# Beyond End-to-End: Dynamic Chain Optimization for Private LLM Adaptation on the Edge

**Yebo Wu**[1†], **Jingguang Li**[1†], **Chunlin Tian**[1], **Kahou Tam**[1], **Zhijiang Guo**[2,3*], **Li Li**[1*]
[1]State Key Laboratory of IOTSC, University of Macau
[2]HKUST, [3]HKUST (Guangzhou)
{yc37926, mc45005, yc27402, yc37436, llili}@um.edu.mo
zhijiangguo@hkust-gz.edu.cn

## Abstract

Federated fine-tuning enables privacy-preserving LLM adaptation but faces a critical bottleneck: the disparity between LLMs' high memory demands and edge devices' limited capacity. To break the memory barrier, we propose **Chain Federated Fine-Tuning (CHAINFED)**, an innovative paradigm that forgoes end-to-end updates in favor of a sequential, layer-by-layer manner. It first trains the initial adapter to convergence, freezes its weights, and then proceeds to the next. This iterative train-and-freeze process forms an optimization chain, gradually enhancing the model's task-specific proficiency. CHAINFED further integrates three core techniques: 1) Dynamic Layer Co-Tuning to bridge semantic gaps between sequentially tuned layers and facilitate information flow; 2) Globally Perceptive Optimization to endow each adapter with foresight beyond its local objective; 3) Function-Oriented Adaptive Tuning to automatically identify the optimal fine-tuning starting point. Extensive experiments on multiple benchmarks demonstrate the superiority of CHAINFED over existing methods, boosting average accuracy by up to 46.46%.

## 1 Introduction

Large Language Models (LLMs) (Liu et al., 2024; Bai et al., 2023) are revolutionizing mobile intelligence, yet adapting them for downstream tasks remains a critical challenge. Centralized training is often rendered infeasible by privacy regulations that isolate task-specific data on user devices (McMahan et al., 2017; Wu et al., 2025c, 2024b,a). While federated fine-tuning (Zhang et al., 2024) offers a promising privacy-preserving solution for collaborative adaptation, its practical deployment on mobile devices is severely bottlenecked by the prohibitive resource demands of LLMs (Tian et al., 2026; Wu et al., 2025a,b).

To mitigate the resource costs, parameter-efficient federated fine-tuning methods (Wu et al., 2025d), particularly adapter-based approaches (Cai et al., 2022), have been widely adopted. By freezing the backbone and updating only lightweight parameters (Bian et al., 2025), these methods successfully reduce computational and communication overheads. However, they fail to alleviate the fundamental memory constraint: the entire model must still be loaded into memory. For instance, LLaMA2-7B (Touvron et al., 2023) requires approximately 25GB of memory, far exceeding the typical 4–12GB capacity of mobile devices (Xu et al., 2023). This memory wall precludes resource-constrained devices from participating, leaving valuable on-device data unutilized.

To this end, we introduce CHAINFED, an innovative federated fine-tuning paradigm that breaks the memory wall via chain optimization. Instead of updating the LLM end-to-end, CHAINFED decouples the optimization into a series of sequential stages, with each stage dedicated to a single adapter. It commences by training the first adapter to convergence, freezes its parameters, and then triggers the training of the next one. This train-and-freeze process continues until all adapters are fully tuned. By focusing resources on one adapter at a time, CHAINFED drastically reduces peak memory usage, making federated fine-tuning feasible on resource-constrained devices. However, this novel paradigm poses several unique challenges, which we address through our core techniques.

• **Challenge 1: Representational Mismatch and Information Flow Bottleneck.** The sequential nature of chain optimization inherently risks representational mismatch. As each adapter is optimized in isolation, it focuses solely on local objectives, neglecting the context of adjacent layers and creating semantic gaps. Furthermore, the train-

[*]Corresponding authors.
[†]Equal contribution.



*Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18419–18435

and-freeze cycle impedes information flow: forward feature propagation is delayed until predecessor convergence, while backward gradient flow is confined to the active layer, preventing cross-layer co-adaptation. To address these, we propose Dynamic Layer Co-Tuning, which coordinates adjacent adapters to align feature spaces and facilitate inter-layer collaboration.

• **Challenge 2: Short-Sighted Optimization Perspective.** Sequential training also suffers from optimization myopia, as adapters are updated without error feedback from downstream layers. This exclusively local focus incentivizes over-specialization, causing the premature discard of generalizable information. Consequently, subsequent layers receive an impoverished feature space, degrading overall performance. To counteract this, we propose Globally Perceptive Optimization, which incorporates the model's holistic objective into each adapter's local loss. This mechanism forces adapters to balance local optimization with global utility, preserving critical information for synergistic, hierarchical learning.

• **Challenge 3: Optimal Fine-Tuning Boundary Identification.** The hierarchical structure of LLMs, transitioning from low-level syntax to high-level semantics, raises a critical question: *At which layer should the fine-tuning chain commence?* Initiating the chain too early is computationally wasteful, while starting too late risks insufficient adaptation. We resolve this dilemma with Function-Oriented Adaptive Tuning, which automatically identifies the optimal boundary by quantifying each layer's task contribution. By pinpointing the precise entry point, our method balances efficiency and effectiveness, retaining foundational knowledge while selectively tuning essential layers.

Our main contributions are summarized as follows: 1) We introduce CHAINFED, an innovative federated fine-tuning paradigm that breaks the memory wall for LLM adaptation via chain optimization. 2) We identify and solve three core challenges in this paradigm with a suite of synergistic techniques. 3) Our extensive experiments show that CHAINFED significantly outperforms existing methods by a large margin.

## 2 Related Work

### 2.1 Adapter-Based Federated Fine-Tuning

Adapters (He et al., 2021) have become a cornerstone of federated fine-tuning, with a significant body of work leveraging them to tackle key challenges such as data heterogeneity and slow convergence (Zhang et al.; Yao et al., 2024). For instance, to address data heterogeneity, C2A (Kim et al., 2023) generates personalized adapters via hypernetworks, while Fed-MNMT (Liu et al., 2023) clusters devices to prevent negative transfer. Other approaches, such as FedAdapter (Cai et al., 2023), focus on accelerating convergence through dynamic configurations. Despite these advancements, a fundamental limitation persists: the memory bottleneck. These methods require loading the entire model into memory, resulting in a prohibitive memory footprint. For instance, fine-tuning even the compact LLaMA3.1-3B (Dubey et al., 2024) demands approximately 14GB of memory, exceeding the capacity of most mobile devices and rendering larger models (e.g., LLaMA2-7B) inaccessible. To address this memory wall, we propose CHAINFED, a chain optimization paradigm that minimizes peak memory usage, thereby enabling participation from resource-constrained edge devices.

### 2.2 Memory-Aware Federated Fine-Tuning

Several approaches aim to mitigate memory costs during local training, primarily by reducing intermediate activations (e.g., FwdLLM (Xu et al., 2023) and FedKSeed (Qin et al., 2023) using zeroth-order optimization) or minimizing trainable parameters (e.g., FLoRA (Wang et al., 2024) via rank reduction). However, these methods overlook the dominant memory consumer: model parameters. For instance, in LoRA-based fine-tuning of LLaMA2-7B, base parameters account for 92.8% of the total memory footprint, dwarfing the contributions of activations (7.2%) and LoRA modules (0.018%). Consequently, targeting these secondary components yields only marginal savings. In contrast, CHAINFED employs a chain optimization paradigm that directly addresses the primary bottleneck—the model parameters themselves.

## 3 Preliminary and Motivation

### 3.1 Basics of Adapter

The core principle of adapters is to keep pre-trained model parameters frozen while inserting lightweight, trainable modules into different parts of the model (Houlsby et al., 2019). As illustrated in Figure 1, an adapter module operates via a bottleneck structure. It first projects the input hidden state $h$ into a low-dimensional space with a down-

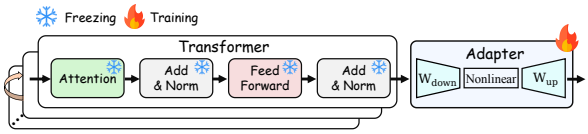
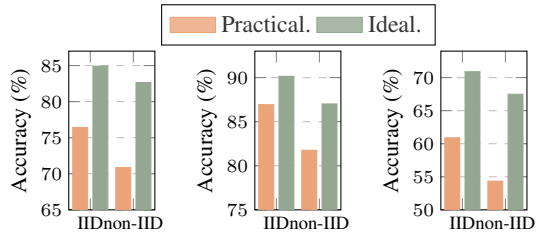



Figure 1: The configuration of the adapter.




(a) YELP-P. (b) AGNEWS. (c) YAHOO.

Figure 2: Performance comparison of BERT fine-tuning under practical versus idealized deployment conditions.

projection matrix ($W_{down} \in \mathbb{R}^{u \times v}$), applies a non-linear activation function $f(\cdot)$, and then restores it to the original dimension using an up-projection matrix ($W_{up} \in \mathbb{R}^{v \times u}$). The output is then added to the input via a residual connection, formulated as:

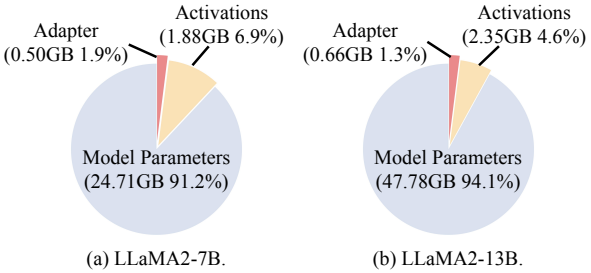
$$h \leftarrow h + f(hW_{down})W_{up}. \quad (1)$$

### 3.2 Memory Wall in Federated Fine-Tuning

This section motivates our work by quantifying the performance disparity between idealized and memory-constrained federated fine-tuning. We then analyze the memory footprint of adapter-based methods on representative LLMs, revealing critical insights that underpin the design of CHAINFED.

**Observation 1: Memory constraints significantly degrade model performance.** Figure 2 contrasts BERT's performance under idealized (full participation) versus practical (memory-constrained) scenarios. We observe severe performance degradation in the practical setting across both IID and non-IID distributions. On YELP-P (Zhang et al., 2015), for instance, accuracy drops by 8.5% (IID) and 11.8% (non-IID). This decline stems directly from the memory wall, which systematically excludes low-end devices from training. These results underscore that memory constraints are not merely a resource hurdle but a fundamental bottleneck to model performance.

**Observation 2: Adapter-based tuning fails to scale with modern LLMs.** We next investigate whether adapters are sufficient to overcome the memory wall. To verify this, we profile the memory usage of fine-tuning contemporary LLMs with adapters. The results in Figure 3 show that even with adapters, fine-tuning LLaMA2-7B still




(a) LLaMA2-7B. (b) LLaMA2-13B.

Figure 3: Memory usage breakdown during adapter-based fine-tuning. The "Adapter" component includes its parameters and gradients.

requires approximately 27.1GB of memory. This demand escalates sharply with model scale, reaching 50.8GB for LLaMA2-13B, a requirement far beyond the capacity of any consumer edge device. This confirms that standard adapter-based tuning, on its own, is an inadequate solution for bringing large-scale federated fine-tuning to the edge.

**Observation 3: Model parameters overwhelmingly dominate the memory footprint.** Our memory breakdown analysis identifies the base model parameters as the primary bottleneck. As shown in Figure 3, for LLaMA2-7B, parameters consume 91.2% of the total memory, dwarfing intermediate activations (6.9%) and adapter modules (1.9%). This dominance intensifies with scale, reaching 94.1% for LLaMA2-13B. This finding exposes a critical limitation in existing techniques that target activations or trainable parameters: since these components constitute a negligible fraction of the total footprint, their potential savings are inherently marginal. To truly overcome the memory bottleneck, an effective strategy must fundamentally reduce the number of model parameters residing in memory during fine-tuning.

## 4 Our Method: CHAINFED

### 4.1 The Chain Optimization Paradigm

Figure 4(b) illustrates the chain optimization paradigm. Unlike conventional methods that update all adapters end-to-end (Figure 4(a)), our approach decomposes the fine-tuning process into sequential stages, each dedicated to a specific layer and its adapter. This process begins by fine-tuning the first adapter, which is subsequently frozen upon convergence. The model then incorporates the second layer and its adapter, initiating the second training stage. This train-and-freeze process continues until all adapters are fully optimized, gradually expanding the model to its complete architecture.

In each stage, only the corresponding adapter

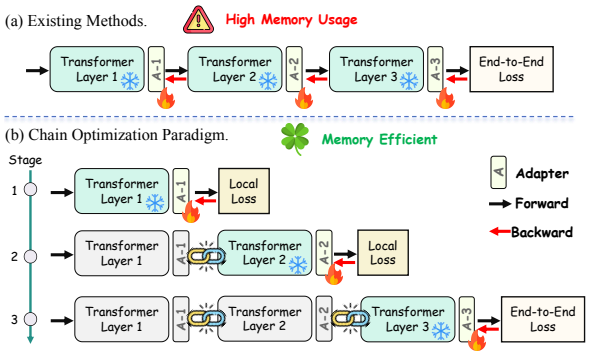



Figure 4: Comparison of existing methods with our chain optimization paradigm on a three-layer transformer model. The local loss is computed by attaching an output layer to each adapter, while the end-to-end loss is derived from the final output of the entire model.

undergoes parameter updates, eliminating the storage overhead for gradients and optimizer states of inactive layers. Specifically, preceding layers operate in inference mode, allowing immediate release of memory after the forward pass, while subsequent layers remain idle to prevent unnecessary allocation. However, this paradigm introduces three unique challenges, which we address below.

### 4.2 Dynamic Layer Co-Tuning

The sequential design of chain optimization inherently induces representational mismatch and bottlenecks bi-directional information flow. First, isolated training drives adapters to over-specialize for their local layers, neglecting adjacent context and causing semantic gaps. Second, the protocol disrupts information exchange: forward propagation is stalled until predecessor convergence, while backward gradients are confined to the active layer, precluding cross-layer co-adaptation.

To address these issues, we propose the Dynamic Layer Co-Tuning (DLCT) to foster inter-layer collaboration. Instead of fine-tuning adapters in isolation, DLCT creates a sliding window that moves sequentially along the chain. Adapters within this window are co-tuned simultaneously. The size of the sliding window is $Q$. When advancing to the next stage, the window then slides forward by one layer, creating an overlap of $Q-1$ adapters between consecutive stages. For instance, with $Q = 2$ (Figure 5), the first stage co-tunes adapters 1 and 2; subsequently, the window shifts to adapters 2 and 3. This overlap allows adapter 2 to be further refined alongside adapter 3, effectively bridging the optimization context between stages.

This co-tuning mechanism offers two key advantages: 1) **Bridging Semantic Gaps.** The inter-stage overlap directly mitigates representational mismatch. By training an adapter alongside both its upstream and downstream neighbors, it is incentivized to function as a semantic anchor, aligning feature representations across layers to ensure coherence. 2) **Breaking Gradient Isolation.** DLCT enables gradient information to freely propagate across adapters using its sliding window. As shown in Figure 5, the shared adapter (e.g., Adapter 2) acts as a conduit: it receives gradients from Adapter 3 in the second stage and influences Adapter 1 from the first, effectively linking non-adjacent layers. Furthermore, to enhance information flow, the window advances continuously in each round rather than waiting for stage-wise convergence. The process cycles iteratively, allowing for multiple passes of holistic refinement across the model.

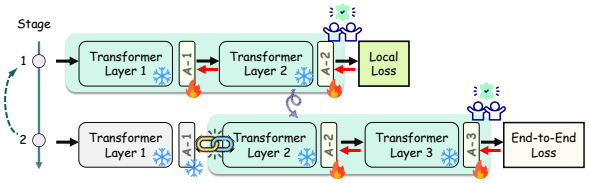



Figure 5: Overview of the Dynamic Layer Co-Tuning mechanism, illustrated with an example where the sliding window size is two ($Q = 2$).

### 4.3 Globally Perceptive Optimization

While DLCT ensures representational coherence and facilitates cross-layer information flow, the optimization perspective remains inherently myopic. Lacking feedback from downstream layers, each adapter greedily maximizes its immediate local objective. This short-sighted behavior induces the premature loss of valuable information, resulting in an impoverished feature space for subsequent layers and ultimately degrading overall performance.

To endow adapters with optimization foresight, we introduce Globally Perceptive Optimization (GPO). This strategy integrates the model's holistic objective into local updates, compelling adapters to align with global goals. To achieve this, we design an auxiliary output branch to compute the global loss. A naive approach would pass the current hidden state through all subsequent layers; however, this would incur substantial computation overhead and memory consumption. To address this, we propose a lightweight alternative: our auxiliary branch comprises only the subsequent adapters and the final output layer. This design leverages adapters as compact, low-rank approximations of layer trans-

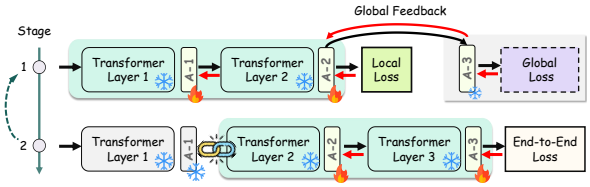



Figure 6: The Globally Perceptive Optimization strategy. An auxiliary output branch, composed only of subsequent adapters and the final output layer, is used to compute a global loss signal.

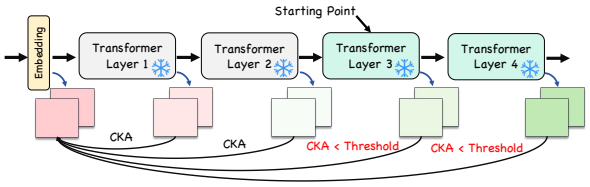



Figure 7: The Function-Oriented Adaptive Tuning scheme. Layer functionality is analyzed using CKA.

formations, enabling accurate estimation of the end-to-end loss with minimal overhead.

As illustrated in Figure 6, the optimization at each stage is guided by a dual-loss signal. For example, when co-tuning adapters 1 and 2, the training objective combines the local loss from adapter 2's output with the global loss computed via the lightweight auxiliary branch. Except for the final stage (which uses only the end-to-end loss), the objective at each stage $m$ is formulated as:

$$\text{Loss}_m = \text{Local Loss} + \lambda \cdot \text{Global Loss}, \quad (2)$$

where $\lambda$ is a hyperparameter balancing local and global objectives. By incentivizing adapters to learn features that are locally optimal yet globally beneficial, GPO promotes a synergistic and foresight-driven optimization process.

### 4.4 Function-Oriented Adaptive Tuning

While DLCT and GPO optimize training dynamics, the inherent functional hierarchy of LLMs, transitioning from shallow syntax to deep semantics, poses a strategic question: *At which layer should the fine-tuning chain commence?* Indiscriminate full-chain tuning is computationally redundant and risks degrading general-purpose representations. Conversely, selectively tuning task-critical layers enhances both efficiency and performance. However, pinpointing the optimal boundary between general and task-specific knowledge is non-trivial, particularly given the data heterogeneity.

To address this, we propose Function-Oriented Adaptive Tuning (FOAT), a data-driven scheme that automatically pinpoints the optimal fine-tuning entry point. Our approach is grounded in the insight that the functional role of a layer correlates with its feature transformation intensity. Layers exhibiting minimal deviation from the initial input are considered general-purpose and kept frozen. Conversely, layers that induce significant feature divergence are deemed task-specific, marking them as prime candidates for adaptation.

To quantify feature transformation, FOAT employs Centered Kernel Alignment (CKA) (Kornblith et al., 2019a) to assess the similarity between layer-wise representations and the initial input. As illustrated in Figure 7, this process initializes before federated training. Each participating device performs a single forward pass with the global model on local data to extract activations, computing the CKA similarity scores as follows:

$$\text{CKA}(Z_i, Z_j) = \frac{\text{HSIC}(Z_i, Z_j)}{\sqrt{\text{HSIC}(Z_i, Z_i) \cdot \text{HSIC}(Z_j, Z_j)}}, \quad (3)$$

where $Z_i$ and $Z_j$ denote layer activations and HSIC is the Hilbert-Schmidt Independence Criterion (Gretton et al., 2005) (detailed in Appendix A). Following local computation, devices upload CKA scores to the server for aggregation, revealing the global feature evolution from general to task-specific. The server then identifies the optimal starting layer, $L_{start}$, as the first layer where the aggregated CKA value falls below a predefined threshold $T$. This data-driven, inference-only strategy is inherently robust to non-IID distributions and ensures efficiency by preserving foundational knowledge while targeting task-critical layers. The complete workflow and convergence analysis are provided in Appendices B and C, respectively.

## 5 Experiments

### 5.1 Experimental Setup

**Models and Datasets.** We evaluate ChainFed on both text classification and instruction tuning tasks. For text classification, we employ three models of varying complexity: DistilBERT-base (Sanh et al., 2020), BERT-base (Devlin et al., 2019), and RoBERTa-large (Liu et al., 2019). These are benchmarked on four datasets: YELP-P (binary sentiment) (Zhang et al., 2015), AGNEWS (4-class news) (Zhang et al., 2015), YAHOO (10-class topic) (Zhang et al., 2015), and 20NEWS (20-class news) (Lang, 1995). For instruction tuning, we fine-tune LLaMA2-7B (Touvron et al.,

2023) and LLaMA3.1-8B (Dubey et al., 2024) using the Alpaca-GPT4 dataset (Peng et al., 2023). Evaluation covers a diverse suite of benchmarks: MMLU (Hendrycks et al., 2020) for knowledge understanding, BBH (Suzgun et al., 2022) and DROP (Dua et al., 2019) for complex reasoning, and CRASS (Frohberg and Binder, 2022) for counterfactual reasoning (Ye et al., 2024).

**Data Distribution and Scale.** We evaluate model performance under both IID and non-IID settings, generating non-IID partitions via a Dirichlet distribution with $\alpha = 1$ (Zhang et al., 2026, 2025). To assess system scalability, we vary the total client population across datasets: 20 for Alpaca-GPT4, 100 for 20NEWS, 1,000 for AGNEWS/YELP-P, and 10,000 for YAHOO. Further implementation details and metrics are provided in Appendix D.

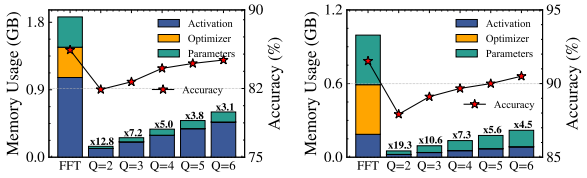
### 5.2 Baselines

We evaluate CHAINFED against a comprehensive suite of baselines, including a performance lower bound (No-FT$^\dagger$, denoting the pre-trained model) and an idealized upper bound (Full Adapters$^\dagger$, representing memory-unconstrained end-to-end training). We further compare against two categories of state-of-the-art approaches: 1) Memory-Unaware Methods: Linear Probing (Kornblith et al., 2019b), FedAdapter (Cai et al., 2022), and C2A (Kim et al., 2023); and 2) Memory-Aware Methods: FwdLLM (Xu et al., 2023), FedKSeed (Qin et al., 2023), FLoRA (Wang et al., 2024), and FedRA (Su et al., 2024). For detailed descriptions of these methods, please refer to Appendix E.

### 5.3 Overall Evaluation

Table 1 summarizes the main results, showing that CHAINFED consistently outperforms baselines across all experimental settings, achieving average accuracy improvements of up to 46.46%.

• **Comparison with Memory-Unaware Methods.** The fundamental limitation of these methods is their inability to scale. As model complexity increases (DistilBERT → RoBERTa), performance degrades sharply due to the memory wall. This barrier excludes resource-constrained devices, causing a catastrophic loss of data diversity that compromises the global model. Moreover, FedAdapter and C2A collapse on RoBERTa, failing to run on most datasets due to memory constraints. Even on smaller models where they are functional, their performance lags significantly behind CHAINFED; on BERT, they are outperformed by 9.56% and 11.74%, respectively. Furthermore, the consistently poor performance of Linear Probing confirms that merely fine-tuning the output layer is insufficient for effective task adaptation.




(a) YELP-P (non-IID). (b) AGNEWS (non-IID).

Figure 8: Impact of the co-tuning window size ($Q$) on model performance and memory usage for BERT.

• **Comparison with Memory-Aware Methods.** FwdLLM and FedKSeed reduce activation memory but ignore the dominant parameter bottleneck. Furthermore, their reliance on imprecise gradient estimation severely hurts performance. Compared to CHAINFED, FwdLLM suffers average drops of 9.68%, 23.89%, and 27.73% on DistilBERT, BERT, and RoBERTa, respectively, and fails to converge on 20NEWS. Similarly, FedKSeed lags by 7.44%–11.31%. FLoRA exhibits degradation patterns similar to memory-unaware methods; since LoRA modules constitute a negligible fraction of memory, reducing their rank offers minimal relief. Although FedRA addresses memory limitations, its random allocation of parameter update tasks leads to synchronization issues, resulting in a significant average performance degradation of up to 10.81% on RoBERTa compared to CHAINFED.

• **Comparison with Upper Bound.** CHAINFED even outperforms Full Adapters$^\dagger$ across all evaluation settings, achieving average performance gains of 2.86%, 2.46%, and 1.61% on DistilBERT, BERT, and RoBERTa, respectively, while reducing peak memory usage by up to 16.87× on RoBERTa. This superior performance is attributed to its efficient parameter updating strategy, which selectively fine-tunes task-critical layers while minimizing interference with other layers, thereby enhancing task adaptation and knowledge retention.

### 5.4 Analysis of the Co-Tuning Window Size

We now analyze the impact of the co-tuning window size ($Q$), which controls the trade-off between model performance and memory usage. We conduct this analysis on BERT, varying $Q$ from 2 to 6. As illustrated in Figure 8, increasing $Q$ yields consistent performance gains but proportionally raises peak memory consumption. This reflects a

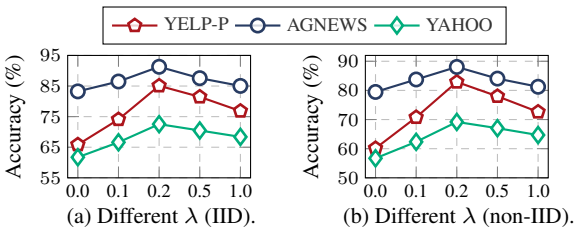
| | Method | YELP-P | | AGNEWS | | YAHOO | | 20NEWS | | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| | | IID | non-IID | IID | non-IID | IID | non-IID | IID | non-IID | |
| **DistilBERT-base** | | | | | | | | | | |
| **Lower Bound** | No-FT† | 50.04 | 50.04 | 25.13 | 25.13 | 10.05 | 10.05 | 5.01 | 5.01 | / |
| **Memory Unaware** | Linear Probing (Kornblith et al., 2019b) | 71.56 | 67.53 | 85.76 | 82.84 | 59.22 | 57.13 | 73.74 | 69.83 | 70.95 (↓ 11.50) |
| | FedAdapter (Cai et al., 2022) | 82.16 | 78.57 | 89.00 | 84.41 | 68.07 | 64.82 | 76.29 | 72.89 | 77.03 (↓ 5.42) |
| | C2A (Kim et al., 2023) | 80.24 | 76.95 | 87.49 | 82.88 | 66.24 | 61.87 | 74.65 | 70.47 | 75.10 (↓ 7.35) |
| **Memory Aware** | FwdLLM (Xu et al., 2023) | 82.00 | 77.61 | 88.79 | 83.03 | 65.77 | 61.93 | 63.70 | 59.34 | 72.77 (↓ 9.68) |
| | FedKSeed (Qin et al., 2023) | 81.91 | 77.42 | 88.51 | 82.86 | 66.92 | 63.11 | 71.49 | 67.83 | 75.01 (↓ 7.44) |
| | FLoRA (Wang et al., 2024) | 82.07 | 77.52 | 88.65 | 82.93 | 67.00 | 63.12 | 72.00 | 67.91 | 75.15 (↓ 7.30) |
| | FedRA (Su et al., 2024) | 81.55 | 77.91 | 88.92 | 84.25 | 68.36 | 65.03 | 76.66 | 72.71 | 76.92 (↓ 5.53) |
| | ChainFed | **86.57** | **84.22** | **92.57** | **90.18** | **73.78** | **70.39** | **82.29** | **79.63** | **82.45** |
| **Upper Bound** | Full Adapters† | 84.76 | 82.58 | 90.05 | 86.96 | 70.85 | 67.42 | 78.66 | 75.45 | 79.59 (↓ 2.86) |
| **BERT-base** | | | | | | | | | | |
| **Lower Bound** | No-FT† | 49.87 | 49.87 | 24.99 | 24.99 | 9.94 | 9.94 | 5.05 | 5.05 | / |
| **Memory Unaware** | Linear Probing (Kornblith et al., 2019b) | 69.09 | 64.96 | 79.88 | 71.55 | 55.23 | 52.82 | 67.62 | 60.97 | 65.27 (↓ 16.85) |
| | FedAdapter (Cai et al., 2022) | 77.84 | 72.59 | 88.41 | 84.03 | 62.67 | 59.74 | 69.42 | 65.76 | 72.56 (↓ 9.56) |
| | C2A (Kim et al., 2023) | 76.39 | 71.16 | 86.81 | 82.11 | 60.78 | 55.73 | 68.14 | 61.92 | 70.38 (↓ 11.74) |
| **Memory Aware** | FwdLLM (Xu et al., 2023) | 81.86 | 77.38 | 87.47 | 80.68 | 66.05 | 62.16 | 5.21* | 5.01* | 58.23 (↓ 23.89) |
| | FedKSeed (Qin et al., 2023) | 82.13 | 77.61 | 87.81 | 82.23 | 67.05 | 63.21 | 71.92 | 68.01 | 75.00 (↓ 7.12) |
| | FLoRA (Wang et al., 2024) | 82.19 | 77.75 | 87.98 | 82.51 | 67.12 | 63.24 | 72.03 | 68.15 | 75.12 (↓ 7.00) |
| | FedRA (Su et al., 2024) | 82.26 | 78.67 | 88.04 | 83.07 | 68.97 | 65.58 | 77.45 | 73.47 | 77.19 (↓ 4.93) |
| | ChainFed | **86.40** | **84.05** | **91.89** | **89.67** | **73.65** | **70.17** | **82.21** | **78.94** | **82.12** |
| **Upper Bound** | Full Adapters† | 84.92 | 82.65 | 90.14 | 87.01 | 70.89 | 67.48 | 78.69 | 75.46 | 79.66 (↓ 2.46) |
| **RoBERTa-large** | | | | | | | | | | |
| **Lower Bound** | No-FT† | 49.95 | 49.95 | 25.24 | 25.24 | 9.98 | 9.98 | 5.02 | 5.02 | / |
| **Memory Unaware** | Linear Probing (Kornblith et al., 2019b) | 66.45 | 62.24 | 69.88 | 62.67 | 50.35 | 48.03 | 60.93 | 51.64 | 59.02 (↓ 22.29) |
| | FedAdapter (Cai et al., 2022) | — | — | 78.59 | 73.87 | — | — | — | — | 35.30 (↓ 46.01) |
| | C2A (Kim et al., 2023) | — | — | 76.93 | 71.98 | — | — | — | — | 34.85 (↓ 46.46) |
| **Memory Aware** | FwdLLM (Xu et al., 2023) | 76.89 | 72.46 | 80.41 | 74.52 | 58.97 | 55.24 | 5.13* | 5.05* | 53.58 (↓ 27.73) |
| | FedKSeed (Qin et al., 2023) | 78.19 | 73.76 | 80.63 | 79.85 | 60.12 | 56.75 | 67.80 | 62.93 | 70.00 (↓ 11.31) |
| | FLoRA (Wang et al., 2024) | — | — | 80.73 | 79.88 | — | — | — | — | 36.31 (↓ 45.00) |
| | FedRA (Su et al., 2024) | 77.45 | 73.71 | 81.56 | 76.63 | 61.69 | 58.37 | 69.27 | 65.31 | 70.50 (↓ 10.81) |
| | ChainFed | **86.47** | **83.08** | **91.96** | **89.52** | **72.77** | **69.24** | **81.12** | **76.32** | **81.31** |
| **Upper Bound** | Full Adapters† | 84.93 | 82.71 | 90.13 | 87.06 | 70.94 | 67.53 | 78.76 | 75.57 | 79.70 (↓ 1.61) |

Table 1: Performance comparison of ChainFed versus baselines on text classification tasks. **Bold** and underlined values indicate the best and second-best results, respectively. "−" denotes methods that failed due to memory constraints (defaulting to No-FT† performance), while "∗" indicates non-convergence.

fundamental trade-off: a larger window facilitates stronger parameter co-adaptation across more layers, but at the cost of loading more components into memory simultaneously. In practice, we set $Q$ based on the device with the lowest capacity to ensure inclusive participation. Notably, with $Q = 6$, ChainFed matches the performance of full-parameter fine-tuning (FFT), while reducing peak memory usage by up to 4.5×, highlighting the effectiveness of the co-tuning mechanism.

## 5.5 Analysis of the Global Loss Weight

In this section, we explore the impact of the global loss weight ($\lambda$), which balances the local and global learning objectives. We conduct experiments on DistilBERT, varying $\lambda$ across the set {0.0,0.1,0.2,0.5,1.0}. Figure 9 shows that setting $\lambda = 0$ yields the lowest performance, confirming that purely local optimization induces myopic learning. Conversely, integrating the global signal



(a) Different $\lambda$ (IID). (b) Different $\lambda$ (non-IID).

Figure 9: Impact of the global loss weight on model performance for DistilBERT.

boosts accuracy by up to 22.76%. However, an excessively high $\lambda$ (e.g., 1.0) degrades performance by 10.27% (compared to $\lambda = 0.2$), as the global objective overshadows layer-specific feature learning. Therefore, we recommend setting $\lambda = 0.1$ for BERT and 0.2 for other models in practice.

## 5.6 Analysis of the Fine-Tuning Threshold

We then analyze the impact of the fine-tuning threshold ($T$), which determines the starting layer for our chain optimization. Our experiments vary

| Threshold | IID | non-IID | Speedup | Comm. Reduction |
|---|---|---|---|---|
| **DistilBERT-base (YELP-P)** | | | | |
| Full Adapters† | 84.76 | 82.58 | ×1 | ×1 |
| $T = 1.0$ | 82.76 | 80.03 | ×1.72 | ×1.66 |
| $T = 0.9$ | 84.43 | 82.37 | ×1.86 | ×2.49 |
| $T = 0.8$ | **86.53** | **84.13** | ×**1.97** | ×**2.75** |
| **DistilBERT-base (AGNEWS)** | | | | |
| Full Adapters† | 90.05 | 86.96 | ×1 | ×1 |
| $T = 1.0$ | 88.45 | 85.82 | ×1.78 | ×2.66 |
| $T = 0.9$ | 90.24 | 87.86 | ×1.94 | ×3.13 |
| $T = 0.8$ | **92.59** | **90.25** | ×**2.02** | ×**3.47** |

Table 2: Impact of the fine-tuning threshold $T$ on model performance. In this set of experiments, we set $Q = 3$.

| Method | MMLU | BBH | DROP | CRASS | Average | Mem. Reduction |
|---|---|---|---|---|---|---|
| **LLaMA2-7B** | | | | | | |
| Full Adapters† | 43.68 | 31.72 | 32.19 | 44.95 | 38.14 | ×1 |
| $Q = 6$ | 45.17 | 32.36 | 33.40 | 50.63 | 40.39 | ×**4.29** |
| $Q = 7$ | 46.04 | 33.19 | 34.05 | 52.87 | 41.54 | ×3.69 |
| $Q = 8$ | **46.76** | **33.65** | **34.79** | **55.34** | **42.64** | ×3.23 |
| **LLaMA3.1-8B** | | | | | | |
| Full Adapters† | 61.15 | 61.28 | 55.72 | 74.63 | 63.20 | ×1 |
| $Q = 6$ | 65.46 | 65.87 | 60.36 | 85.48 | 69.29 | ×**4.60** |
| $Q = 7$ | 67.19 | 68.60 | 62.68 | 90.07 | 72.14 | ×3.94 |
| $Q = 8$ | **68.01** | **70.56** | **63.42** | **93.64** | **73.91** | ×3.45 |

Table 3: Performance of CHAINFED on instruction tuning tasks with varying values of $Q$.

$T$ across the set {1.0,0.9,0.8}, with $T = 1.0$ representing the baseline case of fine-tuning all layers. The results in Table 2 reveal a crucial insight: fine-tuning all layers is suboptimal. Performance peaks at $T = 0.8$, surpassing the full-chain baseline ($T = 1.0$) by up to 4.1% on YELP-P and 4.43% on AGNEWS. This suggests that freezing general-purpose lower layers not only conserves resources but also enhances model generalization. Remarkably, CHAINFED ($T = 0.8$) outperforms even the idealized Full Adapters†, achieving 3.29% higher accuracy on AGNEWS with 2.02× faster convergence and 3.47× lower communication overhead. These results powerfully demonstrate the effectiveness of our Function-Oriented Adaptive Tuning scheme.

### 5.7 Performance on Instruction Tuning

We further evaluate CHAINFED on instruction tuning tasks using LLaMA2-7B and LLaMA3.1-8B, benchmarking against Full Adapters†. As detailed in Table 3, CHAINFED consistently outperforms the baseline while substantially reducing memory usage. For instance, on LLaMA2-7B, when $Q = 6$, CHAINFED yields improvements of 1.49%, 0.64%, 1.21%, and 5.68% on MMLU, BBH, DROP, and CRASS, respectively, while reducing peak memory usage by 4.29×. When $Q = 8$, it achieves a 4.50% average performance gain with a 3.23× memory reduction. These benefits are even more pronounced on LLaMA3.1-8B, where average accuracy improves by up to 10.71% alongside a 3.45× reduction in memory. This consistently superior performance, achieved with low memory usage, validates CHAINFED as a versatile and practical solution for on-device federated fine-tuning.

| Method | YELP-P | | AGNEWS | | Average |
|---|---|---|---|---|---|
| | IID | non-IID | IID | non-IID | |
| **DistilBERT-base** | | | | | |
| w/o DLCT | 73.63 | 69.04 | 83.89 | 80.45 | 76.75 (↓ 11.64) |
| w/o GPO | 70.12 | 63.98 | 83.98 | 80.57 | 74.66 (↓ 13.73) |
| w/o FOAT | 79.95 | 77.14 | 86.78 | 83.59 | 81.87 (↓ 6.52) |
| CHAINFED | **86.57** | **84.22** | **92.57** | **90.18** | **88.39** |
| **BERT-base** | | | | | |
| w/o DLCT | 72.83 | 68.27 | 83.34 | 80.13 | 76.14 (↓ 11.86) |
| w/o GPO | 69.27 | 63.86 | 84.21 | 80.58 | 74.48 (↓ 13.52) |
| w/o FOAT | 78.82 | 77.03 | 86.37 | 81.07 | 80.82 (↓ 7.18) |
| CHAINFED | **86.40** | **84.05** | **91.89** | **89.67** | **88.00** |
| **RoBERTa-large** | | | | | |
| w/o DLCT | 72.12 | 67.98 | 82.50 | 79.96 | 75.64 (↓ 12.12) |
| w/o GPO | 72.31 | 67.63 | 85.22 | 81.65 | 76.70 (↓ 11.06) |
| w/o FOAT | 80.21 | 77.73 | 87.39 | 83.82 | 82.29 (↓ 5.47) |
| CHAINFED | **86.47** | **83.08** | **91.96** | **89.52** | **87.76** |

Table 4: Ablation study of CHAINFED.

### 5.8 Ablation Study

Finally, we conduct an ablation study to quantify the individual contributions of CHAINFED's core components: DLCT, GPO, and FOAT. The results (Table 4) confirm that removing any technique precipitates a substantial performance drop, validating their collective necessity. For DistilBERT, eliminating DLCT, GPO, or FOAT reduces average accuracy by 11.64%, 13.73%, and 6.52%, respectively. This impact persists as model complexity scales: on RoBERTa, performance degrades by 12.12% (w/o DLCT), 11.06% (w/o GPO), and 5.47% (w/o FOAT). These findings underscore that each mechanism plays an indispensable role in ensuring the effectiveness of chain optimization.

## 6 Conclusion

In this paper, we propose CHAINFED, an innovative federated fine-tuning paradigm that effectively addresses the memory constraints of participating devices via chain optimization. CHAINFED further integrates three novel techniques: 1) Dynamic Layer Co-Tuning, 2) Globally Perceptive Optimization, and 3) Function-Oriented Adaptive Tuning. Extensive experiments on multiple benchmarks demonstrate that CHAINFED significantly outperforms state-of-the-art baselines in model performance and resource efficiency.

## Limitations

Despite the promising results and memory efficiency demonstrated by CHAINFED, we acknowledge several limitations that present avenues for future research. First, our evaluation is currently confined to natural language processing tasks using Transformer-based architectures (e.g., BERT and LLaMA series). While CHAINFED is theoretically applicable to other neural networks, its effectiveness on distinct architectures (such as SSMs or Mamba) and other modalities (e.g., Multimodal LLMs) remains to be empirically verified. Second, while CHAINFED inherently preserves data privacy by keeping raw data local, we have not yet integrated it with advanced privacy-enhancing technologies, such as Differential Privacy (DP) or Homomorphic Encryption. Exploring the trade-off between the noise introduced by DP and the precise feature alignment required by our layer-wise tuning remains a valuable direction for future work.

## Ethical Considerations

Our research proposes CHAINFED to enable privacy-preserving LLM fine-tuning on edge devices. This work aligns with ethical guidelines by promoting data privacy; our method ensures that sensitive user data remains on local devices, addressing key privacy concerns in centralized AI training. Furthermore, by drastically reducing memory requirements, CHAINFED promotes sustainability and inclusivity, enabling the deployment of LLMs on consumer-grade hardware rather than relying solely on energy-intensive data centers. All datasets and models used in this study are public benchmarks and pre-trained models (e.g., BERT, RoBERTa, LLaMA2, LLaMA3.1) that are utilized strictly in accordance with their intended research purposes and licensing terms. We strictly adhere to all applicable licenses and usage policies. All evaluation benchmarks are employed exclusively for assessing model capabilities, and all pre-trained models are used solely for research evaluation purposes consistent with their intended use cases.

We acknowledge the general risks associated with LLM generation (e.g., hallucination or toxicity) and emphasize that standard safety alignment procedures should be applied alongside our optimization framework when deploying in production environments.


## Acknowledgements

This work is supported in part by the Science and Technology Development Fund of Macau (0107/2024/RIA2, 0061/2025/RIB2), Joint Science and Technology Research Project with Hong Kong and Macau in Key Areas of Nansha District's Science and Technology Plan (EF2024-00180-IOTSC) and the Multi-Year Research Grant of University of Macau (MYRG-GRG2023-00211-IOTSC-UMDF, MYRG-GRG2024-00180-IOTSC).


## References


Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Jieming Bian, Yuanzhe Peng, Lei Wang, Yin Huang, and Jie Xu. 2025. A survey on parameter-efficient fine-tuning for foundation models in federated learning. *arXiv preprint arXiv:2504.21099*.

Dongqi Cai, Yaozong Wu, Shangguang Wang, Felix Xiaozhu Lin, and Mengwei Xu. 2022. Fedadapter: Efficient federated learning for modern nlp. *arXiv preprint arXiv:2205.10162*.

Dongqi Cai, Yaozong Wu, Shangguang Wang, and Mengwei Xu. 2023. Fedadapter: Efficient federated learning for mobile nlp. In *Proceedings of the ACM Turing Award Celebration Conference-China 2023*, pages 27–28.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jörg Frohberg and Frank Binder. 2022. Crass: A novel data set and benchmark to test counterfactual reasoning of large language models. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2126–2140.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. 2005. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer.

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. *arXiv preprint arXiv:2106.03164*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

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.

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

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019a. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR.

Simon Kornblith, Jonathon Shlens, and Quoc V Le. 2019b. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2661–2671.

Ken Lang. 1995. Newsweeder: learning to filter netnews. In *Proceedings of the Twelfth International Conference on International Conference on Machine Learning*, ICML'95, page 331–339, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Heju Li, Rui Wang, Wei Zhang, and Jun Wu. 2022. One bit aggregation for federated edge learning with reconfigurable intelligent surface: Analysis and optimization. *IEEE Transactions on Wireless Communications*, 22(2):872–888.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Yi Liu, Xiaohan Bi, Lei Li, Sishuo Chen, Wenkai Yang, and Xu Sun. 2023. Communication efficient federated learning for multilingual neural machine translation with adapter. *arXiv preprint arXiv:2305.12449*.

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

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

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.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Zhen Qin, Daoyuan Chen, Bingchen Qian, Bolin Ding, Yaliang Li, and Shuiguang Deng. 2023. Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes. *arXiv preprint arXiv:2312.06353*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *Preprint*, arXiv:1910.01108.

Shangchao Su, Bin Li, and Xiangyang Xue. 2024. Fedra: A random allocation strategy for federated tuning to unleash the power of heterogeneous clients. *Preprint*, arXiv:2311.11227.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Kahou Tam, Li Li, Bo Han, Chengzhong Xu, and Huazhu Fu. 2023. Federated noisy client learning. *IEEE transactions on neural networks and learning systems*, 36(1):1799–1812.

Kahou Tam, Chunlin Tian, Li Li, Haikai Zhao, and ChengZhong Xu. 2024. Fedhybrid: Breaking the memory wall of federated learning via hybrid tensor management. In *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, pages 394–408.

Chunlin Tian, Li Li, Zhan Shi, Jun Wang, and ChengZhong Xu. 2022. Harmony: Heterogeneity-aware hierarchical management for federated learning system. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 631–645. IEEE.

Chunlin Tian, Li Li, Kahou Tam, Yebo Wu, and ChengZhong Xu. 2024. Breaking the memory wall for heterogeneous federated learning via model splitting. *IEEE Transactions on Parallel and Distributed Systems*, 35(12):2513–2526.

Chunlin Tian, Kahou Tam, Yebo Wu, Shuaihang Zhong, Li Li, Nicholas D Lane, and ChengZhong Xu. 2026. Floe: Federated specialization for real-time llm–slm inference. *IEEE Transactions on Parallel and Distributed Systems*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Hui-Po Wang, Sebastian Stich, Yang He, and Mario Fritz. 2022. Progfed: effective, communication, and computation efficient federated learning by progressive training. In *International Conference on Machine Learning*, pages 23034–23054. PMLR.

Jie Wang, Xiaolong Wu, Jindong Tian, Erwu Liu, Yebo Wu, Rucong Lai, and Yong Tian. 2025. Indoor localization fusing inertial navigation with monocular depth estimation in federated learning framework with data heterogeneity. *IEEE Transactions on Instrumentation and Measurement*.

Jie Wang, Yebo Wu, Erwu Liu, Xiaolong Wu, Xinyu Qu, Yuanzhe Geng, and Hanfu Zhang. 2023. Fedins2: A federated-edge-learning-based inertial navigation system with segment fusion. *IEEE Internet of Things Journal*.

Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. 2024. Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations. *arXiv preprint arXiv:2409.05976*.

T Wolf. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yebo Wu, Jingguang Li, Zhijiang Guo, and Li Li. Developmental federated tuning: A cognitive-inspired paradigm for efficient llm adaptation. In *The Fourteenth International Conference on Learning Representations*.

Yebo Wu, Jingguang Li, Zhijiang Guo, and Li Li. 2025a. Elastic mixture of rank-wise experts for knowledge reuse in federated fine-tuning. *arXiv preprint arXiv:2512.00902*.

Yebo Wu, Jingguang Li, Chunlin Tian, Zhijiang Guo, and Li Li. 2025b. Memory-efficient federated fine-tuning of large language models via layer pruning. *arXiv preprint arXiv:2508.17209*.

Yebo Wu, Jingguang Li, Chunlin Tian, Kahou Tam, Li Li, and Chengzhong Xu. 2024a. Bridging memory gaps: Scaling federated learning for heterogeneous clients. *arXiv preprint arXiv:2408.10826*.

Yebo Wu, Li Li, Chunlin Tian, Tao Chang, Chi Lin, Cong Wang, and Cheng-Zhong Xu. 2024b. Heterogeneity-aware memory efficient federated learning via progressive layer freezing. In *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE.

Yebo Wu, Li Li, and Cheng-zhong Xu. 2025c. Breaking the memory wall for heterogeneous federated learning via progressive training. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 1623–1632.

Yebo Wu, Feng Liu, Ziwei Xie, Zhiyuan Liu, Changwang Zhang, Jun Wang, and Li Li. 2026. Tsembed: Unlocking task scaling in universal multimodal embeddings. *arXiv preprint arXiv:2603.04772*.

Yebo Wu, Chunlin Tian, Jingguang Li, He Sun, Kahou Tam, Li Li, and Chengzhong Xu. 2025d. A survey on federated fine-tuning of large language models. *arXiv preprint arXiv:2503.12016*.

Mengwei Xu, Dongqi Cai, Yaozong Wu, Xiang Li, and Shangguang Wang. 2023. Fwdllm: Efficient fedllm using forward gradient. *arXiv preprint arXiv:2308.13894*.

Naen Xu, Hengyu An, Shuo Shi, Jinghuai Zhang, Chunyi Zhou, Changjiang Li, Tianyu Du, Zhihui Fu, Jun Wang, and Shouling Ji. 2026a. When agents" misremember" collectively: Exploring the mandela effect in llm-based multi-agent systems. *arXiv preprint arXiv:2602.00428*.

Naen Xu, Changjiang Li, Tianyu Du, Minxi Li, Wenjie Luo, Jiacheng Liang, Yuyuan Li, Xuhong Zhang, Meng Han, Jianwei Yin, et al. 2024. Copyrightmeter: Revisiting copyright protection in text-to-image models. *arXiv preprint arXiv:2411.13144*.

Naen Xu, Jiayi Sheng, Changjiang Li, Chunyi Zhou, Yuyuan Li, Tianyu Du, Jun Wang, Zhihui Fu, Jinbao Li, and Shouling Ji. 2026b. "i see what you did there": Can large vision-language models understand multimodal puns? *Preprint*, arXiv:2604.05930.

Naen Xu, Jinghuai Zhang, Ping He, Chunyi Zhou, Jun Wang, Zhihui Fu, Tianyu Du, Zhaoxiang Wang, and Shouling Ji. 2026c. Fraudshield: Knowledge graph empowered defense for llms against fraud attacks. *arXiv preprint arXiv:2601.22485*.

Naen Xu, Jinghuai Zhang, Changjiang Li, Hengyu An, Chunyi Zhou, Jun Wang, Boyu Xu, Yuyuan Li, Tianyu Du, and Shouling Ji. 2026d. Bridging the copyright gap: Do large vision-language models recognize and respect copyrighted content? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 35949–35957.

Naen Xu, Jinghuai Zhang, Changjiang Li, Zhi Chen, Chunyi Zhou, Qingming Li, Tianyu Du, and Shouling Ji. 2025. Videoeraser: Concept erasure in text-to-video diffusion models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5965–5994.

Yuhang Yao, Jianyi Zhang, Junda Wu, Chengkai Huang, Yu Xia, Tong Yu, Ruiyi Zhang, Sungchul Kim, Ryan Rossi, Ang Li, et al. 2024. Federated large language models: Current progress and future directions. *arXiv preprint arXiv:2409.15723*.

Rui Ye, Wenhao Wang, Jingyi Chai, Dihan Li, Zexi Li, Yinda Xu, Yaxin Du, Yanfeng Wang, and Siheng Chen. 2024. Openfedllm: Training large language models on decentralized private data via federated learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6137–6147.

Shichen Zhan, Yebo Wu, Chunlin Tian, Yan Zhao, and Li Li. 2024. Heterogeneity-aware coordination for federated learning via stitching pre-trained blocks. In *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE.

Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Tong Yu, Guoyin Wang, and Yiran Chen. 2024. Towards building the federatedgpt: Federated instruction tuning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6915–6919. IEEE.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Xiangtao Zhang, Eleftherios Kofidis, Ruituo Wu, Ce Zhu, Le Zhang, and Yipeng Liu. 2026. Coupled tensor train decomposition in federated learning. *Pattern Recognition*, 170:112067.

Xiangtao Zhang, Sheng Li, Ao Li, Yipeng Liu, Fan Zhang, Ce Zhu, and Le Zhang. 2025. Subspace constraint and contribution estimation for heterogeneous federated learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 20632–20642.

Zixin Zhang, Fan Qi, and Changsheng Xu. Enhancing storage and computational efficiency in federated multimodal learning for large-scale models. In *Forty-first International Conference on Machine Learning*.

## A The Definition of HSIC

The HSIC is computed as:

$$\begin{aligned}\text{HSIC}(Z_i, Z_j) = &\mathbb{E}_{Z_i,Z_i',Z_j,Z_j'}\left[k_{Z_i}(Z_i,Z_i')k_{Z_j}(Z_j,Z_j')\right] \\ &+ \mathbb{E}_{Z_i,Z_i'}\left[k_{Z_i}(Z_i,Z_i')\right]\mathbb{E}_{Z_j,Z_j'}\left[k_{Z_j}(Z_j,Z_j')\right] \\ &- 2\mathbb{E}_{Z_i,Z_j}\left[\mathbb{E}_{Z_i'}[k_{Z_i}(Z_i,Z_i')]\mathbb{E}_{Z_j'}[k_{Z_j}(Z_j,Z_j')]\right],\end{aligned} \tag{4}$$

where $k_{Z_i}$ and $k_{Z_j}$ denote kernel functions.

## B The Workflow of CHAINFED

Algorithm 1 presents the complete workflow of CHAINFED.

**Algorithm 1** The workflow of CHAINFED

**Require:** $N$ devices with memory budgets $\{\text{Mem}_n\}_{n=1}^N$; $R$ communication rounds; Hyperparameters $\lambda$ and $T$; Global model $\Theta$.

**Ensure:** Fine-tuned global model $\Theta^*$.

*Phase 1: Pre-Training Setup*

**All devices**: Compute local CKA scores on $\Theta$ via Eq. (3).

**Server**: Aggregates similarity scores and sets $L_{start}$ based on threshold $T$. ▷ FOAT

**Server**: Sets the co-tuning window size $Q$ based on the minimum device memory, $min(\{\text{Mem}_n\}_{n=1}^N)$. ▷ DLCT

*Phase 2: Iterative Federated Fine-Tuning*

**for** round $r = 1$ to $R$ **do**

  Server samples a subset of devices $\mathcal{S}_r$.

  Server sends relevant parameters to $\mathcal{S}_r$. ▷ DLCT

  **for all** device $n \in \mathcal{S}_r$ **in parallel do**

    Perform local training using Eq. (2). ▷ GPO

    Upload local adapter updates $\Delta\Theta_n^r$.

  **end for**

  Server aggregates local updates to refine the global model: $\Theta_{r+1} \leftarrow \Theta_r + \sum_{n\in\mathcal{S}_r} \frac{|D_n|}{\sum_{n\in\mathcal{S}_r}|D_n|}\Delta\Theta_n^r$.

**end for**

**return** $\Theta^* \leftarrow \Theta$.

## C Theoretical Analysis

In this section, we present a rigorous convergence analysis of CHAINFED in the context of adapter-based federated fine-tuning. Leveraging standard assumptions and established results from the federated optimization literature (Li et al., 2019; Wang et al., 2022), we show that CHAINFED converges to a stationary point, thus providing formal guarantees for its optimization dynamics.

### C.1 Preliminaries

In federated fine-tuning, the global optimization objective is defined as:

$$\min_{\Theta} F(\Theta) = \sum_{n=1}^{N} \frac{|D_n|}{|D|}\,\mathcal{L}_n(\Theta; D_n), \tag{5}$$

where $\Theta$ denotes the set of global model parameters, $D_n$ is the local dataset residing on device $n$ ($n \in [1, N]$), and $|D_n|$ and $|D|$ represent the sizes of the local and total datasets, respectively. The

function $\mathcal{L}_n(\Theta; D_n)$ denotes the local optimization objective on device $n$.

In CHAINFED, adapter modules are integrated into a pre-trained LLM and fine-tuned in a chain, layer-wise manner. The adaptation process at each layer adheres to the transformation defined in Equation (1). Furthermore, the objective in each stage combines a local loss and a globally aligned loss, computed through an auxiliary output branch, as formulated in Equation (2). This strategy facilitates stable layer-wise adaptation while ensuring alignment with the overall global optimization goal.

### C.2 Assumptions

For analytical purposes, we adopt a set of assumptions that follow established formulations in FL (Li et al., 2019), while extending them to accommodate the characteristics of adapter-based fine-tuning.

1. **Smoothness.** Each local objective function $\mathcal{L}_n(\Theta; D_n)$ is differentiable and has an $L$-Lipschitz continuous gradient, meaning there exists a constant $L > 0$ such that for any $\Theta_1, \Theta_2$, the following holds:

$$\|\nabla\mathcal{L}_n(\Theta_1; D_n) - \nabla\mathcal{L}_n(\Theta_2; D_n)\| \leq L\,\|\Theta_1 - \Theta_2\|. \tag{6}$$

   This condition guarantees that the gradient of the local objective varies smoothly with respect to the model parameters, enabling stable gradient-based optimization.

2. **Unbiased Gradients and Bounded Variance.** The stochastic gradient computed on each device provides an unbiased estimate of the true gradient with bounded variance. Formally, there exist constants $\sigma^2$ and $G > 0$ such that:

$$\mathbb{E}[\nabla\mathcal{L}_n(\Theta; D_n, \xi_n)] = \nabla\mathcal{L}_n(\Theta; D_n), \tag{7}$$

$$\mathbb{E}[\|\nabla\mathcal{L}_n(\Theta; D_n, \xi_n) - \nabla\mathcal{L}_n(\Theta; D_n)\|^2] \leq \sigma^2 \tag{8}$$

$$\|\nabla\mathcal{L}_n(\Theta; D_n)\| \leq G, \quad \forall n, \Theta. \tag{9}$$

   This assumption ensures that stochastic optimization does not introduce excessive variance, thereby promoting stable convergence during training.

3. **Bounded Heterogeneity.** Data heterogeneity across devices can lead to divergence in local gradients. We assume that the discrepancy between local gradients across devices is bounded. Specifically, there exists a constant $H > 0$ such that:

$$\mathbb{E}\left[\|\nabla\mathcal{L}_m(\Theta; D_m) - \nabla\mathcal{L}_n(\Theta; D_n)\|^2\right] \leq H^2, \ \forall m, n. \tag{10}$$

   This assumption captures the impact of non-IID data in federated settings. A smaller value of $H$ implies greater homogeneity among devices, which typically leads to faster and more stable convergence.

4. **Adapter Error Bound.** In CHAINFED, fine-tuning is performed on a subset of layers selected by the FOAT mechanism, which introduces an approximation error due to the exclusion of certain layers. We assume this error is bounded by a threshold-dependent constant $\epsilon_{\text{FOAT}} \geq 0$. Formally, there exists:

$$\|\nabla\mathcal{L}_n(\Theta_{\text{FOAT}}; D_n) - \nabla\mathcal{L}_n(\Theta_{\text{Full}}; D_n)\| \leq \epsilon_{\text{FOAT}}, \tag{11}$$

   where $\Theta_{\text{FOAT}}$ denotes the parameters updated in the selected layers, and $\Theta_{\text{Full}}$ corresponds to full-model fine-tuning. With appropriate layer selection, this error can be made arbitrarily small, ensuring FOAT preserves optimization effectiveness while improving efficiency.

### C.3 Proof of Convergence

**One-Round Descent Lemma.** Let $\Theta_t$ denote the global model parameters at the $t$-th iteration, and define the aggregated gradient across participating devices as:

$$g_t = \sum_{n=1}^{N} \frac{|D_n|}{|D|} \nabla\mathcal{L}_n(\Theta_t; D_n). \tag{12}$$

In the analysis of FedAvg-based algorithms (Li et al., 2019, 2022; Tam et al., 2023; Tian et al., 2022), it is well established that under the smoothness assumption, selecting an appropriate learning rate $\eta_t$ ensures the following descent property:

$$F(\Theta_{t+1}) \leq F(\Theta_t) - \eta_t\,\|g_t\|^2 + \frac{L\eta_t^2}{2}\,\|g_t\|^2. \tag{13}$$

Thus, if $\eta_t$ is small enough, i.e., $\eta_t < \frac{2}{L}$, the term $\frac{L\eta_t^2}{2}\,\|g_t\|^2$ remains a higher-order error, ensuring a sufficient decrease in $F(\Theta)$ per iteration.

**Roles of Mechanisms in CHAINFED.** In contrast to FedAvg, CHAINFED incorporates two key mechanisms that reshape the optimization dynamics:

- **Globally Perceptive Optimization (GPO).** The GPO strategy customizes the update objective for each layer by incorporating both the standard local loss and an auxiliary global loss, as defined in Equation (2). Let $g_t^{\text{local}}$ denote the gradient from the local loss, and $g_t^{\text{global}}$ denote the gradient from the auxiliary global branch. The effective gradient becomes:

$$\tilde{g}_t = g_t^{\text{local}} + \lambda\, g_t^{\text{global}}. \tag{14}$$

  The global term $g_t^{\text{global}}$ serves as a regularization force, aligning local updates with the global objective and mitigating optimization drift across devices. With a properly chosen coefficient $\lambda$ and well-designed auxiliary branches, the bias introduced by using $\tilde{g}_t$ instead of the full end-to-end gradient is bounded by a constant $\epsilon_{\text{aux}} \geq 0$.

- **Dynamic Layer Co-Tuning (DLCT).** The DLCT mechanism ensures that the loss landscape observed by concurrently fine-tuned adapters is more coherent, effectively reducing inter-layer gradient isolation. Therefore, the effective descent relation becomes:

$$F(\Theta_{t+1}) \leq F(\Theta_t) - \eta_t \Big( \|\tilde{g}_t\|^2 - \epsilon_{\text{aux}} - \epsilon_{\text{FOAT}} \Big). \tag{15}$$

With appropriate hyperparameter tuning, $\epsilon_{\text{aux}}$ and $\epsilon_{\text{FOAT}}$ can be made arbitrarily small. As a result, there exists a learning rate schedule $\{\eta_t\}$ (or a sufficiently small constant $\eta$) such that the sequence $\{F(\Theta_t)\}$ is monotonically decreasing and lower-bounded, ensuring convergence.

**Theorem 1 (Multi-Round Convergence).** Assume that the global objective $F(\Theta_t)$ converges to a finite limit $F^*$ as $t \to \infty$. Then, from the descent inequality in Equation (15), it follows that:

$$\lim_{t\to\infty} \eta_t \left( \|\tilde{g}_t\|^2 - \epsilon_{\text{aux}} - \epsilon_{\text{FOAT}} \right) = 0. \tag{16}$$

Since the learning rate $\eta_t$ remains positive and does not decay too rapidly (as ensured by standard stochastic approximation conditions), we have:

$$\lim_{t\to\infty} \|\tilde{g}_t\|^2 \leq \epsilon_{\text{aux}} + \epsilon_{\text{FOAT}}. \tag{17}$$

By appropriately tuning hyperparameters, the approximation errors $\epsilon_{\text{aux}}$ and $\epsilon_{\text{FOAT}}$ can be made arbitrarily small, such that:

$$\epsilon_{\text{aux}} + \epsilon_{\text{FOAT}} \to 0. \tag{18}$$

Consequently, we obtain:

$$\lim_{t\to\infty} \|\tilde{g}_t\|^2 = 0, \tag{19}$$

which confirms that $\Theta_t$ converges to a stationary point of the global objective $F(\Theta)$.

### C.4 Conclusion

We have established the convergence of CHAINFED, a federated fine-tuning framework integrating DLCT, GPO, and FOAT. Under standard smoothness and bounded variance assumptions, CHAINFED ensures a monotonically decreasing loss with a well-chosen learning rate $\eta_t$ and properly tuned hyperparameters ($\lambda$ and $T$). This guarantees convergence to a stationary point, validating its theoretical soundness and practical efficiency in resource-constrained federated settings.

## D More Experimental Details

### D.1 Evaluation Metrics

We adopt task-specific evaluation metrics to assess model performance across both text classification and instruction tuning tasks (Xu et al., 2025, 2024, 2026d). For text classification on YELP-P, AGNEWS, YAHOO, and 20NEWS, we report the accuracy, which reflects the proportion of correctly predicted labels and enables consistent comparison across datasets with varying label granularities. For instruction tuning, we follow the official evaluation protocols defined by each benchmark. Specifically, for MMLU (Hendrycks et al., 2020), we compute the average accuracy over 57 sub-tasks to assess general knowledge and language understanding. For DROP (Dua et al., 2019), which targets numerical and discrete reasoning, we report the F1 score to capture partial answer overlap. For BBH (Suzgun et al., 2022), a collection of few-shot reasoning benchmarks, task accuracy is used to evaluate generalization under limited supervision. For CRASS (Frohberg and Binder, 2022), we adopt counterfactual accuracy to measure robustness under causal perturbations. All reported results are averaged over five independent runs with different random seeds to ensure statistical reliability and reproducibility (Wang et al., 2025, 2023).

### D.2 Experimental Testbed

Our experiments are conducted on an on-device FL system composed of heterogeneous hardware (Tam et al., 2024; Tian et al., 2024; Zhan et al., 2024).

These devices possess diverse memory and computational capacities, faithfully emulating real-world deployment conditions. All implementations are built using PyTorch and the Hugging Face Transformers library (Wolf, 2019; Xu et al., 2026a,c,b; Wu et al., 2026).

### D.3 Implementation Details

For CHAINFED, we set the following hyperparameters unless specified otherwise. The GPO loss balancing weight $\lambda$ is set to 0.1 for BERT and 0.2 for other models. The FOAT threshold $T$ is set to 0.9 for BERT and 0.8 for others. The DLCT co-tuning window size, $Q$, is determined based on the number of co-tuning adapters that the device with the minimum memory capacity can support.

### D.4 Task-Specific Setting

The training setting is tailored to each task:

- **For Text Classification:** In each training round, 15 devices are randomly selected based on memory capacity. Each device performs one local epoch using the SGD optimizer with a batch size of 8. The input sequence length is set to 64 tokens for AGNEWS and 256 tokens for other datasets (Wu et al.).
- **For Instruction Tuning:** In each round, 10% of devices are randomly selected. Each device performs 10 local training steps using the AdamW optimizer (Loshchilov and Hutter, 2017) with a batch size of 16. The maximum sequence length is set to 512 tokens.

To ensure a fair comparison, the federated fine-tuning process for all baseline methods is continued until the global model converges.

## E Baseline Description

**Memory-Unaware Methods:** 1) Linear Probing (Kornblith et al., 2019b): This method only fine-tunes the output layer while keeping the rest of the model frozen. 2) FedAdapter (Cai et al., 2022): This method dynamically configures adapter modules to accelerate model convergence. 3) C2A (Kim et al., 2023): This approach leverages a hypernetwork to generate device-specific adapters, thereby addressing data heterogeneity across devices.
**Memory-Aware Methods:** 4) FwdLLM (Xu et al., 2023): A backpropagation-free method that leverages zeroth-order optimization to fine-tune the global model, thereby eliminating the need to store intermediate activations during training. 5) FedKSeed (Qin et al., 2023): This method employs zeroth-order optimization with a finite set of random seeds to perform full-parameter fine-tuning. 6) FLoRA (Wang et al., 2024): This method addresses memory constraints by assigning heterogeneous LoRA ranks across devices based on their available memory capacity. 7) FedRA (Su et al., 2024): This method addresses memory constraints by randomly assigning model update tasks to participating devices based on their memory capacities.

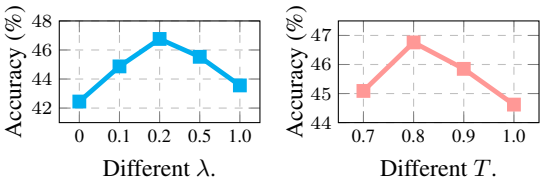



Figure 10: Sensitivity analysis of key hyperparameters on the MMLU evaluated with the LLaMA2-7B.

## F More Experimental Results

### F.1 Sensitivity Analysis

We conduct a sensitivity analysis on our key hyperparameters, $\lambda$ and $T$, to demonstrate the robustness of CHAINFED. The analysis, performed on the LLaMA2-7B model (Figure 10), shows that optimal performance is achieved with $\lambda = 0.2$ and $T = 0.8$. Crucially, these optimal values are consistent with our findings on models in the text classification tasks. This indicates that CHAINFED is robust to hyperparameter settings across different models and tasks, requiring minimal tuning.

### F.2 Model Generalization

To verify the generalization capability of CHAINFED across different model architectures, we conduct additional experiments on TinyLLaMA and LLaMA3.2-3B, with performance evaluated on the MMLU benchmark. The results show that CHAINFED continues to outperform the Full Adapters$^{\dagger}$. Specifically, with a co-tuning window of $Q = 8$, CHAINFED surpasses the upper bound by 1.66% on TinyLLaMA and by a more substantial 4.73% on LLaMA3.2-3B. The ability of CHAINFED to consistently deliver superior performance across a diverse set of model architectures and scales (from BERT to LLaMA) validates it as a general and robust framework for federated fine-tuning.

### F.3 Case Study: Instruction Tuning Quality

To provide a qualitative illustration of CHAINFED's advantages, we conduct a case study compar-

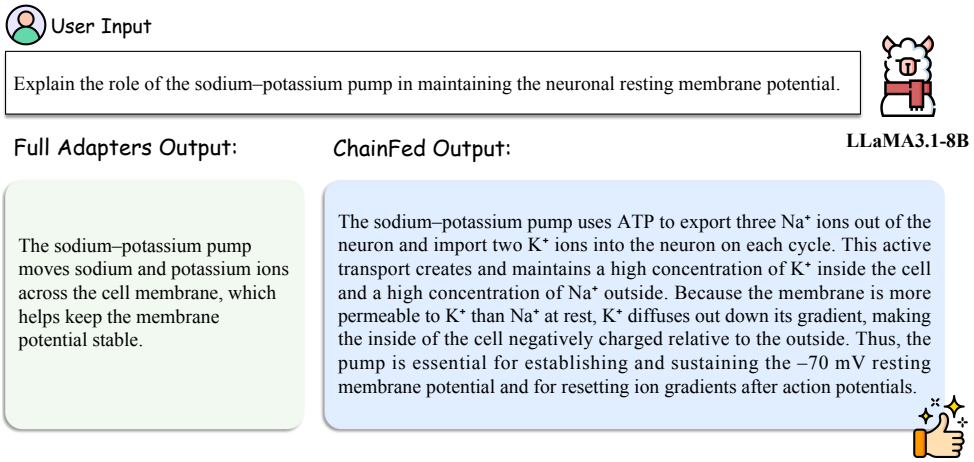



Figure 11: Case Study for LLaMA3.1-8B.

ing its generated responses to those from the Full Adapters† for the same instruction. As shown in Figure 11, CHAINFED's responses are demonstrably superior across several axes. In terms of fluency, CHAINFED's output is more coherent, grammatically natural, and flows more smoothly. For contextual alignment, it demonstrates a deeper comprehension of the user's intent, producing answers that are more precisely tailored to the prompt. Finally, regarding informational content, CHAINFED provides more detailed and nuanced explanations, covering a broader range of relevant details.

These qualitative improvements suggest that CHAINFED's selective, chain-based optimization does more than just save memory; by preserving the foundational knowledge in lower layers, it also leads to better generalization and higher-quality language generation. This highlights the practical effectiveness of CHAINFED, demonstrating its ability to produce superior results even while operating under strict memory constraints.

# G Feasibility and Overhead Analysis

A natural concern regarding the chain optimization paradigm is whether the frequent loading and offloading of model layers introduces significant I/O latency, thereby negating the benefits of memory reduction. Additionally, the computational overhead of adaptive mechanisms like FOAT warrants scrutiny. In this section, we analyze these factors to demonstrate that CHAINFED maintains high runtime efficiency.

## G.1 I/O Latency Analysis

The overhead of swapping layers between storage (disk) and memory is negligible compared to the computational cost of gradient calculation.

- **Magnitude Disparity:** On modern hardware, loading a transformer layer takes milliseconds, whereas the forward and backward propagation for that layer takes seconds.
- **Unified Memory Architecture:** On edge devices (e.g., Jetson Orin, mobile phones) which are the target deployment environment for CHAINFED, the Unified Memory Architecture (UMA) allows the CPU and GPU/NPU to share memory pointers. This eliminates the need for costly data copying between host and device memory during the offloading process, further minimizing write-back latency.

## G.2 Asynchronous Pipelining

To further mitigate I/O costs, CHAINFED employs an asynchronous "Compute-Prefetch-Evict" pipeline that hides I/O latency behind computation.

- **Mechanism:** While the device performs the computation for the current active layer $L_i$, the system asynchronously writes the parameters of the previous layer $L_{i-1}$ back to storage and prefetches the parameters of the next layer $L_{i+1}$ into a reserved memory buffer.
- **Result:** Since the computation time ($T_{comp}$) significantly exceeds the I/O time ($T_{I/O}$), the read/write operations complete before the computation finishes. Thus, the latency is determined by $T_{comp}$, rendering the I/O overhead invisible in the critical path of execution. Consequently, CHAINFED incurs almost zero additional latency attributed to I/O.

### G.3 Efficiency and Feasibility of FOAT

A natural concern is whether computing layer similarity for FOAT imposes prohibitive computational or memory overheads on edge devices. We address this by detailing the efficiency and practical feasibility of our design:

- **Computational Efficiency:** First, the CKA computation is a one-time pre-training setup step (Phase 1 in Algorithm 1). It does not recur during the federated rounds. Furthermore, to minimize overhead, we perform this profiling using only a single mini-batch of local data. Consequently, the cost is negligible compared to the full training process.

- **Resource Feasibility:** To ensure feasibility on devices with limited memory (where loading the full model for profiling is impossible), we implement a block-wise inference strategy. This protocol executes in three phases:

  1. *Budget-Aware Partitioning:* The model is dynamically divided into manageable blocks. The device loads only the specific block of layers that satisfies its current available memory constraints.
  2. *Transient Computation with Immediate Eviction:* Upon performing the forward pass on the current block, the system caches the resulting intermediate activations (feature maps) and immediately evicts the layer parameters from memory to release resources.
  3. *Recursive Activation Passing:* The subsequent block of layers is then loaded, taking the cached activations from the previous step as input.

  By decoupling the inference process from total model size, this strategy ensures that peak memory usage remains strictly bounded by the size of a single block, enabling large-model profiling on constrained devices.

## H Further Analysis

### H.1 Performance Advantages of CHAINFED

Empirically, CHAINFED outperforms the idealized Full Adapters$^{\dagger}$ baseline across all evaluation settings, achieving average gains of 2.86%, 2.46%, and 1.61% on DistilBERT, BERT, and RoBERTa, respectively. This performance advantage stems from two key mechanisms:

- 1) **Selective Layer Adaptation.** By leveraging FOAT to identify and fine-tune only task-critical layers, CHAINFED preserves valuable pre-trained knowledge in lower layers that encode general-purpose linguistic patterns. In contrast, updating all layers indiscriminately can disrupt these foundational representations.

- 2) **Enhanced Generalization.** The combination of DLCT and GPO promotes more stable and generalizable feature learning. DLCT ensures semantic coherence across layers, while GPO prevents over-specialization by maintaining global awareness. This synergistic effect leads to better generalization compared to end-to-end training, which may overfit to local patterns in federated settings.

### H.2 Efficiency Analysis

In addition to significantly reducing the memory footprint, CHAINFED delivers substantial gains in both computational and communication efficiency.

- **Computational Efficiency.** Although adapters are trained sequentially, the total number of forward and backward passes remains comparable to end-to-end training. Moreover, selectively fine-tuning further accelerates model convergence. Empirically, CHAINFED achieves 2.02$\times$ faster convergence than Full Adapters$^{\dagger}$ on AGNEWS.

- **Communication Efficiency.** Communication costs are significantly reduced through two mechanisms: (1) FOAT reduces the number of layers requiring updates, directly decreasing communication volume; (2) the sequential nature ensures that in each round, only adapters within the co-tuning window need synchronization. Our results show that communication can be reduced by up to 3.47$\times$ compared to Full Adapters$^{\dagger}$.

## I The Use of Large Language Models

This manuscript utilized LLMs strictly for the purpose of language editing and textual polishing to enhance presentation quality. We declare that the novel ideas, methodological framework, experimental execution, and data analysis are the original work of the authors. All content modified by AI tools has been carefully reviewed and validated by the authors to ensure accuracy.