FedPETuning: When Federated Learning Meets the Parameter-Efficient Tuning Methods of Pre-trained Language Models

Zhuo Zhang^{1,2,*} **Yuanhang Yang**^{1,*} **Yong Dai**^{4} **Qifan Wang**^{5} **Yue Yu**^{2}
Lizhen Qu^{3,†} **Zenglin Xu**^{1,2,†}
^{1}Harbin Institute of Technology, Shenzhen, China
^{2}Peng Cheng Lab, Shenzhen, China
^{3}Monash University, Melbourne, Australia
^{4}Tencent, Shenzhen, China
^{5}Meta AI, CA, USA
{iezhuo17, ysngkil}@gmail.com daiyongya@outlook.com wqfcr@fb.com
yuy@pcl.ac.cn Lizhen.Qu@monash.edu.cn xuzenglin@hit.edu.cn

Abstract

With increasing concerns about data privacy, there is an increasing necessity of fine-tuning pre-trained language models (PLMs) for adapting to downstream tasks located in end-user devices or local clients without transmitting data to the central server. This urgent necessity therefore calls the research of investigating federated learning (FL) for PLMs. However, large PLMs bring the curse of prohibitive communication overhead and local model adaptation costs for the FL system. To this end, we investigate the parameter-efficient tuning (PETuning) of PLMs and develop a corresponding federated benchmark for four representative PETuning methods, dubbed FedPETuning. Specifically, FedPETuning provides the first holistic empirical study of representative PLMs tuning methods in FL, covering privacy attacks, performance comparisons, and resource-constrained analysis. Intensive experimental results have indicated that FedPETuning can efficiently defend against privacy attacks and maintains acceptable performance with reducing heavy resource consumption. The open-source code and data are available at `https://github.com/SMILELab-FL/FedPETuning`.

1 Introduction

Pre-trained **L**anguage **M**odels (PLMs), such as BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019a), have demonstrated exceptional performance on a multitude of natural language processing (NLP) benchmarks (e.g., GLUE (Radford et al., 2019)). Consequently, PLMs have become *de facto* backbones for real-world applications. Generally, in most real-world NLP applications, PLMs are centrally fine-tuned on the enormous quantity of data collected from individual customers, small businesses, or large enterprises (Qu et al., 2021). However, with the rising privacy concerns and the enactment of data protection laws[1], enterprises or institutions are not allowed to collect data from end devices or local clients to a centralized server for fine-tuning PLMs.

To break this barrier, federated learning (Konečný et al., 2016; McMahan et al., 2017) (FL) has emerged as a privacy-aware technique designed to collaboratively train models without transmitting the numerous end-user or client data to a centralized place. In FL, the decentralized clients only need to periodically compute and send model information (i.e., parameters or gradients) to a server which is responsible for aggregating them to produce a global model. With the notion of privacy preserving, FL is appealing for privacy-sensitive NLP applications (Sui et al., 2020; Basu et al., 2021), as in the case of healthcare (Ge et al., 2020), finance (Long et al., 2020), and mobile keyboard (Ji et al., 2019).

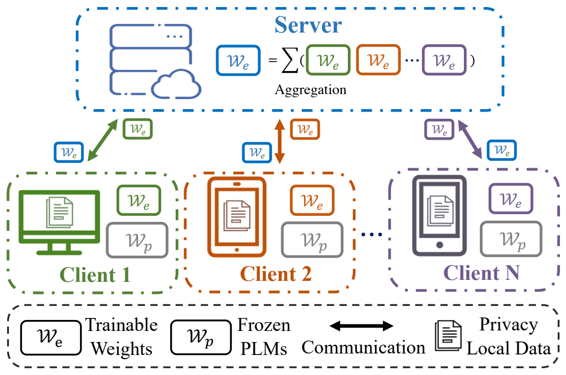

Figure 1: An overview of FedPETuning where a client exchanges a light amount of parameters of PLMs with the server while keeping most parameter frozen.

*Equal contribution.
†Corresponding authors.

[1] Such as the EU's GDPR or the US's HIPAA.

Although fine-tuning PLMs in FL, namely FedFT, presents promising opportunities, there are two significant challenges that cannot be overlooked, including (1) communication overhead in the FL system, and (2) computational and storage costs for local clients. Fine-tuning such PLMs usually requires distributed clients and services high-frequently exchange model gradients or parameters which are usually in the scale of millions or even billions. The limited communication bandwidth[2] in the FL system may cause excessive delays during frequent uploading and downloading phases of the federated fine-tuning procedure. Meanwhile, it is often impractical for local clients to fine-tune the entire PLMs because of their limited computing resources. Moreover, fully fine-tuning PLMs is extremely memory-intensive when a local client wants to store different instances for different tasks (Hu et al., 2022). As such, it is imperative to explore suitable PLMs-empowered FL methods under resource constraints (i.e., communication, parameters adaption, and storage).

To this end, we investigate parameter-efficient tuning (PETuning) methods of PLMs under the FL setting. PETuning methods, such as adapter tuning (Houlsby et al., 2019), prefix tuning (Li and Liang, 2021), LoRA (Hu et al., 2022), BitFit (Zaken et al., 2021), freeze most parameters of PLMs and update only a few additional parameters or a part of the original model parameters for downstream tasks (Ding et al., 2022). These properties make PETuning methods potentially appealing to satisfy the resource constraints since local clients just need to tune lightweight parameters and communicate with the server for updates. Nevertheless, it is crucial to address pertinent concerns. First, as the performance of federated models is greatly affected by the data heterogeneity (Kairouz et al., 2021) among clients, it is not yet known that PETuning methods can achieve acceptable performance in FL with such challenging data heterogeneity. Second, as private data could be recovered from model gradients uploaded by clients via gradient inversion attacks (Zhu et al., 2019), it is unclear whether PETuning methods that upload only partial parameters of the entire model can resist gradient inversion attacks.

To address these concerns, we present the framework of Federated Parameter-Efficient Tuning (named FedPETuning for short), as illustrated in Figure 1, and conduct in-depth experiments of various PETuning methods under the FL setting, measuring privacy-preserving capability, performance, and resource costs. Intensive experimental results on the GLUE benchmark reveal that (1) FedPETuning can reduce the considerable resource costs in FL settings while still achieving acceptable performance (e.g., FedAP reduces 97.4% of communication with only 0.7% performance degradation compared with FedFT) as shown in Table 2, and (2) FedPETuning has an appealing ability to defend against gradient inversion attacks, which can reduce the prevision of recovered words by an average of 40.7% compared to FedFT, as shown in Section 4.2. In summary, the major contributions of this paper are shown as follows:

- FedPETuning is the first benchmark to provide a holistic review of PETuning methods for PLMs under FL settings, covering privacy attacks, performance comparisons, and resource-constrained analysis.
- FedPEtuning can serve as a suite of baselines for efficient-parameter tuning of PLMs in a federated setting, and guide the community to design FL-tailored efficient parameter tuning algorithms.

Our research findings demonstrate the potential of combining large PLMs with FL, providing a promising training paradigm for privacy-preserving learning in the era of large language models (Ouyang et al., 2022; OpenAI, 2023).

2 Related Work

Federated Learning Federated learning (Konečný et al., 2016; McMahan et al., 2017) (FL), a widespread distributed learning technique used in privacy-sensitive tasks, has been hindered by not Independently and Identically Distributed (non-IID) (Kairouz et al., 2021), which results in accuracy discrepancies compared to centralized training. Extensive optimization studies have been conducted to address the non-IID issue, including data optimization (Zhao et al., 2018), model updating optimization (Chai et al., 2020), and model training optimization (Sahu et al., 2018). Recently, some work has tried to solve this problem from a pre-trained model initialization perspective (Chen et al., 2022b; Nguyen et al., 2022) and transformer model structure (Qu et al.,

[2]For example, the communication bandwidth between clients and server is constrained from a hundred Kbps to a few Mbps in most situations (Sui et al., 2020).

2022). Weller et al. (2022) experimentally show that using PLMs could reduce non-IID adverse effects and narrow down its accuracy gap to centralized learning. However, the significant communication overhead of large-scale PLMs are less considered in FL systems, leading to slow and impractical federated training in real-world tasks. Additionally, PLMs can pose challenges for local clients with limited hardware capabilities for computation and storage. In contrast, our study investigates PLMs' training in the FL context under resource constraints.

Injecting parameters-efficient tuning methods into federated learning Parameter-efficient tuning (PETuning) seeks to keep most parameters of PLMs frozen and fine-tune only additional lightweight parameters or a fraction of the parameters for downstream tasks (Houlsby et al., 2019; Li and Liang, 2021; Hu et al., 2022; Zaken et al., 2021). With this trait, PETuning methods can be utilized to mitigate the communication overhead in FL, which primarily relies on the size of model update parameters. In the field of computer vision, Sun et al. (2022) present the FedPEFT framework by injecting three PETuning methods (i.e., Bais, Adapter, Prompt) of the visual pre-trained models into FL, and find that lightweight PETuning methods in FL can significantly reduce the communication burden while maintaining performance and performing better in different FL settings. Meanwhile, Chen et al. (2022c) extend PETuning methods to the visual language model in FL and show that PETuning can facilitate a fast convergence rate. However, these studies ignore the important privacy attack issue existing in FL. With increasing attention to privacy concerns, validation of the privacy-preserving capabilities of federated PETuning methods is paramount and facilitates their practical deployment in real-world scenarios.

In the context of NLP, Zhao et al. (2022) first explore the effect of prompt-tuning under the FL setting and achieve acceptable performance results compared with fine-tuning. Xu et al. (2022) show that it is possible to train large vocabulary language models while preserving accuracy and privacy by adopting the low-rank adaptation (Hu et al., 2022). However, there has been no comprehensive investigation into the FL performance of the PETuning method for PLMs. Our research aims to bridge this gap and provide access to our code and data to inspire further exploration of the potential of this new paradigm for efficient federated NLP.

3 Federated Parameter-Efficient Tuning

In this section, we present how different PETuning methods work, followed by the training process of FedPETuning.

3.1 PETuning Methods

Denote the original PLM parameters by $\mathcal{W}_p = \{w_1, w_2, ..., w_N\}$ and the updated parameters by $\mathcal{W}_p' = \{w_1', w_2', ..., w_M'\}$ after training on the dataset $\mathcal{D}$. Define $\mathcal{W}_e$ as trainable model parameters. In vanilla fine-tuning, $|\mathcal{W}_e|$ are equal to the number of original model parameters and $N = M$, where $|\cdot|$ refers to the number of parameters. In the PETuning methods, most parameters of the PLM keep frozen and only a few added parameters or a part of the original model parameters are updated, that is, $M \geq N$ and $|\mathcal{W}_e| \ll N$. Following the taxonomy of Ding et al. (2022), PETuning methods could be divided into three groups, i.e., addition-based methods, specification-based methods, and reparameterization-based methods.

Addition-based methods introduce new trainable parameters into the frozen PLMs. These methods have two representative branches: **Adapter** tuning and **Prompt** tuning. The adapter tuning proposed by Houlsby et al. (2019) inserts adapter layers to vanilla Transformers. Specifically, two adapter layers are inserted into each Transformer block, wherein each adapter layer contains a down-projection and an up-projection. Given the input feature $h \in \mathcal{R}^d$, the down-projection $\mathcal{W}_d \in \mathcal{R}^{d\times r}$ projects the input h_{in} to a r-dimensional space and the up-projection $\mathcal{W}_u \in \mathcal{R}^{r\times d}$ maps back to the input size. Mathematically, the computation process is as follows,

$$\mathbf{h} \leftarrow \mathcal{W}_u^T f\left(\mathcal{W}_d^T \mathbf{h}\right), \quad (1)$$

where $f(\cdot)$ is the nonlinear activation function. In this strategy, adapter tuning could only fine-tune adapter layers $\mathcal{W}_e = \{\mathcal{W}_u, \mathcal{W}_d\}$ (about 0.5% to 8% parameters of the whole model) during the tuning process while keeping parameters of PLMs frozen.

On the contrary, prompt-tuning (Li and Liang, 2021; Liu et al., 2021) adds extra parameters without modifying the model architecture. Prompt-tuning converts the training objective of downstream tasks into a form similar to the pre-trained stage (Devlin et al., 2019; Liu et al., 2019b), by attaching trainable vectors $\mathcal{P}$, namely prompt, to the original input. During model training, prompt-tuning only adapts light-weight prompt vectors $\mathcal{W}_e = \{\mathcal{P}\}$, which scale to within 10% of the number of PLMs parameters.

Algorithm 1: Training process of FedPETuning

```
Parameters: Client set $\mathcal{C}$; Communication round $\mathcal{T}$; Local epoch number $\mathcal{E}$; The PLMs parameters $\mathcal{W}_p$; The local trainable and efficient parameters $\mathcal{W}_e$ and the local dataset $\mathcal{D}_k$ of the $k$-th client; Local PETuning Method $\mathbf{P}$;
Before Training: Initialize $\mathcal{W}_e^0$ on the server and $\mathcal{W}_p$ on each client in $\mathcal{C}$.
ServerGlobalAggregation:
for each communication round $t = 1$ to $\mathcal{T}$ do
    $\mathcal{C}^t \leftarrow$ (randomly sample K clients from $\mathcal{C}$)
    for each user $k \in \mathcal{C}^t$ in parallel do
        ClientLocalTuning($k$, $\mathcal{W}_e^{t-1}$)
    end
    Receive local updated parameters $\mathcal{W}_e^{k,t}$
    Perform global aggregation by Eq. (3)
end
ClientLocalTuning ($k$, $\mathcal{W}_e^t$) :
    $\mathcal{W}^t \leftarrow$ (assemble $\mathcal{W}_e^t$ and $\mathcal{W}_p$)
    for epoch $e = 1$ to $\mathcal{E}$ do
        $\mathcal{W}_e^{k,t+1} \leftarrow \mathbf{P}(\mathcal{D}_k, \mathcal{W}^t)$
    end
    Send $\mathcal{W}_e^{k,t+1}$ to the server
```

Specification-based methods aim to fine-tune a fraction of the parameters while keeping others frozen. In particular, Zaken et al. (2021) propose **BitFit** and empirically demonstrate that only tuning the bias terms $\mathcal{W}_e = \{\mathbf{b}_{(\cdot)}^{\ell,(\cdot)}\}$ of PLMs could still achieve competitive performance.

Reparameterization-based methods argue that the PLMs' adaptions can be re-parameterized into optimization within a low-dimensional subspace (Aghajanyan et al., 2020). Based on this hypothesis, **LoRA** (Hu et al., 2022) optimizes the low-rank decomposition for weight update matrices $\Delta\mathcal{W}$ during model training. For a pretrained weight matrix $\mathcal{W} \in \mathcal{R}^{d\times k}$, we have

$$\mathcal{W} + \Delta\mathcal{W} = \mathcal{W} + \mathcal{B}\mathcal{A}, \qquad (2)$$

where $\mathcal{B} \in \mathcal{R}^{d\times r}$, $\mathcal{A} \in \mathcal{R}^{r\times k}$, and the rank $r \ll min(d,k)$. In this way, LoRA could make training more efficient with less than 1% trainable parameters $\mathcal{W}_e = \{\mathcal{B}, \mathcal{A}\}$ and match the fine-tuning performance.

3.2 FedPETuning

Our work considers a conventional FL system that comprises a central server responsible for managing participated clients for local training and distributing the shared model parameters. Instead of communicating all cumbersome PLMs parameters, FedPETuning resorts to PETuning methods for exchanging lightweight parameters, as illustrated in Figure 1.

Before training, FedPETuning initializes the backbone PLM with $\mathcal{W}_p$ and the PETuning method $\mathbf{P}$ with efficient-parameter $\mathcal{W}_e$. Then global aggregation in the server and local updating in local clients are executed alternately.

Server Global Aggregation. The third-party service first randomly selects K clients from $\mathcal{C}$ and distributes trainable parameter $\mathcal{W}_e$ to these chosen clients. Then the server performs the federated aggregation based on the received parameters $\mathcal{W}_e^{k,t}$ from clients, and updates the $\mathcal{W}_e$ by

$$\mathcal{W}_e^{t+1} = \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{\sum_{k=1}^{K}|\mathcal{D}_k|}\mathcal{W}_e^{k,t}. \qquad (3)$$

Client Local Tuning. When selected clients download the trainable parameters, they assemble a whole model with lightweight parameters $\mathcal{W}_e$ and local PLM parameters $\mathcal{W}_p$. Then, selected clients train the assembled model with local private data $\mathcal{D}_k$. After local training, the k-th client sends its updated efficient parameters $\mathcal{W}_e^k$ to the central server for federated aggregation.

The training process described above is repeated until a specific criterion (e.g., the maximum number of communication rounds $\mathcal{T}$) is satisfied. This process can be summarized in Algorithm 1.

4 Experiments

In this section, we conduct extensive experiments for evaluating the performance of PETuning in the FL setting, covering privacy attacks (see Section 4.2), performance comparisons (see Section 4.3), and resource-constrained analysis (see Section 4.4). Besides, we also provide an in-depth ablation analysis of FedPETuning in terms of data heterogeneity (see Section 4.5), local training epochs (see Section 4.6), and different FL scenarios (see Section 4.7).

4.1 Experiments Setup

Dataset and non-IID partitioning Our experiments use six datasets from the GLUE benchmark. The reasons are as follows: (1) the GLUE datasets have emerged as the *de facto* standard benchmark for assessing PLMs' effectiveness in natural language understanding; (2) They have been exten-

Task	# Train	# Dev.	# Test	Metric
RTE	2,241	249	277	Accuracy
MRPC	3,301	367	408	F1 Score
SST-2	66,675	674	872	Accuracy
QNLI	103,695	1,048	5,463	Accuracy
QQP	360,210	3,639	40,430	Accuracy
MNLI	388,774	3,928	9,815	Accuracy

Table 1: Dataset descriptions and statistics.

sively leveraged to validate various PETuning methods (Li and Liang, 2021; Liu et al., 2021; Zaken et al., 2021; Houlsby et al., 2019), and (3) these datasets are large enough and convenient for FL data partitioning (non-IID). Due to the limitation of the unpublished test set in GLUE, we follow the previous studies (Liu et al., 2022; Zhao et al., 2022) and use the original validation set as the new test set and split a part of the training set as the validation set. The data breakdown for this benchmark is in Table 1.

For the non-IID data partitioning, we follow Lin et al. (2021) and partition the datasets by using the Dirichlet distribution as the class priors. In particular, we sample $\mathcal{D} \sim Dir(\alpha)$ and allocate data $\mathcal{D}_k$ to k-th client. α determines the degree of non-IID, and a lower value of α generates a high label distribution shift. Unless otherwise specified, we maintain a default setting of α = 1.0 for the Dirichlet parameter throughout our experiments.

Implementation Details We adopt the benchmark FL system FedAvg to simulate the FL setting, which has been applied to commercial products (Bonawitz et al., 2019). In FedAvg, clients upload local model parameters to the central server and download the aggregated model for the next training round. Following Lin et al. (2021), we set the communication round to 100 and the local training epoch to 1 for all tuning methods. Following Chen et al. (2022a), we utilize Roberta-Base (Liu et al., 2019a) as the local model released by Huggingface[3]. The FedAvg implementation is based on `FedLab` (Zeng et al., 2023).

Under the FedAvg training protocol, we primarily evaluate the full fine-tuning (FedFT) and four representative PETuning methods, covering adapter tuning (FedAP), prefix tuning (FedPF), LoRA (FedLR), and BitFit (FedBF). For FedAP, we follow the architecture in Houlsby et al. (2019), which interposes adapter modules into both the multi-head attention module and the feed-forward

[3]https://github.com/huggingface/transformers

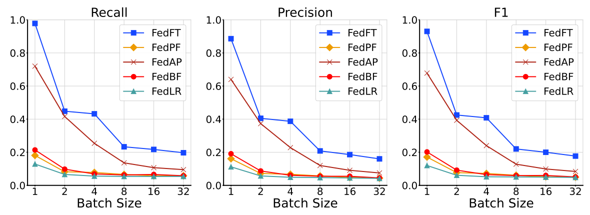

Figure 2: The data reconstruction attack results on the attack dataset with different tuning methods. Low performance is better. FedPETuning can effectively defend against data reconstruction attacks.

network in each Transformer layer. We use prefix tuning (Lester et al., 2021) as the representative of prompt tuning because it has better performance on the small PLMs, e.g., base versions of Roberta. For LoRA and BitFit, we take the architectures from their origin papers (Zaken et al., 2021; Hu et al., 2022). All PETuing methods are based on `OpenDelta`[4], which is a plug-and-play framework for parameter-efficient tuning.

To make a fair and reasonable comparison, we run a hyperparameter sweep for each dataset and tuning method. We select the best model according to the metric on the validation set and report test set metric. Especially, the learning rate is selected from {5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2}. We search the reduction factor from {16, 64} for FedAP, the prompt length from {8, 16, 64} for FedPF, and the scaling factor and rank from {8, 16} for FedLR. All experiments are done on a server with 8 Nvidia Tesla V100 GPUs with 32GB RAM each.

4.2 Privacy-Preserving Results

We first investigate the privacy-preserving capabilities of FedPETuing and FedFT. In this privacy attack experiment, we adopt DLG (Zhu et al., 2019) as our base attack method. Due to space limitations, we have omitted the working process of DLG and recommend the reader read the original paper (Zhu et al., 2019).

Setup and Metrics As the goal for the attacker with DLG is to recover text from client-uploaded gradients, we follow Song and Raghunathan (2020) and evaluate FedPETuning and FedFT in terms of *precision* (the average percentage of recovered words in the target texts), *recall* (the average percentage of words in the target texts are predicted) and *F1 score* (the harmonic mean between preci-

[4]https://github.com/thunlp/OpenDelta

Methods	RTE	MRPC	SST-2	QNLI	QQP	MNLI	Avg.	Rel.	Com.
FedFT	$\mathbf{70.3_{1.2}}$	$\mathbf{90.7_{0.3}}$	$\mathbf{94.0_{0.6}}$	$\mathbf{91.0_{0.4}}$	$\mathbf{89.5_{0.1}}$	$\mathbf{86.4_{0.2}}$	**83.1**	100.0%	1x
FedAP	$69.4_{2.6}$	$89.1_{1.2}$	$93.3_{0.6}$	$90.9_{0.4}$	$88.4_{0.2}$	$86.0_{0.4}$	82.4	99.1%	$\uparrow$60x
FedLR	$67.4_{4.2}$	$84.5_{4.5}$	$93.6_{0.5}$	$90.8_{0.3}$	$87.4_{0.3}$	$84.9_{0.4}$	81.0	97.5%	$\uparrow$141x
FedPF	$58.6_{2.2}$	$86.8_{1.0}$	$93.0_{0.6}$	$87.6_{0.5}$	$85.7_{0.3}$	$82.2_{0.3}$	78.4	94.3%	$\uparrow$12x
FedBF	$61.4_{1.7}$	$84.6_{2.7}$	$92.5_{0.7}$	$87.2_{0.5}$	$84.5_{0.5}$	$81.7_{0.2}$	77.8	93.6%	$\uparrow$190x
CenFT	$73.0_{1.4}$	$90.9_{0.6}$	$92.9_{0.2}$	$90.8_{0.5}$	$\mathbf{91.1_{0.2}}$	$86.0_{0.2}$	83.6	100.0%	-
CenAP	$\mathbf{76.0_{1.8}}$	$90.6_{0.8}$	$\mathbf{94.6_{0.5}}$	$\mathbf{92.9_{0.1}}$	$\mathbf{91.1_{0.1}}$	$\mathbf{87.5_{0.2}}$	**84.7**	101.3%	-
CenLR	$74.4_{2.4}$	$\mathbf{91.7_{0.6}}$	$94.0_{0.4}$	$92.7_{0.6}$	$90.1_{0.3}$	$87.0_{0.2}$	84.4	100.9%	-
CenPF	$65.6_{5.1}$	$90.2_{0.9}$	$93.7_{0.8}$	$91.5_{0.2}$	$89.5_{0.1}$	$86.7_{0.2}$	82.2	98.3%	-
CenBF	$70.9_{1.0}$	$91.3_{0.8}$	$94.1_{0.3}$	$91.3_{0.2}$	$87.4_{0.2}$	$84.6_{0.1}$	82.6	98.8%	-

Table 2: Performance results of the PETuning and FT on GLUE benchmark under the federated (upper half) and the centralized (bottom half) settings. With significantly reducing communication overhead, FedPETuning still maintains acceptable performance. The *Rel.* denotes the percentage of PETuning in terms of performance relative to FT. The blue value indicates more than 95% performance of FT, and red value indicates performance in excess of FT. The *Com.* denotes the normalized values of communication overhead of FedPETuning and FedFT. Mean and standard deviation are computed over 5 runs.

sion and recall). Specifically, we randomly selected 128 samples from the MNLI as the attack dataset.

Results Figure 2 shows the results of DLG on the attack dataset with different tuning methods. The results show that **FedPETuning can effectively defend against data reconstruction attacks compared to FedFT.** Since FedPETuning communicates a fraction of the entire model parameters with the server during the federated training process, it is intractable for the attacker to reconstruct the original text from such lightweight model parameters. Surprisingly, among FedPETuning, DLG is more likely to reconstruct private data from FedAP. We speculate that this result may be related to the design of PETuning methods. Adapter tuning inserts adapter layers to vanilla Transformers, which can encode the input data separately during model training. Compared to other PETuning methods, Adapter tuning is easier to "remember" the model input, thus causing more severe privacy leaks.

There is a clear and consistent trend across all tuning methods: DLG performance decreases as the batch size increases. The reason is that the DLG attack requires more parameters to be optimized in larger batch sizes. Therefore, increasing the batch size is a good defense strategy against the DLG attack. However, clients (e.g., mobile phone) in the FL system are usually resource-constrained in real-world applications. There is a trade-off between limited computing resources and large batches. In contrast, the FedPETuning method does not vary significantly across different batch sizes. In this sense, *FedPETuning can provide a practical defense strategy for clients with restricted computing resources.*

4.3 Performance Comparison

Table 2 shows the performance for four PETuning and FT under the federated (upper half table) and centralized settings (bottom half table). The results demonstrate that **FedPETuning maintains acceptable performance (more than 95% of FedFT's performance) while reducing the communication overhead substantially.**

From the upper half of Table 2, we find that although FedPETuning methods lag behind FedFT in the federated setting, the performance gap is relatively acceptable. For instance, FedAP reduces 60 times the communication overhead by only 0.7% performance degradation compared with FedFT. This result also shows that FedPETuning methods, especially FedAP and FedLR (more than 95% of FedFT's performance), can achieve a good level of trade-off between performance and communication overhead in practice.

In the centralized setting, some PETuning methods achieve competitive performance compared with CenFT. Specifically, CenAP outperforms CenFT on five out of six datasets, and the CenAP and CenLR achieve better performances on average. This result also supports our motivation that PETuning, stimulating colossal models with only a small portion of tunable parameters, can naturally be a promising way for FL under communication and resource constraints.

Comparing the upper and bottom parts of Table 2, we find that all tuning methods endure a decline in performance under the federated setting. Remarkably, the FedPETuning exhibits a significant

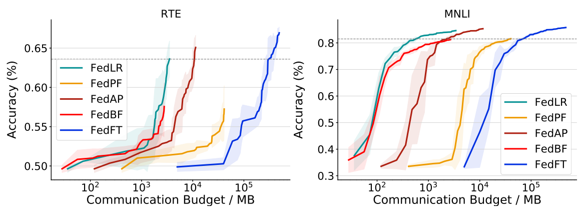

Figure 3: Accuracy versus Communication Budget for all tuning methods. The horizontal dashed line indicates the acceptable performance, which is 95% of the performance of CenFT. FedPETuning makes federated training more efficient by significantly reducing the communication overhead compared to FedFT.

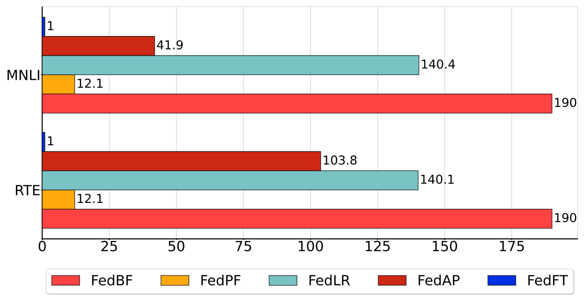

Figure 4: Normalized values of storage efficiencies of FedPETuning and FedFT on RTE and MNLI. Higher is better. FedPETuning lowers the storage cost to entry by 12~190 times.

drop in performance in the data heterogeneous FL context, which aligns with the results observed in data heterogeneity experiments (see Section 4.5). This result suggests that the *FL community may design FL-tailored PETuning methods to bridge the performance gap caused by the non-IID problem.* This will be explored in our future work.

4.4 Resource Costs

We next show the resource cost by different tuning methods under FL settings, including communication budget and client storage overhead. Figure 3 shows accuracy versus communication budget for all tuning methods on RTE and MNLI[5]. As shown in Figure 3, the communication budgets are ranked as FedFT $\gg$ FedPF $>$ FedAP $>$ FedLR $>$ FedBF. The experimental results show that **FedPETuning renders federated training remarkably efficient by significantly reducing the communication overhead as opposed to FedFT.** More communication overhead entails increased training time during uploading and downloading. For FL that necessitates high-frequency communication,

[5]The plots of remaining tasks can be found in Appendix B, which have similar results.

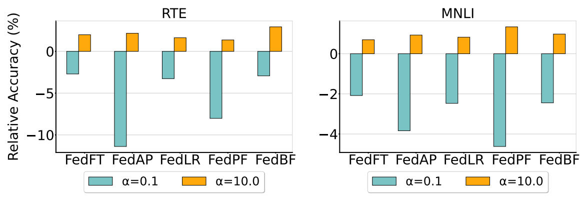

Figure 5: Performance variation of FedPETuning and FedFT under different non-IID with respect to $\alpha = 1.0$ on RTE and MNLI. FedPETuning is more susceptible to data heterogeneity than FedFT.

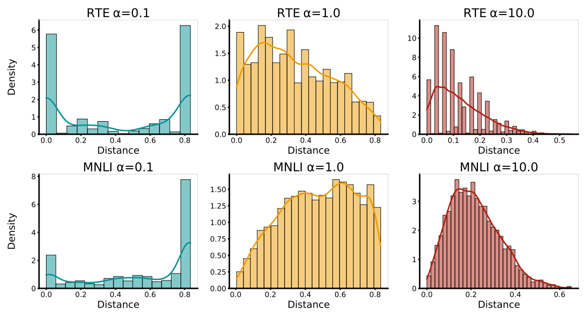

Figure 6: Data distribution under different Dirichlet parameter α. We report the pairwise Jensen–Shannon distance of the label distribution between two clients.

FedFT is extremely time-consuming, making the federated training slow. In this regard, FedPETuning is more pragmatic in real-world applications, particularly in communication-constrained FL challenges.

Figure 4 shows normalized values of storage efficiencies of FedPETuning and FedFT on RTE and MNLI. **FedPETuning lowers the storage cost to entry by 12~190 times.** This appealing characteristic is practiced for local clients of real-world FL systems. When deploying multiple tasks on a local client, FedPETuning can share PLM between different tasks, enabling the client to maintain only a few parameters for each task, thus reducing storage requirements.

4.5 Impact of Data Heterogeneity

As data heterogeneity is a fundamental challenge in FL, we also evaluate the performance of FedPETuning and FedFT under different data heterogeneity. Following Lin et al. (2021), we consider three Dirichlet distributions in this experiment by choosing α from $\{0.1, 1.0, 10.0\}$ where a smaller α indicates a sharper non-IID distribution among clients.

The performance of $\alpha = 1.0$ has already been discussed, Figure 5 illustrates performance changes of FedPETuning and FedFT with respect to the

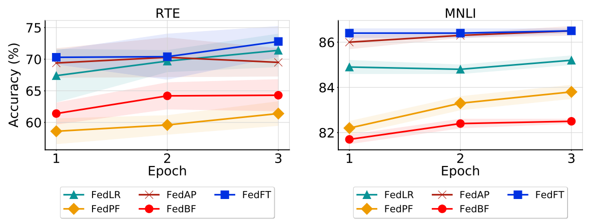

Figure 7: Performance comparison of FedPETuning and FedFT under different epochs on RTE and MNLI. Most tuning methods under the FL setting benefit from increased local training epochs.

Methods	RTE	MRPC	SST-2	Avg.	Rel.	Com.
FedFT	70.4	**90.2**	94.3	**85.0**	100.0%	1x
FedAP	**71.5**	88.5	94.0	84.7	99.7%	↑70x
FedLR	70.8	89.8	**94.4**	**85.0**	100.0%	↑141x
FedPF	66.4	88.1	93.7	82.7	97.3%	↑12x
FedBF	55.2	88.6	92.8	78.9	92.8%	↑189x

Table 3: Performance results of the FedPETuning and FedFT in the cross-silo setting. Significantly reducing the communication overhead, FedPETuning still achieves acceptable performance in the cross-silo FL scenario.

Methods	QNLI	QQP	MNLI	Avg.	Rel.	Com.
FedFT	**87.9**	**88.8**	**85.6**	**87.4**	100.0%	1x
FedAP	85.9	87.0	84.9	85.9	98.3%	↑52x
FedLR	86.0	86.5	84.7	85.7	98.1%	↑140x
FedPF	84.6	81.8	80.4	82.3	94.2%	↑12x
FedBF	80.5	84.0	80.7	81.7	93.5%	↑190x

Table 4: Performance results of FedPETuing and FedFT in the large-scale cross-device setting. With significantly reducing the communication overhead, FedPETuning still achieves acceptable performance in large-scale cross-device FL scenarios.

other two α values. The outcomes corresponding to other tasks can be found in Appendix B. Figure 5 reveals that **FedPETuning is more susceptible to data heterogeneity than FedFT.** As shown in Figure 5, although the increase in data heterogeneity (α from 1.0 to 0.1) degrades the performance of all tuning methods, FedPETuning degrades more dramatically compared to FedFT. This suggests more the federated model necessitates more trainable parameters to tackle intricate data heterogeneity, and *it may be arduous for FedPETuning to confront complex data heterogeneity in FL.*

Another noteworthy phenomenon is the performance of FedPETuning and FedFT do not change much when increasing α from 1.0 to 10.0. To figure out this, we report the Probability Density Function (PDF) of the data distribution under different α in Figure 6 (See Appendix A for other datasets). As shown in Figure 6, the data heterogeneity is similar for α equal to 1.0 and 10.0, while the distribution gap is large for α equal to 0.1. We think similar data heterogeneity between $\alpha = 1.0$ and $\alpha = 10.0$ contributes to this result.

4.6 Impact of Local Training Epochs

Figure 7 shows the performance of FedPETuning and FedFT with different local training epochs on RTE and MNLI. More results are provided in Appendix B. This result reveals that **most tuning methods under the FL setting benefit from increased local training epochs.** Federated models trained with more local epochs consistently yield a performance gain across different tuning methods when clients with heigh-resource data (like MNLI). In contrast, training with more epochs may incur a drop in performance for the low-resource dataset (like FedLR and FedAP on RTE). This may be due to the probability of PLMs becoming more susceptible to overfitting when clients with few data undergo more training epochs. On the contrary, clients with adequate data require more local training epochs to enhance the FL performance.

With the data scale into analysis, we also find that *all tuning methods are more unstable on small datasets* in Figure 7. For instance, the standard deviation performance of FedLR on RTE and MRPC is over 4.0% while other datasets are no more than 0.5%. This phenomenon is consistent with previous work (Chen et al., 2022a), and they experimentally show that increasing training steps can effectively mitigate the instability of the PETuning. However, this strategy fails in the FL setting. Training PLMs more stably in FL is a good research direction we will explore in our future work.

4.7 Various FL Scenarios

To verify the effectiveness of PETuning under different FL scenarios, we mainly consider two FL scenarios in this experiment, i.e., the cross-silo FL scenario (Kairouz et al., 2021) and the large-scale cross-device FL scenario (Lai et al., 2022). Cross-silo (Kairouz et al., 2021) is a vital application scenario for FL, suitable for several users (no more than 100). In Cross-silo, the server selects all clients for training in each communication round. Large-scale cross-device FL (Lai et al., 2022) is another federated scenario for deployment across thousands of clients. In the large-scale cross-device FL, data held by the local client is more scarce. In this experiment, we chose RTE, MRPC, and SST-2 to simulate the cross-silo FL scenarios, while QNLI, QQP, and MNLI to simulate the large-scale cross-device FL scenarios. The total number of clients in the cross-silo and large-scale settings is set to 10 and 1000, respectively. For both FL scenarios, ten clients are involved in each communication round for local training and federated aggregation.

Table 3 and Table 4 show the performance results of PETuning and FT under these two FL scenarios, respectively. The results show that **FedPETuning can significantly reduce the communication overhead while achieving acceptable performance in cross-silo and large-scale cross-device FL scenarios.** More noteworthy between the different FL scenes is the cross-silo setting. As shown in Table 3, the performance gap between the FedPETuning and FedFT diminishes under the cross-silo setting, compared to our standard setting in Section 4.3. For example, FedPF lags behind FedFT by 2.3% and 4.7% under the cross-silo FL scenario and the FL scenario of Table 2, respectively. We attribute this observation to relieving the data scarcity issue since each client has ten times more training samples in the cross-silo setting than in the standard setting.

5 Conclusion

This paper investigates parameter-efficient tuning methods of PLMs in the FL setting with extensive experiments for in-depth measurement of these methods under FL settings, covering privacy attacks, performance comparisons, and resource-constrained analysis. Experimental results unveil that FedPETuning can (1) achieve acceptable performance while reducing the colossal communication overhead and local storage cost, and (2) provide strong privacy-preserving capacity under different FL settings. To facilitate FL-tailored PETuning research, we have released our code and partitioned datasets, aiming to facilitate the use of FedPETuning and inspire the broader community to develop more suitable PETuning methods in future federated learning research.

Limitation

One limitation of this paper is that we do not validate FedPETuning on large scale models, (e.g., T5 (Raffel et al., 2019), LLaMa (Touvron et al., 2023), Vicuna (Chiang et al., 2023), etc). Although the parameter efficiency of PETuning is more impressive on larger pre-trained models, we have to consider the limited computational resources of clients in FL, making it challenging to deploy such a large-scale model. In addition, with the increasing size of modern pre-trained models, the community needs to design FL-friendly PETuning methods. In this sense, our work can serve as a benchmark and guide for future exploration of PETuning in FL.

Acknowledgements

We'd like to thank all the anonymous reviewers for their careful readings and valuable comments. This work was partially supported by the National Key Research and Development Program of China (No. 2018AAA0100204), a key program of fundamental research from Shenzhen Science and Technology Innovation Commission (No. JCYJ20200109113403826), the Major Key Project of PCL (No. 2022ZD0115301), and an Open Research Project of Zhejiang Lab (NO.2022RC0AB04).

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2020. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*.

Priyam Basu, Tiasa Singha Roy, Rakshit Naidu, and Zumrut Muftuoglu. 2021. Privacy enabled financial text classification using differential privacy and federated learning. *arXiv preprint arXiv:2110.01643*.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388.

Zheng Chai, Yujing Chen, Liang Zhao, Yue Cheng, and Huzefa Rangwala. 2020. Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data. *CoRR*, abs/2010.05958.

Guanzheng Chen, Fangyu Liu, Zaiqiao Meng, and Shangsong Liang. 2022a. Revisiting parameter-efficient tuning: Are we really there yet? *arXiv preprint arXiv:2202.07962*.

Hong-You Chen, Cheng-Hao Tu, Ziwei Li, Han-Wei Shen, and Wei-Lun Chao. 2022b. On pre-training for federated learning. *arXiv preprint arXiv:2206.11488*.

Jinyu Chen, Wenchao Xu, Song Guo, Junxiao Wang, Jie Zhang, and Haozhao Wang. 2022c. Fedtune: A deep dive into efficient federated fine-tuning with pre-trained transformers. *arXiv preprint arXiv:2211.08025*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

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*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

Suyu Ge, Fangzhao Wu, Chuhan Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2020. Fedner: Medical named entity recognition with federated learning. *arXiv preprint arXiv:2003.09288*.

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.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Shaoxiong Ji, Shirui Pan, Guodong Long, Xue Li, Jing Jiang, and Zi Huang. 2019. Learning private neural language modeling with attentive aggregation. In *2019 International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE.

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.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

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. *arXiv preprint arXiv:2104.08691*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. 2021. Fednlp: Benchmarking federated learning methods for natural language processing tasks. *arXiv preprint arXiv:2104.08815*.

Ruixuan Liu, Fangzhao Wu, Chuhan Wu, Yanlin Wang, Lingjuan Lyu, Hong Chen, and Xing Xie. 2022. No one left behind: Inclusive federated learning over heterogeneous devices. *arXiv preprint arXiv:2202.08036*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *CoRR*, abs/2110.07602.

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

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

Guodong Long, Yue Tan, Jing Jiang, and Chengqi Zhang. 2020. Federated learning for open banking. In *Federated learning*, pages 240–254. Springer.

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

John Nguyen, Jianyu Wang, Kshitiz Malik, Maziar Sanjabi, and Michael Rabbat. 2022. Where to begin? on the impact of pre-training and initialization in federated learning. *arXiv preprint arXiv:2210.08090*.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Chen Qu, Weize Kong, Liu Yang, Mingyang Zhang, Michael Bendersky, and Marc Najork. 2021. Natural language understanding with privacy-preserving bert. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1488–1497.

Liangqiong Qu, Yuyin Zhou, Paul Pu Liang, Yingda Xia, Feifei Wang, Ehsan Adeli, Li Fei-Fei, and Daniel Rubin. 2022. Rethinking architecture design for tackling data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10061–10071.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. 2018. On the convergence of federated optimization in heterogeneous networks. *CoRR*, abs/1812.06127.

Congzheng Song and Ananth Raghunathan. 2020. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 377–390.

Dianbo Sui, Yubo Chen, Jun Zhao, Yantao Jia, Yuantao Xie, and Weijian Sun. 2020. Feded: Federated learning via ensemble distillation for medical relation extraction. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 2118–2128.

Guangyu Sun, Matias Mendieta, Taojiannan Yang, and Chen Chen. 2022. Exploring parameter-efficient fine-tuning for improving communication efficiency in federated learning. *arXiv preprint arXiv:2210.01708*.

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*.

Orion Weller, Marc Marone, Vladimir Braverman, Dawn Lawrie, and Benjamin Van Durme. 2022. Pretrained models for multilingual federated learning. *arXiv preprint arXiv:2206.02291*.

Mingbin Xu, Congzheng Song, Ye Tian, Neha Agrawal, Filip Granqvist, Rogier van Dalen, Xiao Zhang, Arturo Argueta, Shiyi Han, Yaqiao Deng, et al. 2022. Training large-vocabulary neural language models by private federated learning for resource-constrained devices. *arXiv preprint arXiv:2207.08988*.

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*.

Dun Zeng, Siqi Liang, Xiangjing Hu, Hui Wang, and Zenglin Xu. 2023. Fedlab: A flexible federated learning framework. *Journal of Machine Learning Research*, 24(100):1–7.

Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. 2022. Reduce communication costs and preserve privacy: Prompt tuning method in federated learning. *arXiv preprint arXiv:2208.12268*.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. *CoRR*, abs/1806.00582.

Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in neural information processing systems*, 32.

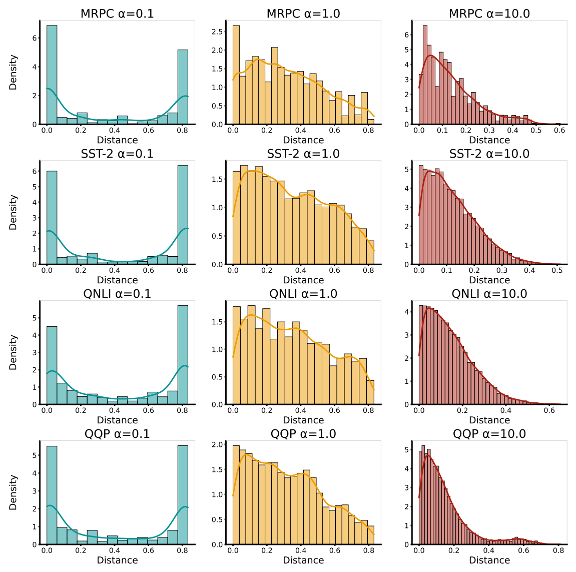
Figure 8: Data distribution under different α. We report the pairwise Jensen–Shannon distance of the label distribution between two clients.

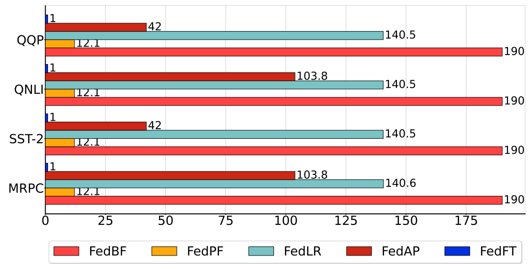
Figure 9: Normalized values of storage efficiencies of FedPETuning and FedFT on the other four tasks. Higher is better.

A Non-IID Partitionings Results

Figure 8 illustrates the data distribution under different α. It is observed that data distributed in clients with α as 0.1 has a large distance with each other, which has impact on the performance. Both α as 1.0 and 10.0 are considered to produce a more uniform distribution.

B Extra Results

Communication Analysis In this section, we illustrate the accuracy given the communication budget on the remaining four tasks in Figure 10. As can be seen, PETuning methods consistently reduce the communication budget over several orders of magnitude while providing comparable performance. Moreover, PETuning methods (apart from FedBF) achieve acceptable accuracy (90% of fine-tuning) on all the tasks, demonstrating the effectiveness of these methods.

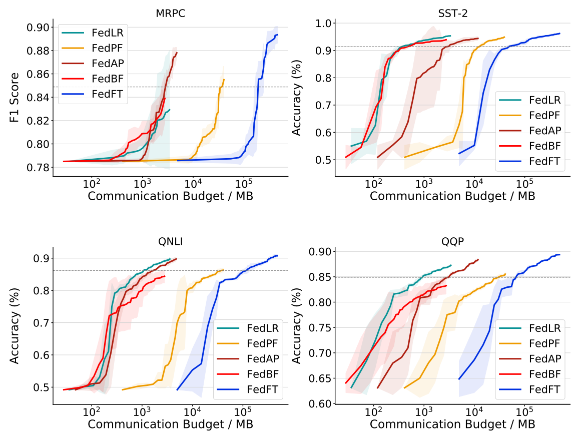
Figure 10: Accuracy versus Communication Budget for all tuning methods. The horizontal dashed line indicates the acceptable performance, which is 95% of the accuracy of CenFT.

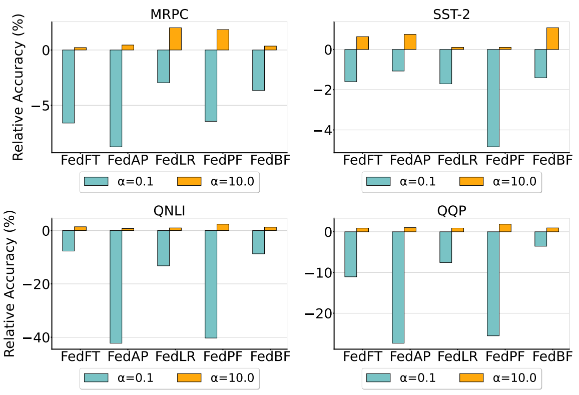
Figure 11: Relative performance of FedPETuning and FedFT on RTE and MNLI under different Dirichlet distributions.

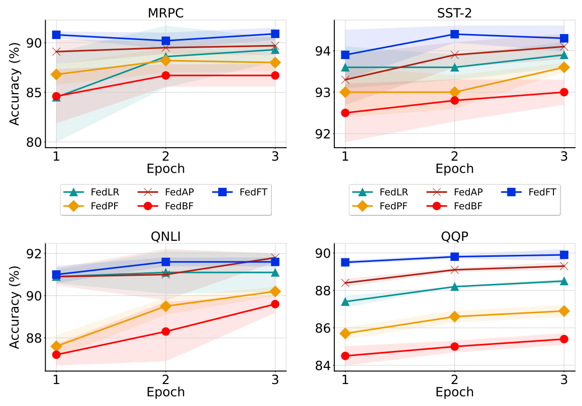
Figure 12: Accuracy comparison of PETuning methods and FT under different epochs.

Non-IID Analysis Figure 11 presents the comparisons of accuracy under different distributions by varying α. All the methods obtain better performance with α as 1.0 than α as 0.1, showing that non-IID distribution has a negative impact on the model's performance. On the contrary, varying α from 1.0 to 10.0 has little impact in most circumstances.

Local Training Epoch Analysis In Figure 12, we show the accuracy of tuning methods trained with different numbers of training epochs on the local clients. The accuracy of the relatively small datasets (MRPC, SST-2) shows a faint decreasing trend because of the over-fitting issue. All the tuning methods benefit from more training epochs on relatively large datasets (QNLI, QQP).

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitation Section

☒ A2. Did you discuss any potential risks of your work?
Our work does not involve potential risks

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and introduction

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
4.1

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
4.1.2

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
4.1.2

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.