FedProxy: Federated Fine-Tuning Large Language Models via Proxy Small Language Models

Anonymous ACL submission

Abstract

Federated fine-tuning of Large Language Models (LLMs) is obstructed by a trilemma of challenges: protecting LLMs intellectual property (IP), ensuring client privacy, and mitigating performance loss on heterogeneous data. Existing methods like Offsite-Tuning (OT) secure the LLMs IP by having clients train only lightweight adapters, yet our analysis reveals they suffer from a fundamental performance bottleneck, leaving a significant gap compared to centralized training. To bridge this gap, we introduce **FedProxy**, a new federated adaptation framework. FedProxy replaces weak adapters with a unified, powerful **proxy Small Language Model (SLM)**, compressed from the proprietary LLM, to serve as a high-fidelity surrogate for collaborative fine-tuning. Our framework systematically resolves the trilemma through a three-stage architecture: (1) **Efficient Representation** via server-guided compression to create a resource-friendly proxy; (2) **Robust Optimization** through an interference-mitigating aggregation strategy to handle data heterogeneity; and (3) **Effortless Fusion** via a training-free "plug-in" mechanism to integrate learned knowledge back into the LLM. Experiments show FedProxy substantially outperforms OT methods and approaches centralized performance, establishing a new benchmark for secure and high-performance federated LLM adaptation.

1 Introduction

The advent of Large Language Models (LLMs) (OpenAI, 2023; Touvron et al., 2023) has revolutionized natural language processing. However, fine-tuning these models on decentralized, privacy-sensitive data via federated learning presents a formidable challenge. This process is hindered by a trilemma: the need to protect the LLMs' intellectual property (IP), ensure client data privacy, and maintain performance across heterogeneous data distributions (Kang et al., 2023; Fan et al., 2023, 2025b,a). To address them, paradigms like Offsite-Tuning (OT) (Xiao et al., 2023) have emerged, where clients train only lightweight components instead of the full model. While OT and its variants (Zhang et al., 2023; Yao et al., 2025b; Kuang et al., 2024) successfully protect the proprietary LLM, our analysis reveals a critical flaw: they suffer from a fundamental performance bottleneck, creating a significant gap compared to standard centralized fine-tuning. This limitation severely hinders their practical utility.

To close this performance gap while upholding privacy and efficiency, we introduce **FedProxy**, a new federated adaptation framework. Instead of relying on weak, disjointed adapters, FedProxy's core innovation is the use of a single, powerful **proxy Small Language Model (SLM)**. This proxy is created on the server by compressing the proprietary LLM and serves as a high-fidelity surrogate for collaborative client-side fine-tuning. The FedProxy framework systematically resolves the key challenges through a three-stage architecture as depicted in Figure 1:

1. Server-Guided Compression: A server-side pipeline uses public data to compress the full LLM into a powerful proxy SLM, which is then distributed to clients. This protects model IP and accommodates resource constraints.

2. Interference-Mitigating Aggregation: A multi-stage aggregation strategy analyzes client heterogeneity and parameter conflicts to guide conflict-aware local training and perform a robust, heterogeneity-aware merge on the server.

3. Training-Free Knowledge Fusion: A simple and effective "plug-in" mechanism integrates the enhanced proxy SLM back into the original LLM, transferring the learned knowledge without costly retraining.

Our key contributions are:

- We identify the performance bottleneck in OT-based federated learning and propose **FedProxy**, a new framework that closes this gap by replacing weak adapters with a powerful proxy SLM.
- We design a three-stage architecture for FedProxy that systematically addresses the core challenges of representation, optimization, and fusion through compression, interference-mitigating aggregation, and training-free knowledge transfer.
- We demonstrate through extensive experiments that FedProxy significantly outperforms OT methods and achieves performance comparable to centralized fine-tuning, setting a new standard for secure and high-performance federated LLM adaptation.

2 Related Work

2.1 Offsite-Tuning

The proliferation of large language models (LLMs) has necessitated robust privacy-preserving mechanisms for fine-tuning, particularly in client-server architectures where direct exposure of the server's proprietary LLM is undesirable. To address this, (Xiao et al., 2023) introduced *Offsite-Tuning* (OT), a parameter-efficient framework that enables task-specific adaptation without transmitting the full model. In the OT paradigm, the server sends a lightweight, frozen LLM *emulator* and trainable adapter modules to the client. The client fine-tunes only the adapters on its local data and returns them to the server for integration into the full LLM. This preserves both client data privacy and server model IP. Subsequent works have aimed to enhance the OT framework. For instance, CRaSh (Zhang et al., 2023) improves emulator generation through layer clustering and dropping, while ScaleOT (Yao et al., 2025b) and GradOT (Yao et al., 2025a) introduce more sophisticated compression techniques based on reinforcement learning and gradient preservation, respectively. The concept has also been extended to federated settings with *FedOT* (Kuang et al., 2024). However, our empirical analysis, detailed in Section 5, reveals that while OT-based methods show improvements over zero-shot performance on models like LLaMA, they consistently fall short of the performance achieved by centralized fine-tuning.

2.2 Model Merging

Model merging seeks to combine multiple fine-tuned models into a single, powerful one without retraining. While naive weight averaging (Wortsman et al., 2022) often fails, advanced techniques address task conflicts. Some methods, like Fisher-Merging (Matena and Raffel, 2022) and RegMean (Jin et al., 2023), compute task-aware coefficients. Others operate on *task vectors* (the difference between fine-tuned and pretrained weights) (Ilharco et al., 2023). More sophisticated approaches, such as TIES-Merging (Yadav et al., 2023) and DARE (Yu et al., 2024), resolve interference between these vectors by trimming redundant parameters and rescaling weights, leading to more robust merged models.

3 Problem Formulation

The goal of federated LLM fine-tuning is to enhance a proprietary, server-held LLM f_θ by leveraging private data $\{\mathcal{D}_k\}_{k=1}^K$ from K clients. This task is fundamentally constrained by the need to protect the LLM's intellectual property (IP), accommodate clients' limited resources, ensure data privacy, and handle statistical data heterogeneity (Xiao et al., 2023; Fan et al., 2025a). We decompose this into three core sub-problems:

1. Efficient and Secure Model Representation. To protect model IP and accommodate client constraints, transmitting the full LLM f_θ is infeasible. The first problem is to create a compact proxy model f_ϕ ($|\phi| \ll |\theta|$) that is powerful enough for fine-tuning yet small enough for efficient on-device training and communication. The objective is to find a compression function g to generate the initial proxy $\phi^{(0)}$:

$$\phi^{(0)} = g(\theta, \mathcal{D}_{\text{pub}}) \tag{1}$$

2. Heterogeneity-Robust Federated Optimization. Due to diverse local data $\mathcal{D}_k$, aggregating client-trained models $\{\phi_k^{(t)}\}$ can lead to "negative interference," where conflicting updates degrade performance. The second problem is to devise a robust federated optimization strategy $\mathcal{A}$ that mitigates this interference. The goal is to produce an improved global proxy model $\phi^{(t+1)}$ at each round:

$$\phi^{(t+1)} = \mathcal{A}(\{\phi_k^{(t)}\}_{k=1}^K, \phi^{(t)}) \tag{2}$$

3. Client Knowledge Fusion into the LLM. Finally, the client knowledge aggregated in the converged proxy model ϕ^* must be transferred back to

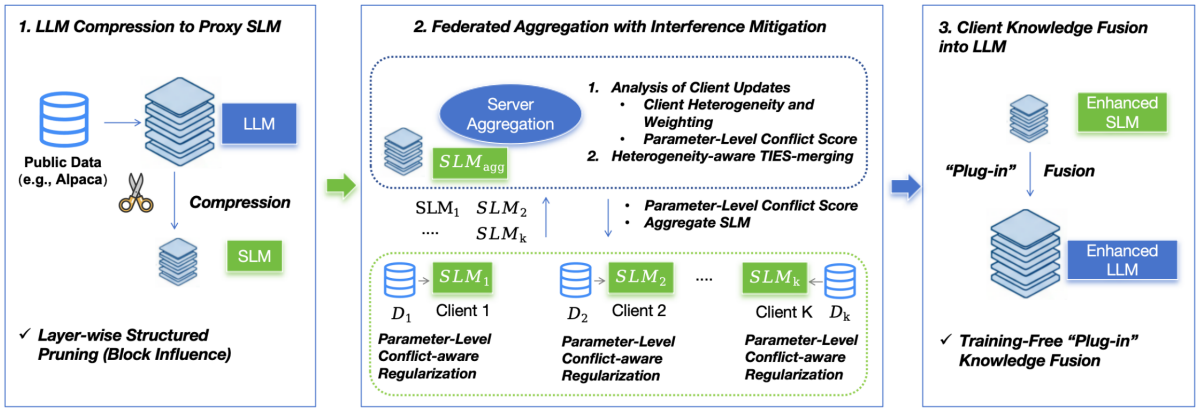

Figure 1: Overview of the FedProxy framework. The framework consists of three main stages: (1) LLM Compression to Proxy SLM, (2) Client-side SLM training and federated aggregation with interference mitigation, and (3) Client knowledge fusion back into the global LLM.

enhance the original LLM f_θ. The third problem is to design a fusion mechanism $\mathcal{F}$ that integrates this specialized knowledge, improving the LLM without direct access to client data:

$$\theta_{\text{new}} = \mathcal{F}(\theta_{\text{old}}, \phi^*) \tag{3}$$

4 The Proposed FedProxy Framework

To address the sub-problems defined in Section 3, we introduce **FedProxy**, a three-stage framework that provides a concrete solution to the problem decomposition. We illustrate the FedProxy architecture in Figure 1, with further details provided in Algorithm 1. The first stage, **Efficient Model Representation via Compression**, is detailed in Section 4.1. The second stage, **Robust Optimization via Interference Mitigation**, is covered in Section 4.2. Finally, the **Training-Free "Plug-in" Knowledge Fusion** is explained in Section 4.3.

4.1 LLM Compression to Proxy SLM

The first phase of the FedProxy framework is to compress a large, general-purpose LLM, denoted as f_θ, into a single, efficient proxy SLM, f_ϕ. This proxy SLM serves as a shared, high-quality foundation for all clients, significantly reducing the computational and communication overhead required for federated fine-tuning. Instead of relying on client-specific data, which can be sensitive and heterogeneous, we perform this compression using a broad, publicly available instruction-following dataset, $\mathcal{D}_{\text{pub}}$ (e.g., Alpaca (Taori et al., 2023)). This approach ensures that the resulting SLM retains a strong, task-agnostic reasoning and instruction-following capability, making it an effective starting point for diverse downstream tasks.

The compression is achieved through structured pruning, where we identify and remove entire transformer blocks from the original LLM. We quantify the importance of each block using the Block Influence (BI) metric, adapted from ShortGPT (Men et al., 2024). The BI score for the i-th transformer block is calculated based on its impact on hidden state representations when processing the public dataset:

$$\text{BI}_i = 1 - \mathbb{E}_{X \sim \mathcal{D}_{\text{pub},t}} \left[\frac{X_{i,t}^\top X_{i+1,t}}{\|X_{i,t}\|_2 \|X_{i+1,t}\|_2} \right] \tag{4}$$

where $X_{i,t}$ denotes the representation of the t-th token at the output of block i. A higher BI score indicates that a block contributes more significantly to the model's representational capacity.

Based on these scores, we generate a global pruning mask by selecting the blocks with the highest BI scores until the desired compression ratio κ is met. This process yields a single, compact ***proxy SLM*** architecture, f_ϕ, where $|\phi| \leq \kappa|\theta|$. This unified Proxy SLM is then used as the initial model for all clients in the subsequent federated fine-tuning stage (Section 4.2).

4.2 Federated Aggregation with Interference Mitigation

Parameter interference represents a critical challenge in federated model merging, particularly when aggregating models trained on heterogeneous tasks. As demonstrated in prior research (Yu et al., 2024; Yadav et al., 2023), unweighted averaging can significantly degrade model performance through two primary interference mechanisms: magnitude interference, where task-critical

Algorithm 1 The FedProxy Framework

Input:
Proprietary LLM f_θ; Public dataset $\mathcal{D}_{\text{pub}}$;
Number of clients K; Communication rounds T;
Client datasets $\{\mathcal{D}_k\}_{k=1}^K$; Compression ratio κ;
Output: Enhanced global LLM θ_{final}

Server-Side Initialization:
Generate proxy SLM: $\phi^{(0)} \leftarrow \text{Compress}(\theta, \mathcal{D}_{\text{pub}}, \kappa)$ (Sec 4.1)
Initialize parameter conflict scores $C_d^{(0)} \leftarrow \mathbf{0}$ for all dimensions d
for each round $t = 0$ **to** $T-1$ **do**
  Server-Side (Distribution):
  Distribute global proxy $\phi^{(t)}$ and conflict scores $\{C_d^{(t)}\}$ to all clients.
  Client-Side Update (Parallel):
  for each client $k \in [K]$ **do**
    Receive $\phi^{(t)}$ and $\{C_d^{(t)}\}$.
    Train ϕ_k on $\mathcal{D}_k$ to minimize $\mathcal{L}_k$ (Eq. 8) $\rightarrow$ get $\phi_k^{(t+1)}$.
    Send updated proxy $\phi_k^{(t+1)}$ to server.
  end for
  Server-Side (Analysis and Aggregation):
  Receive $\{\phi_k^{(t+1)}\}_{k=1}^K$ from clients.
  // Server-Side Analysis of Client Updates
  Compute task vectors $\tau_k^{(t+1)} \leftarrow \phi_k^{(t+1)} - \phi^{(t)}$.
  Compute heterogeneity $\{h_k^{(t+1)}\}$ (Eq. 5) and weights $\{w_k^{(t+1)}\}$ (Eq. 6).
  Compute parameter conflict scores $\{C_d^{(t+1)}\}$ for the next round (Eq. 7).
  // Heterogeneity-Aware Aggregation (H-TIES)
  Aggregate task vectors $\{\tau_k^{(t+1)}\}$ using H-TIES merging (Sec 4.2) to get global update $\Delta\phi^{(t+1)}$.
  Update global proxy: $\phi^{(t+1)} \leftarrow \phi^{(t)} + \Delta\phi^{(t+1)}$.
end for
Final Knowledge Fusion:
Fuse converged proxy $\phi^{(T)}$ into base LLM θ (Sec 4.3).
$\theta_{\text{final}} \leftarrow \mathcal{F}(\theta, \phi^{(T)})$ (Eq. 3)
Return θ_{final}

parameters are diluted, and directional interference, where parameters with opposing signs cancel each other out.

To address this, FedProxy implements a three-stage aggregation strategy in each communication round t. First, the server analyzes all incoming client updates to compute metrics for heterogeneity, weighting, and parameter-level conflict. Second, clients use these metrics for a conflict-aware regularization during their local training. Finally, the server uses the metrics to perform a heterogeneity-aware model merge. This process is detailed below.

4.2.1 Server-Side Analysis of Client Updates

At the end of a training round t, after receiving updated models $\phi_k^{(t)}$ from all clients, the server performs a comprehensive analysis. This step generates a set of metrics that will guide both the next round of client training and the current round's aggregation.

Task Vector and Similarity Calculation. The server first computes each client's task vector, $\tau_k^{(t)} = \phi_k^{(t)} - \phi^{(t-1)}$, representing the update from the previous global model. It then calculates pairwise cosine similarity scores $S_{k,j}^{(t)} = \cos(\tau_k^{(t)}, \tau_j^{(t)})$ between all client pairs.

Heterogeneity and Weighting Metrics. Based on the similarity scores, the server identifies a *task-similarity neighborhood* $\mathcal{N}_k^{(t)} = \{j : |S_{k,j}^{(t)}| > \gamma\}$ for each client, where γ is a similarity threshold. From this, two key metrics are derived:

- A **heterogeneity coefficient** $h_k^{(t)}$, which measures how much a client deviates from its peers. A higher value indicates a more unique or outlier task update.

$$h_k^{(t)} = 1 - \frac{1}{|\mathcal{N}_k^{(t)}|} \sum_{j \in \mathcal{N}_k^{(t)}} \max\left(0, S_{k,j}^{(t)}\right) \quad (5)$$

- An **adaptive aggregation weight** $w_k^{(t)}$, which prioritizes clients that form a strong consensus with their neighbors.

$$w_k^{(t)} = \frac{\exp\left(\sum_{j \in \mathcal{N}_k^{(t)}} |S_{k,j}^{(t)}|\right)}{\sum_{k'=1}^{K} \exp\left(\sum_{j \in \mathcal{N}_{k'}^{(t)}} |S_{k',j}^{(t)}|\right)} \quad (6)$$

The heterogeneity coefficient is also normalized via $h_{k,\text{norm}}^{(t)} = (h_k^{(t)} - h_{\min})/(h_{\max} - h_{\min})$. These two metrics, $h_{k,\text{norm}}^{(t)}$ and $w_k^{(t)}$, are used in the server-side H-TIES merging step.

Parameter-Level Conflict Score. Finally, the server calculates a conflict score $C_d^{(t)}$ for each parameter dimension d, quantifying the disagreement in update signs across all clients.

$$C_d^{(t)} = 1 - \frac{1}{K} \left| \sum_{k=1}^{K} \text{sign}(\tau_k^{(t)}[d]) \right| \quad (7)$$

A score near 1 indicates high conflict (updates pull in different directions), while a score near 0 signifies strong agreement. This conflict score is sent to clients to be used in the next training round.

4.2.2 Parameter-Level Conflict-Aware Client-Side Regularization (PCR)

To proactively mitigate interference, clients incorporate the server-provided conflict scores into their

local training objective for the next round, $t+1$. The composite loss function $\mathcal{L}_k$ for client k combines the standard task loss with a dynamic regularization term:

$$\mathcal{L}_k = \mathcal{L}_{\text{task}}(\phi_k; \mathcal{D}_k) + \lambda_{\text{reg}} \cdot \mathcal{L}_{\text{reg},k} \tag{8}$$

where ϕ_k is the model being trained, and the regularization term $\mathcal{L}_{\text{reg},k}$ penalizes divergence from the previous global model $\phi^{(t)}$. The penalty is scaled by the parameter-level conflict scores $C_d^{(t)}$ from the previous round:

$$\mathcal{L}_{\text{reg},k} = \sum_d C_d^{(t)} \cdot \|\phi_k[d] - \phi^{(t)}[d]\|_2^2 \tag{9}$$

This mechanism encourages clients to align with the global model on consensus parameters (where $C_d^{(t)}$ is high) while allowing more freedom for personalization on conflicting parameters (where $C_d^{(t)}$ is low).

4.2.3 Heterogeneity-aware TIES-merging

To effectively merge client updates, the server employs the **Heterogeneity-aware TIES-merging (H-TIES)** strategy. This approach adapts the aggregation process by dynamically using the heterogeneity coefficient $h_k^{(t)}$ and aggregation weight $w_k^{(t)}$. It consists of three steps:

1. Heterogeneity-Adaptive Sparsification. H-TIES adjusts the sparsity of each client's update based on its heterogeneity. The retention rate $r_k^{(t)}$ is inversely proportional to the heterogeneity coefficient $h_{k,\text{norm}}^{(t)}$. Let r_0 be the base retention rate and δ be the adaptive strength:

$$r_k^{(t)} = \max\left(0, \min(1, r_0 - \delta \cdot h_{k,\text{norm}}^{(t)})\right) \tag{10}$$

Clients with high heterogeneity (large $h_{k,\text{norm}}^{(t)}$) will have their updates made sparser, while homogeneous clients (small $h_{k,\text{norm}}^{(t)}$) retain a larger portion. This yields the ***sparsified update*** $\tilde{\tau}_k^{(t)}$.

2. Pre-Aggregation Scaling. Each client's influence is then scaled by its aggregation weight $w_k^{(t)}$ to amplify the contributions of clients within a strong consensus:

$$\hat{\tau}_k^{(t)} = w_k^{(t)} \tilde{\tau}_k^{(t)} \tag{11}$$

3. Weighted Conflict Resolution and Merging. Finally, the server resolves conflicts and merges the updates using a method inspired by the principles of TIES-merging (Yadav et al., 2023). This is a pure merging step without any server-side learning. For each parameter dimension d, it first computes the total weighted magnitude of positive (P_d) and negative (N_d) updates:

$$P_d = \sum_{k:\hat{\tau}_k^{(t)}[d]>0} |\hat{\tau}_k^{(t)}[d]|, \quad N_d = \sum_{k:\hat{\tau}_k^{(t)}[d]<0} |\hat{\tau}_k^{(t)}[d]| \tag{12}$$

An update $\Delta\phi^{(t)}[d]$ is then computed by performing a weighted average of only the updates that conform to a dominant sign. A sign is dominant if its total magnitude sufficiently outweighs the other, controlled by a threshold $\rho \geq 1$:

$$\Delta\phi^{(t)}[d] = \begin{cases} \frac{\sum_{k:\hat{\tau}_k^{(t)}[d]>0} \hat{\tau}_k^{(t)}[d]}{\sum_{k:\hat{\tau}_k^{(t)}[d]>0} w_k}, & \text{if } \frac{P_d}{N_d+\varepsilon} \geq \rho \\ \frac{\sum_{k:\hat{\tau}_k^{(t)}[d]<0} \hat{\tau}_k^{(t)}[d]}{\sum_{k:\hat{\tau}_k^{(t)}[d]<0} w_k}, & \text{if } \frac{N_d}{P_d+\varepsilon} \geq \rho \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

where ε is a stability constant. This step performs a weighted average of the conforming updates, preventing interference from conflicting signs.

The global ***proxy SLM*** is then updated by directly applying this aggregated delta:

$$\phi^{(t)} = \phi^{(t-1)} + \Delta\phi^{(t)} \tag{14}$$

4.3 Training-Free "Plug-in" Knowledge Fusion

In the final stage, we fuse the knowledge from the aggregated proxy SLM, f_ϕ, back into the original LLM, f_θ. Our approach is simple, effective, and entirely training-free. Leveraging the architectural mapping where the proxy SLM is a direct sub-network of the LLM (i.e., $\phi \subset \theta$), we perform a direct parameter replacement. Specifically, we take the updated layers from the global proxy SLM and use them to overwrite the corresponding layers in the full LLM. This "plug-in" mechanism seamlessly integrates the federated enhancements without requiring any costly retraining or complex merging procedures on the massive LLM, thus efficiently updating its capabilities with the knowledge gained from clients.

5 Experiments

5.1 Experimental Setup

We evaluate our framework using **LLaMA2-7B** (Touvron et al., 2023) and **Mistral-7B-Instruct-v0.2** (Jiang et al., 2023) as the foundational LLMs.

The experiments are designed around two key scenarios to simulate different real-world conditions.

Datasets. We use a total of eight datasets from two widely-adopted benchmarks. For question answering (QA), we use **OBQA** (Mihaylov et al., 2018), **ARC-Challenge (ARC-C)** (Clark et al., 2018), **ARC-Easy (ARC-E)** (Clark et al., 2018), and **CommonsenseQA (CQA)** (Talmor et al., 2019). For general language understanding, we use four tasks from the GLUE benchmark (Wang et al., 2019): **SST2** (Socher et al., 2013), **MRPC** (Dolan and Brockett, 2005), **RTE** (Giampiccolo et al., 2007), and **MNLI** (Williams et al., 2018).

Task Distribution and Scenarios. The datasets are distributed among clients as follows:

- **Homogeneous Tasks**: A 4-client setup where each client is assigned a different partition of the same dataset. The data is distributed randomly and uniformly across the clients.
- **Heterogeneous Tasks**: An 8-client setup where each client is assigned a unique dataset.

Evaluation. All tasks are evaluated based on accuracy. We employ the *lm-evaluation-harness* package (Gao et al., 2023) for standardized zero-shot evaluation across all methods to ensure fair and reproducible comparisons. Detailed configurations are provided in Table 1.

Baselines. We compare FedProxy against four key baselines to benchmark its performance:

- **ZeroShot**: The LLM's performance without any fine-tuning, serving as a **lower bound**.
- **CentSFT**: The LLM fine-tuned on all client data centrally, representing the performance **upper bound**.
- **OT** (Xiao et al., 2023): A standard offsite-tuning approach (the first two and the last two decoders as the adapter).
- **FedOT** (Kuang et al., 2024): The federated extension of offsite-tuning (the first two and the last two decoders as the adapter).

5.2 Main Results

We evaluate FedProxy by creating proxy SLMs with compression ratios of 30% and 50%. The results, detailed in Tables 2 and 3, demonstrate our framework's consistent superiority across both models and task settings.

Scenario	Architecture	Clients	Datasets
Homogeneous	1 Server: LLM 4 Clients: Proxy SLM	Client 0	Partition of Single DataSet
		Client 1	Partition of Single DataSet
		Client 2	Partition of Single DataSet
		Client 3	Partition of Single DataSet
Heterogeneous	1 Server: LLM 8 Clients: Proxy SLM	Client 0	Complete OBQA
		Client 1	Complete ARC-E
		Client 2	Complete ARC-C
		Client 3	Complete CQA
		Client 4	Complete SST2
		Client 5	Complete MRPC
		Client 6	Complete RTE
		Client 7	Complete MNLI

Table 1: Configuration of Homogeneous and Heterogeneous Tasks in FedProxy.

Homogeneous Tasks. In the homogeneous setting (Table 2), FedProxy establishes a commanding performance advantage. With LLaMA2-7B at 30% compression, for instance, it achieves a QA accuracy of **0.6261** and a GLUE score of **0.8678**. This marks a significant leap over OT-based baselines, with absolute improvements of **14.3** and **27.9** percentage points over FedOT, and **14.4** and **30.3** points over OT on QA and GLUE tasks, respectively. Crucially, FedProxy closes the performance gap to the centralized fine-tuning (CentSFT) upper bound, reaching **95.1%** of its performance on QA and **97.9%** on GLUE. This strong performance is consistently mirrored at the 50% compression ratio and across the Mistral-7B model, underscoring the framework's robustness and effectiveness.

Heterogeneous Tasks. FedProxy's superiority is even more striking in the challenging heterogeneous setting (Table 3), where mitigating task interference is paramount. For LLaMA2-7B at 30% compression, FedProxy achieves a QA accuracy of **0.6224** and a GLUE score of **0.8494**. This represents a massive leap over FedOT, with absolute improvements of **13.4** and **19.5** percentage points on QA and GLUE tasks, respectively. Crucially, FedProxy again closes the performance gap to the centralized fine-tuning (CentSFT) upper bound, reaching **95.8%** of its performance on QA and **96.5%** on GLUE. This pattern of significant outperformance is consistently observed with the Mistral-7B model and at the 50% compression ratio, powerfully validating the efficacy of our multi-stage aggregation strategy in neutralizing parameter interference and proving its robustness in diverse federated ecosystems.

Model	Method	Ratio	QA					GLUE				
			OBQA	ARC-E	ARC-C	CQA	*Avg*	SST2	MRPC	RTE	MNLI	*Avg*
LLaMA2-7B	ZeroShot	-	0.322	0.7656	0.4445	0.3079	0.46	0.4954	0.6887	0.6137	0.3996	0.5494
	CentSFT	-	0.522	0.8102	0.5188	0.783	0.6585	0.9633	0.8505	0.8628	0.8677	0.8861
	OT	30	0.362	0.7761	0.4667	0.3219	0.4817	0.4943	0.701	0.5704	0.4938	0.5649
	FedOT	30	0.362	0.7782	0.4642	0.3268	0.4828	0.6468	0.6936	0.5523	0.4609	0.5884
	FedProxy	**30**	**0.446**	**0.7963**	**0.5068**	**0.7551**	**0.6261**	**0.961**	**0.7696**	**0.8809**	**0.8596**	**0.8678**
	OT	50	0.35	0.7753	0.4437	0.3112	0.4701	0.539	0.6936	0.6245	0.36	0.5543
	FedOT	50	0.354	0.7715	0.4462	0.335	0.4767	0.6206	0.7083	0.6101	0.3941	0.5833
	FedProxy	**50**	**0.39**	**0.7866**	**0.471**	**0.7265**	**0.5935**	**0.9656**	**0.7696**	**0.8592**	**0.6207**	**0.8038**
Mistral-7B	ZeroShot	-	0.368	0.8114	0.5468	0.6806	0.6017	0.867	0.7304	0.722	0.5952	0.7287
	CentSFT	-	0.536	0.838	0.5734	0.8288	0.6941	0.9656	0.8922	0.8881	0.8835	0.9074
	OT	30	0.44	0.8009	0.5162	0.7445	0.6254	0.9404	0.7353	0.7834	0.7264	0.7964
	FedOT	30	0.428	0.8266	0.5666	0.7363	0.6394	0.9415	0.75	0.7581	0.7562	0.8015
	FedProxy	**30**	**0.494**	**0.8396**	**0.5657**	**0.7985**	**0.6745**	**0.9667**	**0.8456**	**0.87**	**0.8848**	**0.8918**
	OT	50	0.422	0.7955	0.5452	0.688	0.6127	0.9048	0.7353	0.7509	0.6174	0.7521
	FedOT	50	0.4	0.8182	0.5683	0.6888	0.6188	0.8819	0.7426	0.7329	0.5934	0.7377
	FedProxy	**50**	**0.462**	**0.8573**	**0.5998**	**0.7576**	**0.6692**	**0.9644**	**0.7672**	**0.8123**	**0.834**	**0.8445**

Table 2: Method Performance Comparison in the Homogeneous Tasks Setting.

Model	Method	Ratio	QA					GLUE				
			OBQA	ARC-E	ARC-C	CQA	*Avg*	SST2	MRPC	RTE	MNLI	*Avg*
LLaMA2-7B	ZeroShot	-	0.322	0.7656	0.4445	0.3079	0.46	0.4954	0.6887	0.6137	0.3996	0.5494
	CentSFT	-	0.522	0.8102	0.5188	0.783	0.6585	0.9633	0.8505	0.8628	0.8677	0.8861
	OT	30	0.362	0.7761	0.4667	0.3219	0.4817	0.4943	0.701	0.5704	0.4938	0.5649
	FedOT	30	0.344	0.7803	0.4659	0.362	0.4881	0.828	0.6887	0.6931	0.4107	0.6551
	FedProxy	**30**	**0.424**	**0.7837**	**0.5162**	**0.7658**	**0.6224**	**0.9541**	**0.8113**	**0.8267**	**0.8056**	**0.8494**
	OT	50	0.35	0.7753	0.4437	0.3112	0.4701	0.539	0.6936	0.6245	0.36	0.5543
	FedOT	50	0.35	0.766	0.442	0.3419	0.475	0.6514	0.6863	0.6245	0.4108	0.5933
	FedProxy	**50**	**0.382**	**0.7866**	**0.4872**	**0.7486**	**0.6011**	**0.9541**	**0.7721**	**0.8484**	**0.6029**	**0.7944**
Mistral-7B	ZeroShot	-	0.368	0.8114	0.5468	0.6806	0.6017	0.867	0.7304	0.722	0.5952	0.7287
	CentSFT	-	0.536	0.838	0.5734	0.8288	0.6941	0.9656	0.8922	0.8881	0.8835	0.9074
	OT	30	0.44	0.8009	0.5162	0.7445	0.6254	0.9404	0.7353	0.7834	0.7264	0.7964
	FedOT	30	0.404	0.8405	0.5794	0.7113	0.6338	0.93	0.7377	0.7834	0.7066	0.7894
	FedProxy	**30**	**0.448**	**0.8211**	**0.5794**	**0.8092**	**0.6644**	**0.9564**	**0.8382**	**0.8592**	**0.8408**	**0.8737**
	OT	50	0.422	0.7955	0.5452	0.688	0.6127	0.9048	0.7353	0.7509	0.6174	0.7521
	FedOT	50	0.386	0.827	0.5683	0.7027	0.621	0.9002	0.7181	0.7148	0.6101	0.7358
	FedProxy	**50**	**0.434**	**0.8472**	**0.6109**	**0.7625**	**0.6637**	**0.961**	**0.8039**	**0.8592**	**0.8091**	**0.8583**

Table 3: Method Performance Comparison in the Heterogeneous Tasks Setting.

5.3 Ablation Study

5.3.1 Architectural Advantage of the Proxy SLM

To isolate the architectural benefits of our proxy SLM, we conduct an ablation study against the Offsite-Tuning (OT) paradigm. We introduce a stronger baseline, **FedOT-ET (Emulator Trained)**, where clients train both the emulator and adapters—unlike standard OT, where only adapters are trained. This experiment, conducted in the homogeneous setting with a 50% compression ratio, aims to answer: ***Is FedProxy's advantage merely due to training more parameters, or does its architectural design offer a fundamental benefit?***

As shown in Table 4, while FedOT-ET slightly improves on FedOT, FedProxy significantly outperforms both. This confirms that FedProxy's superiority stems from its cohesive architectural design—fine-tuning an integrated proxy SLM is fundamentally more effective than training a disjointed adapter-emulator structure, thus highlighting the inherent limitations of the OT paradigm.

5.3.2 Impact of Aggregation Components

To dissect the contribution of our multi-stage aggregation strategy, we conduct a systematic ablation study in the challenging heterogeneous setting, using models at a 50% compression ratio. We evaluate the following configurations, with results detailed in Table 5:

- **FedProxy (Full)**: Our complete method, com-

Model	Method	QA	GLUE	ALL
LLaMA2-7B	**FedProxy**	**0.6011**	**0.7944**	**0.6977**
	FedOT-ET	0.4877	0.5751	0.5314
	FedOT	0.4767	0.5833	0.5300
Mistral-7B	**FedProxy**	**0.6637**	**0.8583**	**0.7610**
	FedOT-ET	0.6251	0.7455	0.6853
	FedOT	0.6188	0.7377	0.6783

Table 4: Architectural comparison with Offsite-Tuning (OT) variants. FedOT-ET (Emulator Trained) is a stronger baseline where the emulator is also trained. We report average accuracy on QA, GLUE, and ALL. The results demonstrate that FedProxy's cohesive proxy SLM architecture significantly outperforms the adapter-based OT structure, even when the OT emulator is unfrozen.

Model	Method	QA	GLUE	ALL
LLaMA2-7B	**FedProxy (Full)**	0.6011	0.7944	**0.6977**
	w/o PCR	0.5948	0.7860	0.6904
	w/o H-Ties	0.5962	0.7615	0.6788
	FedAvg	**0.6065**	0.7551	0.6808
	TIES-Merging	0.5911	**0.8008**	0.6959
Mistral-7B	**FedProxy (Full)**	**0.6637**	**0.8583**	**0.7610**
	w/o PCR	0.6626	0.8477	0.7551
	w/o H-Ties	0.6595	0.8369	0.7482
	FedAvg	0.6617	0.8261	0.7439
	TIES-Merging	0.6543	0.8547	0.7545

Table 5: Ablation study of the aggregation components. We report average accuracy on QA, GLUE, and ALL. Our full method achieves the best overall performance, while simpler baselines can excel on individual subtasks.

bining client-side Parameter-level Conflict-aware Regularization (PCR) and server-side Heterogeneity-aware TIES (H-TIES) merging.

- **w/o PCR**: Removes client-side regularization, relying solely on the H-TIES server merge to resolve conflicts.
- **w/o H-TIES**: Retains client-side PCR but replaces our server-side H-TIES merge with the standard FedAvg algorithm, isolating the impact of our heterogeneity-aware analysis.
- **TIES-Merging** (Yadav et al., 2023): A strong baseline using the TIES algorithm alone, without PCR or H-TIES.
- **FedAvg** (McMahan et al., 2017): The standard federated averaging baseline.

The results clearly demonstrate the synergistic value of our design. Both **FedAvg** and **TIES-Merging** show substantially degraded performance, confirming that standard aggregation methods are insufficient to handle parameter interference. Removing either of our key components results in a significant performance drop: **w/o PCR** struggles to align client updates effectively, while **w/o H-TIES** fails to optimally merge models even with regularized clients. The **full FedProxy model** consistently and significantly outperforms all ablated and baseline configurations, proving that both conflict-aware client regularization and heterogeneity-aware server merging are critical, complementary components for achieving robust performance.

6 Conclusions

In this work, we introduced **FedProxy**, a federated LLM fine-tuning framework that addresses the core challenges of LLM IP protection, client-side constraints and data privacy, and data heterogeneity. By decomposing the problem into three stages—compression, aggregation, and fusion—FedProxy offers a systematic solution. Our framework replaces weak adapters with a powerful, compressed proxy SLM, mitigates parameter interference through a multi-stage aggregation strategy, and effectively fuses learned knowledge back into the global model. Extensive experiments show that FedProxy significantly outperforms OT-based methods and approaches the performance of centralized training, establishing a new standard for secure, efficient, and high-performance federated LLM adaptation.

Limitations

Our work on FedProxy opens up several promising avenues for future research. One key direction is exploring the impact of the public dataset on proxy SLM quality, including methods for optimal dataset selection or the creation of more universally effective proxies. Furthermore, to address the quadratic complexity of the server-side analysis, future work could focus on developing more scalable, perhaps approximate, aggregation algorithms for settings with a massive number of clients such as hierarchical aggregation. Finally, investigating the long-term dynamics of the knowledge fusion process presents a rich opportunity to understand how repeated integrations affect the global LLM's

stability and capacity for continuous learning.

References

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*.

Tao Fan, Hanlin Gu, Xuemei Cao, Chee Seng Chan, Qian Chen, Yiqiang Chen, Yihui Feng, Yang Gu, Jiaxiang Geng, Bing Luo, et al. 2025a. Ten challenging problems in federated foundation models. *IEEE Transactions on Knowledge and Data Engineering*.

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

Tao Fan, Guoqiang Ma, Yan Kang, Hanlin Gu, Yuanfeng Song, Lixin Fan, Kai Chen, and Qiang Yang. 2025b. Fedmkt: Federated mutual knowledge transfer for large and small language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 243–255.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. 2023. Dataless knowledge fusion by merging weights of language models. In *The Eleventh International Conference on Learning Representations*.

Yan Kang, Tao Fan, Hanlin Gu, Xiaojin Zhang, Lixin Fan, and Qiang Yang. 2023. Grounding foundation models through federated transfer learning: A general framework. *ACM Transactions on Intelligent Systems and Technology*.

Weirui Kuang, Bingchen Qian, Zitao Li, Daoyuan Chen, Dawei Gao, Xuchen Pan, Yuexiang Xie, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024. Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5260–5271.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Michael S Matena and Colin A Raffel. 2022. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716.

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.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

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*.

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

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR.

Guangxuan Xiao, Ji Lin, and Song Han. 2023. Offsite-tuning: Transfer learning without full model. *arXiv preprint arXiv:2302.04870*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115.

Kai Yao, Zhaorui Tan, Penglei Gao, Lichun Li, Kaixin Wu, Yinggui Wang, Yuan Zhao, Yixin Ji, Jianke Zhu, and Wei Wang. 2025a. Gradot: Training-free gradient-preserving offsite-tuning for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5115–5130.

Kai Yao, Zhaorui Tan, Tiandi Ye, Lichun Li, Yuan Zhao, Wenyan Liu, Wei Wang, and Jianke Zhu. 2025b. Scaleot: Privacy-utility-scalable offsite-tuning with dynamic layerreplace and selective rank compression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 22074–22082.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*.

Kaiyan Zhang, Ning Ding, Biqing Qi, Xuekai Zhu, Xinwei Long, and Bowen Zhou. 2023. Crash: Clustering, removing, and sharing enhance fine-tuning without full large language model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9612–9637.

A Implementation Details

A.1 Hyperparameter Settings

All client-side training was conducted using LoRA fine-tuning with the AdamW optimizer. Key hyperparameters are detailed below:

- **General Training:** We used a batch size of 16, a learning rate of 5e-5, and trained for 10 local epochs. The maximum input and target sequence lengths were set to 64 and 128, respectively.
- **LoRA Configuration:** The LoRA rank was set to 32, alpha to 64, and dropout to 0.1.
- **FedProxy Parameters:** For the Parameter-Level Conflict-Aware Client-Side Regularization (PCR), the regularization coefficient λ was set to 1e-5. For the Heterogeneity-Aware Aggregation (H-TIES), the base retention rate r_0 was 1.0, the heterogeneity penalty δ was 0.2, and the consensus reward ρ was 1.1.

A.2 Data Handling

All datasets were sourced from HuggingFace Datasets (Lhoest et al., 2021). For training sets exceeding 5,000 samples, we used a random subset of 5,000 instances to standardize the training workload.

A.3 Dataset Licenses

All the datasets were downloaded from HuggingFace(Lhoest et al., 2021) and under Apache License, Version 2.0.

A.4 Machine Configuration

The experiments were conducted on machines equipped with 4 and 8 Nvidia V100 32G.