Exploring Sparse Fine-Tuning in Federated Learning under Data Heterogeneity Constraints

Anonymous CVPR submission

Paper ID *****

Abstract

We investigate the application of sparse fine-tuning methods, such as adapters and low-rank techniques, in Federated Learning (FL) environments under non-IID data settings. Our experiments show that sparse methods reduce communication and memory costs while maintaining competitive accuracy and adaptability in personalized and global FL scenarios.

1. Introduction (maatouk)

1.1. Motivation and Overview

The proliferation of smartphones, IoT devices, and edge computing platforms has led to an explosion in user-generated data, much of which is sensitive in nature. Traditional machine learning systems often rely on centralizing this data in cloud servers for training, which raises serious concerns regarding user privacy, data security, and legal compliance. Federated Learning (FL) has emerged as a promising paradigm to address these concerns by enabling collaborative model training across multiple decentralized clients such as mobile phones, wearable devices, or institutions without requiring raw data to leave the local environment. This decentralized approach preserves privacy and offers new opportunities for building intelligent systems that respect data sovereignty. However, despite its promise, deploying FL in real-world applications remains challenging due to the inherent system-level heterogeneity (e.g., varying compute, memory, and network capabilities) and statistical heterogeneity (e.g., non identically distributed data across clients). These factors complicate model training, degrade performance, and hinder convergence, making it imperative to develop methods that can adapt efficiently to the complexities of FL environments.

1.2. Core Challenges in Federated Learning

Despite its privacy-preserving advantages, Federated Learning presents several core challenges that limit its effectiveness in real-world scenarios. A primary obstacle is statistical heterogeneity: clients in FL typically possess non-independent and identically distributed (non-IID) data due to differences in user behavior, local environments, or institutional protocols. This non-uniformity results in divergent local updates, which can severely degrade global model convergence and generalization performance. In addition to statistical issues, resource constraints represent a significant barrier. Many FL participants—such as smartphones or edge devices—are limited in computational power, memory, and bandwidth. Consequently, standard fine-tuning or full-model update strategies incur high communication overhead and computation costs, which are unsustainable for large-scale or frequent participation. Finally, personalization needs are often overlooked in conventional FL approaches that aim to learn a single global model. However, a one-size-fits-all solution is suboptimal in settings where user-specific data patterns are critical for achieving high performance. Addressing these intertwined challenges requires methods that are both efficient and adaptive, capable of tailoring models to local client characteristics while minimizing overhead.

1.3. Gap in Current Approaches

Traditional fine-tuning methods in Federated Learning generally rely on updating the entire set of model parameters, a process that is both computationally intensive and communication-heavy. While effective in centralized settings, such full-model fine-tuning is ill-suited for FL environments, where clients often operate under strict resource limitations. To mitigate these constraints, recent research has explored sparse tuning techniques, such as updating only a subset of parameters (e.g., adapters, low-rank matrices, or selective layers), which have shown promise in centralized transfer learning. However, these methods remain largely underexplored within FL settings, especially in the presence of severe data heterogeneity and personalization demands. Most existing FL frameworks continue to prioritize global performance over personalized efficiency, lacking robust mechanisms to balance client-specific adaptation with communication and computational efficiency. This gap highlights the need for novel sparse tuning strategies that are explicitly designed to operate under realistic FL constraints while maintaining competitive performance.

1.4. Proposal and Objectives

This project proposes to **investigate sparse fine-tuning strategies** tailored to Federated Learning (FL) settings characterized by non-IID data distributions, limited computational resources, and the need for personalized models. By leveraging **parameter-efficient adaptation methods** such as selective layer tuning. The goal is to develop fine-tuning techniques that enable **client-specific model personalization** with minimal communication and computational overhead.

The core objectives of this work are:

- To **design and evaluate sparse fine-tuning approaches** that maintain or improve performance in heterogeneous FL environments.
- To **quantify the trade-offs** between communication cost, computational efficiency, and personalization quality.
- To **establish practical guidelines** for deploying these methods in real-world FL applications, especially on resource-constrained edge devices.

This direction directly addresses the limitations of standard FL fine-tuning practices and aims to contribute to a new class of **adaptive and scalable personalization techniques** for federated systems.

2. Related Work

2.1. Overview of Federated Learning

Federated Learning (FL) is a distributed machine learning paradigm that enables multiple clients to collaboratively train a shared global model without exchanging raw data [**?**]. In its canonical form, a central server orchestrates the training process by periodically aggregating local model updates received from a subset of clients and broadcasting the updated global model back to them. This setup promotes data privacy, as sensitive user data remains local, and complies with regulations such as GDPR. One of the most prominent FL algorithms is *Federated Averaging* (FedAvg), introduced by McMahan et al. [**?**], which reduces the number of communication rounds by averaging locally updated models instead of gradient updates, thereby improving communication efficiency by an order of magnitude compared to naive centralized approaches. FL has garnered increasing attention in privacy-sensitive domains such as healthcare, finance, and mobile computing, where data centralization is either impractical or legally restricted. However, its real-world applicability is hindered by several key challenges inherent to decentralized environments.

2.2. Challenges in Federated Learning

2.2.1 Statistical Heterogeneity

One of the fundamental challenges in FL arises from the non-identically distributed (non-IID) nature of client data. Unlike traditional machine learning settings that assume IID data, clients in FL often operate in diverse environments, leading to data distributions that vary significantly across users due to behavioral, demographic, or contextual factors. This statistical heterogeneity causes divergence in local model updates, which can impair convergence, reduce the stability of training, and degrade the generalization ability of the global model. To address this, methods such as FedProx introduce proximal terms that penalize local models from straying too far from the global model, thus stabilizing updates during aggregation [**?**].

2.2.2 System Heterogeneity

In addition to statistical disparities, FL systems also contend with heterogeneity in hardware, network bandwidth, and client availability. Many FL clients such as smartphones, edge sensors, or IoT devices possess limited computational power, storage, or energy resources. Furthermore, network conditions may be unreliable, and clients may participate asynchronously or intermittently in training rounds. These limitations introduce latency, increase dropout rates, and create imbalanced contributions to model updates, complicating the orchestration and scalability of FL deployments. Techniques such as partial participation and tiered architectures have been proposed to accommodate system-level diversity, but the problem remains an active area of research.

2.3. Sparse Fine-Tuning and Model Editing Techniques

Recent advances in sparse fine-tuning have demonstrated that updating only a small subset of model parameters can achieve transfer performance on par with full-model updates, while dramatically reducing communication and storage costs. In the federated setting, [**?**] introduce task-localized sparse fine-tuning, where each client updates a sparse mask over task-relevant parameters and applies gradient-based model editing to correct specific failure modes. This approach reduces the volume of transmitted updates and enables fine-grained control over which weights are modified, proving effective even under highly non-IID data distributions. However, beyond this work, there is limited exploration of other parameter-efficient techniques — such as low-rank adaptations (e.g., LoRA [**?**]) or adapter modules — in FL contexts. To fully assess the landscape of sparse methods in federated regimes, future studies should integrate these alternative adapters and compare their trade-offs in convergence rate, communication cost, and accuracy.

2.4. Personalization in Federated Learning

Personalization aims to tailor the global model to individual client distributions, addressing the one-size-fits-all limitation of standard FL. Approaches such as FedBN [?] decouple batch-normalization statistics to accommodate feature shifts across clients, yielding improved local accuracy without degrading global aggregation. Meta-learning formulations (e.g., MAML-based FL) have likewise been proposed to rapidly adapt a shared initialization to each client's data, though these are not covered in your provided list and could enrich this discussion [?]. Moreover, adaptive optimization methods — such as those in [?] — can implicitly personalize learning rates per client, bridging the gap between global aggregation and client-specific adaptation. Nonetheless, systematic comparisons between sparse fine-tuning and these personalization schemes remain lacking, motivating combined investigations of parameter-efficient personalization in federated scenarios.

2.5. Relevant Prior Work

Several surveys have synthesized the state of FL, charting its promises and pitfalls. [?] provide a comprehensive overview of FL systems, highlighting practical considerations in data privacy, communication protocols, and deployment challenges. [?] focus on computational heterogeneity, surveying techniques that adapt model size and compute to device capabilities. On the theory side, [?] summarize open problems in statistical and system heterogeneity, offering a taxonomy that frames our work within key research dimensions. Finally, fairness and privacy trade-offs — critical for real-world adoption — are examined by [?], who outline how privacy mechanisms (e.g., differential privacy) may interact with model utility, particularly under non-IID distributions.

2.6. Open Problems and Research Gaps

Despite these advances, several gaps remain. First, the interaction between sparse fine-tuning masks and formal privacy guarantees has not been studied: applying differential privacy noise to sparse updates may affect their support structure in non-trivial ways. Second, fairness across clients under sparse updates is unexplored; it is unclear whether reducing the parameter footprint exacerbates performance disparities for under-represented groups. Third, most current methods assume relatively mild heterogeneity; extreme non-IID settings — such as those in cross-silhouette recognition or highly personalized recommender systems — demand more robust sparse adaptation strategies. Finally, theoretical analyses of convergence rates and optimal mask selection remain nascent. Addressing these challenges will require interdisciplinary efforts, drawing on privacy theory, fairness in ML, and optimization under sparsity constraints.

3. Methodology

Try this outline instead, brainstormed it with Chatgpt, please double check all details from the code, this is just an outline for titles, the numbers and info might not be accurate

3.1. Dataset and Preprocessing

3.1.1 CIFAR-100 Overview

We use the CIFAR-100 dataset, which consists of 60,000 color images of size 32×32 pixels, distributed across 100 classes. Each class contains 600 images, split into 500 training samples and 100 test samples. All images are in RGB format. The dataset is commonly used for image classification tasks and is part of the `torchvision.datasets` module.

3.1.2 Train/Val/Test Splits

The CIFAR-100 dataset provides 50,000 training images and 10,000 test images, with each of the 100 classes represented by 500 training and 100 test samples. Since a separate validation set is not provided, we manually split the original training set into training (80%) and validation (20%) subsets. Stratified sampling is used during this split to preserve the original class distribution. The test set remains untouched and is used exclusively for final model evaluation.

3.1.3 Transformations

To enhance generalization and ensure compatibility with the ViT-S/16 backbone, training images are first resized from 32×32 to 224×224 using `RandomResizedCrop(224, scale=(0.5, 1.0))` and augmented with `RandomHorizontalFlip()`. Validation and test images are resized to 256 pixels on the shorter side using `Resize(256)`, then center-cropped to 224×224 with `CenterCrop(224)`. All images are converted to tensors and normalized using CIFAR-100-specific statistics: mean $(0.5071, 0.4865, 0.4409)$ and standard deviation $(0.2673, 0.2564, 0.2762)$, computed from the training set.

3.1.4 Seed Control

To ensure reproducibility, random seeds are fixed across all relevant libraries, including `random`, `numpy`, `torch`, and `torch.cuda` (if applicable). The seed value, specified in the configuration dictionary (`CFG['seed']`), is passed to `train_test_split()` for stratified dataset splitting. Additional seed-setting commands are included at the beginning of the script to fully stabilize results.

3.2. Model Architecture

3.2.1 Base Model: DINO ViT-S/16

We use a pretrained Vision Transformer (ViT-S/16) from the DINO framework as the encoder in all experiments.

3.2.2 Classifier Head and Fine-tuning

The original classification head is replaced with a custom trainable MLP. Layers 9, 10, and 11 of the backbone are unfrozen; all others remain frozen.

3.2.3 Training Setup

All models are trained using SGD with momentum. Loss is computed using the standard Cross-Entropy Loss.

3.3. Runtime Environment

All experiments are conducted on Google Colab (free tier) using a Tesla T4 GPU. Due to runtime limits, model checkpoints are periodically saved to enable resuming training. No runtime or throughput metrics are reported.

3.4. Centralized Training

3.4.1 Standard Centralized Training

The full training set is used to train the model in a centralized fashion. Validation accuracy is used to tune hyperparameters (learning rate, weight decay, etc.). The same test set is used across all experiments for final evaluation.

3.4.2 Centralized Model Editing

A second experiment is conducted in which the pretrained model is fine-tuned using sparse updates. Model editing is applied after base training, using gradient-based masking to modify only a subset of parameters. Details on specific techniques and calibration strategies are provided in Section 3.6.

3.5. Federated Learning

All federated experiments use $K = 100$ clients, coordinated by a central server. Each communication round involves: Sampling $C = 0.1 \times K = 10$ clients at random. Local training for J local epochs, followed by model aggregation. Aggregated models are validated after each round on the shared validation set. Note: In the IID experiment, J is fixed to 4. In the non-IID experiments, we explore $J \in \{4, 8, 16\}$ and $N_c \in \{1, 5, 10, 50\}$ (number of classes per client).

3.6. FedAvg Aggregation

The global model is updated each round by averaging the parameters of selected client models. Aggregation is performed using a vectorized averaging procedure over model parameters for efficiency. This implementation avoids manual state_dict loops and leverages PyTorch utilities to construct the global model from stacked parameter vectors.

3.7. Federated Training Scenarios

3.7.1 IID Partitioning

Each client receives an IID shard of the training dataset, with approximately uniform class distribution. Training proceeds with the FedAvg algorithm using the parameters described above.

Model Editing Variant An alternative configuration applies model editing to client models. Gradient updates are masked during training, updating only a subset of the model's parameters.

3.7.2 Non-IID Partitioning

Clients are assigned a fixed number Nc of unique classes. We explore Nc = 1, 5, 10, and 50 to simulate increasing levels of heterogeneity. The rest of the FedAvg setup (client sampling, communication, validation) remains unchanged.

Model Editing Variant We also evaluate a version of the non-IID setup with model editing enabled. The impact of editing under skewed data distributions is assessed in terms of test accuracy and convergence behavior.

3.8. Evaluation Protocol and Metrics

Validation Accuracy is computed after each training round using the shared validation set. Final Test Accuracy is measured on the 10,000-image CIFAR-100 test set after the last round. Metrics: Accuracy (%) Loss

4. Methodology ”Pedro”

This study investigates the application of sparse fine-tuning strategies in a federated learning context using the CIFAR-100 dataset and the DINO ViT-S/16 model. CIFAR-100 comprises 60,000 32×32 color images across 100 classes, with 50,000 images allocated for training and 10,000 for testing. For experimental consistency, the training set is further split into training and validation subsets (80%/20%) to monitor generalization during local training.

To simulate federated learning scenarios, the dataset is partitioned across multiple clients under both IID and non-IID settings. In the IID configuration, each client receives an equal and uniform sample of the overall data distribution.

In contrast, the non-IID setup uses a Dirichlet distribution to assign samples such that each client is biased towards a limited subset of classes. The number of classes per client in this setup is controlled by the N_c hyperparameter. Lower N_c values indicate higher statistical heterogeneity, thereby mimicking real-world data silos.

For the model architecture, we adopt the Vision Transformer backbone (ViT-S/16), pretrained using the DINO self-supervised learning framework. This model serves as the basis for both centralized and federated experiments. In the centralized baseline, the model is trained on the global dataset using standard stochastic gradient descent with momentum (SGDM). For the federated setting, we implement the Federated Averaging (FedAvg) algorithm, where each client performs multiple local training steps before sending model updates to a central server for aggregation.

To reduce communication overhead and adapt the model to local data characteristics, we introduce a sparse variant of SGDM, referred to as *SparseSGDM*. This optimizer applies a masking step to the gradient updates before transmission. The masking selects only a subset of parameters to be updated, significantly reducing the number of communicated gradients. The selection of this subset is based on various criteria, including magnitude-based pruning, sensitivity analysis, and Fisher Information Scores, the latter estimating the importance of parameters based on their impact on the loss landscape.

The optimizer's logic involves computing the full gradient, applying a mask to retain only the top-k most relevant elements, and optionally calibrating the remaining updates to preserve convergence. This process is encapsulated in a concise pseudocode, ensuring reproducibility and clarity. Moreover, the model editing step—applying sparse updates to the pretrained global model—is explored using different masking strategies. Calibration techniques, such as gradient rescaling and momentum adjustment, are employed to ensure stability and performance post-editing.

Experiments are conducted separately under IID and non-IID conditions to isolate the effect of data heterogeneity. In each case, we evaluate the performance of federated training with and without sparse updates, focusing on accuracy, convergence speed, and communication efficiency. These experiments provide insights into how sparsity and model editing interact under varying levels of distributional shift, revealing trade-offs between personalization and generalization in federated learning systems.

5. Experimental Setup

- Hardware and Resource Constraints
- Logging and Checkpointing
- Hyperparameter Tuning
- Federated Training Configuration (Rounds, Clients, Local Steps)

6. Results and Discussion

6.1. Centralized and Federated Baselines

- Performance of Centralized Model
- FedAvg under IID Conditions
- Comparative Analysis: Centralized vs Federated (IID)

6.2. Effect of Data Heterogeneity (Non-IID)

- Results Across Different Nc Values
- Impact of Local Training Steps ('J')
- Analysis of Client Drift and Convergence

6.3. Sparse Fine-Tuning in FL

- Accuracy vs Sparsity Trade-offs
- Effect of Calibration Rounds
- Comparison of Masking Strategies: Sensitivity vs Magnitude vs Random
- Robustness Under Heterogeneous Data

7. Conclusion and Future Work

- Summary of findings: Sparse tuning enables efficient, personalized FL.
- Limitations: Dataset diversity, long-term model drift.
- Future directions: Test-time personalization, hierarchical FL, cross-device adaptation.