FedPFT: Federated Proxy Fine-Tuning of Foundation Models

Zhaopeng Peng\textsuperscript{1}, Xiaoliang Fan\textsuperscript{1,*}, Yufan Chen\textsuperscript{1}, Zheng Wang\textsuperscript{1}, Shirui Pan\textsuperscript{2}, Chenglu Wen\textsuperscript{1}, Ruisheng Zhang\textsuperscript{3}, Cheng Wang\textsuperscript{1}

\textsuperscript{1}Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Informatics, Xiamen University
\textsuperscript{2}School of Information and Communication Technology, Griffith University
\textsuperscript{3}School of Information Science and Engineering, Lanzhou University

pengzhaopeng@stu.xmu.edu.cn, fanxiaoliang@xmu.edu.cn, \{yufanchen, zhangrs\}@stu.xmu.edu.cn, s.pan@griffith.edu.au, clwen@xmu.edu.cn, zwang@xmu.edu.cn

Abstract

Adapting Foundation Models (FMs) for downstream tasks through Federated Learning (FL) emerges a promising strategy for protecting data privacy and valuable FMs. Existing methods fine-tune FM by allocating sub-FM to clients in FL, however, leading to suboptimal performance due to insufficient tuning and inevitable error accumulations of gradients. In this paper, we propose Federated Proxy Fine-Tuning (FedPFT), a novel method enhancing FMs adaptation in downstream tasks through FL by two key modules. First, the sub-FM construction module employs a layer-wise compression approach, facilitating comprehensive FM fine-tuning across all layers by emphasizing those crucial neurons. Second, the sub-FM alignment module conducts a two-step distillations—layer-level and neuron-level—before and during FL fine-tuning respectively, to reduce error of gradient by accurately aligning sub-FM with FM under theoretical guarantees. Experimental results on seven commonly used datasets (i.e., four text and three vision) demonstrate the superiority of FedPFT. Our code is available at https://github.com/pzp-dzd/FedPFT.

1 Introduction

In recent years, various transformer-based Foundation Models (FMs) [Bommasani et al., 2021] such as BERT [Kenton and Toutanova, 2019], GPT [Radford et al., 2021], LLaMA [Touvron et al., 2023], and ViT [Dosovitskiy et al., 2020] have attained state-of-the-art performance across a diverse range of natural language processing (NLP) and computer vision (CV) tasks, yet also face both data privacy and FM copyright concerns. For instance, a FM trained on medical data might inadvertently memorize sensitive patient information, and companies that own closed-source FMs may choose not to share FMs with the public. Federated Learning (FL) [McMahan et al., 2017] offers a privacy-preserving approach for collaborative fine-tuning of FMs among multiple participants. This approach is increasingly promising for FM fine-tuning applications, ensuring the adaptation of downstream tasks without directly sharing client data and server FM.

Recent methods [Xiao et al., 2023; Marchisio et al., 2023] mainly aim to fine-tune FMs without using the full model, which leverage layer-drop techniques [Sajjad et al., 2023] to compress a FM and derive a sub-FM, enabling approximate fine-tuning of the original FM. However, these methods still pose two significant challenges that adversely reduce the performance of fine-tuned FMs. On one hand, they failed to fine-tune FMs sufficiently as a result of discarding those intermediate layers of FMs, consequently leading to the performance degradation of fine-tuned FMs. As shown in Fig.1(a), layer-drop methods fail to update intermediate layers of the FM during fine-tuning, due to the mismatch between the FM and the constructed sub-FM. On the other hand, there is a potential defect for the accumulation of gradient errors of FMs due to the lack of alignment between sub-FMs and FMs during FL fine-tuning, subsequently leading to further performance degradation. Fig.1(b) shows that, due to the absence of alignment, existing methods might accumulate significant gradients update errors between the FM and its constructed sub-FM during the FL fine-tuning process.

To address the above two challenges, we propose a framework called Federated Proxy Fine-Tuning (FedPFT) to enhance the adaptation of FMs for downstream tasks, while neither server FMs nor client data are directly shared. First, we design the sub-FM construction module, which performs layer compression on FMs to obtain sub-FMs by measuring neurons saliency of Feed-Forward Network (FFN) in transformer, facilitating comprehensive fine-tuning of FMs by emphasizing those crucial neurons. Second, we design the sub-FM alignment module, which conducts a two-step distillations—layer-level and neuron-level—before and during FL fine-tuning respectively, ensuring the accurate alignment between sub-FMs and FMs with a theoretical guarantee. Extensive experiments on three FMs and seven commonly used datasets demonstrate that FedPFT outperforms existing baselines that fine-tune FMs without using the full model.

Our contributions can be summarized as follows:

\begin{itemize}
  \item We introduce FedPFT, a novel federated fine-tuning of FM method that establishes a sub-FM as a local proxy.
\end{itemize}
FedPFT effectively improves fine-tuning performance while maintaining the critical constraint that neither the server FM nor the client data is directly shared.

- We propose the first module for constructing sub-FMs through layer-wise compression. This technique maintains layer correspondence across sub-FMs and FMs, ensuring the comprehensive fine-tuning of FM layers while also considering the alleviation of training overhead.
- We propose the second module to align sub-FMs with FMs via a two-step distillation—layer-level and neuron-level—before and during FL fine-tuning respectively. Additionally, we offer theoretical insights into the significance of distillation for fine-tuning using sub-model.
- We conduct extensive experiments on three FMs and seven commonly used datasets. Results demonstrate that FedPFT consistently outperforms existing baselines.

2 Related Works

2.1 FM Fine-Tuning Through FL

Traditional centralized fine-tuning faces privacy concerns due to data sharing. Recent works [Chen et al., 2023a; Yu et al., 2023; Zhuang et al., 2023] introduce the concepts of Federated Foundation Models, to alleviate privacy concerns. [Fan et al., 2023; Kuang et al., 2023] propose various Fed-LLM platforms to support federated training of LLMs. [Xu et al., 2023] fine-tune FM via FL on mobile devices. [Chen et al., 2023b] apply FM to federated multi-task learning. [Chen et al., 2023c] save the communication cost during FL training through block-level parameters dropout. [Wang et al., 2023a] reduce the communication and computation cost by training different layers of BERT in each round. [Zhang et al., 2023] apply parameter-efficient fine-tuning (PEFT) methods to federated fine-tuning of FMs for privacy defense. However, most of aforementioned methods rely on sharing the server FM. This limitation may pose risks of FM copyright leakage and impose a substantial computational burden on clients.

2.2 FM Fine-Tuning Without Using the Full Model

Early PEFT methods, including Lora [Hu et al., 2021], Adapter-Tuning [Houlsby et al., 2019], and Prefix-Tuning [Li and Liang, 2021], focus on efficient fine-tuning of complete FMs by reducing the number of tunable parameters. Despite these efforts, the gradient computation for tunable parameters still needs backpropagation through the entire FM [Sung et al., 2022]. Recently, Offsite-Tuning [Xiao et al., 2023] is proposed to achieve fine-tuning without the full model. In this approach, the FM owner sends a light-weight adapter and an emulator constructed through layer-drop and knowledge distillation [Hinton et al., 2015] to clients. Clients then fine-tune the adapter on their data with support from the emulator. The refined adapter is subsequently returned and incorporated into the full model for the fine-tuning process. Similarly, mini-model adaptation [Marchisio et al., 2023] constructs a shallow mini-model from a fraction of the FM’s parameters. However, those methods either discard significant amount of intermediate layers in FM or face the problem of gradient error accumulation, resulting in sub-optimal fine-tuning performance. Different from conventional methods, we construct sub-FMs based on layer-wise compression and mitigate gradient error accumulation by a two-step distillations.

3 FedPFT

3.1 Preliminary

Federated Learning

Given N parties  \( P_i (i = 1, ..., N) \), each party holds data  \( D_i \), Let  \( L(\cdot, \cdot) \) be the loss function. FL aims to train a machine learning model  \( \Theta \) using the dataset  \( D = \cup D_i (i = 1, ..., N) \)
under the coordination of a server $S$, while the raw data of all parties are not directly shared, which is formally described as

$$\Theta = \arg \min_{\Theta} \sum_{i=1}^{N} \frac{|D_i|}{|D|} L(\Theta, D_i). \quad (1)$$

**Foundation Model Fine-tuning**

Given a foundation model $\Theta = \{W_1, W_2, ..., W_n\}$ and a downstream task dataset $D$, the fine-tuning aims to obtain a new model $\Theta^* = \{W_1', W_2', ..., W_n'\}$, it is

$$\Theta^* = \Theta + \Delta \Theta,$$

$$\Delta \Theta = \arg \min_{\Delta \Theta} L(\Theta + \Delta \Theta, D). \quad (2)$$

### 3.2 Problem Definition

For FM fine-tuning using proxy sub-model, we first construct a sub-model $\Theta' = \{W_1', W_2', ..., W_k', W_{k+1}', ..., W_n'\}$ with fewer parameters for $\Theta'$ to act as a proxy. Second, fine-tune the proxy sub-model $\Theta'$ using the dataset $D$. Finally synchronize the updated gradients on $\Theta'$ to $\Theta$. Specifically, we construct $\Theta'$ by compressing $\Theta$, and retain a portion of the parameter matrix in $\Theta$ during the compression process. This compression process is formally described as follows:

$$\Theta' = \Theta_1 \cup C(\Theta_2), \quad (3)$$

where $\Theta_1 \cup \Theta_2 = \Theta$, and $C(\cdot)$ denotes the compression method. During the fine-tuning of $\Theta'$, we update only $\Theta_1$ and synchronize the updated gradient on $\Theta_1$ into $\Theta$ after fine-tuning to obtain $\Theta^*$'s approximation of $\Theta'$, which is formally described as

$$\Theta'' = (\Theta_1 + \Delta \Theta_1') \cup \Theta_2,$$

$$\Delta \Theta_1' = \arg \min_{\Delta \Theta_1'} L((\Theta_1 + \Delta \Theta_1') \cup C(\Theta_2), D). \quad (4)$$

### 3.3 Method Overview

The overall framework of ours FedPFT is shown in Fig. 2. We first derive a proxy sub-FM for the server FM, then collaboratively fine-tune the sub-FM through FL, and finally synchronize the updates on the sub-FM to the FM by plugging-in. FedPFT enhances downstream tasks adaptation of FMs through FL by two key modules: (1) **Sub-FM Construction Module** that constructs sub-FMs by performing layer-wise compression on FMs based on neuron saliency; and (2) **Sub-FM Alignment Module** that reduces the difference between FMs and sub-FMs by layer-level and neuron-level knowledge distillation before and during FL fine-tuning, respectively. We will introduce those two modules in details as follows.

### 3.4 Sub-FM Construction Module Based on Layer-Wise Compression

Transformer-based FM typically consist of three parts: an embedding layer, a task head, and a sequence of transformer layers. Since the size of FM is dominated by all transformer layers, we perform compression for each transformer layer.

Each transformer layer contains two sub-layers: Multi-Head Attention (MHA) and Feed-Forward Network (FFN), each of which applies residual connection and followed by layer normalization. The output of MHA is

$$\text{MHA}(x) = \text{Concat}(\text{Attn}_q(x), ..., \text{Attn}_k(x))W^O,$$

$$\text{Attn}(x) = \text{softmax} \left( xW^Q(xW^K)^T \right) xW^V, \quad (5)$$

where $W^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W^K \in \mathbb{R}^{d_{model} \times d_k}$, $W^V \in \mathbb{R}^{d_{model} \times d_k}$ and $W^O \in \mathbb{R}^{d_{model} \times d_k}$. $h$ is the number of attention heads, $d_k$ and $d_{model}$ are the dimensions of key and FM, respectively, and $d_{model} = d_k \times h$. The parameters number of MHA is about $4d_{model}^2$. 

![Figure 2: The overall framework of FedPFT that enhances FMs adaptation in downstream tasks through FL by two key modules: (1) Sub-FM Construction Module constructs sub-FM by layer-wise compression to facilitate comprehensive FM fine-tuning; and (2) Sub-FM Alignment Module aligns sub-FMs by two-step distillation to ensure accurate alignment between sub-FM and FM with a theoretical guarantee.](image-url)
The output of FFN is
\[ \text{FFN}(x) = \text{gelu}(xW_1 + b_1)W_2 + b_2, \] (6)
where \( W_1 \in \mathbb{R}^{d_{\text{model}} \times d_{ff}} \) and \( W_2 \in \mathbb{R}^{d_{ff} \times d_{\text{model}}} \) are the weight matrices of two linear layers in FFN, respectively, \( b_1 \in \mathbb{R}^{d_{ff}} \) and \( b_2 \in \mathbb{R}^{d_{\text{model}}} \) are the bias, \( d_{ff} \) is the dimensions of FFN and is usually set to \( 4 \times d_{\text{model}} \). The parameters number of FFAs is about \( 8d^2_{\text{model}} \). Obviously, it is that most of the parameters in transformer layer are located in FFN.

Hence, we opt to compress the FFN rather than the MHA of each layer for sub-FM construction. This minimizes the parameters number of sub-FFN while ensuring a consistent set of trainable parameters (i.e., MHA) between the FM and its sub-FFN at each layer. We accomplish layer-wise compression by systematically removing neurons with low saliency in the FFN of each layer, employing a fixed ratio.

First, by further transforming (6), we can represent the output of FFN as the sum of \( d_{ff} \) neurons outputs:
\[ \text{FFN}(x) = \sum_{i=1}^{d_{ff}} (\text{gelu}(xu_i + b_{1i})u_i) + b_2, \] (7)
where \( u_i \in \mathbb{R}^{d_{\text{model}}} \) is the \( i \)th column vector in \( W_2 \), \( u_i \in \mathbb{R}^{d_{\text{model}}} \) is the \( i \)th row vector in \( W_1 \), \( b_{1i} \) is the \( i \)th item in \( b_1 \).

Second, based on (7) and magnitude-based pruning method [Wen et al., 2016], we use the L2-norm of all connect weights of neuron to measure its saliency, that is:
\[ \text{Saliency}(i) = \sqrt{\sum_{j=1}^{d_{\text{model}}} (w_{ij}^2 + u_{ij}^2)}, \] (8)
where \( i \) is the index of neurons in FFN.

Finally, we construct a sub-FFN serving as a proxy for the FM, accomplished by systematically eliminating neurons with low saliency in each layer at a fixed ratio.

3.5 Sub-FM Alignment Module Based on Two-Step Knowledge Distillation

In accordance with the description of FN fine-tuning using proxy sub-model in 3.2, it is evident that the FM fine-tuning is entirely contingent on the gradient descent of its sub-FM. This fine-tuning methodology prompts a fundamental question: How can we ensure the convergence of FM to the optimal value with the assistance of its sub-FM?

**Theorem 1.** Suppose both the function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) and its approximation \( f' : \mathbb{R}^n \rightarrow \mathbb{R} \) are convex and differentiable, and their gradient are Lipschitz continuous with constant \( L_1 > 0 \) and \( L_2 > 0 \), respectively, i.e., we have that \( \| \nabla f(x) - \nabla f(y) \|_2 \leq L_1 \| x - y \|_2 \) and \( \| \nabla f'(x) - \nabla f'(y) \|_2 \leq L_2 \| x - y \|_2 \) for any \( x, y \). Then if we run gradient descent for \( k \) iterations on \( f' \) with a fixed step size \( \eta \leq \frac{1}{L_1} \) and synchronize the gradient to \( f \), let \( \nabla f' - \nabla f = \delta \), when satisfying
\[ \| \delta \|_2^2 \leq \frac{1}{2} \| \nabla f \|_2^2, \] (9)
\[ \eta \sum_{i=1}^{k} \| g(i) \|_2^2 \leq \sum_{i=1}^{k} (g(i), x(i) - x^*), \] (10)
it will yield a solution \( f^{(k)} \) which satisfies
\[ f(x^{(k)}) - f(x^*) \leq \frac{\| x(0) - x^* \|_2^2}{2\eta K}, \] (11)
where \( f(x^*) \) is the optimal value.

**Proof.** See Appendix 1.A

Intuitively, Theorem 1 indicates that when (9) and (10) are satisfied, gradient descent of FM with the help of sub-FM is guaranteed to converge and converges with rate \( O(\frac{1}{k}) \). It is evident that both conditions (9) and (10) are constraints on the difference between the actual and ideal update gradients of FM, and thus how to minimize the difference of the update gradients becomes a problem to be solved in the next step.

**Theorem 2.** For a transformer, let the number of attention head be 1, and ignoring its nonlinear function and residual connection, its output can be expressed as \( y = xW'Q(xW'_K)^T xW'V W'O xW_1 W_2 \), let \( A = W'Q(W'K)^T, B = W'V W'O, C = W_1 W_2 \), then \( y = xAx^T xBC \), and the output of its corresponding sub-layer after compressing FFN layer is expressed as \( y' = xAx^T xBC' \), assuming that the gradient of loss function \( f(y) \) is Lipschitz continuous with constant \( L_3 > 0 \) and \( \| C - C' \|_2 \leq \epsilon_1, \| y - y' \|_2 \leq \epsilon_2 \), there exists the constant \( K_1 > 0 \) and \( K_2 > 0 \) such that
\[ \left\| \frac{\partial \text{loss}}{\partial A} - \frac{\partial \text{loss}}{\partial A'} \right\|_2 \leq K_1 \epsilon_1 + K_2 \epsilon_2, \]
\[ \left\| \frac{\partial \text{loss}}{\partial B} - \frac{\partial \text{loss}}{\partial B'} \right\|_2 \leq K_1 \epsilon_1 + K_2 \epsilon_2. \] (12)

**Proof.** See Appendix 1.B

From Theorem 2, it is evident that shrinking the error of gradients can be achieved by narrowing the difference in output and weights between the sub-FM and FM.

Based on the above analysis, we grasp the importance of narrowing the difference between sub-FM and FM via knowledge distillation to boost the performance of FM that fine-tuned using sub-FM. Therefore, we propose a method to align sub-FM using layer-level and neuron-level distillations in two phases, before and during FL fine-tuning, respectively. These two distillation methods are shown in the Fig.3.

Layer-Level Distillation Before FL Fine-Tuning

Given that our sub-FFMs are constructed based layer-wise compression, where each layer retains a set of tunable parameters (i.e., MHA), we leverage the outputs from all layers to compute the layer-level distillation loss.

Furthermore, based on Theorem 2, we enhance the aforementioned distillation loss by introducing a regularization term. The purpose of this regularization term is to quantify the disparity between the weights of FFN and sub-FFN in each layer, to further facilitate a thorough knowledge transfer during fine-tuning by refining the alignment process. Thus,

\[1\text{https://arxiv.org/abs/2404.11536} \]
the final distillation loss is denoted as:

\[ L_{KD} = \frac{1}{L M_{KD}} \sum_{i=1}^{L} \left( \sum_{j=1}^{M_{KD}} \| O^{(i)}_j - O^{(i)}_j' \|^2 \right) + \mu \| W^{(i)}_1 - W^{(i)}_1' \|^2, \]  

where \( L \) is the number of layers, \( M_{KD} \) is the size of distill dataset, \( O^{(i)}_j \) is the output of the \( j \)th sample in the \( i \)th layer, \( W^{(i)}_1 \) and \( W^{(i)}_1' \) are the weight matrices of the \( i \)th FFN, \( \mu \) is the regularization coefficient.

Neuron-Level Distillation During FL Fine-Tuning

In addition, the absence of alignment between FM and its constructed sub-FM during FL fine-tuning may cause the actual gradient update direction of the FM to gradually deviate from its ideal direction. This deviation can substantially reduce the performance of the fine-tuned FM. To mitigate the problem, we re-align the sub-FM with the FM after FL aggregation in certain rounds.

However, since the datasets for distillation and the datasets for local fine-tuning are typically collected from different domains, excessively aligning sub-FMs through distillation may hinder the adaptation of sub-FMs to downstream tasks. Inspired by [Mallya and Lazebnik, 2018], during the alignment process in FL fine-tuning, we opt to update only a subset of neurons with low saliency in local fine-tuning to prevent the risk of sub-FM forgetting knowledge of local data.

Moreover, since all FFNs of sub-FM are not updated during FL fine-tuning, the effectiveness of magnitude-based neuron saliency measurement methods diminishes. To address this, we opt to select neurons for updating during alignment based on the Average Percentage of Zero activations (APoZ) of outputs on the downstream task dataset [Hu et al., 2016].

The APoZ of the \( k \)th neuron in \( i \)th layer is defined as:

\[ \text{APoZ}_{k}^{(i)} = \frac{1}{M_{DT} S} \sum_{j=1}^{M_{DT}} \sum_{l=1}^{S} \mathbb{1}(O^{(j)}_{kl} = 0), \]  

where \( M_{DT} \) is the size of the downstream task dataset, \( S \) is the sequence length of the \( j \)th sample, \( O^{(i)}_{kl} \) is the output of the \( l \)th token of the \( j \)th sample at the \( k \)th neuron in \( i \)th layer, \( \mathbb{1}(\cdot) \) is the indicator function.

We calculate the APoZ for each neuron on the client using the local dataset before each round that requires alignment, and subsequently select the neurons that need to be updated during alignment based on their APoZ values.

3.6 Cost Analysis

We perform a theoretical analysis of the computational and communication cost of FedPFT based on BERT, and other models such as RoBERTa and ViT are similar. Following the settings in [Wang et al., 2023a], we assume that all FL clients have the same training settings and exclude external differences such as local dataset size and hardware environment.

Computational Cost

Given a BERT model, let \( V \) be the vocabulary size, \( S \) be the sequence length, \( L \) be the number of layers, and \( c_f, c_b \) be the number of forward propagation and backward propagation respectively. Based on the analysis in 3.4, the computational cost of a BERT model is \( \mathcal{O}(d_{model}(V + S) + L(4d^2_{model} + S^2d_{model}) + 2Ld^2_{model}d_{model}) \) where the four terms denote the cost of embedding, MHA, FFN, and Add&Norm, respectively.

Based on the above information, the overall time complexity of the full model is computed as follows. First, the time complexity of embedding is \( \mathcal{O}(d_{model}(V + S)) \). Second, due to \( d_{ff} = 4d_{model} \), the forward propagation time complexity is about \( \mathcal{O}(c_f L(12Sd^2_{model} + S^2d_{model})) \). Identically, the backward propagation time complexity is \( \mathcal{O}(c_b L(12Sd^2_{model} + S^2d_{model})) \). Finally, the overall time cost is \( \mathcal{O}(d_{model}(V + S) + L(c_f + c_b)(12Sd^2_{model} + S^2d_{model})) \) and we have \( d_{model}(V + S) \ll (c_f + c_b)(12Sd^2_{model} + S^2d_{model}) \) and \( S < d_{model} \).

In FedPFT, because \( d_{ff} = d_{model} \) for sub-FM, the time complexity during local training is \( \mathcal{O}(L(c_f + c_b)(6Sd^2_{model} + S^2d_{model})) \). Thus, compared with full model, FedPFT could reduce almost half the computational cost of all clients.

Communication Cost

Following [Vaswani et al., 2017], the space complexity of a BERT model is \( \mathcal{O}(d_{model}(V + S) + 12Ld^2_{model} + 2d_{model}) \).

In FedPFT, if PEFT methods is not used, all model parameters need to be transmitted in each iteration, thus the communication cost will be in the complexity of \( \mathcal{O}(d_{model}(V + S) + 6Ld^2_{model} + 2d_{model}) \), again shrinking nearly half of the cost. If PEFT method is used, take Lora as an example, let \( t \) be the interval between two alignments during FL fine-tuning, \( q \) be the proportion of neurons that need to be updated during alignment, \( R \) be the rank of Lora, and Lora is only added to MHA, then the communication cost will be in the complexity of \( \mathcal{O}(8Lrd_{model} + \frac{2}{7}qLd^2_{model}) \). Since \( \frac{2}{7}qLd^2_{model} \) is usually smaller than \( R \), the complexity is about \( \mathcal{O}(Lrd_{model}) \), which does not impose a significant computational cost.

4 Experiments

4.1 Experimental Setting

Models and Datasets

Our evaluations span a variety of FMs, including two NLP FMs (i.e., BERT-base [Kenton and Toutanova, 2019],
RoBERTa-base [Liu et al., 2019]) and one CV FM (i.e., ViT-base [Dosovitskiy et al., 2020]). Three FMs share consistent hyper-parameters for the number of layers $L$, attention heads $h$, hidden dimension $d_{\text{model}}$, and FFN dimension $d_{\text{ff}}$, all set at $L = 12$, $h = 12$, $d_{\text{model}} = 768$, and $d_{\text{ff}} = 3072$. Our NLP FM evaluations encompass four text datasets: SST-2 [Socher et al., 2013], QNLI [Socher et al., 2013], MNLI [Williams et al., 2017], and QQP². The CV FM is evaluated on three image datasets: CIFAR-10 [Krizhevsky et al., 2009], CIFAR-100 [Krizhevsky et al., 2009] and Flowers [Nilsback and Zisserman, 2008]. Specifically, SST-2 is the text sentiment analysis task, QNLI and MNLI are the natural language inference tasks, QQP is the text matching task, while CIFAR-10, CIFAR-100 and Flowers focus on image recognition tasks. In addition, we employ the Bookcorpus [Zhu et al., 2015] and Wikipedia datasets for distillation of NLP sub-FMs, and the ImageNet-1k [Russakovsky et al., 2015] for the distillation of CV sub-FM, respectively. Detailed dataset descriptions can be found in Appendix 1.C.

**Baselines**

We compare FedPFT with two methods, including: (1) FedPETuning [Zhang et al., 2023] that performs parameter-efficient fine-tuning (PEFT) with full model through FL; and (2) FedOT that performs PEFT without full model through FL. It is worth noting that FedOT is the FL implementation of Offsite-Tuning [Xiao et al., 2023] with multiple clients.

**Hyper-parameters and Implementation**

In the FL scenario, we set up 100 clients with 500 total communication rounds and employ the Dirichlet data partition method [Hsu et al., 2019] to construct different label-skew data heterogeneity scenarios. In each communication round, we randomly select 10 clients for local fine-tuning, using a linear decay of the global learning rate over rounds and AdamW as the local fine-tuning optimizer. We use FedAvg [McMahan et al., 2017] for global model aggregation. For FedOT, following [Xiao et al., 2023], we use 2 layers at the bottom and 2 layers at the top as Adapter, and compress the intermediate 8 layers into 3 layers as Emulator. For FedPFT, we construct sub-FMs by performing layer-wise compression on the intermediate 10 layers. For fair comparison, we keep the number of trainable parameters the same for all three methods. The evaluation metric is the accuracy on the given validation set. All pre-trained models used in experiments are obtained from Hugging Face³. The FL scenarios were implemented using FLGo [Wang et al., 2023b], and all experiments are conducted in PyTorch 2.1 and NVIDIA 3090 GPUs. Other detailed hyper-parameters can be seen in Appendix 1.C.

### 4.2 Overall Comparisons

We first evaluate all three methods in Independent Identically Distributed (I.I.D) data distribution scenarios. Table.1 compares our FedPFT with two baselines for fine-tuning BERT and RoBERTa on four text datasets. We observe that: 1) the performance of all FMs fine-tuned by our FedPFT surpasses that achieved by FedOT and closely approaches the performance achieved by FedPETuning that fine-tuning using full model; and 2) FedOT exhibit a substantial performance gap compared to FedPETuning. This observed discrepancy from FedOT might be attributed to the absence of training the Emulator and the accumulation of gradient errors during fine-tuning.

We further validate the effectiveness of FedPFT for fine-tuning CV FM. Table.2 presents the comparison results on the ViT model with CIFAR-10 dataset, and FedPFT still outperforms FedOT and achieves competitive performance closer to FedPETuning. The results on other two vision datasets (i.e., CIFAR-100, Flowers) are shown in Appendix 1.D.

### 4.3 Impact of Data Heterogeneity

We then evaluate the performance of FMs fine-tuned by three different methods in Non-Independent Identically Distributed (Non-I.I.D) data distribution scenarios. For the datasets used in the data heterogeneity experiments, we unequally partition

---

2. https://huggingface.co/
3. https://huggingface.co/
the dataset into 100 clients following the label distribution $Y_i \sim \text{Dir}(\alpha)$, where $i$ is the client id, $p$ is the global label distribution, $\alpha$ is the degree of Non-I.I.D and a smaller $\alpha$ generates a high label distribution shift. We construct three different label-skewing scenarios by adjusting the value of $\alpha$: Dir-1.0, Dir-5.0, and Dir-10.0. Table 3 shows the comparison of three methods for fine-tuning BERT and RoBERTa in different data-heterogeneous scenarios. We observe that: 1) the performance of all methods declines as the degree of Non-I.I.D increases; 2) our FedPFT still outperforms FedOT and achieves competitive performance closer to FedPETuning in FedPFT in different data-heterogeneous scenarios. We observe that: 1) the performance of all methods declines as the degree of Non-I.I.D increases; 2) our FedPFT still outperforms FedOT and achieves competitive performance closer to FedPETuning. Visualisation of label distributions and results of Non-I.I.D experiments across seven datasets showcase the effectiveness of FedPFT. In the future, we aim to extend the application of FedPFT to larger-scale datasets with superior performance, all without direct sharing of either server FM or client data. Experimental results across seven datasets showcase the effectiveness of FedPFT. In the future, we aim to extend the application of FedPFT to larger-scale datasets.

### 4.4 Ablation Study

To evaluate the efficacy of individual components within FedPFT, we design the following variants for conducting ablation study:

- FedPFT_N which does not perform alignment by knowledge distillation;
- FedPFT_B which perform sub-FM alignment by knowledge distillation only before FL fine-tuning;
- FedPFT_D which perform sub-FM alignment by knowledge distillation only during FL fine-tuning;

Moreover, to validate the effectiveness of the sub-FM construction module of FedPFT, we list the experimental results of FedOT for comparison with FedPFT_B, as both perform sub-FM alignment by knowledge distillation only before FL fine-tuning. Results are shown in Table 4 and we observe that: 1) both FedPFT_N which does not align the sub-FMs entirely and FedPFT_D which lacks the alignment before FL fine-tuning, exhibit notably poor performance. This is consistent with our theoretical analysis (see Theorem 1), indicating that a significant disparity between the gradients of sub-FM and FM can impede the convergence of fine-tuning methods using proxy sub-model; 2) FedPFT_B outperforms FedOT, showing the effectiveness of the sub-FM construction module in FedPFT; and 3) FedPFT outperforms FedPFT_B, emphasizing the necessity of sub-FM alignment during FL fine-tuning. Results on other text and vision datasets are presented in Appendix 1.F.

### 4.5 Parameter Study

In addition, we conduct a study to investigate the impact of two hyper-parameters in the sub-FM alignment module: the interval $t$ between two alignments during FL fine-tuning and the proportion $p$ of neurons that need to be updated for each alignment. The effects of two hyper-parameters on the QNLI dataset are presented in Table 5, suggesting that both $t$ and $p$ should be chosen moderately. Regarding $t$, longer alignment intervals lead to the accumulation of gradient errors, while shorter intervals can prohibit the adaptation of downstream tasks. Concerning $p$, it is crucial to strike a balance between updating an adequate proportion of neurons and avoiding excessive disruption to local fine-tuning.

### 5 Conclusion

This paper introduces FedPFT, a federated fine-tuning framework designed for Foundation Models (FMs). FedPFT addresses critical challenges related to insufficient FM fine-tuning and the accumulation of gradient errors by employing layer-wise compression for sub-FM construction and aligning sub-FM through a two-step distillation process, respectively. This novel framework achieves optimal downstream task adaptation of FM, resulting in an effective fine-tuned FM with superior performance, all without direct sharing of either server FM or client data. Experimental results across seven datasets showcase the effectiveness of FedPFT. In the future, we aim to extend the application of FedPFT to larger-scale FMs for tackling more complex downstream tasks.

### Table 3: Non-IID experimental results on BERT and RoBERTa

<table>
<thead>
<tr>
<th>Method</th>
<th>Model</th>
<th>SST-2 Dir-1.0</th>
<th>SST-2 Dir-5.0</th>
<th>SST-2 Dir-10.0</th>
<th>QNLI Dir-1.0</th>
<th>QNLI Dir-5.0</th>
<th>QNLI Dir-10.0</th>
<th>Transformers Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>FedPETuning</td>
<td>BERT</td>
<td>90.6</td>
<td>90.9</td>
<td>91.1</td>
<td>87.3</td>
<td>87.9</td>
<td>88.1</td>
<td>81M</td>
</tr>
<tr>
<td>FedOT</td>
<td>BERT</td>
<td>89.1</td>
<td>89.9</td>
<td>90.1</td>
<td>82.4</td>
<td>82.9</td>
<td>83.1</td>
<td>47M</td>
</tr>
<tr>
<td>FedPFT</td>
<td>BERT</td>
<td>89.7</td>
<td><strong>91.1</strong></td>
<td><strong>91.6</strong></td>
<td><strong>86.1</strong></td>
<td><strong>86.9</strong></td>
<td><strong>87.2</strong></td>
<td><strong>47M</strong></td>
</tr>
<tr>
<td>FedPETuning</td>
<td>RoBERTa</td>
<td>93.0</td>
<td>93.3</td>
<td>93.5</td>
<td>90.2</td>
<td>90.6</td>
<td>90.8</td>
<td>81M</td>
</tr>
<tr>
<td>FedOT</td>
<td>RoBERTa</td>
<td>91.7</td>
<td>92.1</td>
<td>92.8</td>
<td>84.1</td>
<td>84.6</td>
<td>85.1</td>
<td>47M</td>
</tr>
<tr>
<td>FedPFT</td>
<td>RoBERTa</td>
<td><strong>92.1</strong></td>
<td><strong>92.7</strong></td>
<td><strong>93.1</strong></td>
<td><strong>87.2</strong></td>
<td><strong>87.6</strong></td>
<td><strong>88.1</strong></td>
<td><strong>47M</strong></td>
</tr>
</tbody>
</table>

Table 4: Ablation study of FedPFT on QNLI

<table>
<thead>
<tr>
<th>Method</th>
<th>Model</th>
<th>BERT</th>
<th>RoBERTa</th>
</tr>
</thead>
<tbody>
<tr>
<td>FedOT</td>
<td>BERT</td>
<td>83.9</td>
<td>85.3</td>
</tr>
<tr>
<td>FedPFT_N</td>
<td>BERT</td>
<td>83.0</td>
<td>78.9</td>
</tr>
<tr>
<td>FedPFT_B</td>
<td>BERT</td>
<td>86.9</td>
<td>86.7</td>
</tr>
<tr>
<td>FedPFT_D</td>
<td>BERT</td>
<td>84.3</td>
<td>85.5</td>
</tr>
<tr>
<td>FedPFT(ours)</td>
<td>BERT</td>
<td><strong>87.5</strong></td>
<td><strong>88.7</strong></td>
</tr>
</tbody>
</table>

Table 5: Parameter study of FedPFT on QNLI

<table>
<thead>
<tr>
<th>Hyper-Parameter</th>
<th>Model</th>
<th>BERT</th>
<th>RoBERTa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment interval $t$</td>
<td>5</td>
<td><strong>87.5</strong></td>
<td><strong>88.7</strong></td>
</tr>
<tr>
<td>Updated neurons proportion $p$</td>
<td>0.3</td>
<td><strong>87.4</strong></td>
<td><strong>87.8</strong></td>
</tr>
<tr>
<td>$t$</td>
<td>20</td>
<td>87.1</td>
<td>87.1</td>
</tr>
<tr>
<td>$p$</td>
<td>0.5</td>
<td><strong>87.5</strong></td>
<td><strong>88.7</strong></td>
</tr>
<tr>
<td>$t$</td>
<td>0.8</td>
<td>87.1</td>
<td>87.9</td>
</tr>
</tbody>
</table>
Acknowledgments
The research was supported by Natural Science Foundation of China (62272403) and the Fundamental Research Funds for the Central Universities (No. 20720220064).

References


[Nilback and Zisserman, 2008] Maria-Elena Nilback and Andrew Zisserman. Automated flower classification over a large number of classes. In 2008 Sixth Indian conference


