messages (KB)			
		SST-2	QNLI	QQP	MNLI	SST-2	QNLI	QQP	MNLI
RoLoRA$_{r=2}$	133	3	10	11	11	399	1330	1463	1463
FedTT	199	3	3	3	3	597	597	597	597
FedTT+	**109**	3	3	3	3	**327**	**327**	**327**	**327**

$\mathcal{M} : \mathcal{D} \to \mathcal{R}$ satisfies (ϵ, δ)-differential privacy if for any two adjacent datasets $D, D' \in \mathcal{D}$ that differ in at most one data point, and for any subset of possible outputs $S \subseteq \mathcal{R}$, the following holds:

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in S] + \delta.$$

DP ensures that the mechanism $\mathcal{M}$ provides privacy guarantees by limiting the impact of any single data point on the output, with ϵ controlling the privacy loss and δ allowing for a small probability of failure in the guarantee.

Differentially Private Stochastic Gradient Descent (DP-SGD) Algorithm (Abadi et al., 2016): DP-SGD is a modification of the SGD algorithm designed to provide differential privacy guarantees. It achieves this by introducing two key modifications:

- **Gradient Clipping:** To ensure that individual data points do not have a disproportionate influence on the model update, each per-sample gradient is clipped to a fixed norm C.
- **Noise Addition:** After aggregating the clipped gradients, Gaussian noise $z \sim \mathcal{N}\left(0, C^2\sigma^2 I\right)$ is added to the sum of clipped gradients in a batch $\mathcal{B}$ from the dataset $\mathcal{D}$ before updating the model parameters. This helps obscure the contribution of any single data point.

The noisy sum of clipped gradients is computed as:

$$\bar{g} = \frac{\sum_{i \in \mathcal{B}} \text{Clip}(\nabla f_i, C) + z}{|\mathcal{B}|},$$

which is then used to update the model. Here, the noise scale σ is determined based on sequential composition rules, given the privacy parameters ϵ and δ, the number of iterations T, and the sampling probability $q = |\mathcal{B}|/|\mathcal{D}|$.

Although other techniques like secure aggregation have been explored in privacy-preserving FL (Behnia et al., 2024), our focus here is on DP. In federated learning with DP, the level of privacy protection depends on whether the aggregation server is trusted by the clients. In global DP, the server is trusted, allowing clients to send raw model updates, while the server applies differential privacy by adding noise to the aggregated updates before releasing them (McMahan et al., 2017b). In contrast, local DP assumes an untrusted server, requiring each client to add noise to their updates before transmission, ensuring privacy even if the server is compromised (Wu et al., 2020). In Section 5.6, we adopt the stronger local DP approach to guarantee robust privacy protection without relying on a trusted server.

Proposition 1. The mechanism for updating FedTT using locally run DP-SGD and FedAvg satisfies (ϵ, δ)-DP, given that $\forall i$, the sampling probability is defined as $q = |\mathcal{B}_i| \,/\, |\mathcal{D}_i|$. Furthermore, the total number of local updates per client is denoted as T, and σ is chosen as

$$\sigma = O\left(\frac{q\sqrt{T\log(1/\delta)}}{\epsilon}\right).$$

Proof. Similar to (Sun et al., 2024).