FedMerge: Federated Model Merging for Personalization

Shutong Chen[1], Tianyi Zhou[2], Guodong Long[1*], Jing Jiang[1], Chengqi Zhang[3]
[1]University of Technology Sydney, Australia
[2]University of Maryland, College Park, USA
[3]The Hong Kong Polytechnic University, China
shutong.chen@student.uts.edu.au, tianyi@umd.edu, guodong.long@uts.edu.au, jing.jiang@uts.edu.au, chengqi.zhang@polyu.edu.hk

Abstract

One global model in federated learning (FL) might not be sufficient to serve many clients with non-IID tasks and distributions. Despite recent advances in FL to train multiple global models for better personalization, they only provide limited model choices to clients, so local finetuning of multiple models is still indispensable. This paper proposes a novel "FedMerge" approach that can create a single personalized model per client by simply merging multiple global models with automatically optimized and customized weights. We formulate this problem as a joint optimization of global models and the merging weights per client. Unlike existing FL approaches, where the server broadcasts one or multiple global models to all clients, the server only needs to send a customized, merged model to each client. Moreover, instead of periodically interrupting the local training and re-initializing it to a global model, the merged model aligns better with each client's task and data distribution, smoothing the local-global gap between consecutive rounds caused by client drift. We evaluate FedMerge on different non-IID settings applied to various domains with diverse tasks and data types, in which FedMerge consistently outperforms existing FL approaches, including clustering-based and mixture-of-experts (MoE) based methods.

Code — https://github.com/shutong043/FedMerge
Extended version — https://arxiv.org/abs/2504.06768

Introduction

Federated learning (FL) enables decentralized and collaborative learning of models across various clients without sharing their local data. By exchanging only local models instead of raw data, FL ensures data privacy and security while leveraging distributed datasets for improved performance. In particular, local datasets across clients are often non-identically and independently distributed (non-IID), making the design of algorithms to address such non-IIDness a fundamental challenge.

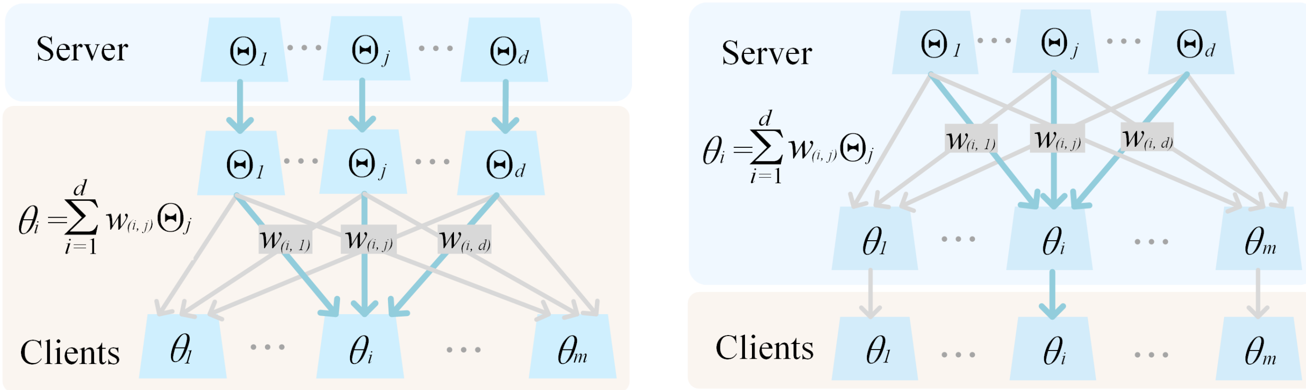
Model merging (Yadav et al. 2024) is a knowledge ensemble technique that integrates multiple models into a single unified model. It has recently gained increasing attention in the context of foundation models (Huang et al. 2023; Chronopoulou et al. 2023; Yang et al. 2024a; Kim, Shin, and Kim 2025; Wang et al. 2025), as it enables efficient knowledge integration across diverse domains. Instead of training a large model from scratch, model merging combines existing pre-trained models into an enriched knowledge space, allowing effective reuse of previously trained experts. This flexibility naturally extends to parameter-efficient tuning methods, such as Low-Rank Adaptation (LoRA), which can be merged with minimal computational overhead. Due to these efficiency advantages, LoRA has been widely explored in FL settings (Yi et al. 2023; Yang et al. 2024b; Sun et al. 2024; Cho et al. 2024). These properties make model merging a promising approach for scaling and personalizing federated learning.

Federated learning with multiple models is an important direction for addressing client heterogeneity (Marfoq et al. 2021; Wu et al. 2023), and a recent dominant approach in this line of work is Mixture-of-Experts (MoE) (Luo, Chen, and Wu 2024; Feng et al. 2025; Jiang et al. 2025; Radwan et al. 2025). In these MoE-like FL approaches, the server maintains a "model soup" of diverse experts that are shared across clients. As illustrated in Fig. 1a, each client downloads multiple (or all) expert models and combines them locally to form a personalized model. However, sharing multiple models across clients introduces an intrinsic conflict: the desire for a rich, diverse set of global models conflicts with the limited computational and communication resources of individual clients. The cost for each client scales with the number of models it downloads, making it impractical to leverage a large model soup. While selecting a subset of models for each client can alleviate this issue (Chronopoulou et al. 2023), it often requires task-specific knowledge, which limits its generalization ability.

Model merging provides an elegant solution to the scalability issues faced by MoE-like FL methods. By integrating multiple expert models into a single personalized model, model merging allows each client to benefit from the richness and diversity of the model soup without having to download or train multiple models locally. This fundamentally eliminates the linear growth of communication and computation costs on the client side, making it feasible to leverage a large number of global models even in resource-constrained environments.

Building on this insight, we propose **Federated Learning with Model Merging (FedMerge)**. As shown in Fig. 1b, the server merges global models using client-specific weights to

*Corresponding author.
Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.



(a) The architecture of **Federated Learning with Mixture of Experts**. Each client receives d global models from the server and optimizes them separately. The client's resource consumption is proportional to the number of global models, making it inefficient.

(b) The architecture of the proposed **Federated Learning with Model Merging (FedMerge)**. The merging operation (weight averaging) is performed on the server side instead of on each client. Only the merged models, θ, are communicated and optimized, making the cost irrelevant to the number of global models.

Figure 1: Comparison between MoE-like FL methods and the proposed FedMerge. Both methods perform weight averaging over global models. MoE-like FL performs weight averaging on each client, while FedMerge performs weight averaging on the server.

construct a unique personalized model for each client. Clients then only need to communicate and optimize a single merged model, while the server handles the merging operation with its typically greater resources. A challenge introduced by this design is how to update both the global models and the merging weights when only merged models are trained on clients. To address this, we formulate FedMerge as a joint optimization problem and derive a backpropagation-like update strategy to enable end-to-end training for both components. Experimental results on various non-IID settings show that FedMerge consistently outperforms clustering-based and MoE-based FL approaches.

We summarize the main contributions of this paper:

- We propose a novel FL framework to solve the conflict between using multiple global models for personalization and the limited resources of clients.
- We introduce the Federated Merging (FedMerge) algorithm, where the server provides a personalized model to each client by merging a set of global models with client-specific weights.
- Convergence analysis is provided in the Appendix E.
- Experimental results demonstrate that FedMerge offers a cost-effective way to improve performance by supporting multiple global models while maintaining a constant and low client-side cost.

Related Work

Model Merging Model merging refers to methods aimed at reusing the knowledge contained in multiple models. The fundamental technique of model merging is to integrate existing models to form new ones. There have been a number of works focusing on merging methodologies, such as simple averaging (Wortsman et al. 2022), merging with Fisher Information (Matena and Raffel 2022), and merging with task vectors (Ilharco et al. 2022). Recently, model merging has become closely related to the development of foundation models (Lu et al. 2023, 2024; Ostapenko et al. 2024; Zhou et al. 2025; Cheng et al. 2025; Du et al. 2025), as such developments have brought numerous reusable expert foundation models online (Face 2023). For example, LoraHub (Huang et al. 2023) trains one LoRA expert per task on a collection of 200 tasks and combines these experts to evaluate downstream tasks. Due to the large number of expert models, model selection plays a practical role in choosing the most relevant experts for each downstream task (Chronopoulou et al. 2023; Zhao et al. 2024). Model merging is also related to Mixture of Experts (MoE)-based methods, where the design of routing weights is a key factor. The router can be neural network-based (Lu et al. 2023), feature similarity-based (Wang et al. 2024), or even SVD-decomposed (Ostapenko et al. 2024). In this paper, rather than directly adopting the various model merging techniques mentioned above in FL, we leverage the fundamental concept of model merging to address the primary challenge in Federated Learning with Multiple Models, enabling a large model soup to be efficiently reused on the server while communicating only a single model to each client.

Federated Learning with Multiple Models The use of multiple models in FL arises from the fact that sharing a single model cannot effectively address the Non-IID problem. Personalized FL methods (T Dinh, Tran, and Nguyen 2020; Li et al. 2021; Collins et al. 2021) apply the concept of multiple models by designing a unique model for each client's personalized data distribution. Our method is more related to a type of multi-model FL, where multiple models are shared on the server (Marfoq et al. 2021; Wu et al. 2023). A recent line of research explores mixture-of-experts (MoE) architectures in FL (Luo, Chen, and Wu 2024; Feng et al. 2025; Jiang et al. 2025; Radwan et al. 2025; Xie et al. 2025), where multiple experts and personalized gating mechanisms are introduced to enhance personalization and handle strong data heterogeneity. These approaches either deploy MoE models on the server to improve scalability or distribute experts across clients for decentralized adaptation, enabling dynamic expert selection and robust personalization across domains. For example, pFedMoE (Yi et al. 2024) employs one local expert and one global expert for each client, while deriving routing weights using personalized router neural networks. Recently, a few methods have incorporated merging concepts into FL (Chen et al. 2024, 2025). In (Tao et al. 2024; Liu et al. 2024), a one-shot federated learning scenario is considered, and model merging methods can be naturally extended to address the challenges rise in this setting. (Salami et al. 2024) proposed an end-to-end optimization framework that leverages RegMean (Jin et al. 2022) to constrain the update distribution of LoRA adapters. While these approaches also utilize model merging to improve federated learning, our work targets a different challenge within this paradigm: balancing the scalability of a large pool of global models with the limited communication and computation capacities of individual clients.

Backgrounds and Motivation

Preliminaries

Federated Learning (FL) enables multiple clients to collaboratively train a model under the coordination of a central server without sharing their private data. Let (X_i, Y_i) denote the data of client i. In standard FL, the goal is to train a single global model Θ that minimizes the weighted sum of local objectives across all m clients:

$$\min_{\Theta} \sum_{i=1}^{m} \frac{n_i}{n} \ell\left(Y_i, f(X_i; \Theta)\right) \tag{1}$$

where n_i and n denote the data sizes of client i and all clients, respectively. This formulation, known as FedAvg (McMahan et al. 2017), aggregates locally trained models to form a unified global model. However, under non-IID data distributions, a single global model often fails to generalize well to each client's local domain.

In parallel, the concept of a model soup (Wortsman et al. 2022) refers to a collection of trained models whose parameters are combined—typically via simple averaging or linear interpolation—to improve generalization performance. The core idea is that interpolating between diverse models trained from different optimization trajectories can reveal complementary information, thereby enhancing both robustness and overall generalization.

Problem Setting

To address the limitations of FedAvg under heterogeneous data, personalized federated learning (PFL) allows each client to maintain a customized model θ_i, optimized with respect to its local data:

$$\min_{\theta} \sum_{i=1}^{m} \frac{n_i}{n} \ell\left(Y_i, f(X_i; \theta_i)\right) \tag{2}$$

where each θ_i learns client-specific knowledge while still being related to some shared global representation.

Motivated by the insight from model soup, we consider how this notion can be adapted to the PFL setting, where the server can maintain a shared model soup composed of models aggregated or collected from different clients. This shared model soup enables model construction that balance personalization and generalization.

Methodology

Objective Function

We place a model soup containing multiple models on the server. Let $\{\Theta_1, \Theta_2, \ldots, \Theta_d\}$ denote a set of d global models, where each model is trained with respect to its own knowledge space. In federated learning (FL), each client is associated with a personalized task, and we assume that the knowledge required to solve each task lies within the set of global models maintained on the server, that is, within the shared model soup.

Therefore, the server can create a personalized model for each client by reusing the global models in the model soup. In this work, we adopt standard weight merging, where the global models are combined with task-specific merging weights to generate a personalized model for each client:

$$\theta_i = \sum_{j=1}^{d} w_{(i,j)} \cdot \Theta_j,$$

where $w_{i,1:d}$ represents the personalized merging weights for client i, determining the contribution of each global model to the merged model.

After this merging operation on the server, the personalized merged models are sent to the clients. The objective of FedMerge is formulated as:

$$\min_{\theta} \sum_{i=1}^{m} \frac{n_i}{n} \ell\left(Y_i, f(X_i; \theta_i)\right) \tag{3}$$

$$\text{s.t.} \quad \theta_i = \sum_{j=1}^{d} w_{(i,j)} \cdot \Theta_j. \tag{4}$$

Note that in Eq (4), each client is optimized only on its merged model θ_i, rather than directly on the global models or the merging weights. This is a key difference between FedMerge and MoE-like FL methods.

Optimizing via Backpropagation-like Strategy

Solving Eq (3) with Eq (4) requires optimizing w and Θ. In the proposed FedMerge, each client only receives the merged model, and only the gradient of the merged model $\frac{\partial \ell}{\partial \theta}$ is directly available. The weights w and global models Θ remain on the server, and their gradients cannot be computed directly.

Inspired by the mechanism of backpropagation in neural networks (Rumelhart, Hinton, and Williams 1986), we propose an update rule for merging weights w and global models Θ. If we consider FedMerge as a single-layer fully connected network, where local models θ are high-level nodes and global models Θ are low-level nodes, then w acts as the propagation weights. The gradient of w and Θ depends on

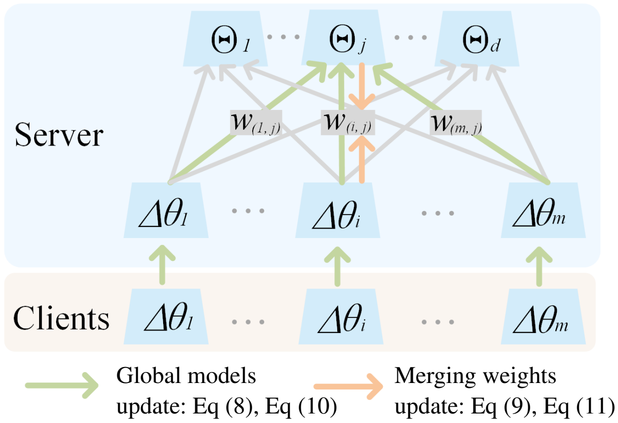


Figure 2: The update information flow of global models and merging weights in FedMerge. Global models are updated from all clients, and merging weights are updated from their associated model pairs.

Algorithm 1 Federated Merging (FedMerge)

```
Initialize: global models {Θ_1, Θ_2, ..., Θ_d}, merging
weights w ∈ ℝ^{m×d}
for each communication round do
    Randomly select a subset A of clients
    for each client i ∈ A do
        Merge models using Eq (4) to obtain θ_i
        Send merged model θ_i to client i
    end for
    for all selected clients (in parallel) do
        Update θ_i using local gradient descent via Eq (7)
        for t steps
        Send local update Δθ_i to the server
    end for
    Compute gradients for Θ via Eq (8)
    Update Θ via Eq (10)
    Compute gradients for w via Eq (9)
    Update w via Eq (11)
end for
```

the gradient of θ. Following the chain rule, we compute $\frac{\partial L}{\partial w}$ and $\frac{\partial L}{\partial \Theta}$ using $\frac{\partial L}{\partial \theta}$ as an intermediate variable:

$$\frac{\partial L}{\partial \Theta_j} = \sum_{i=1}^{m} \frac{n_i}{n} w_{(i,j)} \frac{\partial \ell}{\partial \theta_i}. \tag{5}$$

$$\frac{\partial L}{\partial w_{(i,j)}} = \frac{n_i}{n} \langle \Theta_j, \frac{\partial \ell}{\partial \theta_i} \rangle. \tag{6}$$

Eq (5) and Eq (6) are derived directly from the objective of FedMerge in Eq (3) with Eq (4). The key factor is the use of local gradient $\frac{\partial L}{\partial \theta}$ as an intermediate variable. For the detailed derivation process, please refer to Appendix C.

To better illustrate Eq (5) and Eq (6), Fig. 2 highlights the information flows from local models to global models and merging weights. It shows that the gradient of each global model is influenced by all client updates, while the updates for each merging weight depend on both the global model and the merged model it connects.

We summarize the training procedure as follows:

1. Forward Propagation: The server computes the merged models for each client using Eq (4) and distributes them accordingly. Each client then updates its merged model based on its local objective function:

$$\theta_i^{(t)} - \theta_i^{(t-1)} \leftarrow -\eta \nabla_{\theta_i^{(t-1)}} \ell(Y_i, f(X_i; \theta_i^{(t-1)})), \tag{7}$$

where η is the learning rate for θ. Notably, the local update procedure in FedMerge is the same as in FedAvg.

2. Backward Propagation: After the local update, each client transmits its updated parameters $\Delta\theta_i = \theta_i^{(t)} - \theta_i^{(0)}$ back to the server. The server then updates both the global models Θ and the merging weights w. Using the chain rule in Eq (5) and Eq (6), the updates for the merging weights and global models are computed based on local updates:

$$\Delta\Theta_j = \sum_{i=1}^{m} \frac{n_i}{n} w_{(i,j)} \Delta\theta_i, \tag{8}$$

$$\Delta w_{(i,j)} = \frac{n_i}{n} \langle \Theta_j, \Delta\theta_i \rangle. \tag{9}$$

3. Parameter Update: Once the gradients for the global models and merging weights are obtained, the server updates Θ and w:

$$\Theta_j \leftarrow \Theta_j + \Delta\Theta_j, \tag{10}$$

$$w_{(i,j)} \leftarrow w_{(i,j)} + \Delta w_{(i,j)}. \tag{11}$$

The overall FedMerge process is summarized in Algorithm 1. We also provide detailed convergence analysis, please refer to Appendix E for details.

Technical Implementations

Constraint on Merging Weights Values The original objective of FedMerge in Eq (4) has no constraints on merging weights, allowing the merging weight values to be negative. However, neural network models typically require regularized parameters. To address this, we apply a softmax function to the merging weights to obtain normalized weights. This introduces slight modifications to the updates in Eq (8) and Eq (9). Detailed derivations can be found in Appendix D.

Clarification of Merging Weights Gradient In Eq (9), the update of merging weights involves the inner product of model parameters and gradients. For large models, the effective update directions can be overwhelmed by the inner products of a large number of parameters. To mitigate this issue, we simply use the parameters from the last few layers of the model for inner product—specifically, the classification head in convolutional neural networks. This is because higher-level layers capture more semantic information (Yosinski et al. 2014), and their parameters or gradients are correlated with task heterogeneity across clients.

Classic FL (ResNet-based)							
Multi-Model	**Model Num.**	ResNet-9×5	ResNet-9×10	ResNet-9×15	ResNet-9×20	ResNet-9×25	ResNet-9×30
	Parameter Scale	5	10	15	20	25	30
Single-Model	**Model Arch.**	ResNet-9	ResNet-18	ResNet-34	ResNet-50	ResNet-101	ResNet-152
	Parameter Scale	1	2.27	4.32	4.82	8.73	11.95
Foundation Model FL (LoRA-based)							
Multi-Model	**Description**	Rank-8×2	Rank-8×4	Rank-8×6	Rank-8×8	Rank-8×10	Rank-8×12
	Parameter Scale	2	4	6	8	10	12
Single-Model	**Description**	Rank-8	Rank-16	Rank-24	Rank-32		
	Parameter Scale	1	2	3	4		

Table 1: Parameter scale settings for classic FL (top) and foundation model FL (bottom). The scale is measured relative to Rank-8 for LoRA modules and to ResNet-9 for ResNet architectures.

Server-side Computational Complexity

FedMerge is as computational efficient on the server side as standard baselines Server-side model assembly may raise concerns about computational overhead. We compare the server-side complexity of FedMerge with widely adopted baselines. As in Eq (4), Eq (8) and Eq (9), FedMerge performs weight averaging d times over m models or computes md inner products, resulting in a complexity of $\mathcal{O}(md)$. Cluster FL (Ghosh et al. 2020; Long et al. 2023) and Multi-model FL (Marfoq et al. 2021; Bhuyan and Moharir 2022)—both widely accepted in the literature—also require performing weight averaging d times over m models on the server side, resulting in a complexity of $\mathcal{O}(md)$. Thus, FedMerge does not introduce additional computational complexity on the server side compared to these standard approaches.

Experiments

Classical FL Settings

Baselines FedMerge relates to Cluster FL, Multi-model FL, and MoE-like FL. We compare it with the following representative methods: 1) **Single-model** methods share a single global model, including Local (training local models separately on each client), FedAvg (McMahan et al. 2017), FedAvg+ (FedAvg with local fine-tuning), and pFedMoE (Yi et al. 2024). Although named "MoE," pFedMoE uses only one global model. 2) **Multi-model** methods maintain multiple global models, such as IFCA (Ghosh et al. 2020) (Cluster FL) and FedEM (Marfoq et al. 2021) (MoE-like FL).

Parameter Scale To ensure fair comparison, we control the parameter scale for each method. Multi-model methods (IFCA, FedEM, FedMerge) use ResNet-9 with varying numbers of global models from {5, 10, 15, 20, 25, 30}. Single-model methods (Local, FedAvg, FedAvg+, pFedMoE) adopt different ResNet variants: ResNet-9, ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, whose parameter scales relative to ResNet-9 are {1, 2.27, 4.32, 4.82, 8.73, 11.95}. Full settings are shown in Table 1.

FedMerge Settings Global models are randomly initialized and trained from scratch. Merging weights are initialized uniformly to reflect equal initial contributions from global models.

System Settings These are the universal settings for all methods across all non-IID settings. Each dataset is split into training, validation, and test sets, with the best checkpoint selected based on validation error. We use a constant learning rate of 0.01, local training for 2 epochs, and 500 communication rounds. Each method is executed three times, and the reported results are the average of these runs.

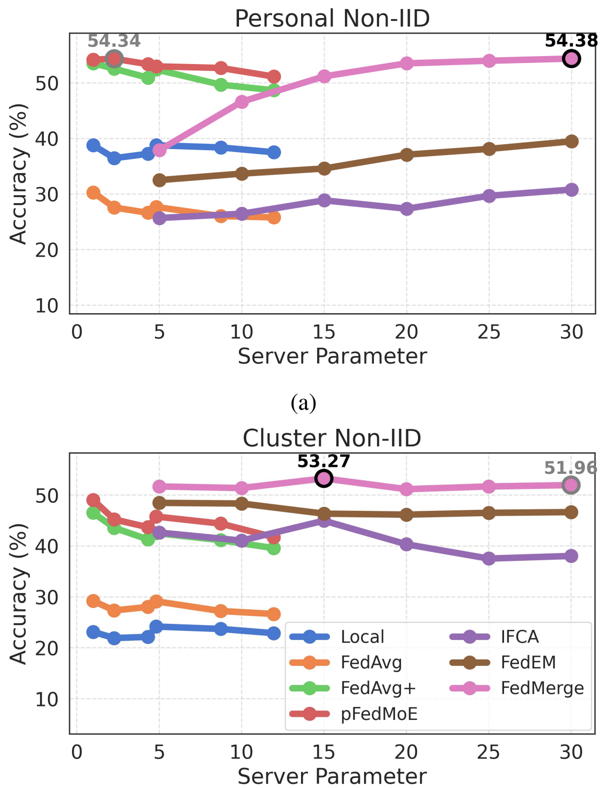
Parameter Usage vs. Performance

FedMerge adapts to different Non-IID settings by adjusting the merging weights adaptively. To validate this adaptability, we consider two data distributions of different non-IID structures:

Non-IID Settings We build two non-IID settings using the CIFAR-100 (Krizhevsky, Hinton et al. 2009) dataset: **Cluster Non-IID**: We split the original CIFAR-100 dataset into 5 clusters based on label categories. Each cluster is evenly assigned to a subgroup of clients out of 50 clients. The ground-truth clustering is $\{\{1, \ldots, 6\}, \{7, \ldots, 11\}, \{12, \ldots, 19\}, \{20, \ldots, 32\}\}, \{33, \ldots, 50\}\}$. **Personal Non-IID**: A commonly used Non-IID setting proposed by (Hsu, Qi, and Brown 2019), where the CIFAR-100 dataset is partitioned using a Dirichlet distribution into 50 clients. We set the concentration parameter $\alpha = 0.1$ for high Non-IIDness.

Following Table 1, we vary Single-model methods by model architecture and vary Multi-model methods by number of models, which results in various server parameter usage. As shown in Fig. 3, we plot the server parameter usage vs. accuracy on Personal Non-IID and Cluster Non-IID settings. The results suggest the following:

Multi-model FL methods provide an alternative way to improve performance beyond scaling model architectures. In Fig. 3a, using larger ResNet architectures for Single-model methods leads to a decrease in performance, while using more ResNet-9 models for Multi-model methods can boost the performance (with the largest architecture considered being ResNet-512 for Single-model methods, and up to 30 models used for Multi-model methods). This is because both Personal and Cluster Non-IID settings involve 50 clients with sparse client data and high non-IID data partition settings, which limits performance gains from scaling up the model. In contrast, Multi-model methods can enhance the base model's ability to handle complex non-IID distributions by increasing the number of models. FedMerge consistently surpasses FedEM and IFCA across various global model numbers, demonstrating the effectiveness of FedMerge com-



(a)

(b)

Figure 3: Server parameter usage vs. accuracy. The x-axis represents multiples of ResNet-9's parameter count. For each method, the variation of server parameters follows Table 1. The highest and second-highest accuracy points are marked with black and gray respectively.

pared to other Multi-Model FL methods.

FedMerge boosts performance by shifting computation to the server, leveraging the server's abundant resources to accommodate the limited client-side capacity. In Fig. 3, FedMerge manages a number of ResNet-9 model on the server while transmitting and optimizing a single model per client. Notably, on Personal Non-IID setting, FedMerge outperforms pFedMoE under the Personal Non-IID setting $(54.38 > 54.34)$ with only one ResNet-9 per client, while pFedMoE needs two ResNet-18 models and additional router weights per client to achieve optimal performance. On Cluster Non-IID, FedMerge consistently surpasses other baselines and achieves optimal performance with 15 global models.

We also provide Non-IID settings with PACS (Li et al. 2017) dataset, where using larger ResNet architectures can boost performance. Additional results are in Appendix A.

Ablation Study

FedMerge adapts to Non-IID settings with dynamic merging weights: To understand the behavior of FedMerge under different Non-IID settings, we visualized the heatmaps of merging weight matrices in Fig. 4. Each row represents the

Non-IID	FedAvg	FedMerge (15 global models)			
		Fixed (1/15)	Fixed (5/15)	Fixed (10/15)	Dynamic
Personal	29.20	40.94	44.70	45.48	**51.18**
Cluster	30.29	32.55	32.52	34.06	**53.27**

Table 2: Ablation study on merging weights. Fixed indicates that each client randomly selects some global models, and the corresponding merging weight is uniformly initialized and frozen during training.

normalized merging weights for a single client. Initially, the merging weight matrix is uniformly initialized, and as the epoch progresses, it begins to exhibit specific distribution patterns. We separate the ground-truth clusters in Cluster Non-IID with white dashed lines. In Cluster Non-IID, rows of merging weights are similar within each ground-truth cluster and distinct between different clusters. This indicates that FedMerge automatically allows clients with similar data distributions to prefer similar subgroups of global models, achieving a soft clustering. In contrast, for Personal Non-IID, rows of merging weights are relatively distinct from one another, and the merging weight matrix does not exhibit a regular structure.

Effect of different weight merging strategies: We further analyze different weight merging strategies in Table 2. As observed, The performance of Fixed strategy decreases compared with dynamic merging, and Cluster FL decreases more than Personal Non-IID because random selection conflicts with the structural information of the cluster.

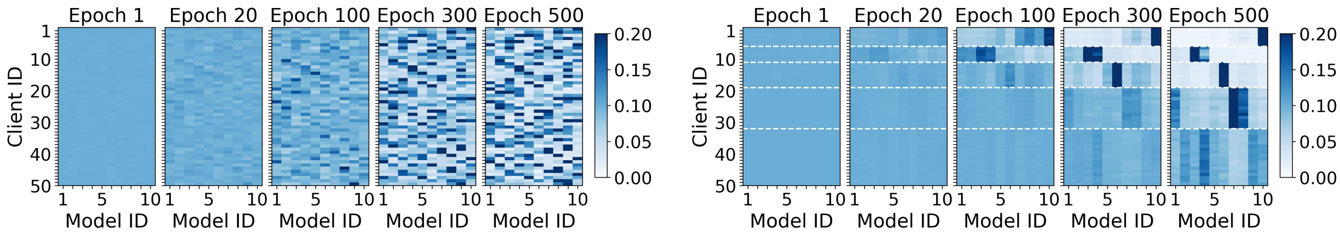
Federated Finetuning of Large Language Models

Baselines We use the same baselines as in classic FL settings, except that FedAvg is replaced with FedIT (Zhang et al. 2024) and FedAvg+ is replaced with FedLoRA (Yi et al. 2023). For pFedMoE and FedEM, we build personalized routers for the LoRA modules. Details can be found in Appendix B.2.

Datasets Several baselines (Ye et al. 2024b; Kuang et al. 2024; Ye et al. 2024a) exist for federated foundation models. However, these models are primarily designed for instruction fine-tuning, and their goal is to learn a global model that follows general instructions rather than to personalize the LLM's behavior for each client. In this paper, we consider two datasets for personalized LLM: **Task-heterogeneous instruction following:** Following (Yang et al. 2024b), we randomly select 8 NLP tasks from Flan (Wei et al. 2021). One task per client, resulting in 8 total clients. Each task randomly selects 300 training samples and 200 test samples. ROUGE-1 is used as the evaluation metric. **Persona-aligned instruction following:** We also provide a more realistic scenario with RoleBench (Wang et al. 2023), where each client takes on a specific role and has a Q&A dataset with clear role characteristics. Details can be found in Appendix B.1.

System Settings We adapt Alpaca-LoRA[1] as the base model, initialized with LLaMA3-8B (Dubey et al. 2024). The rank of LoRA is set to 8. We employ a linear learning rate decay schedule from $5e{-5}$ to $5e{-6}$ over the training process.

[1] https://github.com/tloen/alpaca-lora



(a) Heat map of merge weights for Personal Non-IID.

(b) Heat map of merge weights for Cluster Non-IID.

Figure 4: Visualization of merge weights with 10 global models and 50 clients. Each row denotes one normalized weight vector for each client. In Cluster Non-IID, we separate ground-truth clusters with white dashed lines.

Method	Paraphrase	Entailment	Structure to text	Text formatting	Linguistic acceptability	Word disambiguation	Coreference	Question classification	**Average**
FedEM	73.40	74.30	**69.32**	**87.93**	78.80	**61.31**	66.82	78.33	73.77
FedMerge	**75.49**	**77.02**	68.54	86.97	**80.64**	59.70	**70.61**	**80.03**	**74.86**

Table 3: ROUGE-1 scores (%) of MoE-like method (FedEM) and model merging (FedMerge) in task-heterogeneous instruction following. Results are reported across 8 instruction-following tasks from Flan, with the average shown in the last column.

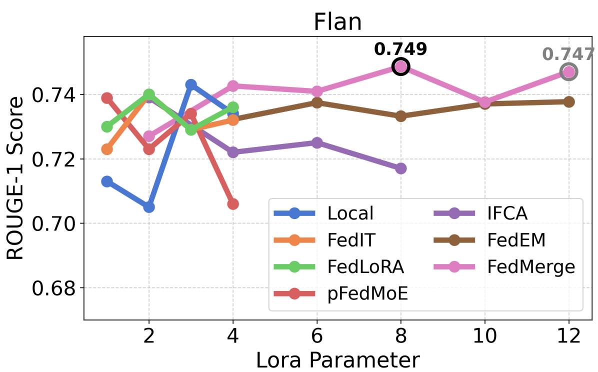


Figure 5: Server parameter usage vs. accuracy on the federated Flan dataset. The values on the x-axis are shown as multiples of the parameter count of a Rank-8 LoRA. The highest and second-highest accuracy points are marked with black and gray respectively.

The number of communication rounds is set to 50 to ensure convergence. For each round, the local training is conducted with 20 steps, and the batch size is set to 4.

Parameter Scale Similar to classic FL settings, we vary the number of LoRA modules for Multi-model methods and vary the rank of LoRA modules for Single-model methods. Details are shown in Table 1. For cluster FL methods like IFCA, the number of global models is limited to at most 8, equal to the number of clients. All experiments are conducted on a single NVIDIA A40 GPU.

Following Table 1, we vary Single-model methods by the LoRA rank and Multi-model methods by the number of LoRA modules. The server parameter usage vs. accuracy for the federated Flan dataset is shown in Fig. 5. FedMerge demonstrates better instruction-following abilities on the federated Flan dataset by sharing up to 12 LoRA modules globally while only transmitting one LoRA module of rank 8 per client. Results for the federated RoleBench dataset can be found in Appendix B.3.

Model Merging (FedMerge) provides a valuable alternative to MoE-like methods (FedEM) for FL with foundation models while being more client-resource friendly. Unlike classic settings, MoE-like methods are widely used in foundation models. A typical setup for a Mixture-of-Experts (MoE) foundation model in a federated learning scenario is that each client hosts several experts and a local router. These experts are sent to the server and aggregated separately to form a pool of global experts. While being simple in optimization strategy, it is not as client-resource-efficient as FedMerge. A detailed comparison between FedEM and FedMerge is presented in Table 3. We select the best performance for FedEM and FedMerge within the rank settings in Table 1, which is 12 shared LoRA modules for FedEM and 8 for FedMerge. From Table 3, FedMerge and FedEM outperform each other on different tasks, while FedMerge is better in average performance across clients.

Conclusion

In this paper, we propose an end-to-end Model Merging framework for Federated Learning, where merging weights and global models are optimized jointly from scratch. The server manages all global models and merging weights, customizing personalized models for each client by weight-averaging the global models. FedMerge can boost performance with a large model soup on the server, while still being client-resource friendly by only communicating and optimizing a single model for each client. FedMerge offers a promising direction for Federated Foundation Models: a large server knowledge pool can match foundation model scales, while each client selects and combines a small subset of parameters without transmitting the full model.

References

Bhuyan, N.; and Moharir, S. 2022. Multi-model federated learning. In *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, 779–783. IEEE.

Chen, J.; Zhang, H.; Zhang, C.; Luo, F.; and Li, Y. 2025. Breaking the Aggregation Bottleneck in Federated Recommendation: A Personalized Model Merging Approach. *arXiv preprint arXiv:2508.12386*.

Chen, M.; Jiang, M.; Zhang, X.; Dou, Q.; Wang, Z.; and Li, X. 2024. Local Superior Soups: A Catalyst for Model Merging in Cross-Silo Federated Learning. *arXiv preprint arXiv:2410.23660*.

Cheng, R.; Xiong, F.; Wei, Y.; Zhu, W.; and Yuan, C. 2025. Whoever started the interference should end it: Guiding data-free model merging via task vectors. *arXiv preprint arXiv:2503.08099*.

Cho, Y. J.; Liu, L.; Xu, Z.; Fahrezi, A.; and Joshi, G. 2024. Heterogeneous lora for federated fine-tuning of on-device foundation models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 12903–12913.

Chronopoulou, A.; Peters, M. E.; Fraser, A.; and Dodge, J. 2023. Adaptersoup: Weight averaging to improve generalization of pretrained language models. *arXiv preprint arXiv:2302.07027*.

Collins, L.; Hassani, H.; Mokhtari, A.; and Shakkottai, S. 2021. Exploiting shared representations for personalized federated learning. In *International conference on machine learning*, 2089–2099. PMLR.

Du, Y.; Wang, X.; Chen, C.; Ye, J.; Wang, Y.; Li, P.; Yan, M.; Zhang, J.; Huang, F.; Sui, Z.; et al. 2025. Adamms: Model merging for heterogeneous multimodal large language models with unsupervised coefficient optimization. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 9413–9422.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Face, H. 2023. Hugging Face - The AI Community Building the Future. https://huggingface.co. Accessed: October 15, 2023.

Feng, Y.; Geng, Y.-a.; Zhu, Y.; Han, Z.; Yu, X.; Xue, K.; Luo, H.; Sun, M.; Zhang, G.; and Song, M. 2025. PM-MOE: Mixture of Experts on Private Model Parameters for Personalized Federated Learning. In *Proceedings of the ACM on Web Conference 2025*, 134–146.

Ghosh, A.; Chung, J.; Yin, D.; and Ramchandran, K. 2020. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33: 19586–19597.

Hsu, T.-M. H.; Qi, H.; and Brown, M. 2019. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*.

Huang, C.; Liu, Q.; Lin, B. Y.; Pang, T.; Du, C.; and Lin, M. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*.

Ilharco, G.; Ribeiro, M. T.; Wortsman, M.; Gururangan, S.; Schmidt, L.; Hajishirzi, H.; and Farhadi, A. 2022. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

Jiang, J.; Chen, Y.; Liu, X.; Jiang, H.; and Fan, C. 2025. Heterogeneous Federated Learning with Scalable Server Mixture-of-Experts. In Kwok, J., ed., *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, 5480–5488. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Jin, X.; Ren, X.; Preotiuc-Pietro, D.; and Cheng, P. 2022. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*.

Kim, K.; Shin, J.; and Kim, J. 2025. Personalized Language Models via Privacy-Preserving Evolutionary Model Merging. *arXiv preprint arXiv:2503.18008*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Kuang, W.; Qian, B.; Li, Z.; Chen, D.; Gao, D.; Pan, X.; Xie, Y.; Li, Y.; Ding, B.; and Zhou, J. 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*, 5260–5271.

Li, D.; Yang, Y.; Song, Y.-Z.; and Hospedales, T. M. 2017. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, 5542–5550.

Li, T.; Hu, S.; Beirami, A.; and Smith, V. 2021. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, 6357–6368. PMLR.

Liu, X.; Liu, L.; Ye, F.; Shen, Y.; Li, X.; Jiang, L.; and Li, J. 2024. Fedlpa: One-shot federated learning with layer-wise posterior aggregation. *Advances in Neural Information Processing Systems*, 37: 81510–81548.

Long, G.; Xie, M.; Shen, T.; Zhou, T.; Wang, X.; and Jiang, J. 2023. Multi-center federated learning: clients clustering for better personalization. *World Wide Web*, 26(1): 481–500.

Lu, K.; Yuan, H.; Lin, R.; Lin, J.; Yuan, Z.; Zhou, C.; and Zhou, J. 2023. Routing to the expert: Efficient reward-guided ensemble of large language models. *arXiv preprint arXiv:2311.08692*.

Lu, Z.; Fan, C.; Wei, W.; Qu, X.; Chen, D.; and Cheng, Y. 2024. Twin-merging: Dynamic integration of modular expertise in model merging. *arXiv preprint arXiv:2406.15479*.

Luo, J.; Chen, C.; and Wu, S. 2024. Mixture of experts made personalized: Federated prompt learning for vision-language models. *arXiv preprint arXiv:2410.10114*.

Marfoq, O.; Neglia, G.; Bellet, A.; Kameni, L.; and Vidal, R. 2021. Federated multi-task learning under a mixture of distributions. *Advances in Neural Information Processing Systems*, 34: 15434–15447.

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

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Ostapenko, O.; Su, Z.; Ponti, E. M.; Charlin, L.; Roux, N. L.; Pereira, M.; Caccia, L.; and Sordoni, A. 2024. Towards modular llms by building and reusing a library of loras. *arXiv preprint arXiv:2405.11157*.

Radwan, A.; Soliman, M.; Abdelaziz, O.; and Shehata, M. 2025. FedDG-MoE: Test-Time Mixture-of-Experts Fusion for Federated Domain Generalization. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 1811–1820.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *nature*, 323(6088): 533–536.

Salami, R.; Buzzega, P.; Mosconi, M.; Bonato, J.; Sabetta, L.; and Calderara, S. 2024. Closed-form merging of parameter-efficient modules for federated continual learning. *arXiv preprint arXiv:2410.17961*.

Sun, Y.; Li, Z.; Li, Y.; and Ding, B. 2024. Improving loRA in privacy-preserving federated learning. *arXiv preprint arXiv:2403.12313*.

T Dinh, C.; Tran, N.; and Nguyen, J. 2020. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33: 21394–21405.

Tao, Z. S.; Mason, I.; Kulkarni, S.; and Boix, X. 2024. Task arithmetic through the lens of one-shot federated learning. *arXiv preprint arXiv:2411.18607*.

Wang, H.; Ping, B.; Wang, S.; Han, X.; Chen, Y.; Liu, Z.; and Sun, M. 2024. LoRA-Flow: Dynamic LoRA Fusion for Large Language Models in Generative Tasks. *arXiv preprint arXiv:2402.11455*.

Wang, H.; Yang, B.; Yin, X.; and Gao, W. 2025. Never Start from Scratch: Expediting On-Device LLM Personalization via Explainable Model Selection. *arXiv preprint arXiv:2504.13938*.

Wang, Z. M.; Peng, Z.; Que, H.; Liu, J.; Zhou, W.; Wu, Y.; Guo, H.; Gan, R.; Ni, Z.; Yang, J.; et al. 2023. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. *arXiv preprint arXiv:2310.00746*.

Wei, J.; Bosma, M.; Zhao, V. Y.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Wortsman, M.; Ilharco, G.; Gadre, S. Y.; Roelofs, R.; Gontijo-Lopes, R.; Morcos, A. S.; Namkoong, H.; Farhadi, A.; Carmon, Y.; Kornblith, S.; et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, 23965–23998. PMLR.

Wu, Y.; Zhang, S.; Yu, W.; Liu, Y.; Gu, Q.; Zhou, D.; Chen, H.; and Cheng, W. 2023. Personalized federated learning under mixture of distributions. In *International Conference on Machine Learning*, 37860–37879. PMLR.

Xie, L.; Luan, T.; Cai, W.; Yan, G.; Chen, Z.; Xi, N.; Fang, Y.; Shen, Q.; Wu, Z.; and Yuan, J. 2025. dFLMoE: Decentralized Federated Learning via Mixture of Experts for Medical Data Analysis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 10203–10213.

Yadav, P.; Raffel, C.; Muqeeth, M.; Caccia, L.; Liu, H.; Chen, T.; Bansal, M.; Choshen, L.; and Sordoni, A. 2024. A survey on model moerging: Recycling and routing among specialized experts for collaborative learning. *arXiv preprint arXiv:2408.07057*.

Yang, E.; Shen, L.; Guo, G.; Wang, X.; Cao, X.; Zhang, J.; and Tao, D. 2024a. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*.

Yang, Y.; Long, G.; Shen, T.; Jiang, J.; and Blumenstein, M. 2024b. Dual-Personalizing Adapter for Federated Foundation Models. *arXiv preprint arXiv:2403.19211*.

Ye, R.; Ge, R.; Zhu, X.; Chai, J.; Yaxin, D.; Liu, Y.; Wang, Y.; and Chen, S. 2024a. Fedllm-bench: Realistic benchmarks for federated learning of large language models. *Advances in Neural Information Processing Systems*, 37: 111106–111130.

Ye, R.; Wang, W.; Chai, J.; Li, D.; Li, Z.; Xu, Y.; Du, Y.; Wang, Y.; and Chen, S. 2024b. 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*, 6137–6147.

Yi, L.; Yu, H.; Ren, C.; Zhang, H.; Wang, G.; Liu, X.; and Li, X. 2024. FedMoE: Data-Level Personalization with Mixture of Experts for Model-Heterogeneous Personalized Federated Learning. *arXiv preprint arXiv:2402.01350*.

Yi, L.; Yu, H.; Wang, G.; and Liu, X. 2023. Fedlora: Model-heterogeneous personalized federated learning with lora tuning. *arXiv preprint arXiv:2310.13283*.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27.

Zhang, J.; Vahidian, S.; Kuo, M.; Li, C.; Zhang, R.; Yu, T.; Wang, G.; and Chen, Y. 2024. Towards building the federatedGPT: Federated instruction tuning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6915–6919. IEEE.

Zhao, Z.; Gan, L.; Wang, G.; Hu, Y.; Shen, T.; Yang, H.; Kuang, K.; and Wu, F. 2024. Retrieval-augmented mixture of lora experts for uploadable machine learning. *arXiv preprint arXiv:2406.16989*.

Zhou, Y.; Karamanolakis, G.; Soto, V.; Rumshisky, A.; Kulkarni, M.; Huang, F.; Ai, W.; and Lu, J. 2025. MergeME: Model merging techniques for homogeneous and heterogeneous MoEs. *arXiv preprint arXiv:2502.00997*.