박 사 학 위 논 문
Ph.D. Dissertation

통신 효율적인 연합학습

Communication-efficient Federated Learning

2025

라히미 모하마드마흐디 (Mohammadmahdi, Rahimi)

한 국 과 학 기 술 원
Korea Advanced Institute of Science and Technology

박 사 학 위 논 문

통신 효율적인 연합학습

2025

라히미 모하마드마흐디

한 국 과 학 기 술 원

전기및전자공학부

통신 효율적인 연합학습

라히미 모하마드마흐디

위 논문은 한국과학기술원 박사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2025년 05월 28일

심사위원장　문재균　(인)

심 사 위 원　정혜원　(인)

심 사 위 원　김준모　(인)

심 사 위 원　성영철　(인)

심 사 위 원　한동준　(인)

Communication-efficient Federated Learning

Rahimi Mohammadmahdi

Advisor: Jaekyun Moon

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering

Daejeon, Korea
May 28, 2025

Approved by

Jaekyun Moon
Professor of Electrical Engineering

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

DEE
20208244

라히미모하마드마흐디. 통신 효율적인 연합학습. 전기및전자공학부 . 2025년. 71+1 쪽. 지도교수: 문재균. (영문 논문)

Rahimi Mohammadmahdi. Communication-efficient Federated Learning. School of Electrical Engineering . 2025. 71+1 pages. Advisor: Jaekyun Moon. (Text in English)

초 록

딥러닝의 발전은 다양한 분야에 혁신을 가져왔으나, 이러한 모델을 실제 환경에 배포하기 위해서는 방대한 양의 데이터를 중앙에 집결시켜야 하는 문제점이 존재하며, 이는 개인 정보 보호와 확장성 측면에서 중대한 이슈를 야기한다. 연합학습(Federated Learning, FL)은 데이터 공유 없이 분산된 클라이언트 장치에서 직접 협력적으로 모델을 학습하여 이 문제를 해결할 수 있다. 그러나 FL의 실질적 활용은 여전히 상당한 통신 오버헤드, 비효율적인 계산, 클라이언트의 이질성 및 개인화 요구 수용의 어려움으로 인해 제한적이다. 본 논문은 이러한 장애 요인들을 세 가지 상호 연결된 연구로 체계적으로 해결한다. 먼저, Chapter 1에서는 전체 모델 파라미터 대신 간결한 유사도 기반의 피트니스 메트릭만을 교환하여 통신 부하를 획기적으로 감소시키는 진화전략 기반의 새로운 FL 기법인 EvoFed를 제안한다. 효율적 통신이라는 개념을 더욱 발전시켜, Chapter 2에서는 모델 전체 파라미터 공간을 저랭크(low-rank) 방식으로 분해하여 통신과 계산 자원을 동시에 최적화하는 모델 독립적 프로젝션 적응법(Model-Agnostic Projection Adaptation, MAPA)을 제안한다. 나아가 이질적인 클라이언트 환경에서 개인화된 성능 향상의 필요성을 인식하여, Chapter 3에서는 특이값 분해(SVD)를 활용해 클라이언트별 모델 업데이트를 정렬함으로써 클라이언트 드리프트를 효과적으로 완화하고 추가적 통신 부담 없이 개인화를 향상시키는 Principal-Aligned LoRA (PA-LoRA)를 개발한다.

이러한 혁신들을 통합한 본 논문의 방법론은 효율적이고 확장 가능하며 개인화된 연합학습의 실용적 프레임워크를 제공하여, 개인 정보가 민감하거나 자원이 제한된 환경에서도 광범위한 FL 배포를 가능하게 한다.

핵 심 낱 말 연합학습, 통신 효율성, 저랭크 적응, 모델 정렬

Abstract

Advances in deep learning have revolutionized numerous fields, yet deploying these models often requires aggregating massive datasets in a central location, raising critical privacy and scalability concerns. Federated Learning (FL) addresses this by enabling collaborative model training directly on distributed client devices without sharing private data. However, practical adoption of FL remains constrained by substantial communication overhead, computational inefficiencies, and difficulties in accommodating client heterogeneity and personalization. This thesis systematically addresses these barriers through three interconnected contributions. First, in Chapter 1, we introduce EvoFed, a novel evolutionary-strategy-based FL method that drastically reduces communication by exchanging compact fitness-based similarity metrics instead of full model parameters. Building on this concept of efficient communication, we extend our approach in Chapter 2 by proposing Model-Agnostic Projection Adaptation (MAPA), a unified low-rank factorization method that compresses the entire model parameter space, further optimizing both communication and computational resources. Recognizing the need to address personalization in heterogeneous client settings, in Chapter 3, we develop Principal-Aligned LoRA (PA-LoRA), a personalized FL approach leveraging Singular Value Decomposition (SVD) to align client-specific updates, effectively mitigating client drift and enhancing personalization without additional communication overhead. Collectively, these innovations form a coherent and practical framework, significantly advancing the efficiency, scalability, and personalization capabilities of federated learning, paving the way for widespread deployment in privacy-sensitive and resource-constrained environments.

Keywords Federated Learning, Communication Efficiency, Low-Rank Adaptation, Model Alignment

Contents

List of Tables

List of Figures

Introduction

The exponential growth of data and computational capabilities has accelerated advancements in artificial intelligence, enabling unprecedented progress in machine learning and, in particular, deep learning. Centralized learning frameworks, while powerful, inherently require data consolidation, raising substantial privacy, security, and regulatory concerns. Federated Learning (FL) has emerged as a transformative paradigm, designed to address these concerns by enabling distributed model training across decentralized data sources. In FL, multiple clients collaboratively train models locally, sharing only model updates rather than raw data, thereby enhancing data privacy and reducing risks associated with centralized data storage.

Despite its significant promise, the practical deployment of FL faces notable challenges, particularly related to communication efficiency, client heterogeneity, and model personalization. Communication overhead, resulting from frequent and large model updates, restricts the feasibility of FL for resource-constrained clients and networks. Moreover, client heterogeneity—manifesting as diverse data distributions, computational resources, and learning objectives—introduces critical alignment and convergence issues. Addressing these challenges requires innovative solutions that optimize communication, efficiently align decentralized models, and effectively personalize large-scale models for individual clients.

This thesis comprehensively addresses these fundamental challenges by developing novel methodological frameworks and techniques across three interconnected chapters:

- **Chapter 1** provides foundational context, critically reviews existing methods, and outlines key limitations in current FL frameworks, emphasizing communication bottlenecks, alignment difficulties arising from model parameter symmetries, and the challenge of effectively personalizing pre-trained large-scale models.

- **Chapter 2** introduces **Model-Agnostic Projection Optimization (MAPO)**, a flexible, universal framework aimed at significantly enhancing communication efficiency. MAPO transcends architecture-specific constraints through a unified, model-level gradient decomposition approach. It dynamically explores subspace optimization strategies, leveraging random matrix reinitialization to efficiently balance communication overhead with convergence performance, validated through rigorous theoretical analyses and extensive empirical evaluations.

- **Chapter 3** presents **Aggregation of Principals in Low-rank via SVD (APriLS)**, addressing the critical misalignment problem encountered in federated low-rank adaptation strategies, particularly LoRA. APriLS employs singular value decomposition (SVD) to implicitly align client parameters, significantly mitigating alignment issues caused by permutation and transformation symmetries inherent in neural networks. APriLS ensures robust convergence, enables effective personalization, and substantially reduces communication frequency, demonstrated through comprehensive theoretical and experimental validation.

Collectively, these contributions offer transformative advancements to the field of federated learning, providing robust, scalable, and efficient solutions essential for practical deployment across diverse real-world scenarios.

Chapter 1. Evolutionary Strategies as Low-rank Compression

1.1 Introduction

Federated Learning (FL) provides a decentralized machine learning framework that enables model training across many devices, known as clients, without needing to collect and process sensitive client data on the centralized server [1]. The typical FL process begins with each client downloading an identical initialized model from a central server, performing model updates with local data, and then uploading the updated local model for the next communication round. Subsequently, the server combines the uploaded models to refine the global model, typically using a technique like FedAvg [2]. This iterative cycle repeats for a fixed number of rounds, ensuring collaborative model improvement across clients.

Although FL provides notable benefits, such as a certain level of privacy preservation and the utilization of diverse data sources, one of the major challenges associated with FL is the significant communication overhead involved in transmitting model updates between clients and the server, especially when dealing with models that have a large number of parameters.

Various strategies have been developed to mitigate the communication burden in FL. These techniques can be broadly classified into three categories: i) Compressing updates: sparsification [1], structured updates [3], and quantization [4, 5] reduce the size of transmitted model updates, ii) Local computation: performing multiple local epochs at the clients [6] lessens the frequency of communication with the server, and iii) Advanced aggregation methods and client selection: MOCHA [7] enhances the efficacy of update aggregation and [8, 9] reduce communication by only selecting a subset of clients to participate in each training round.

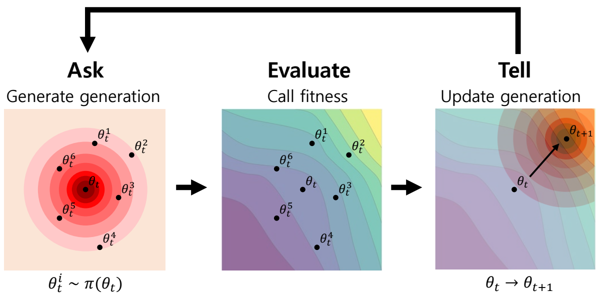
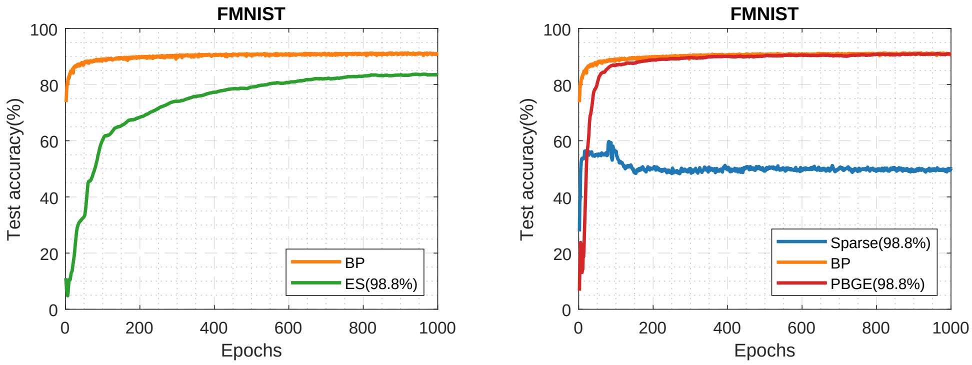


Figure 1.1: ES follows an iterative ask, evaluate, and tell approach for optimization. The strategy starts by generating candidate solutions of a base model(*ask*), which are then assessed using a fitness measure (*evaluate*). The base model is updated towards a better solution using the evaluation results (*tell*).

Motivation. Existing FL techniques primarily rely on transmitting gradient signals or model updates, which are computed through backpropagation (BP). On the other hand, Evolutionary Strategies (ES) [10–12] update model parameters by utilizing fitness values obtained from evaluating a population of models. This approach eliminates the need for a gradient signal, as depicted in Figure 1.1. Recent advances in neuroevolution have shown promise in supervised settings and competitive performance with reinforcement learning in control tasks [13–18]. In this context, ES offers a distinct advantage compared to traditional BP methods.



(a) Estimation with ES (b) The proposed method

Figure 1.2: Test accuracy on FMNIST dataset comparing (a) ES against BP, and (b) PBGE with BP and Sparse with 98.8% compression.

This paradigm shift opens up the possibility of searching for novel solutions beyond the gradient-based methods. However, it is critical to note that ES, despite its potential, still lags behind gradient-based methods in certain problems like supervised learning. Fig. 1.2(a) reveals the performance gap between ES and BP. This emphasizes the need for a balanced approach that would leverage the strengths of both ES and gradient-based methods to achieve optimal results.

Our proposed method operates on the premise of incorporating high-quality gradient signals into the evaluation process of ES. The process can be formalized as follows:

Given a base model, denoted by θ, we instantiate a population P comprised of N model samples. Each individual sample, θ^i, is derived by adding random perturbations to θ. Unlike traditional ES, where the fitness of θ^i corresponds to its performance on the task, we instead assess the fitness of each sample θ^i by measuring its similarity to θ', the model parameters updated through gradient descent steps. This operation effectively exploits the gradient signal to construct a fitness vector, a process we term Population-Based Gradient Encoding (PBGE). Fig. 1.2(b) shows the results of a representative experimental setting where PBGE is able to follow BP closely on the FMNIST dataset while maintaining an effective compression rate of over 98.8%. In particular, this method significantly outperforms sparsification strategies at equivalent compression rates.

In the context of FL, the gradient signal can be encoded into a fitness vector for the population of models and communicated to the server. For global synchronization, a well-established approach like FedAvg would involve reconstruction and aggregation of the clients' models and sharing of the aggregated model with the clients. However, by utilizing shared random seeds, we can ensure uniformity in the generated populations between clients and the server. This allows us to transmit and aggregate only the small fitness vectors without reconstructing the models, reinforcing communication efficiency. In addition, there is an advantage in implementing extra privacy measures. When encryption is desired, the required overhead would be much smaller with the fitness vectors than with the model parameter vectors because of the size difference. A brief overview of EvoFed is shown in Fig. 1.3 (the detailed methodology is provided in Section 1.4).

Contributions. To summarize, our main contribution is to introduce a novel concept of Population-Based Gradient Encoding (PBGE), which allows an accurate representation of the large local gradient vector using a relatively small fitness vector, significantly reducing the communication burden as well as the encryption overhead in FL environments. We also propose and verify the EvoFed framework that integrates PBGE into federated learning based on the exchange and aggregation of the fitness vectors between clients and the server.

Our approach achieves similar performance to FedAvg on both FMNIST and CIFAR-10 datasets. The advantage is that our scheme achieves very high compression, exceeding 98.8% on FMNIST and 99.7% on CIFAR-10 in key practical settings. Significantly, our EvoFed model also outperforms traditional compression methods, offering superior results at similar compression rates. The price we pay is the overhead of generating a population of perturbed models and computing similarity measures. This overhead could be substantial in the form of increased local processing time depending on the size of the population, but the client computational load can be traded with local memory space by employing parallel processing. In addition, as discussed later, reduce population size without affecting performance.

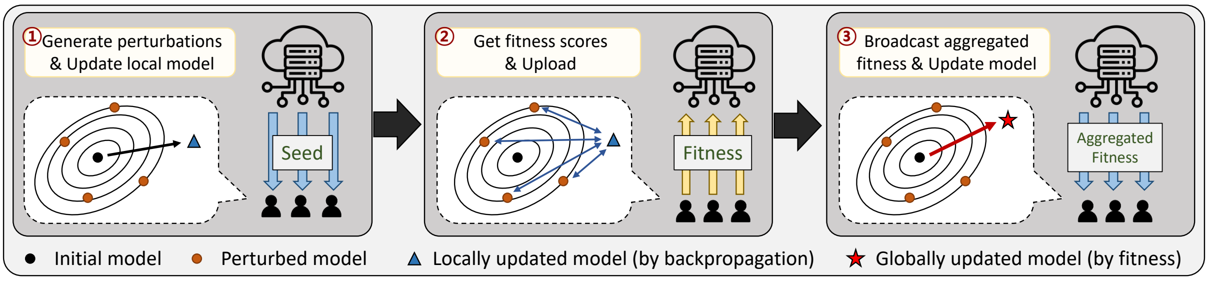


Figure 1.3: Overview of the proposed EvoFed: (1) Using the shared random seed, each client generates a population of perturbations around the local model. Each client also performs a gradient update of the local model using the local data as in conventional FL. (2) Each client evaluates the fitness of each perturbed model with respect to the updated model. The fitness values are communicated to the server. (3) The server aggregates the fitness values. Clients update their local models using broadcast aggregated fitness.

1.2 Related Work

Federated Learning. Several techniques have been proposed to alleviate the communication overhead in FL, such as model compression [1, 19, 20], distillation techniques [21–25], and client update subsampling [26, 27]. However, these approaches often come with trade-offs, including increased convergence time, lower accuracy, or additional computational requirements. More importantly, they still require the exchange of model parameters, which requires substantial communication bandwidth. Furthermore, [28] shows that the gradient sparsity of all participants has a negative impact on global convergence and communication complexity in FL.

Evolutionary Strategies. ES are black-box optimization algorithms inspired by biological evolution [29]. ES algorithms iteratively refine a population of solutions based on fitness evaluations. Natural Evolution Strategies (NES) is a specific variant of ES [12, 30–35]. Within the NES framework, the distribution $p_\psi(\theta)$, parameterized by ψ, is adopted to represent the population and maximize the average objective value $\mathbb{E}_{\theta\sim p_\psi}[f(\theta)]$ via stochastic gradient ascent, where $f(\theta)$ is the fitness function. NES algorithms leverage a score function estimator as in [10]. Our method follows the guidelines provided by [36], where the parameter distribution p_ψ is a factored Gaussian. Accordingly, we can represent $\mathbb{E}_{\theta\sim p_\psi}[f(\theta)]$ using the mean parameter vector θ, such that $\mathbb{E}_{\theta\sim p_\psi}[f(\theta)] = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[f(\theta+\sigma\epsilon)]$. Given a differentiable function estimator, the well-known conversion procedure allows optimization over θ to be rewritten as (see, for example, [11, 37, 38])

$$\nabla_\theta \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[f(\theta+\sigma\epsilon)] = \frac{1}{\sigma}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\{f(\theta+\sigma\epsilon)\epsilon\}. \tag{1.1}$$

The strategy of sharing seeds for random number generators to synchronize parallel processes and maintain consistency is a well-established practice in areas ranging from parallel simulations to cryptographic protocols [39]. The appeal of shared random seeds stems from their ability to offer deterministic randomness, ensuring that multiple entities, operating independently, can produce synchronized and identical random outputs. Within the realm of ES, this concept was effectively harnessed by [36] to address the challenge of scalability. We also utilize the shared random seeds to scale down the communication load in a distributed setting. However, the key difference is that the work of [36] distributes the perturbations into multiple workers dealing with a single global objective, whereas our method generates identical perturbations across all clients, with each having a distinct objective. An important consequence is that with our method, each client only needs to communicate N fitness values to update the model, instead of MN values in [36] with M denoting the number of nodes, enabling scalability regardless of the number of clients.

Federated Learning and Evolutionary Strategies. Several studies have explored the optimization of FL using ES. For instance, [40] introduced an evolutionary approach for network architecture search (NAS) in real-time FL, which minimizes local payload while optimizing model performance. Sparse evolutionary training (SET) [41] substitutes fully connected layers in neural networks with sparse layers to decrease the number of model parameters. Furthermore, [42] presented the SET algorithm that optimizes neural networks in FL via a bi-objective method to maximize accuracy performance while minimizing communication overhead. Additionally, [43] introduces the MOEA/D framework [44] to the environment of FL and FLEA [45] utilized Evolutionary Algorithms in FL setup at the client-level to evolve models.

While these studies have made significant contributions, the present work establishes a unique way of using ES as a method to reduce communication overhead in FL by transmitting fitness values instead of model parameters. In particular, the new gradient-driven fitness function essentially separates our work from the traditional utilization of ES for compression.

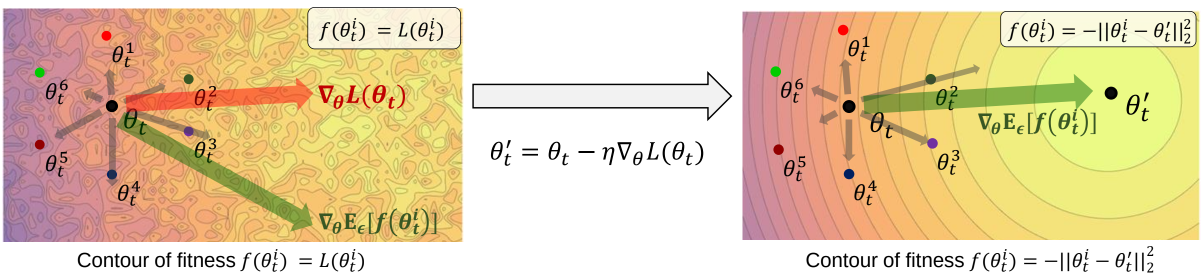


Figure 1.4: Illustration of one update step of typical ES and the proposed Population-Based Gradient Encoding (PBGE) strategy. **Left**: using loss value $L(\theta)$ directly for the fitness $f(\theta)$, as in existing ES, leads to an inherently noisy estimate of the gradient signal $\nabla_\theta\mathbb{E}_\epsilon[f(\theta+\sigma\epsilon)]$ due to the noisy loss surface. **Right**: PBGE defines fitness as the distance to the updated model θ' obtained through BP (i.e., $f(\theta) = -||\theta-\theta'||_2^2$). This enables the reliable estimate of θ' on the convex surface. By sampling a sufficient number of perturbations ϵ_i, a decent gradient signal can be obtained, which aligns to the true gradient signal $\nabla_\theta L(\theta)$ computed from the BP.

1.3 Population-Based Gradient Encoding

This work focuses on the process of encoding gradient information through an identical population of models generated at both ends of some network link. Population distribution is a zero mean isotropic multivariate Gaussian with fixed covariance $\sigma^2 I$. We also adopt 'mirrored sampling' [46,47] for variance reduction where the Gaussian noise vector ϵ is instantiated with pairs of perturbations ϵ, $-\epsilon$.

Given the reference point $\theta' = \theta - \eta\nabla L(\theta)$ where η is the learning rate in the BP-based gradient update and $\nabla L(\theta)$ represents the gradient derived from the data, we define the fitness function $f(\theta)$ to measure the similarity between the model parameters θ and θ': $f(\theta) = -||\theta - \theta'||_2^2$. This choice ensures that the gradient of the expectation, $\nabla_\theta \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[f(\theta + \sigma\epsilon)]$, aligns with the actual gradient $\nabla L(\theta)$, effectively encoding the gradient information in the fitness values:

$$\nabla_\theta \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[-||(\theta + \sigma\epsilon) - \theta'||_2^2] = -\nabla_\theta ||\theta - \theta'||_2^2 = 2(\theta - \theta') \tag{1.2}$$

where the first equality simply follows from the assumption that ϵ is zero-mean. The visual representation of this process is illustrated in Fig. 1.4. Eq. 1.2 gives $\theta' = \theta - \frac{1}{2}\nabla_\theta \mathbb{E}_\epsilon[-||(\theta + \sigma\epsilon) - \theta'||_2^2]$, and comparing with the BP operation $\theta' = \theta - \eta\nabla L(\theta)$, we have

$$\eta\nabla_\theta L(\theta) = \frac{1}{2}\nabla_\theta \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[-||(\theta + \sigma\epsilon) - \theta'||_2^2].$$

Now also utilizing Eq. 1.1, we write

$$\eta\nabla_\theta L(\theta) = \frac{1}{2\sigma}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\{f(\theta + \sigma\epsilon)\epsilon\} \tag{1.3}$$

where f is the specific distance-based fitness function defined above. Approximating the expectation by sampling, we further write

$$\eta\nabla_\theta L(\theta) \approx \frac{1}{2N\sigma}\sum_{i=1}^{N} f(\theta + \sigma\epsilon_i)\epsilon_i \tag{1.4}$$

which shows how the gradient update information can be encoded using the fitness values $f(\theta + \sigma\epsilon_i)$ corresponding to the perturbations ϵ_i. Consequently, this removes the need to evaluate each perturbation with the dataset, as is done in existing ES, and the entire update process is encoded with low-cost distance measurement.

Finally, the update on θ itself based on the fitness values is naturally given by

$$\theta' \approx \theta + \frac{1}{2N\sigma}\sum_{i=1}^{N} f(\theta + \sigma\epsilon_i)\epsilon_i \tag{1.5}$$

allowing a remote model update based on the transmitted $f(\theta + \sigma\epsilon_i)$ values with the shared knowledge of the ϵ_i values.

The implemented algorithm consistently executes three steps: (i) Compute the target θ' through gradient descent, (ii) Implement perturbations to the model parameters and assess the perturbed parameters by computing their Euclidean distance to θ', and (iii) Utilize the assessment results and encode the gradient with the fitness measures.

Partitioning. In the context of PBGE, the fitness function indicates the model distance, not the model performance on client data. This feature enables a unique partition-based approach for handling a large number of model parameters. Here, the parameters, flattened into a vector θ, are split into K partitions: $\theta[1], \theta[2], ..., \theta[K]$. Each partition is then effectively encoded individually using PBGE, which is advantageous when working with large models, as storing a large population of those may be

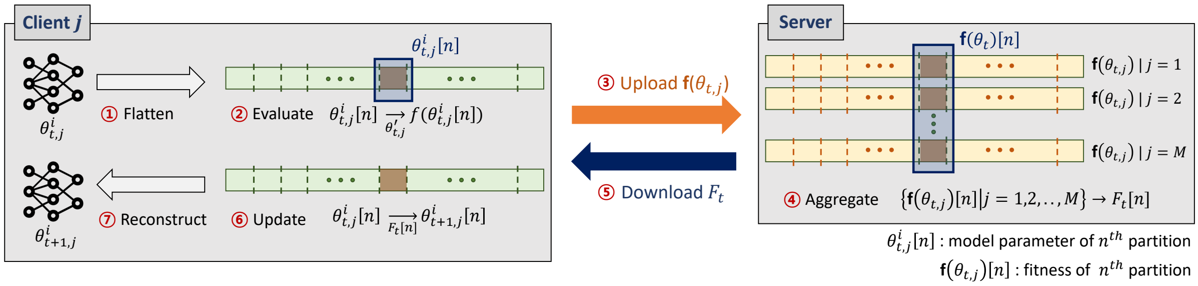


Figure 1.5: Partition-based Model Parameter Compression. Illustration of dividing model parameters θ into K partitions and compressing them individually using PBGE.

burdensome for clients with limited memory. Partitioning allows us to compute K fitness values per each perturbation, providing K times more reference points to encode the gradient information given a population size. Hence, even with a small population size (requiring less memory), we achieve robust performance as validated by the empirical results provided in Supplementary Materials. Essentially, partitioning provides a tradeoff of memory with communication as more fitness values now need to be communicated per each perturbation. This partitioning process is visualized in Fig. 1.5. Notably, this partitioning technique can be used with any model architecture, regardless of its specific design, providing a practical and efficient means of compressing large models.

1.4 EvoFed

EvoFed operates on the principle that the evolutionary update step depends only on the fitness values given the perturbation samples. In the FL context, we can leverage this characteristic to devise an accurate yet communication-efficient strategy for model updates. This section provides a detailed exposition of our methodology, breaking it down into stages for clarity. This iterative process, as outlined in Algorithm 1, aims to gradually converge the model parameters at all nodes to an optimal solution while minimizing data transmission during each update. An overall view of our proposed methodology is depicted in Fig. 1.3.

1.4.1 Initial Setup

The initialization of the server and clients in EvoFed begins with the same baseline model, denoted by θ_0. A key assumption is that the server and all clients share the same seed (e.g., via broadcasting) for identical random population generation. This approach ensures consistent baseline models and populations across all nodes.

In this step, a population of candidate solutions (in this case, models) is generated by adding random perturbations to the current best solution (the baseline model). The generation of an i-th member of the model population can be formulated as follows:

$$\theta_t^i = \theta_t + \mathcal{N}(0, \sigma I) \tag{1.6}$$

Here, $\mathcal{N}(0, \sigma I)$ represents the perturbations sampled from a multivariate normal distribution with zero mean and a shared covariance matrix σI. We also denote the population at each node at any given time

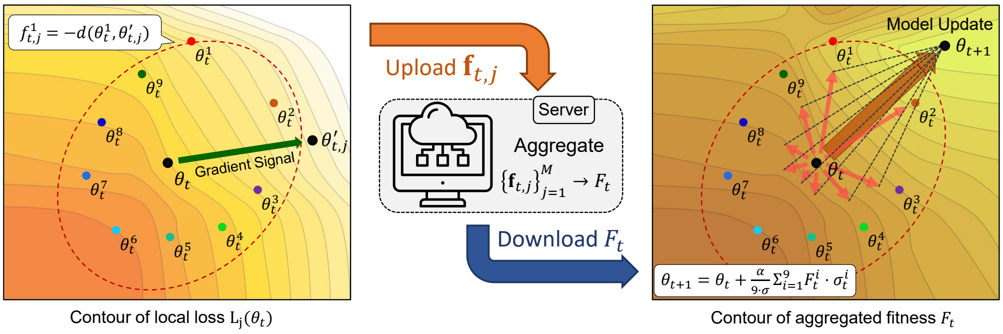


Figure 1.6: Local Updates and Fitness Evaluation (left) and Server/Client-side Update (right). Left: Client performs BP on local data θ_t to obtain $\theta'_{t,j}$, evaluates fitness $f^i_{t,j}$ by distance measure (e.g. L2) with θ^i_t, and uploads $\mathbf{f}_{t,j}$ to the server. Right: After obtaining the aggregated fitness $\mathbf{F}_t$, all nodes update the baseline model θ_t according to Eq. 1.9.

t as $\mathbf{P}_t = \{\theta^1_t, \theta^2_t, ..., \theta^N_t\}$, where θ^i_t represents the i-th model in the population generated by adding the i-th perturbation to the baseline model parameters, and N is the size of the population.

1.4.2 Local Updates and Fitness Evaluation

Each client node begins by executing BP on its local dataset, using the baseline model, θ_t, resulting in an updated model, θ'_t. Following this, the fitness of each member θ^i_t in the local population, $\mathbf{P}_t$, is evaluated. This evaluation is done by measuring the similarity between θ'_t and θ^i_t. The L2 norm or Euclidean distance serves as the measure of this similarity. The fitness of θ^i_t is represented as $f(\theta^i_t)$:

$$f(\theta^i_t) = -||\theta'_t - \theta^i_t||^2_2 \tag{1.7}$$

The process of local update and fitness evaluation is illustrated in Fig. 1.6. The fitness values are the only information that needs to be communicated among nodes. The fitness vectors are significantly smaller in size compared to the model parameter vectors, which helps reduce the communication overhead. Hence, each client sends a fitness vector, $\mathbf{f}_t = \{f(\theta^1_t), f(\theta^2_t), ..., f(\theta^N_t)\}$ corresponding to all population members, to the server.

1.4.3 Server-side Aggregation and Update

The server's responsibility is to aggregate the fitness values reported by all client nodes, forming a global fitness vector $\mathbf{F}_t$ comprising N elements, with $\mathbf{F}^i_t$ representing the fitness value of the i^{th} member of the population. Each client's contribution to $\mathbf{F}_t$ is weighted by their respective batch size b_j, giving a larger influence to clients with potentially more accurate local updates.

The global fitness vector $\mathbf{F}_t$ is computed as follows:

$$\mathbf{F}_t = \frac{1}{\sum_{j=1}^M b_j} \sum_{j=1}^M b_j \mathbf{f}_{t,j} \tag{1.8}$$

where M is the total number of clients and $\mathbf{f}_{t,j}$ is the fitness value vector from client j at time t. After the aggregation, the server broadcasts $\mathbf{F}_t$ so that each client updates the baseline model θ_t.

Algorithm 1 EvoFed: Federated Learning with Evolutionary Strategies

Input: Learning rates η, α, population size N, noise std σ, seed s
Initialize: server and M clients with seed s and identical parameters θ_0 and $\theta_{0,j}$ respectively
for each communication round $t = 0, 1, ..., T-1$ in parallel **do**
  for each client j in parallel **do**
    $\theta'_{t,j} = \theta_{t,j} - \eta \nabla_\theta L(\theta'_{t,j})$ //Backpropagation Update
    Sample $\epsilon_t^i \sim \mathcal{N}(0, \sigma I)$ //Sample perturbations
    $\theta_{t,j}^i = \theta_{t,j} + \epsilon_t^i$ for $i = 1, ..., N$ //Initialize population
    $\mathbf{f}_{t,j}^i = -\|\theta_{t,j}^i - \theta'_{t,j}\|_2^2$ for $i = 1, ..., N$ // Compute fitness vector using L_2 calculation
  end for
  $\mathbf{F}_t^i = \frac{1}{\sum_j^M b_j} \sum_j^M b_j \mathbf{f}_{t,j}^i$ for $i = 1, ..., N$ // Server averages fitness vectors
  $\theta_{t+1} = \theta_t + \frac{\alpha}{N\sigma} \sum_{i=1}^N \mathbf{F}_t^i \cdot \epsilon_t^i$ // Server updates model using the aggregated fitness
  Broadcast $\{\mathbf{F}_t^1, \mathbf{F}_t^2, \ldots, \mathbf{F}_t^N\}$
  for each client j in parallel **do**
    $\theta_{t+1,j} = \theta_{t,j} + \frac{\alpha}{N\sigma} \sum_{i=1}^N \mathbf{F}_t^i \cdot \epsilon_t^i$ // Client updates model using the aggregated fitness
  end for
end for

1.4.4 Broadcasting Fitness

Once the aggregation is complete, the server broadcasts $\mathbf{F}_t$ to all client nodes, maintaining synchronicity across the network. This only involves the transmission of the fitness vector, again reducing communication overhead. The aggregated fitness vector can be used by the local clients (as well as by the server, if needed) to update the local models.

1.4.5 Client-side Update

When the global fitness vector $\mathbf{F}_t$ is received from the server, the clients can update their local model following Eq. 1.5. This strategy involves the addition of a weighted sum of noise vectors to the current model parameters, where the weights are aggregated fitness values:

$$\theta_{t+1} = \theta_t + \frac{\alpha}{N\sigma} \sum_{i=1}^{N} \mathbf{F}_t^i \cdot \epsilon_t^i \tag{1.9}$$

where ϵ_t^i is the i-th noise vector from the population at time t.

Notably, the right side of Eq. 1.9 can be shown equivalent to the average of all locally updated models akin to FedAvg, i.e., it is straightforward to show that

$$\theta_{t+1} = \frac{1}{\sum_j^M b_j} \sum_{j=1}^{M} b_j \theta_{t+1,j} \tag{1.10}$$

where $\theta_{t+1,j}$ is the updated model of client j at time $t+1$ defined as

$$\theta_{t+1,j} = \theta_t + \frac{\alpha}{N\sigma} \sum_{i=1}^{N} f_{t,j}^i \cdot \epsilon_t^i. \tag{1.11}$$

1.4.6 Convergence Analysis

Theorem 1.4.1. *Suppose that $L_j(\theta)$ is the β-smooth function, i.e., $\|\nabla L_j(u) - \nabla L_j(v)\| \leq \beta \|u - v\|$ for any u, v, and also suppose that the variance of the stochastic gradient of D_j is bounded, i.e., $\mathbb{E}\|\nabla L_j(\theta) - \widetilde{\nabla} L_j(\theta)\|^2 \leq B^2$ for all j. When perturbation ϵ^i is sampled, a conditioned mirrored sampling is applied*

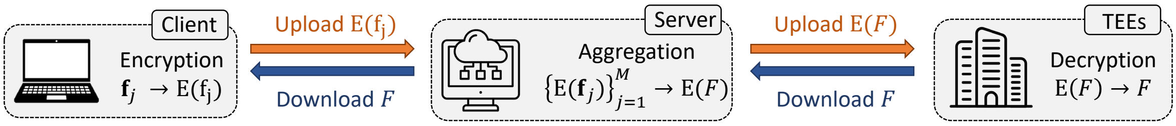


Figure 1.7: Privacy enhancement through Fully Homomorphic Encryption and Trusted Execution Environments.

such that $\frac{1}{N}\sum_{i=1}^{N}\epsilon^i = 0$, $\frac{1}{N}\sum_{i=1}^{N}\left(\epsilon^i\right)^2 \leq G^2$, $\frac{1}{N}\sum_{i=1}^{N}\left(\epsilon^i\right)^3 = 0$. *Given a decreasing learning rate* $\eta_t < \frac{1}{4\alpha\beta}$, *EvoFed converges in the sense of*

$$\frac{1}{H_T}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\left[\|\nabla L(\theta_t)\|^2\right] \leq \frac{\mathbb{E}\left[L(\theta_0)\right] - L^*}{\alpha G^2 H_T} + 4\alpha\beta B^2\left(\frac{1}{H_T}\sum_{t=0}^{T-1}\eta_t^2\right)$$

where $H_T = \sum_{t=0}^{T-1}\eta_t$, *and* L^* *represents the minimum value of* $L(\theta)$.

Given a decreasing learning rate (e.g., $\eta_t = \frac{\eta_0}{1+t}$), it can be seen that $H_T = \sum_{t=0}^{T-1}\eta_t \rightarrow \infty$ as T increases, while $\sum_{t=0}^{T-1}\eta_t^2 < \infty$. Consequently, the upper bound stated in Theorem 1.4.1 approaches 0 as T grows, ensuring convergence towards a stationary point. The detailed discussions and the proof can be found in Supplementary Materials.

1.4.7 EvoFed and Privacy Enhancements

Unlike other FL methods, EvoFed operates on fitness measures. Encryption on smaller fitness vectors requires a lower overhead compared to encryption on large model parameter vectors. Fig. 1.7 shows Fully Homomorphic Encryption (FHE) [48–50] that allows aggregation of encrypted data on the server while keeping individual client data confidential. For the decryption of the aggregated fitness vector, EvoFed can leverage third-party Trusted Execution Environments (TEEs), such as Intel SGX [51], providing a secure space for sensitive computations.

1.4.8 Partial Client Participation

When FL has to operate with a large client pool, a typical practice is to select only a subset of clients in each global round. In EvoFed, this means that the newly joined clients in a given round can either download the latest model or else the last k fitness vectors from the server, where k is the time gap from the last participation. In the latter case, the model updates from Equation (1.9) can be modified to accommodate k-step updates:

$$\theta_t = \theta_{t-k} + \frac{\alpha}{N\sigma}\sum_{l=1}^{k}\sum_{i=1}^{N}\mathbf{F}_{t-l}^{i}\cdot\epsilon_{t-l}^{i}. \tag{1.12}$$

Note that even in the former case where the latest model is downloaded by the newly joined clients, the level of security is not compromised as far as the client-to-server messages are concerned.

1.5 Experiments

Our algorithm's effectiveness is assessed on three image classification datasets: FMNIST [52], MNIST [53], and CIFAR-10 [54]. Both MNIST and FMNIST contain 60,000 training samples and 10,000 test samples, whereas CIFAR-10 is composed of 50,000 training samples and 10,000 test samples. We employ a CNN model having 11k parameters for the MNIST and FMNIST datasets and a more substantial model with 2.3M parameters for CIFAR-10. A grid search has been conducted to identify the optimal performance hyperparameters for each baseline, as outlined in the results section. We take into account both global accuracy and communication costs to ascertain hyperparameters that maximize accuracy while minimizing communication overhead.

Data Distribution. We distribute the training set of each dataset among clients for model training, and the performance of the final global model is evaluated using the original test set. Our experimental setup contains $M = 5$ clients with non-IID data distribution (assigning two classes to each client).

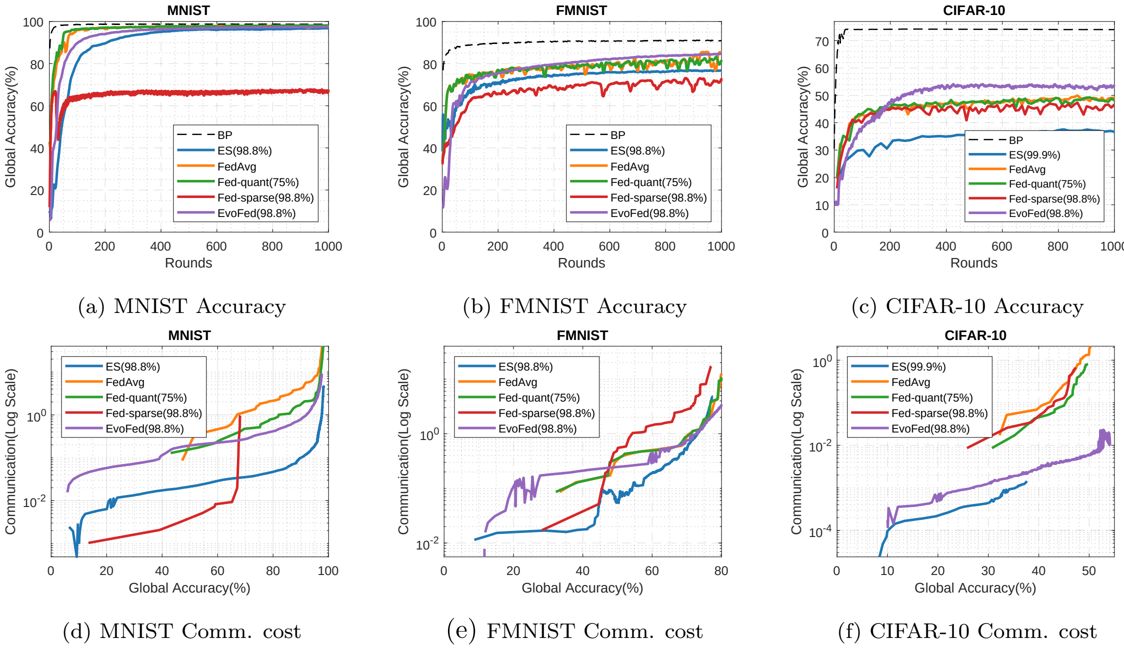
Implementation Details. Our EvoFed framework is built using JAX [55], which facilitates extensive parallelization and, in particular, consistent random number generation across a large number of nodes. We have implemented our framework on the Evosax [56] library, a convenient tool for the ES algorithm. EvoFed is configured with a population size of 128 and a mini-batch size of 256 for MNIST / FMNIST and 64 for CIFAR-10. We perform ten local epochs (performing ten BP steps before fitness calculation) and train over 1,000 global rounds.

Baselines. We compare the performance of the proposed EvoFed with BP, FedAvg, ES, FedAvg with quantization (Fed-quant), and FedAvg with Sparsification (Fed-sparse). In each scenario, we push for maximum compression, stopping right before the model starts to show performance degradation relative to FedAvg with no compression. BP provides the upper-performance baseline, while ES serves as a reference emphasizing the significance of PBGE.

1.6 Results and Discussions

In this section, we discuss the experimental results in detail and provide further insights into the performance of EvoFed. The accuracy of EvoFed, compared with multiple baseline methods and different datasets, is shown in Fig. 1.8 (a), (b), and (c). Efficiently encoding and exchanging gradient information, EvoFed enhances the effectiveness of the ES algorithm across all tasks, delivering results comparable to FedAvg. Also, EvoFed achieves superior accuracy at an equivalent compression rate compared to sparsification. This suggests that utilizing a shared population of samples can reduce the information necessary for gradient compression, thereby enhancing the efficiency of the process. Fig. 1.8 (d), (e), and (f) shows the performance of each method as a function of communication load for all three datasets. It can be seen that EvoFed tends to utilize significantly less communication resources as compared to other high-accuracy techniques.

Table 1.1 summarizes the performance of different schemes on MNIST, FMNIST, and CIFAR-10 datasets, focusing on communication cost and accuracy. EvoFed achieves significantly lower communication costs compared to FedAvg while maintaining competitive accuracy levels. In the MNIST dataset, EvoFed achieves an accuracy of 97.62% with a mere 9.2 MB of communication load, while FedAvg achieves 98.09% accuracy at a considerably high communication cost of 73.7 MB. The effective compression achieved is an impressive 98.8% which indicates that the gradient vector is condensed into just 1.2% of the fitness vector that is communicated between clients and the server. Similarly, for the FMNIST dataset, EvoFed achieves an accuracy of 84.72% with only 7.78 MB of communication, whereas FedAvg's accuracy is 85.53% with a communication cost of 40.99 MB. The efficiency of EvoFed becomes even more apparent in the CIFAR-10 dataset where the model compression is over 99.7%. EvoFed achieves an accuracy of 54.12% with a low communication cost of only 0.023 GB, surpassing FedAvg, which performs



(a) MNIST Accuracy (b) FMNIST Accuracy (c) CIFAR-10 Accuracy

(d) MNIST Comm. cost (e) FMNIST Comm. cost (f) CIFAR-10 Comm. cost

Figure 1.8: Performance comparison of EvoFed and baseline methods on MNIST, FMNIST, and CIFAR-10 datasets. The top row displays the accuracy achieved by each method on the respective datasets, while the bottom row illustrates the communication cost associated with each method.

50.22% at a communication cost of 2.134 GB. The simpler ES method actually gives better performance as well as higher communication efficiency than EvoFed for MNIST but its accuracy for other data sets is highly limited.

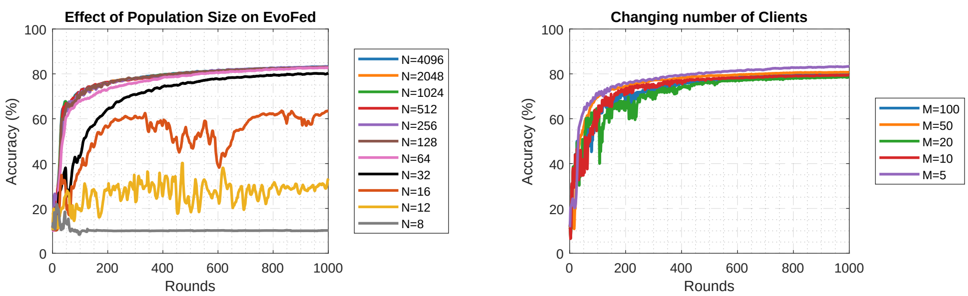
Additional Experiments. Figure 1.9 illustrates the performance of MAPA extension (MAPAX) for varying numbers of k partitions. As shown model's performance is significantly influenced by the reconstruction matrix, which has the same amount of communication; the factorization with a larger reconstruction matrix performs better and leads to less communication per accuracy. We observe a performance improvement as we increase the population size, although at the cost of increased computation and memory usage. Nonetheless, this improvement is not linear and plateaus once sufficient samples are generated to explore the parameter space. For instance, in the case of the FMNIST dataset and a model with 11K parameters, any performance enhancement beyond a generated sample size of 128 is marginal.

Figure 1.9(b) showcases EvoFed's performance trend as the number of clients increases. A minor performance decline is observable for some larger values of M relative to M=5. This decline could potentially be attributed to the limited data available per client and the increased variance associated with local training. Nonetheless, we believe that carefully tuning the hyperparameters based on data distribution could assist in achieving more robust performance.

Supplementary Materials provide further details regarding hyperparameters, model architectures, and experiments. We also include a detailed ablation study with different evolutionary strategies, the population size, and the number of partitions, as well as detailed communication and computational complexity analyses.

Table 1.1: Communication cost (at target accuracy) and maximum accuracy for different methods across datasets.

Dataset: MNIST	Comm. @ 90% Acc. (MB)	Comm. @ Max Acc. (MB)	Max Accuracy
ES	0.10	4.60	97.62%
FedAvg	4.20	73.70	98.09%
Fed-quant	1.70	41.80	98.15%
Fed-sparse	–	1.00	67.85%
EvoFed (ours)	0.80	9.20	98.30%
Dataset: FMNIST	**Comm. @ 70% Acc. (MB)**	**Comm. @ Max Acc. (MB)**	**Max Accuracy**
ES	0.48	4.85	76.86%
FedAvg	0.87	40.99	85.53%
Fed-quant	0.94	37.98	83.23%
Fed-sparse	2.78	17.13	73.17%
EvoFed (ours)	0.75	7.78	84.72%
Dataset: CIFAR-10	**Comm. @ 45% Acc. (GB)**	**Comm. @ Max Acc. (GB)**	**Max Accuracy**
ES	–	0.021	37.47%
FedAvg	0.266	2.134	50.22%
Fed-quant	0.086	0.800	49.78%
Fed-sparse	0.129	0.671	47.36%
EvoFed (ours)	0.004	0.023	54.12%



(a) Varying number of population from 8 to 4096 (b) Varying number of clients from 5 to 100

Figure 1.9: Effect of population size (left) and number of clients (right) on EvoFed

1.7 Conclusion

EvoFed, the novel paradigm presented herein, unites FL and ES to offer an efficient decentralized machine learning strategy. By exchanging compact fitness values instead of extensive model parameters, EvoFed significantly curtails communication costs. Its performance parallels FedAvg while demonstrating an impressive equivalent model compression of over 98.8% on FMNIST and 99.7% on CIFAR-10 in representative experimental settings. Consequently, EvoFed represents a substantial stride forward in FL, successfully tackling the critical issue of high communication overhead.

Chapter 2. Model-Agnostic Projection Optimization

2.1 Introduction

Federated Learning (FL) is a distributed framework that enables model training across many clients without centralizing data. In each communication round, clients download a global model, update it using local data, and send modifications back to the server, which aggregates them (e.g., via FedAvg [2]). While this iterative process enables collaborative learning, frequent transmission of model updates incurs significant communication overhead, limiting FL application, particularly with large models or resource-constrained clients.

Communication-Efficient Federated Learning (CEFL) literature [57] proposes a vast range of strategies to reduce communication load. These methods are typically categorized into *sketched updates*, which compress the total model update after optimization (e.g., subsampling, quantization, random projection), and *structured updates*, which restrict the trainable parameters to a lower-dimensional subspace before optimization (e.g., random masks, weight-sharing, and low-rank decomposition) [1].

Low-rank decomposition is a widely used approximation technique that expresses model gradients or parameters as the product of low-rank matrices [58]. *Parameter decomposition* is particularly effective for Parameter-Efficient Fine-Tuning (PEFT), where auxiliary low-rank adaptation (LoRA) modules are added to each layer to reduce computation and storage overhead of full-model fine-tuning [59]. Although LoRA alleviates communication burdens in FL, constraining model parameters to a low-rank subspace can degrade performance. In contrast, *gradient decomposition* preserves full-rank model representations during inference and restricts only the gradients to a low-rank form during backpropagation [60–64]. A visual comparison is shown in Figure 2.1.

Challenges. While CEFL methods for gradient decomposition [65–69], parameter decomposition [70–74], or LoRA variants [75–79] offer notable benefits, they face several key challenges: 1) The layer-wise decomposition that adheres to the structural constraints (e.g., fully connected or convolutional), requiring *architecture-dependent* implementation for each layer decomposition. 2) Given a decomposition $\Delta W_i \in I\!R^{d_1 \times d_2} \approx B_i A_i$, where $A_i \in I\!R^{r \times d_2}$ and $B_i \in I\!R^{d_1 \times r}$, the number of transmitted parameters is $\mathbf{C} = |A_i| + |B_i| = r(d_1 + d_2)$ for $r \in I\!N$, restricting the communication rate to multiples of $(d_1 + d_2)$, imposing a *rigid communication granularity* as $C \in (d_1 + d_2)I\!N$. 3) Given M number of clients and (A_i^j, B_i^j) denoting the low-rank decomposition of layer i from client j, averaging these low-rank matrices is *not equivalent to full-rank aggregation* as:

$$\frac{1}{M}(B_i^1 A_i^1 + B_i^2 A_i^2 + \cdots + B_i^M A_i^M) \neq \frac{1}{M}(B_i^1 + B_i^2 + \cdots + B_i^M)\frac{1}{M}(A_i^1 + A_i^2 + \cdots + A_i^M).$$

4) Although fixing all $\{A_i^j\}_{j=1}^M$ matrices to the same values can mitigate the aggregation problem and improve the communication granularity to $\mathbf{C} \in d_1 I\!N$, as shown in FA-LoRA [75] and EvoFed [80], it restricts the model's ability to explore richer subspaces, often leading to *suboptimal solutions* [79]. Thus, we aim to answer the following key question:

How can we develop an architecture-independent model-wide decomposition that offers flexibility on communication rate, address the low-rank averaging problem, and suboptimality of freezing A?

Key Ideas. We propose a novel Model-Agnostic Projection Optimization (**MAPO**) that streamlines gradient projection and addresses its challenges while being computationally lighter than layer-wise

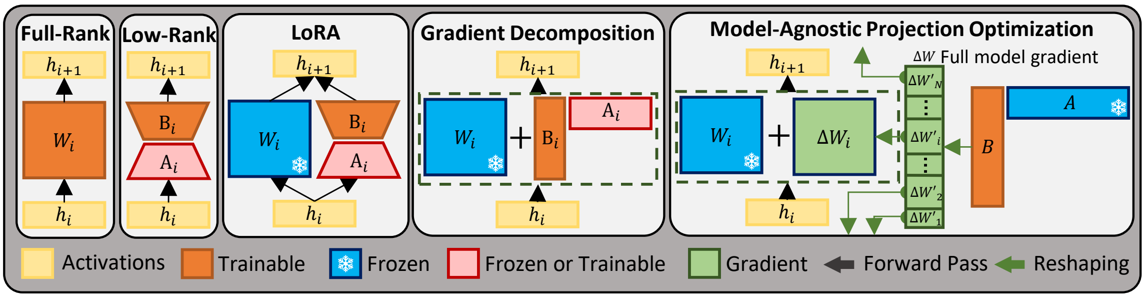


Figure 2.1: Comparison of various decomposition methods, from left: no decomposition, low-rank parameter decomposition, frozen model with low-rank adapter (LoRA), low-rank gradient decomposition, and MAPO.

methods. Our key ideas are described as follows:

(i) Firstly, MAPO reimagines low-rank gradient projection by treating the entire model gradient as a single matrix rather than layer-by-layer decomposition. It eliminates architecture-specific constraints by merging the flattened gradients of all layers, constructing the *universal gradient vector* $\Delta W \in \mathbb{R}^d$. where d is the total number of parameters, making MAPO applicable to any model architecture.

(ii) Secondly, given any communication budget k, MAPO pads ΔW with zeros so the length becomes divisible by k. Afterwards, padded ΔW will be reshaped to $\Delta W' \in \mathbb{R}^{k \times \lceil d/k \rceil}$ which further can be decomposed it into a $A \in \mathbb{R}^{1 \times \lceil d/k \rceil}$ and $B \in \mathbb{R}^{k \times 1}$ matrices, as $\Delta W' = BA$.

(iii) Lastly, instead of relying on a fixed A, MAPO explores new subspaces in each federated round through reinitialization of A, mitigating the risk of suboptimal convergence. Synchronization of A is achieved efficiently via a shared seed, removing the need to transmit A.

Summary of Contributions. By integrating **(i)** model-level decomposition, **(ii)** flexible communication rate, and **(iii)** subspace exploration, MAPO offers a flexible trade-off between communication cost and performance while remaining more efficient than low-rank decomposition methods. Figure 2.3 illustrates the distinction between MAPO and other paradigms. Our main contributions are:

- We introduce model-agnostic optimization of gradient projections that enhances communication and computation efficiency, boosts performance through exploration, and offers more flexibility in balancing communication and error rate.
- We provide theoretical analysis for MAPO convergence behavior, and establish its computation efficiency compared to layer-wise factorization with the same communication and error rates.
- We conduct extensive experiments across diverse datasets, model architectures, and baselines, demonstrating that MAPO surpasses existing methods in full training and fine-tuning scenarios.

2.2 Background and Related Works

In this section, we review key CEFL approaches in relation to MAPO. We begin with sketched update techniques that project model updates into subspaces, outlining their limitations. Then, we examine structured update methods, particularly projection optimization, highlighting the unique opportunities and challenges introduced by operating within a fixed subspace.

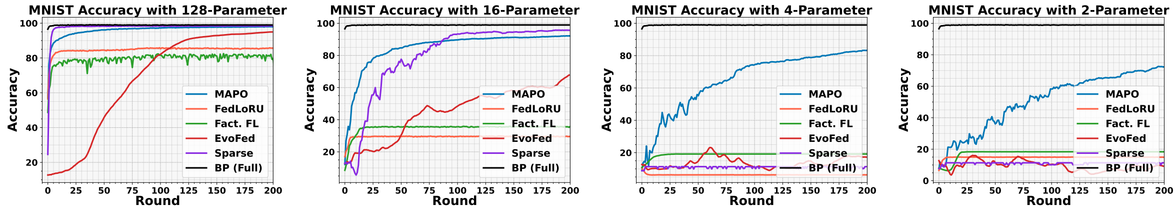


Figure 2.2: MNIST performance for varying trainable parameters.

2.2.1 Sketched update vs. Structured update

Sketched update includes techniques such as sparsification [1], quantization [4, 5, 81–85], gradient subspace projection [86–88], and random subspace projection [80, 89]. They aim to compress the information in the update vector $\Delta W \in \mathbb{R}^d$ defined as the difference between the locally optimized and the global model $\Delta W = W^* - W_g$, where W^* can be the result of multiple local epochs.

The subspace projection process [89–92] defines a random matrix $A \in \mathbb{R}^{p \times d}$, and finds the projection vector $B \in \mathbb{R}^p$, which minimizes the reconstruction error $\|\Delta W - BA\|_2$, where d denotes the total number of model parameters and $p \ll d$ is compressed length:

$$B^* = \arg \min_{B \in \mathbb{R}^p} \|\Delta W - BA\|_2 \quad ; \quad B^* \approx \Delta W \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1}.$$

As the matrix A is considerably large ($p \times d$), various methods propose novel designs for A to adapt it for large-scale models. Notably, defining A as a subset of seen gradient vectors results in a significantly lower rank of A suffices for an effective projection [86–88]. More recently, EvoFed [80] utilizes evolutionary strategies to evolve A, improving its representation and efficiency.

Sketching Limitations. Although sketched methods benefit from a full-rank training, their shortcoming is blindness to the loss surface $\mathcal{L}(W; \mathcal{D})$ and alternative solutions besides ΔW that can be reconstructed from the projection subspace. They typically perform well, given a sufficient communication budget, but as the compression rate increases, the reconstruction of the projection vector ends up far off from ΔW. In contrast, subspace optimization directly finds the steepest direction within the subspace, leading to a more effective reduction in loss. Figure 2.2 presents an example of centralized MNIST training, illustrating the performance degradation of sketched update techniques such as EvoFed [80] and Top-k Sparsification [1] compared to MAPO. As sparsity increases, MAPO continues to converge, even having 2 or 4 trainable parameters out of 11,274.

Structured update techniques reduce the number of trainable parameters and communication cost by constraining the weights or gradients to a low-rank subspace by structural modification such as pruning [93–96], weight–sharing [97–99], low-rank gradient [65–69], and parameter decomposition [70–74], including LoRA and its variants [59, 75–78]. Although parameter decomposition techniques reduce the model size and representation, resulting in subpar performance for general training, as shown in Figure 2.2 for Factorized-FL [72]. Therefore, CEFL generally adopts a gradient decomposition direction. In particular, gradient decomposition methods with freezing A, also known as *projection optimization*, remain popular owing to strong theoretical foundations, reduced communication, and hardware friendliness [60–64].

Prior works on gradient decomposition relied on each layer's shape and architecture, producing a unique A_i and B_i matrices for each layer, limiting the feasibility of sharing a projection matrix A across layers. MAPO overcomes this limitation by evenly partitioning the whole model gradient vector $\Delta W \in \mathbb{R}^d$

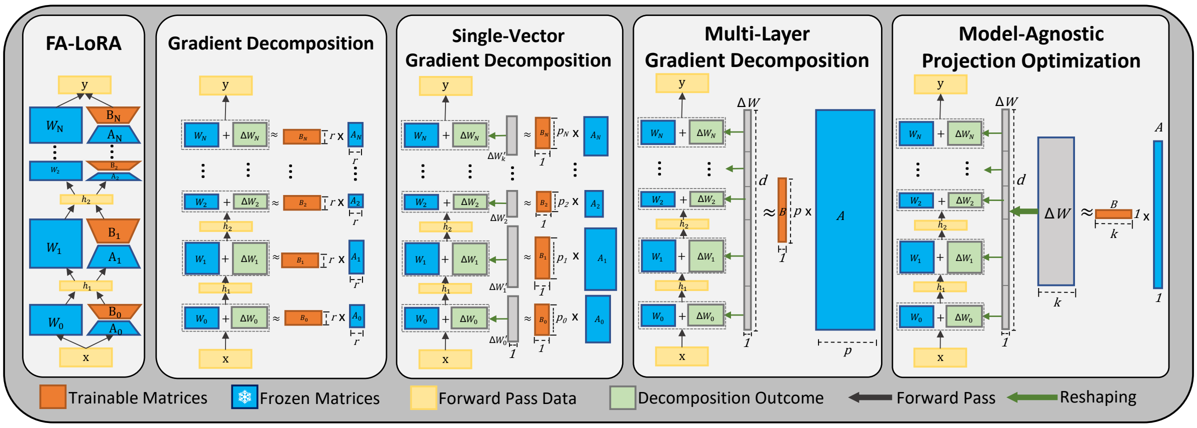


Figure 2.3: Step-by-Step illustration of methodology based on propositions, demonstrating how each step will contribute to designing MAPO factorization and differing from LoRA architecture.

into k segments $\{\Delta W_i'\}_{i=1}^{k} \in \mathbb{R}^{k\times\lceil d/k\rceil}$, allowing the use of a shared random reconstruction matrix $A \in \mathbb{R}^{1\times\lceil d/k\rceil}$ across all partitions, maintaining the benefits of model-wide projection while substantially reducing memory costs.

2.2.2 Parameter-efficiency vs. Communication-efficiency

Despite their apparent similarities, parameter decomposition and gradient decomposition methods differ fundamentally in assumptions and objectives. Parameter decomposition directly imposes a low-rank structure on the model parameters, effectively replacing the original model with a compressed version. Although this reduces the total number of parameters and computational overhead, it still requires transmitting all parameters at each communication round, resulting in no relative reduction in communication per parameter. In contrast, gradient decomposition methods maintain the original model architecture and computational complexity but substantially reduce communication overhead by transmitting compressed updates that are significantly smaller than the full model.

In this work, to ensure a fair assessment of communication efficiency, we evaluate MAPO against gradient-based compression baselines under consistent model architectures. Additional experiments with parameter decomposition and LoRA-based methods are provided in Sections B.2 and B.3 for completeness. Key methodological distinctions among related works are summarized in Table 2.1.

Table 2.1: Summary of CEFL methods and objectives. The column "Comm. Flex" indicates support for arbitrary bitrates, and "Agg. Eq." denotes equivalence between low-rank and full-rank averaging.

Method	Scope	Target	Full-rank Inference	Agg. Eq.	PEFT	Fixed Subspace	Arch-Agnostic	Comm Flex	Personalized FL
Sparsification [1]	Model	Update	✓	✓	✗	✗	✓	✓	✗
Quantization [84]	Model	Update	✓	✓	✗	✗	✓	✓	✗
EvoFed [80]	Model	Update	✓	✓	✗	✓	✓	✓	✗
Factorized-FL [72]	Layer	Parameter	✗	✗	✗	✗	✗	✗	✓
LoRA [59]	Layer	Adapter	✗	✗	✓	✗	✗	✗	✗
FA-LoRA [75]	Layer	Adapter	✗	✓	✓	✓	✗	✗	✗
SA-LoRA [79]	Layer	Adapter	✗	✗	✓	✗	✗	✗	✓
FedLoRU [67]	Layer	Gradient	✓	✓	✗	✓	✗	✗	✗
MAPO (Ours)	Model	Gradient	✓	✓	✗	✓	✓	✓	✗

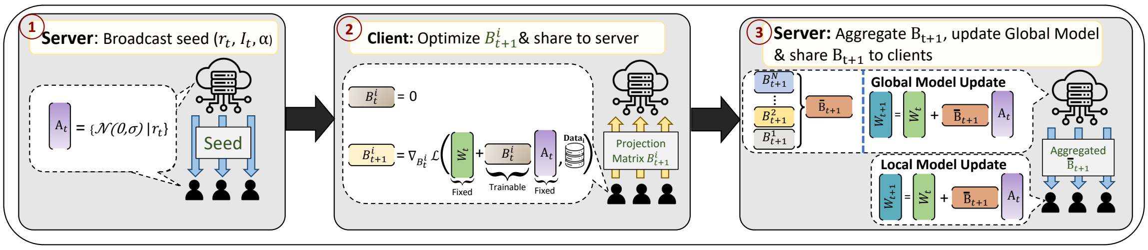


Figure 2.4: Application of MAPO to communication-efficient FL.

2.3 Proposed Method

In this section, we introduce MAPO and its application in FL. We first present the MAPO factorization technique and discuss its key properties regarding communication efficiency and error rate. Subsequently, we detail how MAPO can be effectively integrated into the FL training process.

2.3.1 Model-Agnostic Projection Optimization (MAPO)

MAPO Description. MAPO performs a black-box, model-agnostic factorization of the global model gradient $\Delta W \in \mathbb{R}^d$, avoiding architecture-specific constraints and enabling continuous subspace exploration during optimization. Specifically, MAPO partitions ΔW into k segments $\{\Delta W'_i\}_{i=1}^k \in \mathbb{R}^{k \times \lceil d/k \rceil}$ and employs a shared random reconstruction matrix $A \in \mathbb{R}^{1 \times \lceil d/k \rceil}$ across all partitions. This design preserves model-wide projection benefits while substantially reducing memory overhead. As illustrated in Figure 2.1, MAPO reshapes the universal gradient $\Delta W \in \mathbb{R}^{d \times 1}$ into $\Delta W' \in \mathbb{R}^{k \times \lceil d/k \rceil}$, which is then decomposed into a reconstruction vector A and a projection vector $B \in \mathbb{R}^{k \times 1}$. Figure 2.3 shows a step-by-step visualization analogous to Theorems 2.3.4 to 2.3.6.

MAPO Properties. MAPO aims to construct an expressive subspace, enabling a small B to encode sufficient information for updating the model efficiently. First, we formally define the concepts of communication overhead rate and reconstruction error rate in the context of matrix factorization in Theorems 2.3.2 and 2.3.3. Using these definitions, Theorem 2.3.4 establishes that reshaping a single layer preserves both the factorization error and communication rates. Extending this, Theorem 2.3.5 demonstrates that vectorizing multiple layers into a single matrix similarly maintains these properties. Finally, this leads to the proof of Theorem 2.3.6, which introduces a computationally and communication-efficient, model-agnostic factorization method as an alternative to traditional layer-wise gradient projection techniques. Section B.6 presents the formal proofs.

Assumption 2.3.1 (Gaussian Matrices are Full Rank). Let $A \in \mathbb{R}^{m \times n}$ be a random matrix with entries drawn independently from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$. Then, A is almost surely of full rank, i.e., $\text{rank}(A) = \min(m, n)$, as the probability of A being rank deficient is zero. This result follows from standard properties of random matrices [100, 101].

Definition 2.3.2 (Communication Overhead Rate). Let $\Delta W_i \in \mathbb{R}^{d_1 \times d_2}$ be the update matrix of a model. Suppose the factorization of ΔW_i as $\Delta W_i = B_i A_i$, where $A_i \in \mathbb{R}^{q \times d_2}$ is a fixed random matrix and $B_i \in \mathbb{R}^{d_1 \times q}$ is a trainable matrix with $q \leq \min(d_1, d_2)$ being the factorization rank. The **communication overhead rate** CO_{rate} is defined as the ratio of the size of B_i to the size of ΔW:

$$\text{CO}_{rate} = \frac{\text{size}(B_i)}{\text{size}(\Delta W_i)} = \frac{q}{d_2}.$$

Definition 2.3.3 (Reconstruction Error Rate). Using the same factorization as Theorem 2.3.2, the *reconstruction error rate* is the expected ratio of the reconstruction error to the original model update. Given full-rank random reconstruction (Theorem 2.3.1), it is expressed as:

$$\frac{\mathbb{E}_{A_i}\left[\|\Delta W_i - B_i A_i\|_2^2\right]}{\|\Delta W_i\|_2^2} = 1 - \frac{q}{d_2}.$$

Proposition 2.3.4 (Single-Vector Factorization). *Let ΔW_i, A_i, and B_i be factorizations of a single layer of the network as in Theorem 2.3.2. By reshaping ΔW_i into $\Delta W_i' \in \mathbb{R}^{1\times d_1 d_2}$ the factorization of $\Delta W_i' = B_i' A_i'$ where $A_i' \in \mathbb{R}^{p\times d_1 d_2}$ and $B_i' \in \mathbb{R}^{1\times p}$ can achieve the same **reconstruction error** and **communication overhead** to the conventional factorization of ΔW_i when $p = q d_1$.*

Proposition 2.3.5 (Multi-Layer Factorization). *Let ΔW_i, A_i, and B_i be **single-vector factorization** of i-th layer of the N-layered network as in Theorem 2.3.4. By concatenating the reshaped weights ΔW_i into $\Delta W' \in \mathbb{R}^{1\times d}$, where $d = \sum_{i=1}^{N} d_1^i d_2^i$. The factorization of $\Delta W' = B' A'$ where $A' \in \mathbb{R}^{p\times d}$ and $B' \in \mathbb{R}^{1\times p}$ can achieve the same **reconstruction error** and **communication overhead** to the single-vector factorization applied to each ΔW_i when $p = N q d_1$.*

Proposition 2.3.6 (MAPO Factorization). *Let ΔW, A, B, and rank p be a multi-layer factorization of a network as defined in Theorem 2.3.5. By reshaping $\Delta W \in \mathbb{R}^{1\times d}$ into $\Delta W' \in \mathbb{R}^{k\times\lceil d/k\rceil}$, and the factorization of $\Delta W' = B' A'$ where $A' \in \mathbb{R}^{1\times\lceil d/k\rceil}$ and $B' \in \mathbb{R}^{k\times 1}$, we can achieve the same **reconstruction error** and **communication overhead** to the multi-layer factorization of ΔW when $k = p$,* while reducing the memory by a factor of k^2.

2.3.2 Application to Communication-Efficient FL

This subsection explains how our method, outlined in Section 2.3.1, is utilized in FL. The procedure pseudo-code is provided in Algorithm 2, and visualized in Figure 2.4.

Matrix Construction and Broadcasting. To ensure consistency across the network, the server and all clients start from an identical condition at each round. We guarantee identical model parameters W_t and reconstruction matrix A_t by broadcasting a random seed r_t and the aggregated projection vector $\overline{B}_t$ at the beginning of round t. The initial aggregated projection vector is set to $\overline{B}_0 = \mathbf{0}$.

In the first round $(t = 0)$, all clients and the server initialize the model W^0 using the same seed. The reconstruction matrix $A^0 \in \mathbb{R}^{1\times\lceil d/k\rceil}$ is drawn from Gaussian $A^0 \sim \mathcal{N}(0, I)$, and the client j's projection vector $B^{0,j} \in \mathbb{R}^{k\times 1}$ is set to 0 for all $1 \leq j \leq M$, where M is the total number of clients.

In subsequent rounds $(t \geq 1)$, clients update their local model W^t using the previous round's matrix A^{t-1}, the model parameters W^{t-1}, and the broadcasted projection vector $\overline{B}^t$ as follows:

$$W^t = W^{t-1} + \mathbf{vec}(\overline{B}^t A^{t-1})_{[0:d]}, \tag{2.1}$$

where $\mathbf{vec}(\cdot)$ and $(\cdot)_{[0:d]}$ denotes vectorization and truncating to the first d elements. Clients then regenerate $A^t \sim \mathcal{N}(0, I)$ using the seed r^t and reset $B^{t,j} \leftarrow \mathbf{0}$, ensuring A^t and W^t synchronization.

Algorithm 2 Federated Learning with MAPO

Input: Initial seed r^0, global model W^0, reconstruction matrix A^0, projection vector $\overline{B}^0$
Initialize $W^0 \in \mathbb{R}^d$, $A^0 \in \mathbb{R}^{1\times\lceil d/k\rceil}$, $\overline{B}^0 \leftarrow \mathbf{0} \in \mathbb{R}^{k\times 1}$
for $t = 1$ to $T-1$ **do**
    Server: Broadcast $\overline{B}^{t-1}$ and seed r^{t-1} to all clients
    for all clients $j = 1$ to M (in parallel) **do**
        Receive $\overline{B}^{t-1}$ and r^{t-1}
        Update local model: $W^t \leftarrow W^{t-1} + \mathbf{vec}(\overline{B}^t A^{t-1})[0:d]$
        Re-generate $A^t = \mathcal{N}(0, \sigma^2 I_d) \mid r^{t-1}$
        Initialize $B^{t,j} \leftarrow \mathbf{0} \in \mathbb{R}^{k\times 1}$
        for $e = 1$ to E **do**
            Compute gradient: $\nabla B^{t,j} \leftarrow \nabla_{B^{t,j}} \mathcal{L}^j(W^t + \mathbf{vec}(B^{t,j}A^{t-1})[0:d], \mathcal{D}^j)$
            Update projection vector: $\hat{B}^{t,j} \leftarrow B^{t,j} - \eta \nabla B^{t,j}$
            $B^{t,j} \leftarrow \hat{B}^{t,j}$
        end for
        Send $\hat{B}^{t,j}$ to the server
    end for
    Server: Re-generate $A^t = \mathcal{N}(0, \sigma^2 I_d) \mid r^{t-1}$
    Aggregate: $\overline{B}^t \leftarrow \frac{1}{S}\sum_{j=1}^M b_j \hat{B}^{t,j}$, where $S = \sum_j b_j$
    Update global model: $W^{t+1} \leftarrow W^t + \mathbf{vec}(\overline{B}^t A^{t-1})[0:d]$
    Generate new seed r^t (e.g., $r^t = \text{hash}(r^{t-1})$)
end for
Return W^T

Local Projection Optimization. This step optimizes the projection $\hat{B}^{t,j}$ to minimizes the client loss $\mathcal{L}(W^t + \mathbf{vec}(B^{t,j}A^{t-1})_{[0:d]}, \mathcal{D}^j)$, where $\mathcal{D}^j$ denotes client j's local dataset, and model weights are derived as $W^t + \mathbf{vec}(B^{t,j}A^t)_{[0:d]}$ given the random matrix A^t.

At each communication round $t \geq 1$, after initializing A_t and $B^{t,j}$, clients perform local training to optimize $B^{t,j}$ using their local data $\mathcal{D}^j$. The gradient of the projection vector is computed as:

$$\nabla B^{t,j} = \nabla_{B^{t,j}} \mathcal{L}^j(W^t + \mathbf{vec}(B^{t,j}A^{t-1})_{[0:d]}) \quad \textbf{for} \quad \mathcal{L}^j(W) = \frac{1}{|\mathcal{D}^j|}\sum_{x\in\mathcal{D}^j} \ell(W, x). \tag{2.2}$$

where $\ell(W, x)$ is the loss function (e.g., cross-entropy loss) given model W and data point x. Therefore, given the learning rate η, only the projection $\hat{B}^{t,j}$ is updated using gradient descent as:

$$\hat{B}^{t,j} \leftarrow B^{t,j} - \eta \nabla B^{t,j}, \tag{2.3}$$

After optimization, clients send their optimized projection vector $\hat{B}^{t,j}$ to the server. The low dimensionality of $\hat{B}^{t,j}$ compared to W^t results in communication efficiency.

Server-Side Aggregation and Global Model Update. Upon receiving the projection vectors $\hat{B}^{t,j}$ and their corresponding weights $b^j = |D^j|$ (e.g., batch sizes or number of local samples) from the clients, the server aggregates them to form the global projection vector:

$$\overline{B}^t = \frac{1}{S}\sum_{j=1}^M b^j \hat{B}^{t,j}\ , \quad \text{for} \quad S = \sum_{j=1}^M b_j \tag{2.4}$$

Table 2.2: Summary of datasets and models used in our experiments.

Dataset	Client Distribution	Train/Test	# Classes	Model	# Parameters
MNIST [53]	Non-IID (2 classes)	60K / 10K	10	CNN - 2 Layers	11,274
FMNIST [52]	Non-IID (2 classes)	60K / 10K	10	CNN - 2 Layers	11,274
CIFAR-10 [102]	Non-IID (2 classes)	50K / 10K	10	CNN - 4 Layers	1,146,634
CIFAR-100 [102]	Non-IID (10 classes)	50K / 10K	100	WideResNet 16d4w	2,854,420
TinyImageNet [103]	Non-IID (10 classes)	100K / 10K	200	WideResNet 16d4w	2,880,120
Shakespeare [104]	Distributed by Roles	14K / 2K	65	LSTM	814,957
Sentiment140 [104]	Distributed by Users	1.4M / 200K	2	Transformer	2,221,570
GLUE Tasks [105]	Non-IID	*differ per task*	*differ per task*	RoBERTa-Large	357,199,876

This weighted averaging captures the collective contribution of all clients, proportional to their data sizes. The server then broadcasts the aggregated projection vector $\overline{B}^t$ to all clients. After receiving $\overline{B}^t$, the server and all clients update their local models using the reconstruction matrix A^t and the aggregated projection vector $\overline{B}^t$ as:

$$W^{t+1} = W^t + \mathbf{vec}(\overline{B}^t A^{t-1})_{[0:d]}. \tag{2.5}$$

This update integrates the clients' optimized directions into their local models and ensures synchronization across the network. This process is repeated until the global model converges.

2.4 Convergence Analysis

We analyze the convergence behavior of FL with MAPO.

Assumption 2.4.1. For each j, $\mathcal{L}^j(v)$ is β-smooth, i.e., $\left\|\nabla\mathcal{L}^j(u) - \nabla\mathcal{L}^j(v)\right\| \leq \beta\|u - v\|$ for any u, v.

Assumption 2.4.2. Variance of the stochastic gradient of D^j is bounded for each client j, i.e.,

$$\mathbb{E}\left[\left\|\nabla\mathcal{L}^j(W) - \widetilde{\nabla}\mathcal{L}^j(W)\right\|^2\right] \leq \sigma_l^2$$

.

Theorem 2.4.3. *Let the learning rate satisfy $\eta_t \leq \frac{1-4\epsilon}{4\beta(1+\epsilon)}$. Then, the algorithm achieves the bound:*

$$\frac{1}{4H_T}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\left[\left\|\nabla\mathcal{L}(W^t)\right\|^2\right] \leq \frac{\mathbb{E}\left[\mathcal{L}(W^0)\right] - \mathcal{L}^*}{H^T} + 2(\epsilon + \beta + \beta\epsilon)\sigma_l^2\frac{1}{H^T}\sum_{t=0}^{T-1}\eta_t^2,$$

where $H_T = \sum_{t=0}^{T-1}\eta_t$, ϵ is JL Lemma distortion parameter, and $\mathcal{L}^*$ is the minimum value of $\mathcal{L}(W)$.

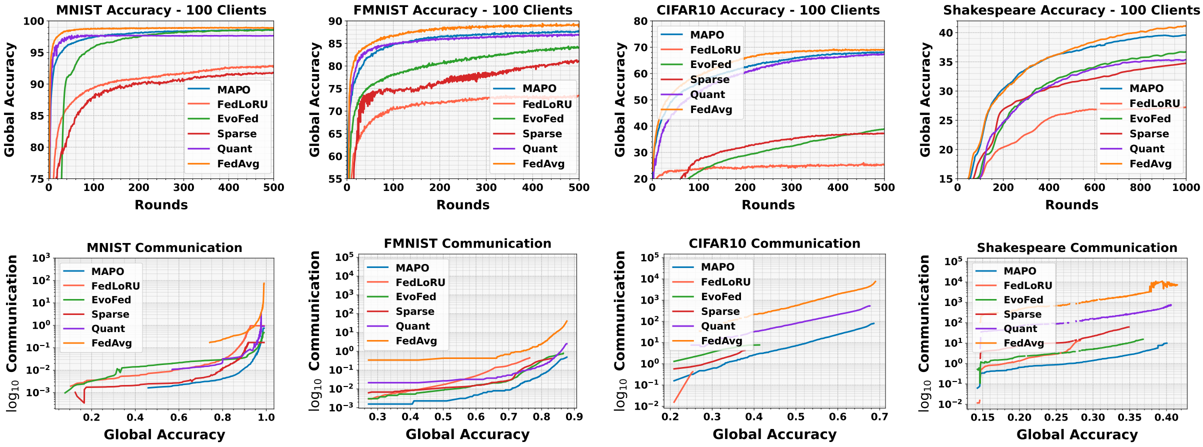
With a decreasing learning rate satisfying $\sum_{t=0}^{\infty}\eta_t \to \infty$, $\sum_{t=0}^{\infty}\eta_t^2 < \infty$ ($\eta_t = \frac{\eta_0}{t+c}$ for some constants $\eta_0 > 0$, $c > 0$), the term $H_T = \sum_{t=0}^{T-1}\eta_t$ grows unbounded, while the weighted sum $\sum_{t=0}^{T-1}\eta_t^2$ remains finite. Therefore, the right-hand side of Theorem 2.4.3's bound satisfies:

$$\frac{\mathbb{E}[\mathcal{L}(W^0)] - \mathcal{L}^*}{H_T} \to 0, \quad \frac{1}{H_T}\sum_{t=0}^{T-1}\eta_t^2 \to 0 \quad \text{as} \quad T \to \infty.$$

Thus, confirming convergence to a stationary point, as the gradient norm average satisfies:

$$\frac{1}{H_T}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\left[\|\nabla\mathcal{L}(W^t)\|^2\right] \to 0,$$

As shown above, the convergence bound is influenced by the factor $\epsilon+\beta+\beta\epsilon$. In particular, the bound becomes tightest and achieves the highest communication efficiency when there is no reconstruction error, i.e., when $\epsilon = 0$. The complete proof of Theorem 2.4.3 is located in Section B.7.



(a) MNIST (b) FMNIST (c) CIFAR-10 (d) Shakespeare

Figure 2.5: **Performance comparison** of all methods on MNIST, FMNIST, CIFAR-10, and Shakespeare datasets. The top row shows the accuracy, while the bottom row illustrates the communication cost per accuracy.

Table 2.3: Summary of maximum accuracy (%) and communication cost (% relative to FedAvg). Accuracy values report mean (±std) over 3 runs, estimated from observed variance.

	MNIST		FMNIST		CIFAR-10		CIFAR-100		Shakespeare		Sent140		TinyImageNet	
Method	Com.	Acc.	Com.	Acc.	Com.	Acc.	Com.	Acc.	Com.	Acc.	Com.	Acc.	Com.	Acc.
FedAvg	100	98.9 (±0.1)	100	89.2 (±0.2)	100	69.0 (±0.2)	100	43.47 (±0.3)	100	41.86 (±0.3)	100	74.90 (±0.3)	100	36.48 (±0.4)
Sparse	15.3	92.1 (±0.4)	24.1	81.1 (±0.4)	2.7	37.15 (±0.5)	1.20	33.72 (±0.5)	1.73	34.86 (±0.4)	1.93	74.21 (±0.3)	1.32	25.34 (±0.5)
Quantize	31.3	97.6 (±0.2)	24.1	87.1 (±0.3)	15.2	67.40 (±0.3)	6.10	40.05 (±0.4)	10.11	35.45 (±0.4)	13.85	73.70 (±0.3)	8.75	34.47 (±0.4)
EvoFed	9.40	98.5 (±0.2)	7.60	84.7 (±0.3)	3.4	39.50 (±0.4)	20.4	37.62 (±0.4)	0.23	36.76 (±0.3)	0.40	70.50 (±0.3)	1.85	15.40 (±0.5)
FedLoRU	30.2	93.8 (±0.4)	17.9	74.1 (±0.5)	1.7	23.52 (±0.5)	1.20	19.10 (±0.5)	1.67	28.07 (±0.5)	1.30	66.61 (±0.4)	1.27	7.31 (±0.5)
MAPO	**2.95**	**98.6 (±0.1)**	**3.10**	**88.0 (±0.2)**	**1.20**	**68.3 (±0.2)**	**0.91**	**40.16 (±0.3)**	**0.13**	**39.96 (±0.3)**	**0.19**	**74.50 (±0.2)**	**0.97**	**35.22 (±0.3)**

2.5 Experimental Setup

We evaluate MAPO across diverse model architectures, tasks, and baselines. The benchmarks span five image classification datasets—MNIST [53], FMNIST [52], CIFAR-10, CIFAR-100 [102], and TinyImageNet [103]—as well as sequential tasks, including next-character prediction on Shakespeare and sentiment analysis on Sentiment140, both drawn from the LEAF benchmark suite [104], tailored for FL. Additionally, we evaluate MAPO as a fine-tuning method, alongside LoRA baselines on various GLUE [105] tasks, highlighting the communication and computation efficiency in Section B.2. The dataset specifications and corresponding model architectures are summarized in Table 2.2, highlighting MAPO's adaptability across varying data modalities, model scales, and application domains.

Non-IID Distribution. To simulate realistic FL conditions, we partition the training datasets in a non-IID manner across 100 clients. For image classification and GLUE tasks, each client is assigned a distinct subset of classes. For LEAF tasks, we follow the natural user-based partitioning, where individual Shakespearean roles and Twitter users correspond to separate clients.

Model Architectures. We evaluate MAPO across diverse architectures of varying complexity, including CNNs (2-layer for MNIST and FMNIST; 4-layer for CIFAR-10), WideResNet (width 4, depth 16) for CIFAR-100 and TinyImageNet, LSTM for next-character prediction, Transformer for sentiment analysis, and RoBERTa for GLUE tasks. Detailed architecture specifications and hyperparameters are in Section B.4.

Baselines. We compare MAPO against multiple baselines, including standard compression methods with Top=k subsampling (Sparse) [1], and quantization (Quant) [84]. Additionally, we evaluate MAPO against EvoFed [80], a state-of-the-art gradient compression, and FedLoRU [67], a representative gradient projection approach. Subsampling and quantization serve as references to establish MAPO's performance compared to conventional compression techniques. EvoFed provides a strong comparison to demonstrate the effectiveness of MAPO's subspace optimization relative to methods applying compression post-optimization. FedLoRU allows us to highlight MAPO's dynamic subspace exploration and its benefits over static layer-wise gradient projections. Results comparing MAPO with additional parameter-factorization (Factorized-FL [72]) and adapter-based fine-tuning baselines (LoRA [59], FA-LoRA [75], and SA-LoRA [79]) are included in Sections B.2 and B.3.

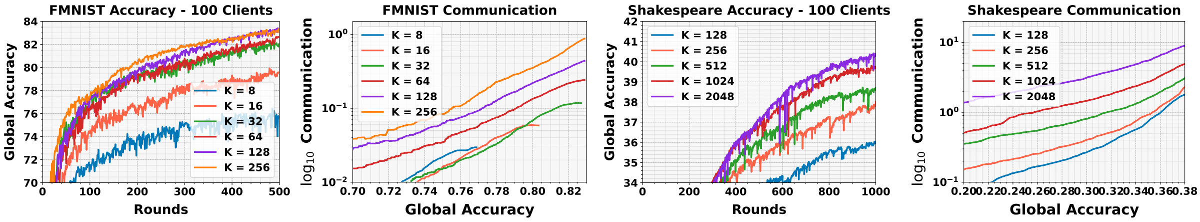
Federated Learning Setting. In each training round, 10% of the clients are randomly selected to participate. Selected clients train locally in parallel and transmit their updates to the central server, which aggregates these updates and redistributes the resulting global model back to the clients. Model performance is evaluated centrally using the test dataset at the server.

2.6 Results and Discussions

We now discuss our experimental results in detail and provide insights into MAPO's performance. Figure 2.5 (top row) shows the accuracy of MAPO compared to multiple baseline methods across various datasets. MAPO consistently outperforms all other methods and achieves accuracy comparable to FedAvg, despite transmitting only a fraction of the parameters. This improvement results from MAPO's dynamic subspace optimization, which promotes effective exploration and efficient use of the communication budget to minimize the loss function directly. Additionally, Figure 2.5 (bottom row) illustrates the minimal communication cost required by each method to reach a given accuracy level, highlighting MAPO's significantly lower communication demands (logarithmic scale on the y-axis). Additional results on CIFAR-100, TinyImageNet, and Sentiment140 are presented in Section B.1.

Table 2.3 summarizes experimental results by comparing the maximum accuracy of each baseline and their communication cost relative to FedAvg. To ensure fair comparison, communication costs are reported as the percentage required to reach the accuracy of the worst-performing baseline. MAPO consistently achieves competitive accuracy with significantly lower communication overhead. Specifically, on MNIST and FMNIST, MAPO achieves 99.6% and 98.6% of FedAvg's accuracy, respectively, using only 3% of FedAvg's communication cost. For CIFAR-10, CIFAR-100, and TinyImageNet, MAPO attains 98.9%, 92.4%, and 96.5% of FedAvg accuracy, respectively, while consuming approximately 1% of the communication. Finally, in sequential tasks (Shakespeare and Sentiment140), MAPO retains up to 95.5% and 99.5% of FedAvg's accuracy, respectively, while dramatically reducing communication to less than 0.2%.

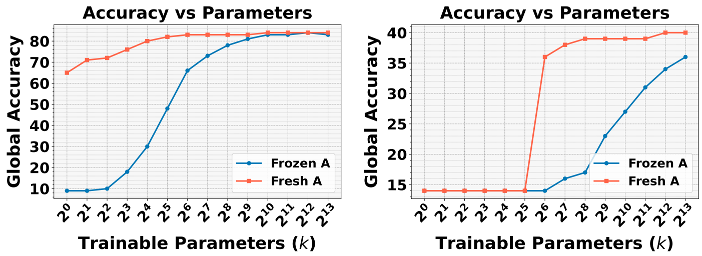
MAPO Hyperparameter. MAPO simplifies gradient projection by applying a single factorization across all model parameters, thus replacing per-layer rank selection with a single hyperparameter, k, directly controlling communication cost and model accuracy. Figure 2.6 illustrates the effect of varying k on performance and communication efficiency for the FMNIST and Shakespeare datasets. While a smaller k significantly reduces communication overhead, it slows the convergence, requiring more training rounds. Conversely, increasing k improves convergence speed and accuracy but rapidly raises communication costs, often with diminishing returns. Therefore, the optimal k achieves a target accuracy with minimal total communication. Figure 2.6(b) and (c) show communication costs associated with



(a) FMNIST Accuracy (b) FMNIST Communication (c) Shakespeare Accuracy (d) Shakespeare Communication

Figure 2.6: Accuracy and communication cost per accuracy level for FMNIST and Shakespeare datasets. Demonstrating the effect of a number of trainable parameters (k) on the communication efficiency of MAPO.

specific accuracy levels, guiding the selection of optimal k. We use the same guidelines for all baselines to fairly tune hyperparameters.



(A) FMNIST (B) Shakespeare

Figure 2.7: Comparison of having a fresh A vs. frozen A.

Fresh Reconstruction Matrix. A key factor in MAPO's performance is using a dynamically generated reconstruction matrix A rather than a fixed one. This approach promotes the exploration of new subspaces throughout training. Figure 2.7 illustrates the benefits of using a fresh A on the FMNIST and Shakespeare datasets. We evaluate MAPO across varying numbers of trainable parameters, ranging from 2^0 to 2^{13}. For FMNIST, this corresponds to 0.009% to 72.27% of the total model parameters, while for Shakespeare, it spans from 0.0001% to nearly 1%. In both cases, MAPO with a fresh A achieves superior convergence with fewer parameters, effectively leveraging the search space. In contrast, when A is frozen, performance follows a logarithmic correlation with the number of trainable parameters, requiring an exponentially larger parameter count to match the results obtained with a fresh A.

Additional Results. Comparisons with LoRA-based methods and Factorized-FL are provided in Sections B.2 and B.3. Section B.5 supplements our main experiments with evaluations under IID distributions and without client sampling. Additionally, Section B.8 presents a detailed memory complexity analysis, emphasizing MAPO's computational efficiency and flexibility compared to layer-wise low-rank factorization.

Limitations. MAPO's improved communication efficiency comes with additional computational overhead from gradient projection optimization. While significantly reduced compared to prior methods, MAPO still requires $\lceil d/r \rceil + r$ memory and computation (instead of $dr + r$; see Section B.8). MAPO complements, but does not replace, PEFT methods like LoRA, as it reduces communication overhead without decreasing the trainable parameters or storage requirements (see Section B.2).

2.7 Conclusion

We introduced *Model-Agnostic Projection Optimization* (MAPO), a novel approach for CEFL. Unlike layer-wise decomposition, MAPO factorizes the entire gradient using a projection vector and a random reconstruction matrix, regenerated at each round. MAPO balances communication efficiency and accuracy without imposing architecture-specific constraints or fixed-subspace limitations. Our theoretical analysis establishes convergence guarantees, and empirical results demonstrate superior performance and scalability across diverse datasets, confirming its practical value for FL.

Chapter 3. Aggregation of Principals in Low-rank via SVD

3.1 Introduction

Recent advances in Pre-trained Foundation Models (PFMs) have demonstrated exceptional performance across diverse downstream applications, benefiting from massive-scale computational resources and large datasets [106–110]. However, full-rank fine-tuning (FFT) of FMs remains computationally expensive and data-intensive for users with limited resources. To overcome this, Parameter-Efficient Fine-Tuning (PEFT) approaches, notably Low-Rank Adaptation (LoRA) [59], have emerged as a practical solution, reducing both computational costs and storage requirements significantly [111]. While LoRA considerably alleviates fine-tuning overhead, individuals with limited data struggle to adapt these models effectively to their specific tasks [112].

Personalized Federated Learning (PFL) presents a viable solution, allowing clients to collaboratively fine-tune their models by leveraging distributed local datasets without compromising privacy [113]. Clients periodically upload model gradients, which the server aggregates and broadcasts back to them [2]. However, communication overhead from the frequent transmission of model gradients poses a major challenge for clients with limited resources.

Communication-Efficient Federated Learning (CEFL) [57] proposes a vast range of strategies to reduce communication load, typically categorized into *sketched updates*, which compress the gradients after optimizing the local model (e.g., sparsification, quantization [84], random projection [80]), and *structured updates*, which reduce the number of trainable parameters to a lower-dimensional subspace before optimization (e.g., random masks, weight-sharing, and low-rank factorization) [1]. On the other hand, performing a few local epochs in each round [6] can reduce the communication frequency. However, the aggregation of models after multiple local epochs results in a noticeable performance degradation due to the accumulated client drift.

Client drift is the divergence in clients' parameters trained by stochastic gradient descent (SGD) due to data heterogeneity [114–116] and overparameterization of neural networks (NN) [117–120]. To reduce client drift, early works such as FedProx [121] and SCAFFOLD [122] propose regulations on client drift by modifying the optimization objective in the clients, while improving the stability are not very effective, as they failed to address the root cause of client drift, as explained with Linear Mode Connectivity (LMC) [123].

Linear Mode Connectivity (LMC) barriers refer to the obstacles encountered when linearly interpolating (e.g., averaging) between two independently trained NN, typically resulting in suboptimal performance due to significant differences in their parameter spaces. Prior literature indicates that both the NN *permutation symmetry* [124] and *client heterogeneity* [115] lead SGD solutions to reside in distinct loss basins, creating barriers to linear connectivity and complicating model alignment. Furthermore, even when two aligned models lie within the same loss basin, linearly interpolating their weights can lead to *variance collapse* [125], resulting in a sharp loss barrier at the midpoint.

Permutation symmetry, visualized in Figure 3.1 (a), refers to the property that applying a permutation transformation P and its inverse $P^\top$ between any two consecutive hidden layers preserves the network's functionality. However, identifying the exact optimal permutation for all layers and clients that minimizes their distance involves computationally expensive pairwise matching [126], making it

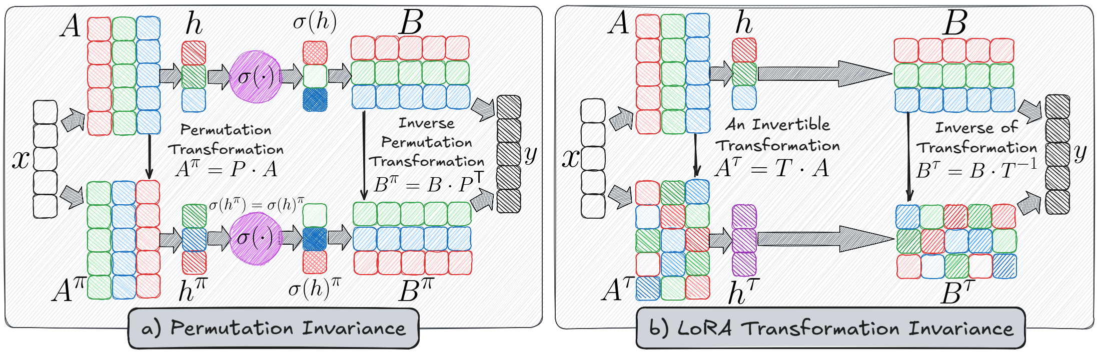


Figure 3.1: Models exhibiting identical functionality may differ significantly in parameters due to NN symmetries.

impractical for large models [3, 127], thus, optimization methods have been proposed to approximate a soft permutation [128–131]. However, prior work does not suffice for LoRA symmetry, as consequence layers without non-linearity introduce *invertible transformation symmetry*.

LoRA symmetry extends beyond permutation invariance because the low-rank decomposition $W = BA$ is inherently non-unique [132]. Specifically, there exists a basis ambiguity, as any invertible matrix T can produce an equivalent decomposition: $B \times A = BT \times T^{-1}A$, as shown in Figure 3.1 (c). To address this symmetry, methods include *freezing A* [76] and *full-rank merging* [133] are proposed.

Freezing A matrix of LoRA and limiting the training of only to B matrix [76], avoids the symmetry problem as LoRA aggregation of N clients will be equivalent to full-rank aggregation:

$$\frac{1}{N}\sum_{i=0}^{N}(B_i A^*) = \Big(\frac{1}{N}\sum_{i=1}^{N} B_i\Big)\Big(\frac{1}{N}\sum_{i=1}^{N} A^*\Big).$$

However, freezing A limits the representation of LoRA and shows suboptimal convergence [79].

Full-rank merging reconstructs and aggregates the full-rank representation $W_i = B_i \times A_i$, followed by a low-rank SVD decomposition to find the resulting low-rank $\bar{A}$ and $\bar{B}$ matrices [133]:

$$\bar{A} = \sqrt{S}V^{\top} \quad ; \quad \bar{B} = U\sqrt{S} \quad ; \quad USV^{\top} = \textbf{SVD}(\overline{W}) \quad ; \quad \overline{W} = \frac{1}{N}\sum_{i=0}^{N} W_i$$

However, full-rank merging introduces expensive reconstruction and decomposition of $\overline{W}$, and it is impractical for heterogeneous scenarios with partial parameter sharing and aggregation [79, 134].

Client heterogeneity refers to scenarios where clients train on vastly different tasks or data distributions, resulting in divergent loss surfaces whose basins do not align [135]. In such cases, interpolating between client models often leads to degraded performance across all tasks, rendering collaborative training counterproductive [3, 128–131]. Nevertheless, while the full parameter spaces may remain misaligned, several works have shown that clients can still share a common low-dimensional subspace where their loss basins overlap [136–140]. Interpolation within these subspaces can capture shared knowledge, while the remaining parameters are preserved as task-specific components, enabling personalization without harming global collaboration. Common approaches to build a shared subspace are using the same backbone model with personalized heads [136, 137] or limiting the aggregation within aligned features [131, 139], linearly connected [138], or correlated components [140].

LoRA Asymmetry suggests that matrices A and B play distinct roles in training. Specifically, the optimal choice of A^* is shown to be independent of the data distribution, implying that A captures *task-agnostic* general knowledge, while B encodes *task-specific* personalized preferences [77]. Building upon this, FedSA-LoRA [79] proposes a personalized FL approach that addresses client heterogeneity by aggregating only the matrix A (for common knowledge) while personalizing matrix B, reducing the communication cost and outperforms existing baselines. However, it does not address the LoRA symmetry and maintains training consistency by limiting each round to only a few local steps.

In this work, we introduce **APriLS** (Aggregation of Principals in Low-rank via SVD), a novel aggregation method that implicitly aligns client models by approximating full-rank aggregation without reconstructing full-rank weights. APriLS follows a two-stage process: it first aligns and aggregates the A_i matrices, then determines and aggregates the corresponding B_i matrices. In the PFL setting, where only A_i matrices are communicated, A alignment occurs on the server, while B alignment is performed locally on each client.

Our approach mitigates the LoRA misalignment and transformation symmetry in low-rank representation, and extensive theoretical analysis and empirical validation demonstrate that APriLS significantly reduces communication frequency, enhances convergence stability, and provides robust personalization capabilities. The primary contributions of our work are:

- **Principal-Aligned LoRA Mechanism:** We propose APriLS, an innovative aggregation technique using SVD on LoRA's low-rank parameter matrices (A), aligning client models implicitly without requiring full-rank reconstruction, external datasets, and averaging.
- **Enhanced Communication Efficiency:** Our federated protocol substantially reduces communication frequency by enabling extended local training intervals.
- **Rigorous Convergence Analysis:** We provide a comprehensive theoretical analysis, establishing improved convergence guarantees and bounds.
- **Extensive Empirical Validation:** Our method is rigorously evaluated through comprehensive experiments on GLUE benchmarks, affirming APriLS's superior performance in accuracy, communication reduction, and personalization efficacy.

3.2 Related work

In this section, we outline the recent studies in PEFT of FMs and integration of this method in FL settings, particularly focusing on communication-efficacy and personalization. We further discuss the alignment problem and the limitations of current solutions in PFL for the LoRA architecture, highlighting the importance of an aggregation method that works to mitigate the alignment problem directly in the low-rank space.

Pre-Trained Foundation Model refers to large-scale pretrained models trained on broad data that can be adapted to many tasks [141]. Early breakthroughs like BERT (110M parameters) [142] and GPT-2 paved the way, but it was GPT-3 (175B) that epitomized the leap in scale. GPT-3 demonstrated astonishing few-shot learning capabilities by its size [107]. Subsequent models such as LLaMA (7B–65B) showed that state-of-the-art performance is possible even with open models, given enough training data [143]. These PFMs achieve broad knowledge but are extremely expensive to train and deploy, and fine-tuning such models for each downstream task is often infeasible [144].

Parameter-Efficient Fine-Tuning strategies have evolved to address the above issues, allowing for the adaptation of large models by training only a few additional parameters [145–149]. LoRA is a prominent PEFT technique, especially for large Transformer models, that freezes the original network weights W and learns a pair of low-rank matrices B and A such that the weight update can be expressed as $\Delta W = BA$ with $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$ for $W \in \mathbb{R}^{d\times k}$ and $r \ll \min(d,k)$ [59]. In other words, rather than directly adjusting each of the $d \times k$ parameters in W, LoRA sketch ΔW in a low-dimensional subspace spanned by A matrix, resulting in B projection matrix, drastically reducing the trainable parameters and the memory needed to store updates, since only the much smaller A and B matrices are learned.

LoRA in Federated Learning. Following LoRA's success in PEFT, various studies integrated LoRA architecture in FL for communication-efficiency [150], privacy-preservation [76], and personalization [79]. However, the federated averaging [2] of LoRA faces a significant problem, as the low-rank representation of ΔW is not unique and there are infinite possibilities to describe same weight matrix as $W = \{BT \times T^{-1}A \ \forall\ A, B, T \mid det(T) \neq 0 \wedge BA = W\}$. Therefore, averaging the projection matrices B_i from misaligned subspaces A_i results in poor correlation with actual full-rank weight averaging as: $\frac{1}{N}(B_1A_1 + B_2A_2 + \cdots + B_NA_N) \neq \frac{1}{N}(B_1 + B_2 + \cdots + B_N) \times \frac{1}{N}(A_1 + A_2 + \cdots + A_N)$.

Model merging refers to various approaches that have been proposed for overcoming LMC barriers by addressing the *permutation symmetry*, *client heterogeneity*, and *variance collapse* to improve the connectivity and compatibility of independently trained models [151]. In FL, addressing permutation symmetry was first discussed in Probabilistic Federated Neural Matching (PFNM) [127] for linear models and then extended for CNN and LSTM layers in Federated Matched Averaging (FedMA) [3]. In parallel to these works, Optimal Transport (OT) [128] and Neuron Alignment [129] proposed an optimization technique to find a continuous transformation as an approximation of soft permutation. Git-Rebasin [130] estimates the permutation of the whole model layers via a fast discrete optimization, and offers a practical solution for merging large-scale models. However, despite the success of these methods, minor errors in weight reconstruction might lead to a significant difference in activations and model behavior.

Variance Collapse refers to the phenomenon where, even when two permutation-aligned models lie within the same loss basin, linearly interpolating their weights can degrade performance [152]. This occurs because matched neurons are only partially correlated, so averaging their weights reduces each channel's activation variance. The reduction intensifies with network depth, compressing the feature dynamic range, distorting normalization statistics, and causing a sharp accuracy drop at the interpolation midpoint. REPAIR addresses this issue by re-scaling each channel to a convex combination of the endpoint variances, effectively restoring performance; however, this post-hoc renormalization is applied after weight alignment and does not fundamentally resolve the underlying mismatch [125]. In contrast, activation alignment provides a more direct solution by learning a data-dependent transformation that minimizes activation differences across models. As introduced in Fed^2 [131], activation alignment avoids variance collapse altogether, since the transformation implicitly normalizes activations and preserves interpolation performance without requiring explicit variance adjustments.

Client heterogeneity is another critical factor in LMC barriers, as merging models across different tasks or data distributions leads to poor performance on all of them. ZipIt! [139] generalizes the Git-Rebasin idea that lets unmatched features stay separate and lets layers be partially merged. Canonical Correlation Analysis (CCA) [140] drops the one-to-one permutation assumption entirely by averaging correlated directions rather than individual neurons and scales easily to merging many models simultaneously. **Client heterogeneity** refers to scenarios where clients train on vastly different tasks or data distributions, resulting in divergent loss surfaces whose basins do not align [135]. In such cases, interpolating between client models often leads to degraded performance across all tasks, rendering collaborative training counterproductive [3, 127–131]. Nevertheless, while the full parameter spaces may remain misaligned, several works have shown that clients can still share a common low-dimensional subspace where their loss basins overlap [7, 136–140]. Interpolation within these subspaces can capture shared knowledge, while the remaining parameters are preserved as task-specific components, enabling personalization without harming global collaboration. Common approaches to build a shared subspace are using the same backbone model with personalized heads [136, 137] or limiting the aggregation within aligned features [131, 139], linearly connected [138], or correlated components [140]. **Merging LoRA**

layers pose new challenges as doubling the number of layers, introducing a new type of symmetry, and enforcing low-rank representation amplify the LMC barriers [153]. TIES-Merging [154] shows that including sign flipping symmetries in addition to the permutation can significantly improve the misalignment introduced by LoRA layers, LoRA-LEGO

LoRA asymmetry suggests that matrix A and B have different roles in training, as B creates new features and A mainly routes them, hence tuning or merging B matters far more and freezing a random A barely hurts accuracy [77]. Table 3.1 shows the difference between various methods introduced.

Method	Objectives (fixed = ✓)				Properties / limitations (✓ desirable)								Complexity $O(\cdot)$
	LoRA	Perm.	WtAlign	ActAlign	LowR	NoFrz	DataF	GradF	ContT	Scal$_N$	Scal$_d$	1-Shot/Inv	
FedAvg	✗	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✗	$NLrd$
FedProx	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	$NLrd$
SCAFFOLD	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	$NLrd$
FedMA	✗	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓	✗	$N^2L(r^3+d^3)$
OT	✗	✓	✓	✓	✓	✗	✗	✗	✓	✗	✓	✓	$NLr(nd+Nr)$
Git-Rebasin	✓	✓	✓	✗	✓	✗	✓	✓	✓	✗	✗	✓	$\binom{N}{2}L(r^3+d^2r)$
ZipIt!	✗	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✗	$NLrd$
REPAIR	✗	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✗	$NLrd$
CCA	✗	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✗	$NLrd\log d$
TIES-Merge	✗	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✓	$NLrd$
LoRA-LEGO	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✗	$TNLr^2d$
KnOTS	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓	✗	✓	$NL(r^2d+d^2)$
LoRM	✓	✗	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓	$NLr^2+L(r^3+rd)$
IterIS	✓	✗	✗	✓	✓	✗	✗	✗	✓	✓	✓	✓	$TnNLrd$
Mix-of-Show	✓	✗	✗	✓	✓	✗	✗	✗	✓	✓	✓	✓	$TnNLrd$
Twin-Merging	✓	✓	✓	✗	✓	✗	✓	✓	✓	✓	✗	✗	$Ld^2r+NLrd$
APriLS (ours)	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	NLr^2d+Lr^3

Table 3.1: .

3.3 Methodology

In this section, we start by formally defining the misalignment problem due to LoRA symmetry and formalize an optimal solution for aggregation to mitigate the misalignment issue for both FL and PFL settings. Subsequently, we describe the APriLS process for constructing the optimal solution and its integration within the FL process. Lastly, we provide the convergence analysis of APriLS, establishing a tighter bound for client drift that allows less frequent communication and improves the efficiency of PEFT in the FL framework.

3.3.1 Theoretical Analysis: Optimal solution and error bounds

The standard integration of LoRA within FL settings factorizes each model layer $W \in \mathbb{R}^{d\times d}$ into two matrices as $W = BA$, where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times d}$, with factorization rank $r \ll d$. During each FL round, client i uploads its local matrices A_i and B_i to the server. The server then aggregates these parameters—typically by averaging—and broadcasts the aggregated results back to clients [2]. However, separately averaging A_i and B_i does not equate to the full-rank averaging of the original layer parameters. Formally, this inequality is given by:

$$\frac{1}{N}\sum_{i=0}^{N}(B_iA_i) \neq \Big(\frac{1}{N}\sum_{i=0}^{N}B_i\Big)\Big(\frac{1}{N}\sum_{i=0}^{N}A_i\Big).$$

Even in PFL, where clients only share their A_i matrices for aggregation [72, 79], separate low-rank averaging differs from full-rank averaging. Specifically, when clients have equivalent full-rank parameters $W_i = BA\ \forall i$, one expects no change in A_i, B_i after aggregation as $W_i = \sum_{j=0}^{N} W_j/N$. However, due to the non-uniqueness of LoRA factorization (i.e., $W_i = BA = (BT_i^{-1})(T_iA)$), low-rank averaging of the $A_i = T_iA$ matrices yields:

$$W_i = BA \neq BT_i^{-1}\Big(\frac{1}{N}\sum_{j=0}^{N} T_jA\Big) = B_i\Big(\frac{1}{N}\sum_{j=0}^{N} A_j\Big).$$

Recent literature suggests solutions to avoid these issue through freezing A_i matrices in the clients (FA-LoRA) [75–78], or averaging on full-rank reconstruction of $W_i = B_iA_i$ and refactorization of $\bar{W} = \sum_{i=0}^{N} W_i/N$ into $\bar{A} = \sqrt{S}V^\top$ and $\bar{B} = U\sqrt{S}$, where $USV^\top = \mathbf{SVD}(\bar{W})$. However, the FA-LoRA approach suffers from suboptimality [79] and full-rank averaging, besides additional computation overhead, requires access to all LoRA parameters in the server, and does not apply to PFL strategies that benefit from partial parameter sharing.

To mitigate these issues, we first define the aggregation error in FL and PFL settings as the deviation from full-rank aggregation in Theorems 3.3.1 and 3.3.2. Given formal definitions, Theorem 3.3.3 shows that the expected error of FedAvg for N clients, τ local steps, and the step size of η is bounded by $\mathbb{E}[\|E\|_F^2] \leq N^{-1/2}\tau\eta\sigma_A\sigma_B$, where σ_A and σ_B assumed as the client drift bound for A and B matrices in each step. One must reduce the error bound to a manageable threshold to stabilize the training process. Therefore, the common practice is to select a modest τ and η. However, Theorems 3.3.4 and 3.3.5 establishes the optimal solution and error bound that reduce the client drifts for FL and PFL without reconstruction of $W_i = B_iA_i$, allowing a faster convergence and higher τ. The proofs for Theorems 3.3.3 to 3.3.5 are in Section C.1.

Definition 3.3.1 (Aggregation Error in FL)**.** Let $W_i = B_iA_i$ with $B_i \in \mathbb{R}^{d\times r}$ and $A_i \in \mathbb{R}^{r\times d}$ be the rank-r parameters of client i. The FL aggregation error is defined as:

$$E_{\mathbf{FL}} := \Big\|\frac{1}{N}\sum_{i=1}^{N} B_iA_i - B'A'\Big\|_F,$$

where $(B', A') \in \mathbb{R}^{d\times r} \times \mathbb{R}^{r\times d}$, denoting an aggregation outcome for B_is and A_is.

Definition 3.3.2 (Aggregation Error in PFL)**.** Following notation in Theorem 3.3.1. The PFL aggregation error, where the clients only share A_is globally, is defined as:

$$E_{\mathbf{PFL}} = \frac{1}{N}\sum_{i=1}^{N}\big\|B_iA_i - B_i'A'\big\|_F$$

where $(B_i^\star, A^\star) \in \mathbb{R}^{d\times r} \times \mathbb{R}^{r\times d}$, denoting a transformation of B_i and the aggregation of A_is.

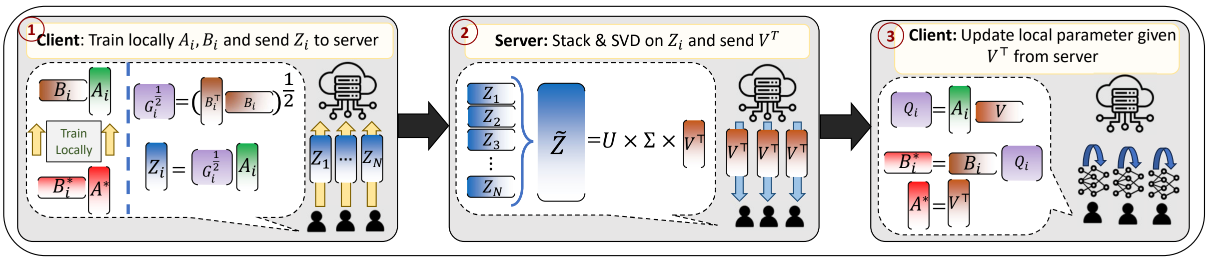


Figure 3.2: Models exhibiting identical functionality may differ significantly in parameters due to NN symmetries.

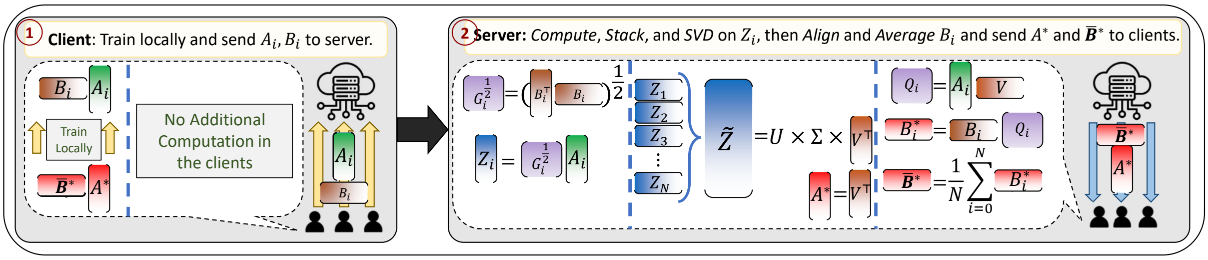


Figure 3.3: Models exhibiting identical functionality may differ significantly in parameters due to NN symmetries.

Proposition 3.3.3 (Expected error bound for factor-wise averaging). *Let $W_i = B_i A_i$ with $B_i \in \mathbb{R}^{d\times r}$ and $A_i \in \mathbb{R}^{r\times d}$ be the rank-r parameters of client i after τ local steps with stepsize η starting from a common initialization. Assume each per-step stochastic gradient is unbiased and has a finite second moment $\mathbb{E}\big[\|\nabla_A \ell_i\|_F^2\big] \le \sigma_A^2$, $\mathbb{E}\big[\|\nabla_B \ell_i\|_F^2\big] \le \sigma_B^2$. Then the expected values for $E_{\boldsymbol{FL}}$ and $E_{\boldsymbol{PFL}}$ are:*

$$\boxed{\mathbb{E}\Big[\|E_{\boldsymbol{FL}}\|_F^2\Big]^{1/2} \le \frac{\tau\eta\sigma_A\sigma_B}{\sqrt{N}}} \quad ; \quad \boxed{\mathbb{E}\Big[\|E_{\boldsymbol{PFL}}\|_F^2\Big]^{1/2} \le \frac{\tau\eta\sigma_A\sigma_B}{N}}$$

Proposition 3.3.4 (Factor-Wise Aggregation for Global FL). *Following the notation Theorem 3.3.1 after the local training, form the Gram factor $G_i := (B_i^\top B_i)^{1/2} \in \mathbb{R}^{r\times r}$ and define the weighted row-stack as:*

$$\widetilde{A} = \Big[G_1 A_1 \quad G_2 A_2 \quad \dots \quad G_N A_N\Big]^\top \in \mathbb{R}^{Nr\times d}.$$

Let the truncated rank-r SVD be $\widetilde{A} = U\Sigma V_r^\top$, with $V_r \in \mathbb{R}^{d\times r}$ whose columns are the r dominant right singular vectors. Then the optimal solution (A^*, B^*) and error bound of Theorem 3.3.1 are:

$$A^\star = V_r^\top \in \mathbb{R}^{r\times d}, \qquad B^\star = \frac{1}{N}\sum_{i=1}^{N} B_i\big(A_i V_r\big) \in \mathbb{R}^{d\times r}.$$

$$\boxed{\mathbb{E}\Big[\Big\|\frac{1}{N}\sum_{i=1}^{N} B_i A_i - B^\star A^\star\Big\|_F^2\Big]^{1/2} \le \frac{\tau\eta\,\sigma_B}{\sqrt{N}}\Big(\sum_{j>r}\sigma_j^2(\widetilde{A})\Big)^{1/2} \le \frac{\tau\eta\,\sigma_B\sigma_A}{\sqrt{N}}}$$

where $\sigma_j(\widetilde{A})$ are the singular values of $\widetilde{A}$ and $\sigma_B^2 := \frac{1}{N}\sum_{i=1}^{N}\sigma_{B,i}^2$ with $\sigma_{B,i}^2$ the second moment of the per-step gradient on B_i.

Proposition 3.3.5 (Factor-Wise Aggregation for Personalized FL). *Following the notation Definition 3.3.2 after the local training, let each client form the Gram factor $G_i := (B_i^\top B_i)^{1/2} \in \mathbb{R}^{r\times r}$ locally and uploads $G_i A_i$ to the server. We define the weighted row-stack as:*

$$\widetilde{A} = \Big[G_1 A_1 \quad G_2 A_2 \quad \dots \quad G_N A_N\Big]^\top \in \mathbb{R}^{Nr\times d}.$$

Let the truncated rank-r SVD be $\widetilde{A} = U\Sigma V_r^\top$, with $V_r \in \mathbb{R}^{d\times r}$ whose columns are the r dominant right singular vectors. The optimal A^* that is computed and broadcast by the server, and the optimal B_i^* that is calculated at the client i given A^*, with the error bound of Theorem 3.3.2 are as:

$$A^\star = V_r^\top \in \mathbb{R}^{r\times d}, \qquad B_i^\star = B_i\big(A_i A^{\star\top}\big) \in \mathbb{R}^{d\times r}.$$

$$\boxed{\mathbb{E}\Big[\, \big\|B_i A_i - B_i^\star A^\star\big\|_F^2\Big]^{1/2} \;\le\; \frac{\tau\eta\,\sigma_B}{N}\,\Big(\sum_{j>r}\sigma_j^2(\widetilde{A})\Big)^{1/2} \;\le\; \frac{\tau\eta\,\sigma_B\sigma_A}{N}}$$

where $\sigma_j(\widetilde{A})$ are the singular values of $\widetilde{A}$ and $\sigma_B^2 := \frac{1}{N}\sum_{i=1}^{N}\sigma_{B,i}^2$ with $\sigma_{B,i}^2$ the second moment of the per-step gradient on B_i.

Algorithm 3 APriLS for PFL

Input: N clients, R rounds, LoRA rank r, datasets $\{D_i\}_{i=1}^N$, and the same initialization
$A_i \leftarrow \mathcal{N}(0,\sigma)_{r\times d}, \quad B_i \leftarrow 0_{d\times r},$

Procedure:

```
for r in 1, ..., R do
  Client-Side: For all clients in parallel i ∈ [1, N]
    A_i, B_i ← Train(D_i, B_i, A_i)
    G_i = SafeSQRT(B_i^⊤ × B_i)
    Z_i = G_i × A_i
    Send Z_i to the server
  Server-Side:
    collect {Z_i}_{i=1}^N
    stack into Z^stack ← {Z_i}_{i=1}^N ∈ IR^{Nr×d}
    (U, S, V^⊤) ← TruncatedSVD(Z^stack, r)
    broadcast V^⊤ ∈ IR^{r×d}
  Client-Side: For all clients in parallel i ∈ [1, N]
    Q_i ← A_i × V
    B_i ← B_i × Q_i          # update B matrix
    A_i ← V^⊤                # update A matrix
end for
```

Algorithm 4 APriLS for FL

Input: N clients, R rounds, LoRA rank r, datasets $\{D_i\}_{i=1}^N$, and the same initialization
$A_i \leftarrow \mathcal{N}(0,\sigma)_{r\times d}, \quad B_i \leftarrow 0_{d\times r},$

Procedure:

```
for r in 1, ..., R do
  Client-Side: For all clients in parallel i ∈ [1, N]
    A_i, B_i ← Train(D_i, B_i, A_i)
    Send A_i and B_i to the server
  Server-Side:
    for i in 1, ..., N do
      G_i = SafeSQRT(B_i^⊤ × B_i)
      Z_i = G_i × A_i
    end for
    stack into Z^stack ← {Z_i}_{i=1}^N ∈ IR^{Nr×d}
    (U, S, V^⊤) ← TruncatedSVD(Z^stack, r)
    for i in 1, ..., N do
      Q_i ← A_i × V
      B'_i ← B_i × Q_i
    end for
    B̄ = (1/N) Σ_{i=0}^N B'_i
    broadcast B̄ and V^⊤ ∈ IR^{r×d} as B_i and A_i.
end for
```

3.3.2 Aggregation of Principals in Low-rank via SVD (APriLS)

This subsection explains how the aggregation outlined in Proposition 3.3.4 and 3.3.5 in Section 3.3.1 is achieved in practice. Figures 3.2 and 3.3 visualizes the outline of these procedures as described in Algorithms 3 and 4.

Personalized FL. In each round, clients train the model locally, resulting in low-rank matrices for each layer as A_i and B_i. Instead of directly sharing A_i matrices with the server, in APriLS, clients compute the Gram factor $G_i = \sqrt{(B_i^\top B_i)}$ and then compute the weighted row-stack $Z_i = G_i A_i$, and the server receives the Z_i matrices, which are then stacked into a single matrix $Z^{\text{stack}} \in \mathbb{R}^{Nr\times d}$. The server then computes the truncated SVD of Z^{stack} to obtain the right singular vectors $V_r \in \mathbb{R}^{d\times r}$, which are broadcasted to clients. Each client then computes the transformation $Q_i = A_i V_r^\top$ and updates its B_i matrix as $B_i' = B_i Q_i$. This allows clients to update their B_i matrices without needing to reconstruct the full-rank parameters, as they only need to compute the product of B_i and Q_i.

Globalized FL. In the global FL setting, clients train their models locally and share both A_i and B_i matrices with the server. The server computes the Gram factor $G_i = \sqrt{(B_i^\top B_i)}$ and the weighted row-stack $Z_i = G_i A_i$, similar to the personalized FL setting. The server then stacks these matrices into a single matrix $Z^{\text{stack}} \in \mathbb{R}^{Nr\times d}$ and computes the truncated SVD to obtain the right singular vectors $V_r \in \mathbb{R}^{d\times r}$. Next, the server computes the transformation $Q_i = A_i V_r^\top$ for each client and updates the B_i matrices as $B_i' = B_i Q_i$. Finally, the server averages the updated B_i' matrices to obtain $\bar{B} = \frac{1}{N}\sum_{i=0}^{N} B_i'$, which is broadcasted back to clients along with $V_r^\top$ for updating their A_i matrices.

3.4 Results and Discussion

We evaluated APriLS on various tasks from GLUE in both personalized and global FL settings, comparing it against state-of-the-art methods such as FedAvg, FedProx, FedMA, FFA-LoRA and KnOTS. The results demonstrate that APriLS achieves superior performance in terms of accuracy and convergence speed across all tasks and allows for a larger number of local steps per round, leading to more efficient training. We use Roberta-Large with 357M parameters and LoRA rank of 16 for all experiments. The datasets are split into 100 clients and the results are averaged over 3 runs with different random seeds.

Figure 3.4 shows the accuracy per number of local steps for each method in personalized FL settings. It shows that APriLS consitantly outperforms other methods, given same amount of local steps, and it can perform more local steps per round than other methods, leading to faster convergence. The results for global FL settings are similar, with APriLS achieving better performance and faster convergence compared to other methods as shown in Figure 3.5. Appendix C.2 provides include the tables including the results for all tasks in both personalized and global FL settings, showing that APriLS consistently outperforms other methods across all tasks.

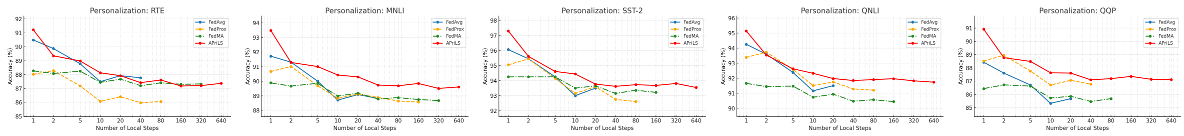


Figure 3.4: Accuracy per number of local steps in personalized FL settings.

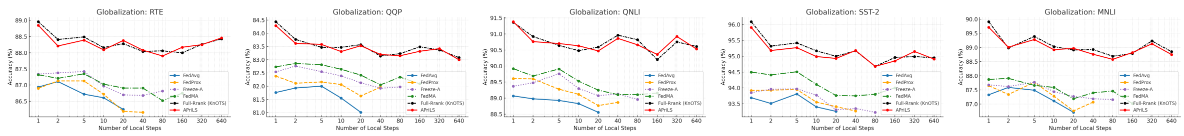
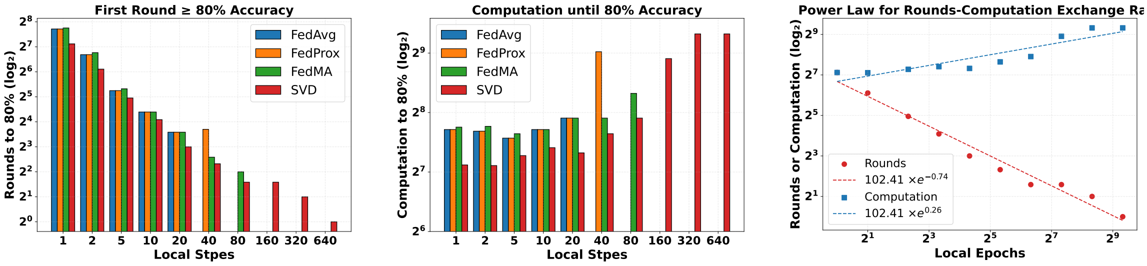


Figure 3.5: Accuracy per number of local steps in global FL settings.

3.5 Ablation Study

We conducted an ablation study to evaluate the impact of different components of APriLS on the performance and convergence speed. The results are shown in Figure 3.6. The first plot shows the effect of varying the number of rounds to reach 80% accuracy, demonstrating that APriLS requires fewer rounds compared to other methods. The second plot shows the computation time required to reach 80% accuracy, indicating that APriLS is more efficient in terms of computation. The third plot shows the power law relation between computation and communication required to reach 80% accuracy, confirming that APriLS exhibits around 3 to 1 exchange rate between computation and communication, which is significantly better than other methods that require more communication for the same level of computation. Therefore, increasing the number of rounds to reach 80% accuracy leads to a significant reduction in communication while adding around 1/3 of computation overhead.



(a) Rounds to 80% (b) Computation to 80% (c) Power Laws

Figure 3.6: Ablation study of APriLS. (a) Number of rounds to reach 80% accuracy, (b) Computation time to reach 80% accuracy, (c) Power law relation between computation and communication to reach 80% accuracy.

3.6 Conclusion

In this chapter, we introduced APriLS, a novel method for merging LoRA layers in federated and personalized learning settings. APriLS addresses the misalignment issue caused by LoRA symmetry by leveraging the properties of low-rank matrices and truncated SVD to achieve optimal aggregation without reconstructing full-rank parameters. The theoretical analysis shows that APriLS achieves a tighter convergence bound than existing methods, allowing for more efficient training with fewer communication rounds. The experimental results demonstrate that APriLS consistently outperforms state-of-the-art methods in terms of accuracy and convergence speed across various tasks in both personalized and global FL settings.

Conclusion

This thesis has addressed key challenges in federated learning by introducing comprehensive methodologies to enhance communication efficiency, model alignment, and personalization capabilities. The developed frameworks systematically address critical bottlenecks limiting the scalability, robustness, and effectiveness of federated learning systems.

Chapter 1 provided a foundational overview, systematically identifying existing limitations in FL methodologies related to communication overhead, model misalignment, and personalization. The extensive literature review and critical analysis set the stage for subsequent methodological innovations presented in the thesis.

Chapter 2 proposed the **Model-Agnostic Projection Optimization (MAPO)** framework, an innovative, architecture-independent gradient decomposition technique that dynamically adjusts communication costs through model-level optimizations. MAPO demonstrated substantial improvements in communication efficiency, convergence stability, and scalability through rigorous theoretical analyses and empirical validations across diverse datasets and model architectures.

Building upon MAPO, **Chapter 3** introduced **Aggregation of Principals in Low-rank via SVD (APriLS)**, directly addressing alignment and symmetry challenges prevalent in federated low-rank adaptations. APriLS employed principal-aligned SVD aggregation, effectively mitigating client drift and enabling stable, personalized federated learning. Theoretical convergence analyses and extensive empirical studies on widely recognized benchmarks affirmed APriLS's superiority over traditional FL and personalized FL methods.

Together, these methodological advances significantly contribute to the practical deployment and theoretical understanding of federated learning, addressing critical real-world challenges of communication efficiency, decentralized model alignment, and personalized adaptation. Future research directions include further enhancements in optimization strategies, tighter integration with privacy-preserving techniques, and adaptation to emerging federated learning scenarios, reinforcing the foundational contributions of this thesis and promoting continued innovation in decentralized learning systems.

Appendices

Chapter A. Supplementary Material for Chapter 1

A.1 Complexity Analysis

Our proposed method significantly reduces communication overhead in federated learning. However, this reduction in communication comes at the cost of an increase in computation and memory usage on the client side. In this section, we provide a comprehensive analysis of these complexity trade-offs and discuss potential strategies to alleviate the added burdens. Specifically, we present an analysis of the computational, memory, and communication complexities of our proposed model and provide a comparative assessment against existing baselines.

In this analysis, E stands for the number of local epochs executed within a single communication round, M indicates the total number of clients, N represents the population size per client, and $|\theta|$ denotes the dimension of the model parameter vector.

Table A.1 presents the order of time and memory complexities for each method where we do not employ parallelization for ES and EvoFed computation. Here, clients generate perturbations one by one and reuse the memory after the fitness measurement of a perturbed sample. As shown in Table A.1, EvoFed without parallelization has a similar order of memory complexity in clients to conventional FL, i.e., FedAvg, and reduced memory complexity in the server. However, in this scenario, the time complexity for EvoFed grows linearly with the number of perturbations as compared to FedAvg.

Table A.1: Comparison of time and memory complexities for ES, EvoFed, and FedAvg, without parallel processing of N individual perturbed models.

Method	Client Time	Client Memory	Server Time	Server Memory
ES	$O(N\|\theta\|)$	$O(\|\theta\|)$	$O(N(\|\theta\| + M))$	$O(\|\theta\| + MN)$
EvoFed	$O(N\|\theta\| + E\|\theta\|)$	$O(\|\theta\|)$	$O(N(\|\theta\| + M))$	$O(\|\theta\| + MN)$
FedAvg	$O(E\|\theta\|)$	$O(\|\theta\|)$	$O(M\|\theta\|)$	$O(M\|\theta\|)$

To mitigate the time complexity in EvoFed, one approach is to generate and evaluate a batch of T perturbations in parallel. This method poses a trade-off between time and memory complexity.

Partitioning (as discussed in the main text) is an alternative strategy that enables computing a higher number of fitness values for each perturbation by dividing it into K partitions. Consequently, the algorithm requires a fewer perturbations N' to obtain sufficient fitness values for gradient encoding, resulting in a reduction in memory complexity. This necessitates the transmission of $N'K$ fitness values to the server. While partitioning does not introduce additional time complexity, having a small number of perturbations restricts the algorithm's exploration capabilities in parameter space, as discussed in Section A.4.1. Therefore, in practice, we choose the population size to be $\frac{N}{K} \leq N' \leq N$, trading memory complexity with communication cost.

Table A.2 provides a comparative analysis of the time and memory complexities for EvoFed and FedAvg when both T individual perturbed models are processed in parallel, and each perturbed model sample is divided into K partitions. In scenarios with enough memory, it is feasible to execute all perturbations in a parallel setting $T = N$ and without partitioning $K = 1$ and $N = N'$.

As discussed before, the communication complexity of ES and EvoFed is limited to transferring fitness values $O(N'K)$ while FedAvg is to the gradient signal $O(|\theta|)$. However we can apply additional

Table A.2: Comparison of time and memory complexities for EvoFed and FedAvg, with parallel processing of T individual perturbed models and where each perturbation is partitioned to K segments.

Method	**Client Time**	**Client Memory**	**Server Time**	**Server Memory**
EvoFed (Parallel)	$O(\frac{N}{T}\|\theta\| + E\|\theta\|)$	$O(T\|\theta\|)$	$O(\frac{N}{T}(\|\theta\| + M))$	$O(N\|\theta\| + MN)$
EvoFed (Partitioned)	$O(N'\|\theta\| + E\|\theta\|)$	$O(N'\|\theta\|)$	$O(N'(\|\theta\| + M))$	$O(N'\|\theta\| + MKN')$
EvoFed (Both)	$O(\frac{N'}{T}\|\theta\| + E\|\theta\|)$	$O(T\|\theta\|)$	$O(\frac{N'}{T}(\|\theta\| + M))$	$O(N'\|\theta\| + MKN')$
FedAvg	$O(E\|\theta\|)$	$O(\|\theta\|)$	$O(M\|\theta\|)$	$O(M\|\theta\|)$

compression on each method to reduce this complexity as explored in Section A.4.1.

A.2 Model Architecture and Optimization Hyperparameters

We used a CNN model with 11k parameters for the MNIST and FMNIST datasets and a bigger model with 2.3M parameters for CIFAR-10, with architectural details provided in Table A.5 and Table A.6 respectively. We also provide detailed information about the optimization hyperparameters e.g. learning rate (lr), momentum and batch size, etc. for MNIST and FMNIST in Table A.3 and for Cifar-10 in Table A.4:

Table A.3: Hyperparameters used in experiments on dataset MNIST & FMNIST

Model	Methods	Hyperparameters										
		batch size	lr	momentum	optimizer	*lr_es*	*momentum_es*	*optimizer_es*	*w_decay*	sigma	eps	$\beta\_1 \& \beta\_2$
CNN	ES	128	-	-	-	0.0148	0.9	sgd	0.0	0.27	1e-8	0.99 & 0.999
	FedAvg	256	0.0111	0.8099	sgd	-	-	-	-	-	-	-
	Fed-quant	256	0.0111	0.8099	sgd	-	-	-	-	-	-	-
	Fed-sparse	256	0.0111	0.8099	sgd	-	-	-	-	-	-	-
	EvoFed (ours)	256	0.0873	0.9074	sgd	0.0427	0.9	sgd	0.0152	0.27	1e-8	0.99 & 0.999

Table A.4: Hyperparameters used in experiments on dataset CIFAR-10

Model	Methods	Hyperparameters										
		batch size	lr	momentum	optimizer	*lr_es*	*momentum_es*	*optimizer_es*	*w_decay*	sigma	eps	$\beta\_1 \& \beta\_2$
CNN	ES	32	-	-	-	0.04	0.4815	sgd	0.0	0.35	1e-8	0.99 & 0.999
	FedAvg	128	0.0009	0.6132	sgd	-	-	-	-	-	-	-
	Fed-quant	128	0.0009	0.6132	sgd	-	-	-	-	-	-	-
	Fed-sparse	128	0.0009	0.6132	sgd	-	-	-	-	-	-	-
	EvoFed (ours)	64	0.0148	0.3011	sgd	0.0275	0.5239	sgd	0.0824	0.35	1e-8	0.99 & 0.999

A.3 Convergence Analysis

Assumption A.3.1. For each $j, L_j(v)$ is β-smooth, i.e., $\|\nabla L_j(u) - \nabla L_j(v)\| \leq \beta\|u - v\|$ for any u, v.

Assumption A.3.2. Variance of the gradient of D_j is bounded, $\mathbb{E}\left[\left\|\nabla L_j(\theta) - \widetilde{\nabla} L_j(\theta)\right\|^2\right] \leq B^2$.

Assumption A.3.3. When perturbation ϵ^i is sampled from the population distribution p_ψ, a conditioned mirrored sampling is applied such that $\frac{1}{N}\sum_{i=1}^{N} \epsilon^i = 0$, $\frac{1}{M}\sum_{i=1}^{N} \left(\epsilon^i\right)^2 \leq G^2$, $\frac{1}{N}\sum_{i=1}^{N} \left(\epsilon^i\right)^3 = 0$.

Table A.5: Detailed information of the CNN architecture used in MNIST & FMNIST experiments

Layer	Parameter & Shape (cin, cout, kernal size) & hyper-parameters	#
layer1	conv1: $1 \times 8 \times 5 \times 5$, stride:(1, 1); padding:0	$\times 1$
	avgpool	$\times 1$
layer2	conv1: $8 \times 16 \times 5 \times 5$, stride:(1, 1); padding:0	$\times 1$
	avgpool	$\times 1$
	fc: 16×10	$\times 1$

Table A.6: Detailed information of the CNN architecture used in CIFAR-10 experiments

Layer	Parameter & Shape (cin, cout, kernal size) & hyper-parameters	#
layer1	conv1: $3 \times 64 \times 5 \times 5$, stride:(1, 1); padding:0	$\times 1$
	avgpool	$\times 1$
layer2	conv1: $64 \times 128 \times 5 \times 5$, stride:(1, 1); padding:0	$\times 1$
	avgpool	$\times 1$
	fc: 128×256	$\times 1$
	fc: 256×10	$\times 1$

Theorem A.3.4. *Given a decreasing learning rate $\eta_t < \frac{1}{4\alpha\beta}$, EvoFed has the convergence bound as:*

$$\frac{1}{H_T}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\left[\|\nabla L(\theta_t)\|^2\right] \leq \frac{\mathbb{E}\left[L(\theta_0)\right]-L^*}{\alpha G^2 H_T} + 4\alpha\beta B^2\left(\frac{1}{H_T}\sum_{t=0}^{T-1}\eta_t^2\right)$$

where $H_T = \sum_{t=0}^{T-1}\eta_t$, and L^* represents the minimum value of $L(\theta)$.

By β-smoothness of $L(\theta)$ and taking expectation on both sides, we have

$$\mathbb{E}\left[L(\theta_{t+1})-L(\theta_t)\right] \leq \mathbb{E}\left[\langle\nabla L(\theta_t), \theta_{t+1}-\theta_t\rangle\right] + \frac{\beta}{2}\mathbb{E}\left[\|\theta_{t+1}-\theta_t\|^2\right] \tag{A.1}$$

Proof. By utilizing the proof of Lemma 1 and recognizing $\langle\cdot,\cdot\rangle$ as the inner product operation, we rewrite the first term $\mathbb{E}\left[\langle \nabla L(\theta_t), \theta_{t+1} - \theta_t \rangle\right]$ as follows:

$$
\begin{aligned}
\mathbb{E}\left[\langle \nabla L(\theta_t), \theta_{t+1} - \theta_t \rangle\right] &\underset{(a)}{=} \mathbb{E}\left[\left\langle \nabla L(\theta_t), \frac{1}{M}\sum_{j=1}^{M} \frac{\alpha}{N\sigma}\sum_{i=1}^{N} f(\theta_{t,j} + \sigma\epsilon_t^i)\epsilon_t^i \right\rangle\right] \\
&\underset{(b)}{=} \mathbb{E}\left[\left\langle \nabla L(\theta_t), \frac{1}{M}\sum_{j=1}^{M} \frac{\alpha}{N\sigma}\sum_{i=1}^{N} \|(\theta_{t,j} + \sigma\epsilon_t^i) - (\theta_{t,j}^{'})^i\|^2 \epsilon_t^i \right\rangle\right] \\
&= -\mathbb{E}\Bigg[\Bigg\langle \nabla L(\theta_t), \frac{1}{M}\sum_{j=1}^{M} \frac{\alpha}{N\sigma}\sum_{i=1}^{N} \|(\theta_{t,j} + \sigma\epsilon_t^i) \\
&\qquad\qquad - (\theta_{t,j} - \eta_t \widetilde{\nabla} L_j(\theta_{t,j}))\|^2 \epsilon_t^i \Bigg\rangle\Bigg] \\
&= -\mathbb{E}\left[\left\langle \nabla L(\theta_t), \frac{1}{M}\sum_{j=1}^{M} \frac{\alpha}{N\sigma}\sum_{i=1}^{N} \|(\sigma\epsilon_t^i) + \eta_t \widetilde{\nabla} L_j(\theta_{t,j})\|^2 \epsilon_t^i \right\rangle\right] \\
&= -\mathbb{E}\Bigg[\Bigg\langle \nabla L(\theta_t), \frac{1}{M}\sum_{j=1}^{M} \frac{\alpha}{N\sigma}\sum_{i=1}^{N} \Big(\sigma^2 \left(\epsilon_t^i\right)^3 + 2\sigma \left(\epsilon_t^i\right)^2 \eta_t \widetilde{\nabla} L_j(\theta_t) \\
&\qquad\qquad + \epsilon_t^i \eta_t^2 \|\widetilde{\nabla} L_j(\theta_t)\|^2 \Big) \Bigg\rangle\Bigg] \\
&\underset{(c)}{\leq} -\mathbb{E}\left[\left\langle \nabla L(\theta_t), \frac{1}{M}\sum_{i=j}^{M} \frac{\alpha}{\sigma}\left(2\sigma G^2 \eta_t \widetilde{\nabla} L_j(\theta_t)\right) \right\rangle\right] \\
&\underset{(d)}{=} (-2\alpha\eta_t G^2)\, \mathbb{E}\left[\left\langle \nabla L(\theta_t), \frac{1}{M}\sum_{j=1}^{M} \nabla L_j(\theta_t) \right\rangle\right] \\
&\underset{(e)}{=} (-\alpha\eta_t G^2) \Bigg\{ \mathbb{E}\left[\|\nabla L(\theta_t)\|^2\right] + \mathbb{E}\left[\|\frac{1}{M}\sum_{j=1}^{M} \nabla L_j(\theta_t)\|^2\right] \\
&\qquad\qquad - \underbrace{\mathbb{E}\left[\|\nabla L(\theta_t) - \frac{1}{M}\sum_{j=1}^{M} \nabla L_j(\theta_t)\|^2\right]}_{=0} \Bigg\}
\end{aligned}
$$

where (a) comes from the Lemma 1, (b) is due to $f(\theta_t) = -\|\theta_t - \theta_t^{'}\|^2$, (c) follows from Assumption 3, (d) is from taking expectation for the mini-batch, and (e) is due to the well-known equality $\|z_1 - z_2\|^2 = \|z_1\|^2 + \|z_2\|^2 - 2\langle z_1, z_2 \rangle$.

On the other hand, we can bound the second term $\mathbb{E}\left[\|\theta_{t+1}-\theta_t\|^2\right]$ as follows:

$$
\begin{aligned}
\mathbb{E}\left[\|\theta_{t+1}-\theta_t\|^2\right] &= \mathbb{E}\left[\left\|\frac{1}{M}\sum_{j=1}^{M}\frac{\alpha}{N\sigma}\sum_{i=1}^{N} f(\theta_t+\sigma\epsilon_t^i)\epsilon_t^i\right\|^2\right] \\
&= \mathbb{E}\left[\left\|\frac{1}{M}\sum_{j=1}^{M}\frac{\alpha}{N\sigma}\sum_{i=1}^{N} \|(\theta_t+\sigma\epsilon_t^i)-\theta_t^{'}\|^2\epsilon_t^i\right\|^2\right] \\
&= \mathbb{E}\left[\left\|\frac{1}{M}\sum_{j=1}^{M}\frac{\alpha}{N\sigma}\sum_{i=1}^{N} \|(\theta_t+\sigma\epsilon_t^i)-(\theta_t-\eta_t\widetilde{\nabla}L_j(\theta_t))\|^2\epsilon_t^i\right\|^2\right] \\
&= \mathbb{E}\left[\left\|\frac{1}{M}\sum_{j=1}^{M}\frac{\alpha}{N\sigma}\sum_{i=1}^{N} \left(\sigma^2\left(\epsilon_t^i\right)^3+2\sigma\left(\epsilon_t^i\right)^2\eta_t\widetilde{\nabla}L_j(\theta_t)+\epsilon_t^i\eta_t^2\|\widetilde{\nabla}L_j(\theta_t)\|^2\right)\right\|^2\right] \\
&\underset{(a)}{\leq} \mathbb{E}\left[\left\|\frac{1}{M}\sum_{j=1}^{M}\frac{\alpha}{\sigma}\left(2\sigma G^2\eta_t\widetilde{\nabla}L_j(\theta_t)\right)\right\|^2\right] = \mathbb{E}\left[(4\alpha^2G^2\eta_t^2)\left\|\frac{1}{M}\sum_{j=1}^{M}\widetilde{\nabla}L_j(\theta_t)\right\|^2\right] \\
&\underset{(b)}{\leq} (8\alpha^2G^2\eta_t^2)\mathbb{E}\left[\left\|\frac{1}{M}\sum_{j=1}^{M}\nabla L_j(\theta_t)\right\|^2+\left\|\frac{1}{M}\sum_{j=1}^{M}\nabla L_j(\theta_t)-\frac{1}{M}\sum_{j=1}^{M}\widetilde{\nabla}L_j(\theta_t)\right\|^2\right] \\
&\underset{(c)}{\leq} (8\alpha^2G^2\eta_t^2)\left\{\mathbb{E}\left[\left\|\frac{1}{M}\sum_{j=1}^{M}\nabla L_i(\theta_t)\right\|^2\right]+B^2\right\}
\end{aligned}
$$

where (a) comes from Assumption 3, (b) is due to $\|a+b\|^2 \leq 2\|a\|^2+2\|b\|^2$, and (c) is by Assumption 2.

By applying the aforementioned bounds of $\mathbb{E}\left[\langle\nabla L(\theta_t),\theta_{t+1}-\theta_t\rangle\right]$ and $\mathbb{E}\left[\|\theta_{t+1}-\theta_t\|^2\right]$ to (A.1), we obtain:

$$
\begin{aligned}
\mathbb{E}\left[L(\theta_{t+1})-L(\theta_t)\right] &\leq \mathbb{E}\left[\langle\nabla L(\theta_t),\theta_{t+1}-\theta_t\rangle\right]+\frac{\beta}{2}\mathbb{E}\left[\|\theta_{t+1}-\theta_t\|^2\right] \\
&\leq \mathbb{E}\left[(-\alpha\eta_tG^2)\left\{\|\nabla L(\theta_t)\|^2+\|\frac{1}{M}\sum_{j=1}^{M}\nabla L_j(\theta_t)\|^2\right\}\right. \\
&\qquad\left.+(4\alpha^2\beta G^2\eta_t^2)\left\{\left\|\frac{1}{M}\sum_{j=1}^{M}\nabla L_j(\theta_t)\right\|^2+B^2\right\}\right] \\
&= -\alpha\eta_tG^2\mathbb{E}\left[\|\nabla L(\theta_t)\|^2\right] \\
&\qquad\underbrace{+\,\alpha\eta_tG^2(4\alpha\beta\eta_t-1)\mathbb{E}\left[\left\|\frac{1}{M}\sum_{j=1}^{M}\nabla L_j(\theta_t)\right\|^2\right]}_{\leq 0 \text{ if we choose } \eta_t\leq\frac{1}{4\alpha\beta}}+(4\alpha^2\beta G^2\eta_t^2)B^2 \\
&\leq -\alpha\eta_tG^2\mathbb{E}\left[\|\nabla L(\theta_t)\|^2\right]+(4\alpha^2\beta G^2\eta_t^2)B^2
\end{aligned}
$$

Eventually, through the telescoping sum for $t = 0, 1, ..., T-1$, we obtain

$$L^* - \mathbb{E}\left[L(\theta_0)\right] \;\leq \sum_{t=0}^{T-1}(-\alpha\eta_t G^2)\mathbb{E}\left[\|\nabla L(\theta_t)\|^2\right] + \sum_{t=0}^{T-1}(4\alpha^2\beta G^2\eta_t^2)B^2$$

where L^* represents the minimum value of $L(\theta)$.

After performing division on both sides by $H_T = \sum_{t=0}^{T-1}\eta_t$, and employing some manipulations, we obtain

$$\frac{1}{H_T}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\left[\|\nabla L(\theta_t)\|^2\right] \leq \frac{\mathbb{E}\left[L(\theta_0)\right] - L^*}{\alpha G^2 H_T} + 4\alpha\beta B^2\left(\frac{1}{H_T}\sum_{t=0}^{T-1}\eta_t^2\right) \tag{A.2}$$

By utilizing a decreasing learning rate (e.g., $\eta_t = \frac{\eta_0}{1+t}$), it can be seen that $H_T = \sum_{t=0}^{T-1}\eta_t \to \infty$ as T increases, while $\sum_{t=0}^{T-1}\eta_t^2 < \infty$. Consequently, the upper bound stated in Equation (A.2) approaches 0 as T grows, ensuring convergence towards a stationary point.

A.4 Additional Experimental Result

In this section, we delve into the impacts of various parameters on both the training and communication rate. We first study the role of population size and the number of clients. Subsequently, we investigate the effect of additional compression techniques, such as sparsification, ranking, and quantization, on the model's performance. Lastly, we assess the efficacy of partitioning on clients in attaining better accuracy, and its relationship with the population size.

A.4.1 Sparsification, Quantization, and Partitioning

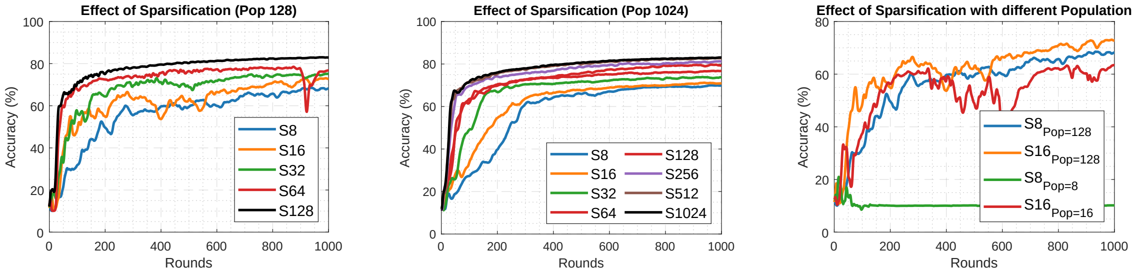
In this section, we explore the effect of fitness sparsification i.e. selecting top-k fitness values from the fitness vector of the whole population based on magnitude. We examined the effects of sparsification on two distinct population sizes: 128 and 1024. Without any sparsification, both populations demonstrated comparable performance. However, when we select the top-k most fit values, the denser population (comprising 1024 members) could tolerate a higher degree of sparsification compared to the less populous one (with 128 members).

To enable a fair and insightful comparison between the two population sizes, our focus was on assessing performance based on the number of members remaining post-sparsification rather than directly contrasting sparsification rates. We placed particular emphasis on the best and worst performing members, as they exert the most significant influence on the model update process in ES.

Fig. A.1(a) and (b) visualize the sparsification process for populations of 128 and 1024, respectively, illustrating the performance decline that occurs as the number of remaining members diminishes.

Fig. A.1(c) provides further insights into the performance improvements achieved by selecting top-8 or top-16 members from the initial set of 128 or 1024, as compared to optimizing with the whole population of 8 or 16.

Our results underline the crucial role that population size plays in exploring optimal solutions, overshadowing even the significance of compression rate. A larger population allows for broad exploration that can later be compressed to a smaller number of members without a performance loss. However, initiating the process with a smaller population cannot achieve equivalent performance due to the restricted exploration. Therefore, population size is a critical factor affecting the efficacy of exploration in evolutionary strategies.



(a) Sparse (pop=128) (b) Sparse (pop=1024) (c) Sparse vs Pop

Figure A.1: Effect of sparsification on EvoFed

In this section, we examine the sensitivity of EvoFed to the precise value of fitness. We propose two techniques to reduce the bits required to represent the fitness vector, thus enhancing compression without compromising performance. For a clearer understanding of these methods' impacts, we chose a population size of 32, which is relatively less populated and has minimal redundancy, highlighting the insensitivity of EvoFed to precise fitness values.

Fig. A.2(a) depicts the effect of quantization with varying bit numbers. The legend represents the number of bits used for quantization as a numeral followed by the letter Q, where $Q32$ indicates no compression and $Q1$ signifies transmitting a single bit (either 0 or 1) in place of the fitness value. The result exhibits a marginal performance loss even with $Q2$, illustrating EvoFed's insensitivity to precise fitness values and the potential for further compression gains through quantization.

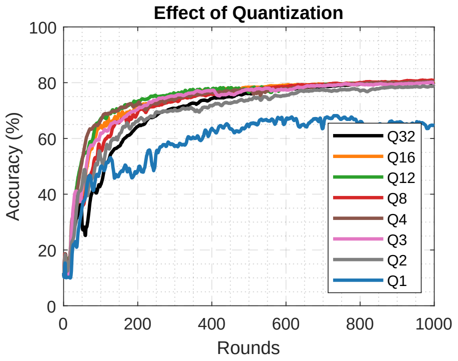
Fig. A.2(b) presents the performance when we transmit the member's rank within the population instead of the fitness value. In the legend, the number of samples assigned the same rank is denoted as a numeral following the letter R; $R32$ indicates assigning 32 different ranks to all members, and $R1$ implies assigning the same rank to every member. This ranking technique, a common practice in the Evolutionary Strategies literature, is typically employed when fitness values derived from the environment are noisy, and the quality of the solution can be improved by transmitting the ranking instead. However, where we have high-quality fitness measures derived from L2 loss, this technique only slightly improves the performance while reducing compression gains. By assigning the same rank to neighbouring samples within the fitness ranking, we can further enhance compression performance.

Comparing ranking and quantization, it is observed that quantization delivers superior performance with the same number of bits. Additionally, the number of bits used in quantization is independent of the population size, making quantization a more appropriate approach for compressing fitness values.

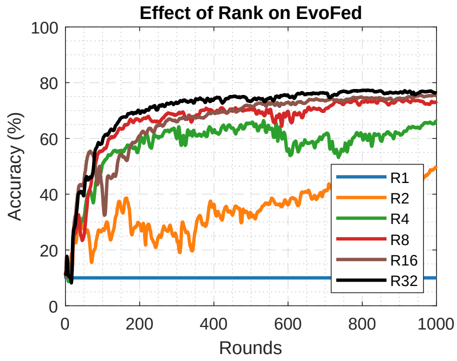
The EvoFed's partitioning technique, as described in the main text, features a unique attribute that enhances performance. This technique maintains a fixed number of population samples at each client, thereby addressing memory limitations on the clients but necessitating increased communication as a trade-off. Although sparsification results underscore the importance of population size for exploration, partitioning presents an additional approach that navigates the limitation posed by the compression rate to improve performance.

Figure A.3 illustrates the impact of partitioning in four scenarios, each with a different population size. The results emphasize that partitioning is most effective when the clients cannot manage a sufficient number of samples to attain satisfactory performance. Partitioning enables us to gather more information from the limited sample size.

Each sub-figure in Figure A.3 includes baselines without partitioning, allowing for the comparison



(a) Quantization (pop=32)

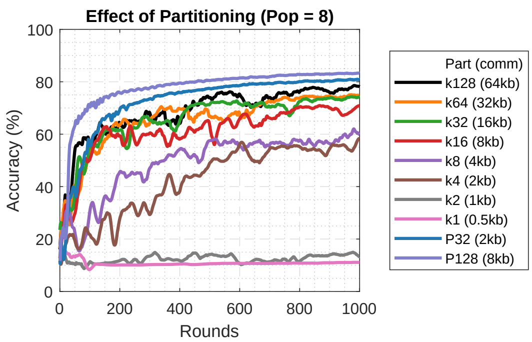


(b) Rank (pop=32)

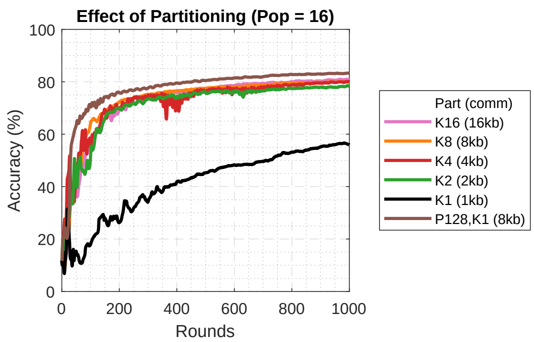
Figure A.2: Effect of Quantization on EvoFed

of improvements achievable either through an increased population size or the use of more partitions while maintaining a consistent communication rate. The legend of each figure specifies the number of partitions (the number following the letter k) and the population of the baselines (the number following the letter p). The volume of information required to be communicated for one round is also depicted for each method in the legend.

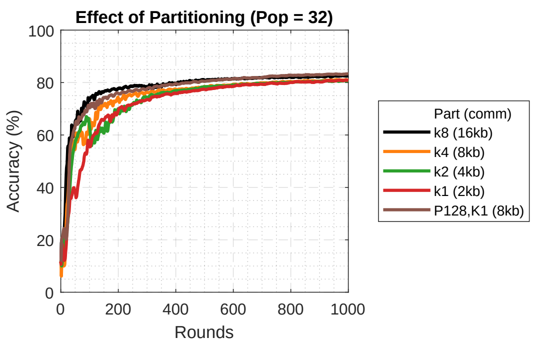
Figure A.3 clearly shows that using population sizes of 32 and 128 results in only a marginal improvement in performance. However, when utilizing population sizes of 8 and 16, a significant and noticeable improvement can be observed.



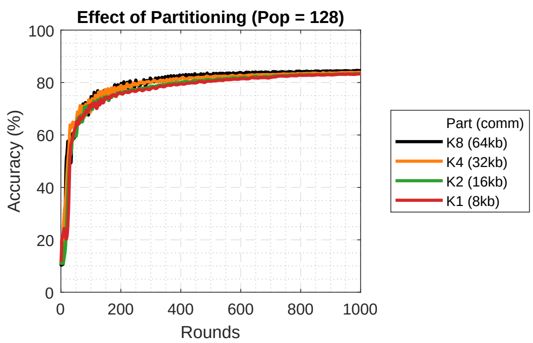
(a) EvoFed with Population Size = 8



(b) EvoFed with Population Size = 16



(c) EvoFed with Population Size = 32

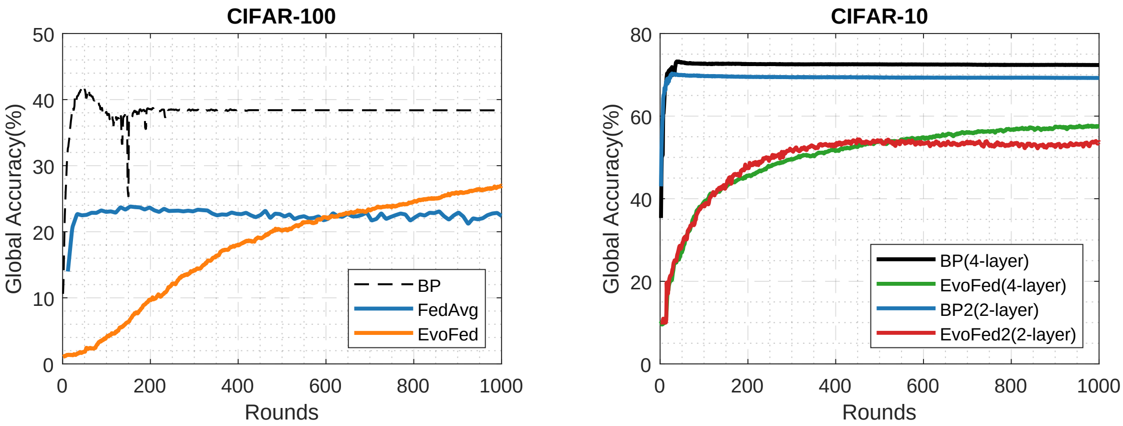


(d) EvoFed with Population Size = 128

Figure A.3: Effect of partitioning on EvoFed

A.4.2 Larger Dataset and Model

Additionally, we investigate the efficacy of EvoFed in the context of a larger model and a bigger dataset. Figure A.4(a) showcases the performance on CIFAR-100 dataset with the same model parameters as those used in the CIFAR-10 experiment. The results show that EvoFed, although having a slower convergence rate, achieves higher performance than FedAvg eventually, with a significant compression rate. Figure A.4(b) illustrates the performance gain on CIFAR-10 dataset when the CNN layers are doubled. As the experimental result shows, having a larger model generally leads to better performance with slower convergence. EvoFed follows the same trend as BP in a centralized setting, suggesting the compression has not been affected by model size. All experiments were conducted with a population size of 32 and 50 partitions.



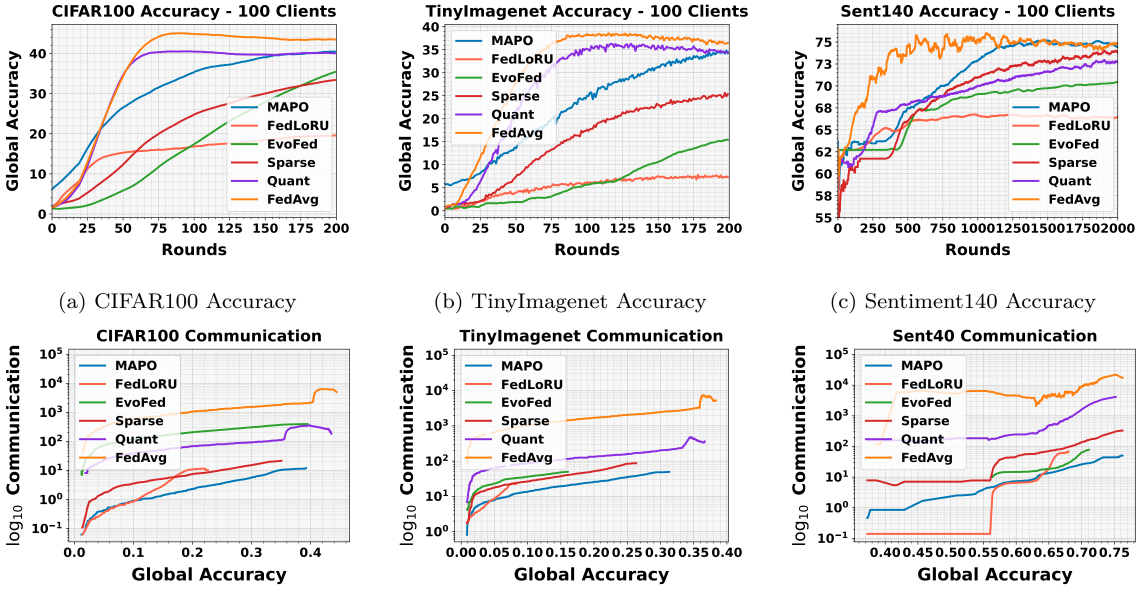
(a) (b)

Figure A.4: Larger dataset and Model: (a) shows performance on CIFAR-100, and (b) depicts the impact of having a larger model on CIFAR-10.

Chapter B. Supplementary Material for Chapter 2

B.1 Accuracy and Communication Learning curves

This appendix provides extended experimental results that complement the main findings discussed in Section 2.5. We include detailed evaluations of MAPO and baseline methods on CIFAR-100, TinyImageNet, and Sentiment140 datasets. Similar to the main results, Figure B.1 reports both maximum test accuracy and the communication cost required to reach a given accuracy threshold. These additional experiments further demonstrate MAPO's superior communication efficiency and consistent performance gains across more challenging and large-scale tasks.



(a) CIFAR100 Accuracy (b) TinyImagenet Accuracy (c) Sentiment140 Accuracy

(d) CIFAR100 Comm. cost (e) TinyImagenet Comm. cost (f) Sentiment140 Comm. cost

Figure B.1: **Performance comparison** of MAPO and baseline methods on CIFAR100, TinyImagenet, and Sentiment140 datasets. The top row shows the accuracy achieved by each method on the respective datasets, while the bottom row illustrates the communication cost associated with each method.

B.2 Comparison with Low-Rank Adaptation in Fine-tuning

We conduct fine-tuning experiments using RoBERTa-large on five GLUE tasks to evaluate MAPO alongside LoRA, FA-LoRA, and SA-LoRA. Table B.1 compares the number of trainable parameters and the communication load per round for each method. Table B.2 summarizes fine-tuning results under federated settings, reporting communication efficiency based on the number of rounds and total communication required to reach 80% accuracy. Overall, the results indicate that MAPO improves communication efficiency without compromising performance.

Table B.1: Number of trainable and communication parameters per round for different methods.

Method	Number of trainable parameters	Number of communication parameters per round
LoRA	1.83M	0.78M
FA-LoRA	1.44M	0.39M
SA-LoRA	1.83M	0.39M
MAPO$_{d/1k}$	357M	0.36M
MAPO$_{d/10k}$	357M	35.70K
MAPO$_{d/100k}$	357M	3.57K
MAPO$_{d/1m}$	357M	357

Table B.2: Comparison of model accuracies, communication rounds, and total communication cost.

Model	SST2			QNLI			RTE			MNLIm			MNLImm		
	Acc	Round	Total	Acc	Round	Total	Acc	Round	Total	Acc	Round	Total	Acc	Round	Total
LoRA	84.86	36	28.08M	91.72	85	66.30M	86.62	180	140.40M	87.41	86	67.08M	87.34	82	63.96M
FA-LoRA	94.15	44	17.16M	91.63	76	29.64M	57.28	—	—	85.92	76	29.64M	86.46	213	83.07M
SA-LoRA	95.41	19	7.41M	91.04	55	21.45M	70.01	—	—	89.44	29	11.31M	85.49	126	49.14M
MAPO$_{d/1k}$	96.79	5	1.78M	93.14	11	3.93M	87.91	23	8.21M	88.90	17	6.07M	88.26	22	7.85M
MAPO$_{d/10k}$	96.10	5	178.50K	92.57	8	285.60K	89.57	23	821.10K	88.81	18	642.60K	87.43	25	892.50K
MAPO$_{d/100k}$	95.53	5	17.85K	89.24	7	24.99K	84.38	24	85.68K	85.04	20	71.40K	84.60	29	103.53K
MAPO$_{d/1m}$	90.37	7	2.50K	80.09	34	12.14K	57.04	—	—	72.46	—	—	37.76	—	—

B.3 Comparison with Factorized-FL

In this section, we present a detailed comparison between MAPO and Factorized-FL as a representative of the parameter decomposition methods. Factorized-FL can be interpreted as a variant of rank-1 LoRA, where a sparse bias matrix substitutes for LoRA's frozen fine-tuned weights, initialized to zero. Table B.3 reports the communication efficiency of MAPO and Factorized-FL on CIFAR-10 and SVHN datasets, evaluated under both IID and non-IID partitions. Each column denotes the total communication in GB required to reach X% of FedAvg's final test accuracy. Results show that MAPO achieves significantly lower communication costs compared to Factorized-FL while maintaining competitive performance across both datasets and data distributions.

Table B.3: Communication cost comparison across different methods on SVHN and CIFAR-10 under IID and Non-IID settings.

Method	SVHN				CIFAR-10				Com/Round
	IID@80%	IID@90%	Non-IID@80%	Non-IID@90%	IID@80%	IID@90%	Non-IID@80%	Non-IID@90%	
FedAvg	183.51	244.68	285.46	509.75	305.85	407.80	326.24	652.48	20.39GB
Factorized-FL	127.75	182.50	146.00	219.00	182.50	292.00	200.75	310.25	18.25GB
MAPO$_{2k}$	0.32	0.79	0.56	–	0.32	–	0.94	–	**0.78MB**
MAPO$_{16k}$	**0.08**	**0.18**	**0.12**	**0.27**	**0.08**	**0.18**	**0.23**	**0.45**	6.25MB
MAPO$_{40k}$	3.84	8.64	5.76	13.12	3.84	8.64	10.88	21.12	0.32GB

B.4 Implementation details and Hyperparameters

All experiments were conducted on a single NVIDIA RTX 3090 with 24 GB of memory. The main experiments and baselines are implemented with JAX [55]. The GLUE tasks and LLM fine-tuning implementation use Hugging Face libraries and models implemented in FederatedScope [155] with half precision (i.e., 16-bit float). The model configuration and training used in this work are provided in Tables B.4 and B.5.

Table B.4: Neural network configurations for different datasets.

Dataset	Model type	# Conv	Kernel	Hidden features	# Linear	# Output	# Parameters
MNIST	CNN	2	5×5	8, 16	1	10	11.3K
FMNIST	CNN	2	5×5	8, 16	1	10	11.3K
CIFAR-10	CNN	4	5×5	64, 64, 128, 128	2	10	1.1M
CIFAR-100	WideResNet	16	3×3	64×4, 128×4	2	100	2.8M
TinyImageNet	WideResNet	16	3×3	64×4, 128×4	2	200	2.88M
Shakespeare	LSTM	-	-	256, 8 (embed)	2	65	814K
Sentiment140	Transformer	-	-	512, 96 (embed)	2	2	2.2M
SVHN	CNN	4	5×5	64, 64, 128, 128	2	10	1.1M
GLUE	RoBERTa-large	-	-	1024 (hidden)	2	Varies	357M

Table B.5: Training hyperparameters for FedAvg and variants.

Hyperparameter	MNIST	FMNIST	CIFAR-10	CIFAR-100	TinyImageNet	Sentiment140	Shakespeare	SVHN	GLUE
Batch size	32	32	32	32	32	32	32	32	128
Optimizer	SGD	SGD	SGD	AdamW	AdamW	SGD	SGD	SGD	SGD
Learning rate	0.2	0.2	0.03	0.1	0.2	0.001	0.2	0.03	0.02
Momentum	0.9	0.9	0.4	0.9	0.9	0.9	0.9	0.4	0.0
L1 regularization	0.0	0.0	1e-4	0.0	1e-5	0.0	5e-6	1e-4	0.0
L2 regularization	0.0	0.0	1e-5	3e-3	1e-4	0.0	5e-5	1e-5	0.0

B.5 IID and Client Sampling

This section includes the results of additional experiments on IID distribution and client sampling for MNIST, FMNIST, and CIFAR-10. Across all three datasets, we observe consistent trends. Reducing the fraction of clients participating (from all clients to 10%) moderately decreases accuracy for all methods, and non-IID settings introduce additional accuracy penalties. However, MAPO's performance remains robust in these more demanding scenarios; it routinely stays close to FedAvg's high-accuracy results while maintaining significant communication savings. This resilience suggests that MAPO's approach scales well to heterogeneous data distributions and partial-participation regimes, crucial in large-scale FL deployments.

Table B.6: Extrapolated MNIST results for IID vs. non-IID and full vs. 10% client participation.

	IID				**Non-IID**			
	All clients		**10% clients**		**All clients**		**10% clients**	
Method	Com.	Acc.	Com.	Acc.	Com.	Acc.	Com.	Acc.
FedAvg	100%	99.6%	100%	99.5%	100%	99.3%	100%	98.9%
Sparse	10.0%	93.9%	12.0%	93.6%	13.3%	93.4%	15.3%	92.1%
Quantize	22.0%	98.8%	25.0%	98.5%	29.0%	98.2%	31.3%	97.6%
EvoFed	6.5%	99.4%	7.0%	99.2%	8.5%	**99.0%**	9.4%	**98.5%**
FedLoRU	22.0%	95.0%	25.0%	94.7%	28.2%	94.3%	30.2%	93.8%
MAPO	**2.0%**	**99.5%**	**2.3%**	**99.3%**	**2.7%**	**99.0%**	**2.9%**	**98.5%**

Table B.7: Extrapolated FMNIST results for IID vs. non-IID and full vs. 10% client participation.

	IID				**Non-IID**			
	All clients		**10% clients**		**All clients**		**10% clients**	
Method	Com.	Acc.	Com.	Acc.	Com.	Acc.	Com.	Acc.
FedAvg	100%	91.5%	100%	91.0%	100%	90.0%	100%	89.2%
Sparse	16.0%	84.0%	19.0%	83.5%	21.0%	82.0%	24.1%	81.1%
Quantize	16.0%	89.7%	19.0%	89.2%	21.0%	88.0%	24.1%	87.1%
EvoFed	4.5%	87.0%	5.5%	86.5%	6.8%	85.5%	7.6%	84.7%
FedLoRU	12.0%	76.8%	14.0%	76.2%	15.5%	75.0%	17.9%	74.1%
MAPO	**2.0%**	**90.0%**	**2.3%**	**89.6%**	**2.7%**	**88.8%**	**3.1%**	**88.0%**

Table B.8: Extrapolated CIFAR-10 results for IID vs. non-IID and full vs. 10% client participation.

	IID				**Non-IID**			
	All clients		**10% clients**		**All clients**		**10% clients**	
Method	Com.	Acc.	Com.	Acc.	Com.	Acc.	Com.	Acc.
FedAvg	100%	73.0%	100%	72.0%	100%	70.0%	100%	69.0%
Sparse	1.8%	41.0%	2.0%	40.0%	2.4%	38.0%	2.7%	37.2%
Quantize	10.0%	71.0%	12.0%	70.0%	13.0%	68.5%	15.2%	67.4%
EvoFed	2.0%	43.0%	2.5%	42.0%	3.0%	40.5%	3.4%	39.5%
FedLoRU	1.1%	27.0%	1.3%	26.0%	1.5%	24.5%	1.7%	23.5%
MAPO	**0.8%**	**71.5%**	**0.9%**	**70.8%**	**1.0%**	**69.2%**	**1.2%**	**68.3%**

B.6 Proof of Definitions and Propositions

Definition B.6.1 (Communication Overhead Rate). Let $\Delta W \in \mathbb{R}^{d_1 \times d_2}$ be the update matrix of a model. Suppose the factorization of ΔW as $\Delta W = BA$, where $A \in \mathbb{R}^{q \times d_2}$ is a fixed random matrix and $B \in \mathbb{R}^{d_1 \times q}$ is a trainable matrix with $q \leq \min(d_1, d_2)$ being the factorization rank. The **communication overhead rate** CO_{rate} is defined as the ratio of the size of B to the size of ΔW:

$$\text{CO}_{rate} = \frac{\text{size}(B)}{\text{size}(\Delta W)} = \frac{q}{d_2}.$$

Definition B.6.2 (Reconstruction Error Rate). Using the same factorization as Theorem 2.3.2, the *reconstruction error rate* is the expected ratio of the reconstruction error to the original model update. Given full-rank random reconstruction (Theorem 2.3.1), it is expressed as:

$$\frac{\mathbb{E}_A\left[\|\Delta W - BA\|_2^2\right]}{\|\Delta W\|_2^2} = 1 - \frac{q}{d_2}.$$

Proof. Let $\Delta W = [\Delta w_1\ \Delta w_2\ \cdots\ \Delta w_{d_1}]$, where each column $\Delta w_i \in \mathbb{R}^{d_2}$. Similarly, the reconstruction BA can be written as $[b_1 A\ b_2 A\ \cdots\ b_{d_1} A]$, where each $b_i \in \mathbb{R}^q$ is a trainable matrix. The reconstruction error is given by:

$$\|\Delta W - BA\|_2^2 = \sum_{i=1}^{d_1} \|\Delta w_i - b_i A\|_2^2.$$

The projection of Δw_i onto the subspace spanned by A is $P_A \Delta w_i$. The error rate E is defined as:

$$E = \frac{\|\Delta w_i - \Delta w_i P_A\|_2^2}{\|\Delta w_i\|_2^2}.$$

Using the Pythagorean theorem:

$$\|\Delta w_i\|_2^2 = \|\Delta w_i P_A\|_2^2 + \|w_i - \Delta w_i \ P_A|_2^2,$$

we rewrite E as:

$$E = \frac{\|\Delta w_i\|_2^2 - \|\Delta w_i P_A\|_2^2}{\|\Delta w_i\|_2^2} = 1 - \frac{\|\Delta w_i P_A\|_2^2}{\|\Delta w_i\|_2^2}.$$

The expected value of $\|\Delta w_i P_A\|_2^2$ for a full-rank random Gaussian projection is:

$$\mathbb{E}[\|\Delta w_i P_A\|_2^2] = \frac{q}{d_2}\|\Delta w_i\|_2^2.$$

Substituting this into E:

$$\mathbb{E}[\|\Delta w_i - b_i A\|_2^2] = 1 - \frac{\mathbb{E}[\|\Delta w_i P_A\|_2^2]}{\|\Delta w_i\|_2^2} = 1 - \frac{\frac{p}{d}\|\Delta w_i\|_2^2}{\|w_i\|_2^2} = 1 - \frac{q}{d_2}.$$

Applying this to each column $\Delta\Delta w_i$ of ΔW, we obtain:

$$\mathbb{E}_A\left[\sum_{i=1}^{d_1} \|\Delta w_i - b_i A\|_2^2\right] = \sum_{i=1}^{d_1} \mathbb{E}_A\left[\|\Delta w_i - (\Delta w_i) P_A\|_2^2\right].$$

Using the expected error formula:

$$= \sum_{i=1}^{d_1}\left(1 - \frac{q}{d_2}\right)\|\Delta w_i\|_2^2 = \left(1 - \frac{q}{d_2}\right)\sum_{i=1}^{d_1}\|\Delta w_i\|_2^2.$$

Since $\|\Delta W\|_2^2 = \sum_{i=1}^{d_1} \|\Delta w_i\|_2^2$, we get:

$$\mathbb{E}_A\left[\|\Delta W - BA\|_2^2\right] = \left(1 - \frac{q}{d_2}\right)\|\Delta W\|_2^2.$$

□

Proposition B.6.3 (Single-Vector Factorization). *Let ΔW, A, and B be factorizations of a single layer of the network as in Theorem 2.3.2. By reshaping ΔW into $\Delta W' \in \mathbb{R}^{1\times d_1 d_2}$ the factorization of $\Delta W' = B'A'$ where $A' \in \mathbb{R}^{p\times d_1 d_2}$ and $B' \in \mathbb{R}^{1\times p}$ can achieve the same **reconstruction error** and **communication overhead** to the conventional factorization of ΔW when $p = qd_1$.*

Proof of Error Preservation. In the single-vector setup, $\Delta W' \in \mathbb{R}^{d_1 d_2}$ is projected onto a subspace of dimension p. From random projection theory (as used in Theorem 2.3.3), if A' is sampled such that $\operatorname{rank}(A') = p$, then:

$$\mathbb{E}\left[\frac{\|\Delta W' - B'A'\|_2^2}{\|\Delta W'\|_2^2}\right] = 1 - \frac{p}{d_1 d_2}.$$

Substituting $p = qd_1$ gives:

$$1 - \frac{qd_1}{d_1 d_2} = 1 - \frac{q}{d_2}.$$

Hence, the expected reconstruction error satisfies:

$$\mathbb{E}\left[\|\Delta W' - B'A'\|_2^2\right] = \left(1 - \frac{q}{d_2}\right)\|\Delta W'\|_2^2,$$

which matches the original factorization. □

Proof of Communication Preservation. For $\Delta W' \in \mathbb{R}^{d_1 d_2}$, with the total size $\text{size}(\Delta W') = d_1 d_2$, we have the communication overhead as:

$$\text{size}(B') = p = qd_1.$$

Thus, the communication overhead is:

$$\text{CO}'_{rate} = \frac{\text{size}(B')}{\text{size}(\Delta W')} = \frac{qd_1}{d_1 d_2} = \frac{q}{d_2},$$

which matches the original overhead.

Since both the expected reconstruction error and the communication overhead remain unchanged, the single-vector factorization with $p = qd_1$ is equivalent in terms of efficiency. □

Proposition B.6.4 (Multi-Layer Factorization). *Let ΔW_i, A_i, and B_i be **single-vector factorization** of i-th layer of the n-layered network as in Theorem 2.3.4. By concatenating the reshaped weights ΔW_i into $\Delta W' \in \mathbb{R}^{1\times d}$, where $d = \sum_{i=1}^{n} d_1^i d_2^i$. The factorization of $\Delta W' = B'A'$ where $A' \in \mathbb{R}^{p\times d}$ and $B' \in \mathbb{R}^{1\times p}$ can achieve the same **reconstruction error** and **communication overhead** to the single-vector factorization applied to each ΔW_i when $p = nqd_1$.*

Proof of Error Preservation. For each layer i, a random full-rank matrix $A_i \in \mathbb{R}^{q\times d_2^i}$ yields an expected squared reconstruction error

$$\mathbb{E}\Big[\|\Delta W_i - B_i A_i\|_F^2\Big] \;=\; \Big(1 \;-\; \frac{q}{d_2^i}\Big)\|\Delta W_i\|_F^2.$$

Flattening ΔW_i into $\Delta W_i' \in \mathbb{R}^{(d_1^i d_2^i)\times 1}$, a single-vector projection of dimension $q\, d_1^i$ preserves this same error ratio (cf. Theorem 2.3.4).

When we concatenate all $\Delta W_i'$ into $\Delta W' \in \mathbb{R}^{1\times d}$, we form a block-structured vector. Let $p := n\, q$ and let $A' \in \mathbb{R}^{p\times d}$ be constructed from a Gaussian distribution. By the standard random-projection argument in dimension d with subspace size p,

$$\mathbb{E}\Big[\|\Delta W' - B'A'\|_2^2\Big] \;=\; \Big(1 \;-\; \frac{p}{d}\Big)\|\Delta W'\|_2^2 \;=\; \Big(1 \;-\; \frac{p}{N d_1 d_2}\Big)\|\Delta W'\|_2^2.$$

Since $p = Nqd_1$, the overall ratio matches applying single-vector factorizations of rank q to each $\Delta W_i'$ individually. □

Proof of Communication Preservation. For each layer i, the single-vector factorization of ΔW_i introduces

$$\text{size}(B_i) \;=\; q\, d_1^i, \quad \text{size}(\Delta W_i) \;=\; d_1^i\, d_2^i, \quad \text{hence} \quad \frac{\text{size}(B_i)}{\text{size}(\Delta W_i)} \;=\; \frac{q}{d_1^i}.$$

Concatenating all $\Delta W_i'$ into $\Delta W' \in \mathbb{R}^{1\times d}$ gives $\text{size}(\Delta W') = d$, with

$$d \;=\; \sum_{i=1}^{N} d_1^i\, d_2^i.$$

Meanwhile, in the multi-layer factorization, the new trainable vector $B' \in \mathbb{R}^{1\times p}$ has

$$\text{size}(B') \;=\; p \;=\; N\, q.$$

Thus

$$\frac{\text{size}(B')}{\text{size}(\Delta W')} \;=\; \frac{N\, q}{\sum_{i=1}^{N}\big(d_1^i\, d_2^i\big)},$$

which matches the *total* overhead of N individual rank-q factorizations (one per layer) in aggregate. Consequently, the communication overhead rate is also preserved.

Since both the expected reconstruction error (per layer or in total) and the communication overhead remain the same, choosing $p = N\,q$ for $\Delta W'$ is equivalent to applying single-vector factorization of rank q separately to each layer. $\square$

Proposition B.6.5 (MAPO Factorization). *Let ΔW, A, B, and rank p be a multi-layer factorization of a network as defined in Theorem 2.3.5. By reshaping $\Delta W \in \mathbb{R}^{1\times d}$ into $\Delta W' \in \mathbb{R}^{k\times\lceil d/k\rceil}$, and the factorization of $\Delta W' = B'A'$ where $A' \in \mathbb{R}^{1\times\lceil d/k\rceil}$ and $B' \in \mathbb{R}^{k\times 1}$, we can achieve the same* ***reconstruction error*** *and* ***communication overhead*** *to the multi-layer factorization of ΔW when $k = p$,* while reducing the memory by a factor of k^2*.*

Proof of Error Preservation. Since $\Delta W \in \mathbb{R}^{1\times d}$ is reshaped into $\Delta W' \in \mathbb{R}^{k\times\lceil d/k\rceil}$, we still have $\|\Delta W'\|_F^2 = \|\Delta W\|_2^2$. When $A' \in \mathbb{R}^{1\times\lceil d/k\rceil}$ is a suitable random projection (and $B' \in \mathbb{R}^{k\times 1}$ is fit accordingly), the rank-1 subspace of dimension 1 within $\lceil d/k\rceil$ induces the known expected error ratio

$$\mathbb{E}\Big[\|\Delta W' - B'A'\|_F^2\Big] \;=\; \big(1 - \tfrac{1}{\lceil d/k\rceil}\big)\,\|\Delta W'\|_F^2,$$

since the ambient dimension is $k \times \lceil d/k\rceil \;\approx d$. By taking $k = p$, we obtain (via standard random-projection arguments) the matching error ratio $1 - p/d$, up to negligible rounding. Therefore:

$$\mathbb{E}\Big[\|\Delta W' - B'A'\|_F^2\Big] \;=\; \big(1 - \frac{p}{d}\big)\,\|\Delta W'\|_F^2,$$

$\square$

Proof of Communication Preservation. The matrix $B' \in \mathbb{R}^{k\times 1}$ has size k in total. Meanwhile, $\Delta W' \in \mathbb{R}^{k\times\lceil d/k\rceil}$ has size $k \times \lceil d/k\rceil \approx d$. Thus

$$\frac{\text{size}(B')}{\text{size}(\Delta W')} \;=\; \frac{k}{\lceil d/k\rceil\,k} \;\approx\; \frac{k}{d} = \frac{p}{d}.$$

Setting $k = p$ matches the original ratio $\frac{p}{d}$ from $B \in \mathbb{R}^{p\times 1}$ in the multi-layer factorization. $\square$

Proof of Memory Reduction by Factor k^2. In standard rank-p factorizations for $\Delta W \in \mathbb{R}^{1\times d}$, one typically stores a $p \times d$ projection plus a $1 \times p$ vector, whose total size scales as $dp + p$. By contrast, $A' \in \mathbb{R}^{1\times\lceil d/k\rceil}$ plus $B' \in \mathbb{R}^{k\times 1}$ has combined size $\lceil d/k\rceil + k$. When $k = p$, the ratio of these sizes can be shown to drop by a factor of approximately k^2. Hence, the approach allocates k^2 times less memory than a naive $p \times d$ plus $1 \times p$ arrangement. As $p = k$

$$\frac{dp + p}{\lceil d/k\rceil + k} = \frac{dk + k}{\lceil d/k\rceil + k} \approx \frac{d + 1}{d/k^2 + 1} \approx k^2$$

Thus, the factorization $\Delta W' = B'A'$ with $k = p$ exactly preserves the original rank-p error and overhead while using k^2-fold less memory. $\square$

B.7 Proof of Theorem

B.7.1 Assumptions and Preliminaries

We restate the key assumptions required for the convergence analysis.

Assumption B.7.1. For each $j, \mathcal{L}^j(v)$ is β-smooth, i.e., $\left\|\nabla\mathcal{L}^j(u)-\nabla\mathcal{L}^j(v)\right\| \leq \beta\|u-v\|$ for any u, v.

Assumption B.7.2. Variance of the stochastic gradient of D^j is bounded for each client j, i.e.,

$$\mathbb{E}\Big[\Big\|\nabla\mathcal{L}^j(W) - \widetilde{\nabla}\mathcal{L}^j(W)\Big\|^2\Big] \leq \sigma_l^2$$

.

Lemma B.7.3 (Johnson-Lindenstrauss Lemma)**.** *Given $0 < \epsilon < 1$, a set of points $\{x_1, x_2, \ldots, x_M\} \subset \mathbb{R}^d$, and a target dimension $k = O\left(\frac{\log M}{\epsilon^2}\right)$, there exists a random linear mapping $P \in \mathbb{R}^{d\times k}$ such that for all i, j:*

$$(1-\epsilon)\|x_i - x_j\|^2 \leq \|x_iP - x_jP\|^2 \leq (1+\epsilon)\|x_i - x_j\|^2.$$

In our context, the random projection matrices $B^{t,j}$ and reconstruction matrices A^t satisfy the JL property with high probability.

B.7.2 Proof of Theorem 1

Theorem B.7.1. *Let the learning rate satisfy $\eta_t \leq \frac{1-4\epsilon}{4\beta(1+\epsilon)}$. Then, the algorithm achieves the bound:*

$$\frac{1}{4H_T}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\left[\left\|\nabla\mathcal{L}(W^t)\right\|^2\right] \leq \frac{\mathbb{E}\left[\mathcal{L}(W^0)\right] - \mathcal{L}^*}{H^T} + 2(\epsilon+\beta+\beta\epsilon)\sigma_l^2\frac{1}{H^T}\sum_{t=0}^{T-1}\eta_t^2,$$

where $H_T = \sum_{t=0}^{T-1}\eta_t$, ϵ is JL Lemma distortion parameter, and $\mathcal{L}^*$ is the minimum value of $\mathcal{L}(W)$.

Proof. By the β-smoothness of $\mathcal{L}(W)$ and taking expectation on both sides, we have

$$\mathbb{E}\left[\mathcal{L}(W^{t+1}) - \mathcal{L}(W^t)\right] \leq \mathbb{E}\left[\langle\nabla\mathcal{L}(W^t), W^{t+1} - W^t\rangle\right] + \frac{\beta}{2}\mathbb{E}\left[\left\|W^{t+1} - W^t\right\|^2\right]. \tag{B.1}$$

Using the update rule $W^{t+1} = W^t - \eta_t\overline{B}_tA^t$, where $\overline{B}_t = \frac{1}{M}\sum_{j=1}^{M}B^{t,j}$, we can rewrite the first term as:

$$\begin{aligned}
\mathbb{E}\left[\langle\nabla\mathcal{L}(W^t), W^{t+1} - W^t\rangle\right] &= -\eta_t\mathbb{E}\left[\left\langle\nabla\mathcal{L}(W^t), \overline{B}^tA^t\right\rangle\right] \\
&= -\eta_t\mathbb{E}\left[\left\langle\nabla\mathcal{L}(W^t), \left(\frac{1}{M}\sum_{j=1}^{M}B^{t,j}\right)A^t\right\rangle\right] \\
&= -\eta_t\mathbb{E}\left[\left\langle\nabla\mathcal{L}(W^t), \frac{1}{M}\sum_{j=1}^{M}B^{t,j}A^t\right\rangle\right].
\end{aligned}$$

We decompose $B^{t,j}A^t$ as:

$$\widetilde{\nabla}\mathcal{L}^j(W^t) = B^{t,j}A^t + e^{t,j},$$

where $e^{t,j} = \widetilde{\nabla}\mathcal{L}^j(W^t) - B^{t,j}A^t$ is the projection error.

Substituting back, we have:

$$\mathbf{E} = \mathbb{E}\left[\langle\nabla\mathcal{L}(W^t), W^{t+1} - W^t\rangle\right] = -\eta_t\mathbb{E}\left[\left\langle\nabla\mathcal{L}(W^t), \frac{1}{M}\sum_{j=1}^{M}\left(\widetilde{\nabla}\mathcal{L}^j(W^t) - e^{t,j}\right)\right\rangle\right]$$

Separating it into A_1 and A_2:

$$\mathbf{E} = \underbrace{-\eta_t \mathbb{E}\left[\left\langle \nabla \mathcal{L}(W^t), \frac{1}{M}\sum_{j=1}^{M} \widetilde{\nabla}\mathcal{L}^j(W^t)\right\rangle\right]}_{A_1} + \underbrace{\eta_t \mathbb{E}\left[\left\langle \nabla \mathcal{L}(W^t), \frac{1}{M}\sum_{j=1}^{M} e^{t,j}\right\rangle\right]}_{A_2}.$$

We will now concentrate on A_1 as:

$$\begin{aligned}
A_1 &= -\eta_t \mathbb{E}\left[\left\langle \nabla \mathcal{L}(W^t), \frac{1}{M}\sum_{j=1}^{M} \nabla\mathcal{L}^j(W^t)\right\rangle\right] \\
&= -\frac{\eta_t}{M}\sum_{j=1}^{M}\mathbb{E}\left[\left\langle \nabla \mathcal{L}(W^t), \nabla\mathcal{L}^j(W^t)\right\rangle\right] \\
&\underset{(a)}{=} -\frac{\eta_t}{2M}\sum_{j=1}^{M}\left\{\mathbb{E}\left[\|\nabla \mathcal{L}(W^t)\|^2\right] + \mathbb{E}\left[\left\|\nabla\mathcal{L}^j(W^t)\right\|^2\right]\right\} \\
&\quad + \frac{\eta_t}{2}\mathbb{E}\left[\left\|\underbrace{\nabla \mathcal{L}(W^t) - \frac{1}{M}\sum_{j=1}^{M}\nabla\mathcal{L}^j(W^t)}_{=0}\right\|^2\right] \\
&= -\frac{\eta_t}{2}\mathbb{E}\left[\|\nabla \mathcal{L}(W^t)\|^2\right] - \frac{\eta_t}{2M}\sum_{j=1}^{M}\mathbb{E}\left[\left\|\nabla\mathcal{L}^j(W^t)\right\|^2\right]
\end{aligned}$$

where (a) uses $\langle a, b\rangle = \frac{1}{2}\{||a||^2 + ||b||^2 - ||a-b||^2\}$. We now turn our attention to A_2 as:

Next, we focus on A_2:

$$\begin{aligned}
A_2 &= \eta_t \mathbb{E}\left[\left\langle \nabla \mathcal{L}(W^t), \frac{1}{M}\sum_{j=1}^{M} e^{t,j}\right\rangle\right] \\
&\underset{(a)}{\leq} \frac{\eta_t}{4}\mathbb{E}\left[\left\|\nabla \mathcal{L}(W^t)\right\|^2\right] + \eta_t \mathbb{E}\left[\left\|\frac{1}{M}\sum_{j=1}^{M} e^{t,j}\right\|^2\right] \\
&\underset{(b)}{\leq} \frac{\eta_t}{4}\mathbb{E}\left[\left\|\nabla \mathcal{L}(W^t)\right\|^2\right] + \frac{\eta_t}{M} \mathbb{E}\left[\left\|\sum_{j=1}^{M} e^{t,j}\right\|^2\right] \\
&\underset{(c)}{\leq} \frac{\eta_t}{4}\mathbb{E}\left[\left\|\nabla \mathcal{L}(W^t)\right\|^2\right] + \frac{\epsilon\eta_t}{M} \mathbb{E}\left[\left\|\sum_{j=1}^{M} \widetilde{\nabla}\mathcal{L}^j(W^t)\right\|^2\right] \\
&\underset{(d)}{\leq} \frac{\eta_t}{4}\mathbb{E}\left[\left\|\nabla \mathcal{L}(W^t)\right\|^2\right] + \frac{2\epsilon\eta_t}{M} \sum_{j=1}^{M}\left\{\mathbb{E}\left[\left\|\nabla\mathcal{L}^j(W^t)\right\|^2\right] + \mathbb{E}\left[\left\|\widetilde{\nabla}L_i(W^t) - \nabla\mathcal{L}^j(W^t)\right\|^2\right]\right\} \\
&\underset{(e)}{\leq} \frac{\eta_t}{4}\mathbb{E}\left[\left\|\nabla \mathcal{L}(W^t)\right\|^2\right] + \frac{2\epsilon\eta_t}{M} \sum_{j=1}^{M}\mathbb{E}\left[\left\|\nabla\mathcal{L}^j(W^t)\right\|^2\right] + 2\epsilon\eta_t^2\sigma_l^2
\end{aligned}$$

where (a) uses $\langle a,b\rangle \leq \frac{1}{4}\|a\|^2 + \|b\|^2$, and (b) follows Jensen's inequality, (c) comes from JL Lemma, (d) follows the inequality $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$, and (e) is based on Assumption 2. On the other hand, we can also place a bound on the second term $\mathbb{E}\left[\|W^{t+1} - W^t\|^2\right]$ as shown below:

$$
\begin{aligned}
&\mathbb{E}\left[\|W^{t+1} - W^t\|^2\right] = \mathbb{E}\left[\left\|\eta_t \overline{B_t} A^t\right\|^2\right] = \mathbb{E}\left[\left\|\eta_t \left(\frac{1}{M}\sum_{j=1}^{M} B^{t,j}\right) A^t\right\|^2\right] \\
&\underset{(a)}{\leq} 2\eta_t^2 \mathbb{E}\left[\left\|\frac{1}{M}\sum_{j=1}^{M} \widetilde{\nabla}\mathcal{L}^j(W^t)\right\|^2\right] + 2\eta_t^2 \mathbb{E}\left[\left\|\frac{1}{M}\sum_{j=1}^{M}\left\{B^{t,j}A^t - \widetilde{\nabla}\mathcal{L}^j(W^t)\right\}\right\|^2\right] \\
&\underset{(b)}{\leq} \frac{2\eta_t^2}{M}\mathbb{E}\left[\left\|\sum_{j=1}^{M} \widetilde{\nabla}\mathcal{L}^j(W^t)\right\|^2\right] + \frac{2\eta_t^2}{M}\mathbb{E}\left[\left\|\sum_{j=1}^{M}\left\{B^{t,j}A^t - \widetilde{\nabla}\mathcal{L}^j(W^t)\right\}\right\|^2\right] \\
&= \frac{2\eta_t^2}{M}\mathbb{E}\left[\left\|\sum_{j=1}^{M} \widetilde{\nabla}\mathcal{L}^j(W^t)\right\|^2\right] + \frac{2\eta_t^2}{M}\mathbb{E}\left[\left\|\sum_{j=1}^{M} e^{t,j}\right\|^2\right] \\
&\underset{(c)}{\leq} \frac{4\eta_t^2}{M}\sum_{j=1}^{M}\left\{\mathbb{E}\left[\left\|\nabla\mathcal{L}^j(W^t)\right\|^2\right] + \mathbb{E}\left[\left\|\widetilde{\nabla}L_i(W^t) - \nabla\mathcal{L}^j(W^t)\right\|^2\right]\right\} + \frac{2\eta_t^2}{M}\mathbb{E}\left[\left\|\sum_{j=1}^{M} e^{t,j}\right\|^2\right] \\
&\underset{(d)}{\leq} \frac{4\eta_t^2}{M}\sum_{j=1}^{M}\mathbb{E}\left[\left\|\nabla\mathcal{L}^j(W^t)\right\|^2\right] + \frac{2\eta_t^2}{M}\mathbb{E}\left[\left\|\sum_{j=1}^{M} e^{t,j}\right\|^2\right] + 4\eta_t^2\sigma_l^2 \\
&\underset{(e)}{\leq} \frac{4\eta_t^2}{M}\sum_{j=1}^{M}\mathbb{E}\left[\left\|\nabla\mathcal{L}^j(W^t)\right\|^2\right] + \frac{2\epsilon\eta_t^2}{M}\mathbb{E}\left[\left\|\sum_{j=1}^{M} \widetilde{\nabla}\mathcal{L}^j(W^t)\right\|^2\right] + 4\eta_t^2\sigma_l^2 \\
&\underset{(f)}{\leq} \frac{4\eta_t^2}{M}\sum_{j=1}^{M}\mathbb{E}\left[\left\|\nabla\mathcal{L}^j(W^t)\right\|^2\right] \\
&+ \frac{4\epsilon\eta_t^2}{M}\sum_{j=1}^{M}\left\{\mathbb{E}\left[\left\|\nabla\mathcal{L}^j(W^t)\right\|^2\right] + \mathbb{E}\left[\left\|\widetilde{\nabla}\mathcal{L}^j(W^t) - \nabla\mathcal{L}^j(W^t)\right\|^2\right]\right\} + 4\eta_t^2\sigma_l^2 \\
&\underset{(g)}{\leq} \frac{4\eta_t^2}{M}\sum_{j=1}^{M}\mathbb{E}\left[\left\|\nabla\mathcal{L}^j(W^t)\right\|^2\right] + \frac{4\epsilon\eta_t^2}{M}\sum_{j=1}^{M}\mathbb{E}\left[\left\|\nabla\mathcal{L}^j(W^t)\right\|^2\right] + 4\epsilon\eta_t^2\sigma_l^2 + 4\eta_t^2\sigma_l^2 \\
&= \frac{4(1+\epsilon)\eta_t^2}{M}\sum_{j=1}^{M}\mathbb{E}\left[\left\|\nabla\mathcal{L}^j(W^t)\right\|^2\right] + 4(1+\epsilon)\eta_t^2\sigma_l^2
\end{aligned}
$$

where (a), (c), and (f) are based on the inequality $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$, (b) comes from Jensen's inequality, (d), (g) derive from Assumption 2, and (e) comes from JL Lemma.

By utilizing the established bounds for $\mathbb{E}\left[\langle \nabla \mathcal{L}(W^t), W^{t+1} - W^t\rangle\right]$ and $\mathbb{E}\left[\|W^{t+1} - W^t\|^2\right]$ to Equation (B.1), we derive the following:

$$
\begin{aligned}
&\mathbb{E}\left[\mathcal{L}(W^{t+1}) - \mathcal{L}(W^t)\right] \le \mathbb{E}\left[\langle \nabla \mathcal{L}(W^t), W^{t+1} - W^t\rangle\right] + \frac{\beta}{2}\mathbb{E}\left[\|W^{t+1} - W^t\|^2\right] \\
&\le \underbrace{-\frac{\eta_t}{2}\mathbb{E}\left[\|\nabla \mathcal{L}(W^t)\|^2\right] - \frac{\eta_t}{2M}\sum_{j=1}^{M}\mathbb{E}\left[\left\|\nabla \mathcal{L}^j(W^t)\right\|^2\right]}_{A_1} \\
&+ \underbrace{\frac{\eta_t}{4}\mathbb{E}\left[\left\|\nabla \mathcal{L}(W^t)\right\|^2\right] + \frac{2\epsilon\eta_t}{M}\sum_{j=1}^{M}\mathbb{E}\left[\left\|\nabla \mathcal{L}^j(W^t)\right\|^2\right] + 2\epsilon\eta_t^2\sigma_l^2}_{A_2} \\
&+ \frac{2\beta(1+\epsilon)\eta_t^2}{M}\sum_{j=1}^{M}\mathbb{E}\left[\left\|\nabla \mathcal{L}^j(W^t)\right\|^2\right] + 2\beta(1+\epsilon)\eta_t^2\sigma_l^2 \\
&= -\frac{\eta_t}{4}\mathbb{E}\left[\|\nabla \mathcal{L}(W^t)\|^2\right] \\
&+ \frac{\eta_t}{M}\underbrace{\left\{-\frac{1}{2} + 2\epsilon + 2\beta(1+\epsilon)\eta_t\right\}}_{\le 0 \text{ if we choose } \eta_t \le \frac{1-4\epsilon}{4\beta(1+\epsilon)}}\sum_{j=1}^{M}\mathbb{E}\left[\left\|\nabla \mathcal{L}^j(W^t)\right\|^2\right] + 2\eta_t^2(\epsilon + \beta + \beta\epsilon)\sigma_l^2 \\
&\le -\frac{\eta_t}{4}\mathbb{E}\left[\|\nabla \mathcal{L}(W^t)\|^2\right] + 2\eta_t^2(\epsilon + \beta + \beta\epsilon)\sigma_l^2
\end{aligned}
$$

Ultimately, by applying the telescoping sum over $t = 0, 1, \ldots, T-1$, we arrive at the following result:

$$
\mathcal{L}^* - \mathbb{E}\left[\mathcal{L}(W^0)\right] \le \sum_{t=0}^{T-1} -\frac{\eta_t}{4}\mathbb{E}\left[\left\|\nabla \mathcal{L}(W^t)\right\|^2\right] + \sum_{t=0}^{T-1} 2\eta_t^2(\epsilon + \beta + \beta\epsilon)\sigma_l^2
$$

In this case, $\mathcal{L}^*$ stands for the minimum of $\mathcal{L}(W)$.

By performing a division by $H_T = \sum_{t=0}^{T-1}\eta_t$ on both sides and utilizing some algebraic adjustments, we arrive at the following expression:

$$
\frac{1}{4H_T}\sum_{t=0}^{T-1}\eta_t\mathbb{E}\left[\left\|\nabla \mathcal{L}(W^t)\right\|^2\right] \le \frac{\mathbb{E}\left[\mathcal{L}(W^0)\right] - \mathcal{L}^*}{H_T} + 2(\epsilon + \beta + \beta\epsilon)\sigma_l^2\left(\frac{1}{H_T}\sum_{t=0}^{T-1}\eta_t^2\right) \tag{B.2}
$$

With a decreasing learning rate such as $\eta_t = \frac{\eta_0}{t+1}$, we observe that $H_T = \sum_{t=0}^{T-1}\eta_t$ tends towards infinity as T grows, while $\sum_{t=0}^{T-1}\eta_t^2$ remains bounded. Therefore, as $T \to \infty$, the upper bound in Equation (B.2) converges to 0, confirming the convergence to a stationary point. $\square$

B.8 Complexity Analysis and MAPO Flexibility

Theorems 2.3.4 to 2.3.6 discussed how the error rate and accuracy of low-rank factorization are only determined by the size of the projection vector, regardless of reshaping and vectorization of layers. Although they prove that MAPO can achieve the same performance as layer-wise factorization given the same projection (communication) budget, we did not discuss the memory and computation complexity. In this section, we show that MAPO can effectively reduce computation. Furthermore, we show how layer-wise low-rank adaptation (LoRA and FA-LoRA) limits the model trade-offs and how MAPO can offer more flexibility.

B.8.1 Computational Complexity

We compute the memory and computation cost for matrix allocation and multiplication in terms of standard matrix multiplication. Given matrices $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{p\times n}$, the complexities for computing $C = BA$ are:

$$\text{Memory}_{C=AB} = O(nm + pn + pm),$$

$$\text{Time}_{C=BA} = O(mnp).$$

We aim to demonstrate that factorization under MAPO, where $W \in \mathbb{R}^{k\times\lceil \frac{d}{k}\rceil}$ is factorized into $A \in \mathbb{R}^{1\times\lceil \frac{d}{k}\rceil}$ and $B \in \mathbb{R}^{k\times 1}$, reduces the memory and time complexity of the LoRA factorization for an n-layered model. In LoRA, each layer i is factorized as $w_i \in \mathbb{R}^{d_i^1\times d_i^2}$ into $A \in \mathbb{R}^{q\times d_i^1}$ and $B \in \mathbb{R}^{d_i^2\times q}$.

We demonstrate that, given the same communication budget and factorization error rate, MAPO significantly reduces the computational cost compared to LoRA. This reduction becomes more pronounced as the number of layers or the selected rank increases. Specifically, MAPO achieves a **memory reduction** by a factor of q^2 and a **computation reduction** by a factor of q, where q is the chosen LoRA rank. Furthermore, even when $q = 1$, MAPO still achieves memory savings as $\sum_{i\neq j}^{n} d_i^1 d_i^2$ scales with the number of layers. The only scenario where MAPO and LoRA yield identical efficiency is when the model consists of a single layer ($n = 1$) and a rank-1 factorization ($q = 1$).

Memory Complexity

Given these definitions, the memory complexities for MAPO and LoRA are:

$$\text{Memory}_{MAPO} = O\left(\left\lceil \frac{d}{k}\right\rceil + k + \left\lceil \frac{d}{k}\right\rceil k\right) \approx O\left(\frac{d}{k} + k + d\right),$$

$$\text{Memory}_{LoRA} = O\left(\sum_{i=1}^{n}(d_i^1 q + d_i^2 q + d_i^1 d_i^2)\right) = O\left(\sum_{i=1}^{n} d_i^1 q + \sum_{i=1}^{n} d_i^2 q + \sum_{i=1}^{n} d_i^1 d_i^2\right).$$

Given the same communication budget $k = \sum_{i=1}^{n} q d_i^1$ and $d = \sum_{i=1}^{n} d_i^1 d_i^2$, we rewrite LoRA's memory complexity as:

$$\text{Memory}_{LoRA} = O\left(q\sum_{i=1}^{n} d_i^2 + k + d\right).$$

For MAPO to have lower memory usage than LoRA, the following condition must hold:

$$\begin{gathered}
\text{Memory}_{MAPO} \leq \text{Memory}_{LoRA}, \\
\frac{d}{k} + k + d \leq q \sum_{i=1}^{n} d_i^2 + k + d, \\
\frac{d}{k} \leq q \sum_{i=1}^{n} d_i^2.
\end{gathered}$$

Replacing k and d with their respective summation terms:

$$\begin{aligned}
\sum_{i=1}^{n} d_i^1 d_i^2 &\leq q^2 \sum_{i=1}^{n} d_i^1 \sum_{i=1}^{n} d_i^2, \\
&\leq q^2 \sum_{i=1}^{n} d_i^1 d_i^2 + q^2 \sum_{i \neq j}^{n} d_i^1 d_i^2.
\end{aligned}$$

Thus, the inequality always holds under the conditions $d_i^1, d_i^2, q, n \geq 1$, and equality occurs if $q = n = 1$, which corresponds to a model with a single layer and rank-1 factorization. In this case, MAPO and LoRA perform the same decomposition.

Time Complexity

Given the definitions, we can express the time complexities for MAPO and LoRA as follows:

$$\begin{gathered}
\text{Time}_{MAPO} = O\left(\left\lceil \frac{d}{k} \right\rceil k\right) \approx O(d), \\
\text{Time}_{LoRA} = O\left(\sum_{i=1}^{n} q d_i^1 d_i^2\right).
\end{gathered}$$

Since $d = \sum_{i=1}^{n} d_i^1 d_i^2$, we can rewrite LoRA's time complexity as:

$$\text{Time}_{LoRA} = O(qd).$$

For MAPO to have a lower time complexity than LoRA, the following condition must hold:

$$\begin{gathered}
\text{Time}_{MAPO} \leq \text{Time}_{LoRA}, \\
d \leq qd.
\end{gathered}$$

This condition is always true for $d, q \geq 1$, and equality occurs when $q = 1$.

B.8.2 MAPO Flexibility

Suppose our neural network has n layers. Let:

$$W_i \in I\!R^{d_i^1 \times d_i^2} \quad \text{for each layer } i = 1, \dots, n.$$

Let $d = \sum_{i=1}^{n} d_i^1 d_i^2$ be the total number of parameters (i.e., the sum of the entries across all layers). Let

$$d_1 \;=\; \sum_{i=1}^{n} d_i^1.$$

In many treatments of LoRA, the main communication or factor-size bottleneck arises from a factor that scales linearly with $q \cdot d_i^1$.

LoRA Factorization Per Layer. LoRA factorizes each layer W_i of dimension $d_i^1 \times d_i^2$ with a fixed rank q. Concretely,

$$W_i \;\approx\; W_i + B_i A_i, \qquad A_i \in I\!R^{q \times d_i^2}, \quad B_i \in I\!R^{d_i^1 \times q}.$$

The number of additional parameters introduced by each low-rank pair (A_i, B_i) is

$$\underbrace{d_i^1 \cdot q}_{\text{size of } B_i} \;+\; \underbrace{q \cdot d_i^2}_{\text{size of } A_i} \;=\; q\,(\,d_i^1 + d_i^2).$$

Summing over all n layers,

$$\sum_{i=1}^{n} \bigl(d_i^1 \cdot q + q \cdot d_i^2\bigr) \;=\; q \sum_{i=1}^{n} \bigl(d_i^1 + d_i^2\bigr).$$

Therefore, we can write the communication cost as:

$$\text{Communication cost} \;\approx\; q \sum_{i=1}^{n} d_i^1 \;=\; q\,d_1.$$

Since q must be an integer, we see that the communication overhead comes in integer multiplesd_1, as:

$$\text{LoRA total communication} \;\in\; \{\,q\,d_1 \;|\; q = 1, 2, \dots\}.$$

There is no way to select a non-integer q. Hence communication budgets strictly between d_1 and $2\,d_1$ (or between $q\,d_1$ and $(q+1)d_1$) are not possible in layer-wise LoRA. Therefore, Any attempt to finely tune the communication or factor budget (e.g., to $1.5\,d_1$) is disallowed by LoRA's integral-rank requirement. This **rigidity** is precisely what we seek to overcome in MAPO.

MAPO Factorization. MAPO flattens or reshapes all parameters into one large matrix and then performs a single low-rank factorization with rank 1. A simplified abstraction is:

1. Reshape $w_1, \dots, w_n$ into a single matrix $W \in I\!R^{k \times \lceil d/k \rceil}$, where $d = \sum_{i=1}^{n} d_i^1\, d_i^2$ is the total parameter count. 2. Factor $W \approx A\,B$, with

$$A \in I\!R^{1 \times \lceil d/k \rceil}, \quad B \in I\!R^{k \times 1},$$

Once all parameters are merged, MAPO can proportionally allocate any communication budget as k can be selected freely.

$$\underbrace{\lceil d/k \rceil}_{\text{size of } A} \;+\; \underbrace{k}_{\text{size of } B}.$$

Therefore, we can write the total communication as:

$$\text{MAPO total communication} \;\in\; \{\,k \;|\; k = 1, 2, \dots\}.$$

This is particularly important in communication-efficient FL since viable solutions can be found with communication cost $k < d_1$ or $d_1 < k < 2d_1$, which architecture-dependent layer-wise factorization can not offer.

Chapter C. Supplementary Material for Chapter 3

C.1 Proof of Proposition 3.3.3

Proposition C.1.1 (Factor-wise aggregation error). *Let $W_i = B_i A_i$ with $B_i \in \mathbb{R}^{d\times r}$ and $A_i \in \mathbb{R}^{r\times d}$ be the rank-r parameters of client i after τ local SGD steps with stepsize η starting from a common initialization, and denoting the aggregation error as:*

$$E := \frac{1}{N}\sum_{i=1}^{N} B_i A_i - \Big(\frac{1}{N}\sum_{i=1}^{N} B_i\Big)\Big(\frac{1}{N}\sum_{i=1}^{N} A_i\Big).$$

The expected and worst-case deterministic error bounds are:

$$\boxed{\textit{(P1)}\ \Big(\mathbb{E}\|E\|_F^2\Big)^{1/2} \ \le\ \frac{\tau\eta\sigma_A\sigma_B}{\sqrt{N}} \quad ; \qquad \textit{(P2)}\ \|E\|_F \ \le\ \tau^2\eta^2 G_A G_B.}$$

*where **(P1)** assumes unbiased gradients with second moments σ_A^2, σ_B^2, and **(P2)** assumes deterministic per-step bounds G_A, G_B. Proof details appear in Appendix C.1.*

Proof. Write $A_{\text{avg}} = \frac{1}{N}\sum_i A_i$, $B_{\text{avg}} = \frac{1}{N}\sum_i B_i$ and the centred deviations $\Delta A_i := A_i - A_{\text{avg}}$, $\Delta B_i := B_i - B_{\text{avg}}$. Because $\sum_i \Delta A_i = \sum_i \Delta B_i = 0$,

$$E = \frac{1}{N}\sum_{i=1}^{N} B_i A_i - B_{\text{avg}} A_{\text{avg}} = \frac{1}{N}\sum_{i=1}^{N} \Delta B_i\, \Delta A_i. \tag{A-1}$$

(P-1) Unbiased stochastic gradients. For each client and step, the SGD update is $\Delta A_i = -\eta\sum_{t=1}^{\tau} g_{i,t}^A$ with $\mathbb{E}[g_{i,t}^A] = 0$ and $\mathbb{E}\|g_{i,t}^A\|_F^2 \le \sigma_A^2$; similarly for B. Independence across clients implies $\mathbb{E}[\Delta B_i] = \mathbb{E}[\Delta A_i] = 0$ and $\mathbb{E}\|\Delta A_i\|_F^2 \le \tau\eta^2\sigma_A^2$. Applying Jensen and Cauchy–Schwarz to (A-1):

$$\mathbb{E}\|E\|_F^2 \le \frac{1}{N^2}\sum_{i=1}^{N} \mathbb{E}\big[\|\Delta B_i\|_F^2\big]\,\mathbb{E}\big[\|\Delta A_i\|_F^2\big] \le \frac{\tau^2\eta^2\sigma_A^2\sigma_B^2}{N},$$

which yields (P-1) after taking the square root.

(P-2) Deterministic bound. With per-step gradient norms bounded by G_A, G_B, $\|\Delta A_i\|_F \le \tau\eta G_A$, $\|\Delta B_i\|_F \le \tau\eta G_B$. Insert into (A-1) and apply Cauchy–Schwarz:

$$\|E\|_F \le \frac{1}{N}\sum_{i=1}^{N} \|\Delta B_i\|_F\,\|\Delta A_i\|_F \le \tau^2\eta^2 G_A G_B,$$

which is (P-2). □

C.2 Table of Results

For the convenience of the reader, we summarize the results of personalization and global training in Section C.2 and Section C.2, respectively.

Task	**Method**	**1**	**2**	**5**	**10**	**20**	**40**	**80**	**160**	**320**	**640**
MNLI	FedAvg	91.72	91.31	90.01	88.71	89.09	88.82	34.27	33.88	34.11	34.60
	FedProx	90.67	91.01	89.69	88.83	89.12	88.89	88.65	88.58	34.36	33.70
	FedMA	89.89	89.67	89.86	88.99	89.18	88.79	88.88	88.75	88.68	34.72
	APriLS	**93.48**	**91.29**	**91.01**	**90.43**	**90.30**	**89.74**	**89.69**	**89.85**	**89.51**	**89.61**
SST-2	FedAvg	96.06	95.45	94.25	93.00	93.50	50.38	49.94	51.31	49.43	52.22
	FedProx	95.05	95.45	94.15	93.15	93.60	92.75	92.61	50.01	49.93	50.13
	FedMA	94.25	94.25	94.25	93.50	93.65	93.15	93.36	93.22	50.18	52.67
	APriLS	**97.30**	**95.61**	**94.61**	**94.44**	**93.77**	**93.62**	**93.73**	**93.68**	**93.81**	**93.54**
QNLI	FedAvg	94.26	93.60	92.39	91.16	91.51	50.04	49.74	50.20	50.29	50.11
	FedProx	93.39	93.74	92.59	91.52	91.75	91.29	91.21	54.43	52.17	48.21
	FedMA	91.66	91.45	91.48	90.74	90.94	90.48	90.57	90.45	51.29	57.56
	APriLS	**95.15**	**93.54**	**92.63**	**92.33**	**91.98**	**91.85**	**91.91**	**91.97**	**91.83**	**91.74**
QQP	FedAvg	88.43	87.61	86.71	85.33	85.68	49.86	49.91	51.13	54.63	43.96
	FedProx	88.52	88.95	87.76	86.71	87.06	86.78	49.91	50.10	50.03	49.97
	FedMA	86.44	86.72	86.63	85.73	85.86	85.47	85.68	53.15	55.33	43.55
	APriLS	**90.92**	**88.77**	**88.49**	**87.63**	**87.61**	**87.10**	**87.19**	**87.36**	**87.14**	**87.11**
RTE	FedAvg	90.49	89.87	88.78	87.49	87.92	87.77	49.83	52.12	50.09	49.95
	FedProx	88.02	88.29	87.18	86.08	86.41	85.98	86.06	53.57	55.85	45.97
	FedMA	88.27	88.08	88.25	87.44	87.67	87.20	87.41	87.30	58.69	48.77
	APriLS	**91.22**	**89.35**	**88.98**	**88.13**	**87.92**	**87.42**	**87.61**	**87.18**	**87.21**	**87.37**
AVG	FedAvg	92.19	91.57	90.43	89.14	89.54	65.37	46.74	47.30	47.71	46.17
	FedProx	91.13	91.49	90.27	89.26	89.59	89.14	81.69	59.34	48.47	45.60
	FedMA	90.10	90.03	90.09	89.28	89.46	89.02	89.18	82.57	60.83	47.45
	APriLS	**93.61**	**91.71**	**91.14**	**90.59**	**90.32**	**89.95**	**90.03**	**90.01**	**89.90**	**89.87**

Task	Method	1	2	5	10	20	40	80	160	320	640
MNLI	FedAvg	87.33	87.59	87.49	87.11	86.71	33.81	33.54	33.97	34.56	33.96
	FedProx	87.65	87.34	87.77	87.28	86.76	87.07	33.57	34.07	34.11	33.90
	Freeze-A	87.68	87.62	87.77	87.44	87.27	87.19	87.16	34.28	34.50	33.64
	FedMA	87.87	87.91	87.66	87.59	87.19	87.40	87.46	87.02	34.64	34.13
	Full-Rrank (KnOTS)	**89.91**	88.98	**89.39**	**89.03**	**88.91**	**88.93**	**88.69**	**88.79**	**89.23**	**88.85**
	APriLS	89.71	**89.01**	89.28	88.92	88.97	88.76	88.58	88.82	89.13	88.75
SST-2	FedAvg	93.69	93.51	93.81	93.40	93.26	50.77	50.16	49.77	49.53	50.38
	FedProx	93.92	93.93	93.95	93.55	93.41	93.27	49.63	50.30	50.03	49.99
	Freeze-A	93.85	93.96	93.97	93.77	93.31	93.35	93.23	49.99	49.63	50.30
	FedMA	94.50	94.41	94.51	94.11	93.76	93.75	93.80	94.00	49.62	49.76
	Full-Rrank (KnOTS)	**96.09**	95.32	**95.42**	**95.17**	95.00	95.17	94.68	94.96	**94.99**	**94.95**
	APriLS	95.91	**95.18**	95.27	94.99	**94.93**	**95.18**	**94.68**	**94.85**	95.15	94.91
QNLI	FedAvg	89.07	88.98	88.93	88.83	88.56	50.73	49.80	50.03	50.09	50.02
	FedProx	89.61	89.60	89.28	89.12	88.77	88.87	49.77	49.64	50.02	50.13
	Freeze-A	89.37	89.48	89.76	89.30	89.08	89.13	88.96	50.40	49.72	50.27
	FedMA	89.92	89.69	89.91	89.53	89.25	89.11	89.11	89.16	49.60	50.40
	Full-Rrank (KnOTS)	**91.36**	**90.92**	**90.64**	90.48	**90.59**	**90.96**	**90.82**	**90.20**	**90.75**	**90.61**
	APriLS	91.38	90.76	90.70	**90.63**	90.47	90.86	90.66	90.36	90.92	90.53
QQP	FedAvg	81.76	81.93	82.00	81.55	81.01	50.60	50.05	49.68	49.66	50.45
	FedProx	82.38	82.11	82.16	82.06	81.63	81.96	49.67	50.05	50.08	50.25
	Freeze-A	82.54	82.76	82.55	82.39	82.13	81.92	81.97	50.29	50.06	50.18
	FedMA	82.73	82.86	82.81	82.64	82.42	82.05	82.34	82.09	50.03	50.37
	Full-Rrank (KnOTS)	**84.44**	**83.77**	**83.47**	**83.47**	**83.57**	**83.14**	**83.23**	**83.49**	**83.37**	**83.08**
	APriLS	84.29	83.62	83.58	83.31	83.53	83.20	83.15	83.32	83.42	83.00
RTE	FedAvg	86.94	87.11	86.72	86.61	86.25	50.68	50.13	50.17	50.09	50.02
	FedProx	86.90	87.13	87.13	86.72	86.19	86.16	49.95	50.01	50.17	50.25
	Freeze-A	87.34	87.38	87.42	86.97	86.70	86.68	86.82	50.07	49.56	50.10
	FedMA	87.32	87.21	87.35	87.03	86.91	86.91	86.52	86.75	49.63	50.15
	Full-Rrank (KnOTS)	**88.96**	**88.41**	**88.49**	88.16	88.28	88.04	88.06	88.00	**88.26**	88.43
	APriLS	88.85	88.21	88.39	**88.09**	**88.38**	**88.08**	**87.90**	**88.17**	88.25	**88.46**
AVG	FedAvg	87.16	88.18	87.79	87.14	87.15	43.61	42.72	43.19	42.79	43.29
	FedProx	87.29	88.02	87.86	87.35	87.35	87.46	42.12	42.99	42.88	43.08
	Freeze-A	87.76	88.04	87.89	87.57	87.30	87.65	87.23	54.61	46.29	46.46
	FedMA	88.07	88.22	87.84	87.78	87.51	87.44	87.45	87.00	46.19	46.56
	Full-Rrank (KnOTS)	**90.15**	**89.88**	**89.88**	**89.86**	**89.67**	**89.85**	**89.70**	**89.89**	**89.96**	**89.99**
	APriLS	90.03	89.76	89.84	89.67	89.66	89.75	89.58	89.64	89.87	89.93

Bibliography

[1] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[2] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[3] Jianyu Wang, Ananda Theertha Yu, and Gregory Wornell. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[4] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.

[5] Yuzhu Mao, Zihao Zhao, Guangfeng Yan, Yang Liu, Tian Lan, Linqi Song, and Wenbo Ding. Communication-efficient federated learning with adaptive quantization. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–26, 2022.

[6] Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

[7] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

[8] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. In *IEEE transactions on neural networks and learning systems*, 2019.

[9] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497*, 2020.

[10] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32, 1992.

[11] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

[12] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

[13] Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[14] Joel Lehman and Risto Miikkulainen. Neuroevolution. *Scholarpedia*, 8(6):30977, 2013.

[15] Sebastian Risi and Julian Togelius. Neuroevolution in games: State of the art and open challenges. *CoRR*, abs/1410.7326, 2014.

[16] Sebastian Risi and Julian Togelius. Neuroevolution in games: State of the art and open challenges. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(1):25–41, 2015.

[17] Daniel Golovin, John Karro, Greg Kochanski, Chansoo Lee, Xingyou Song, and Qiuyi Zhang. Gradientless descent: High-dimensional zeroth-order optimization. *arXiv preprint arXiv:1911.06317*, 2019.

[18] Robert Tjarko Lange, Tom Schaul, Yutian Chen, Chris Lu, Tom Zahavy, Valentin Dalibard, and Sebastian Flennerhag. Discovering attention-based genetic algorithms via meta-black-box optimization. *arXiv preprint arXiv:2304.03995*, 2023.

[19] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Sparse binary compression: Towards distributed deep learning with minimal communication. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[20] Jinjin Xu, Wenli Du, Yaochu Jin, Wangli He, and Ran Cheng. Ternary compression for communication-efficient federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[21] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[22] Sohei Itahara, Takayuki Nishio, Yusuke Koda, Masahiro Morikura, and Koji Yamamoto. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *arXiv preprint arXiv:2008.06180*, 2020.

[23] Felix Sattler, Arturo Marban, Roman Rischke, and Wojciech Samek. Communication-efficient federated distillation. *arXiv preprint arXiv:2012.00632*, 2020.

[24] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

[25] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. Communication-efficient federated learning via knowledge distillation. *Nature communications*, 13(1):1–8, 2022.

[26] Mingzhe Chen, Nir Shlezinger, H Vincent Poor, Yonina C Eldar, and Shuguang Cui. Communication-efficient federated learning. *Proceedings of the National Academy of Sciences*, 118(17):e2024789118, 2021.

[27] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE international conference on communications (ICC)*, pages 1–7. IEEE, 2019.

[28] Canh T Dinh, Nguyen H Tran, Minh NH Nguyen, Choong Seon Hong, Wei Bao, Albert Y Zomaya, and Vincent Gramoli. Federated learning over wireless networks: Convergence analysis and resource allocation. *IEEE/ACM Transactions on Networking*, 29(1):398–409, 2020.

[29] Alois Huning. Evolutionsstrategie. optimierung technischer systeme nach prinzipien der biologischen evolution, 1976.

[30] Daan Wierstra, Tom Schaul, Jan Peters, and Jürgen Schmidhuber. Fitness expectation maximization. In *Parallel Problem Solving from Nature–PPSN X: 10th International Conference, Dortmund, Germany, September 13-17, 2008. Proceedings 10*, pages 337–346. Springer, 2008.

[31] Sun Yi, Daan Wierstra, Tom Schaul, and Jürgen Schmidhuber. Stochastic search using the natural gradient. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1161–1168, 2009.

[32] Yi Sun, Daan Wierstra, Tom Schaul, and Jürgen Schmidhuber. Efficient natural evolution strategies. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 539–546, 2009.

[33] Tobias Glasmachers, Tom Schaul, and Jürgen Schmidhuber. A natural evolution strategy for multi-objective optimization. In *Parallel Problem Solving from Nature, PPSN XI: 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I 11*, pages 627–636. Springer, 2010.

[34] Tobias Glasmachers, Tom Schaul, Sun Yi, Daan Wierstra, and Jürgen Schmidhuber. Exponential natural evolution strategies. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 393–400, 2010.

[35] Tom Schaul, Tobias Glasmachers, and Jürgen Schmidhuber. High dimensions and heavy tails for natural evolution strategies. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 845–852, 2011.

[36] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[37] James C Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341, 1992.

[38] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17:527–566, 2017.

[39] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[40] Hangyu Zhu and Yaochu Jin. Real-time federated evolutionary neural architecture search. *IEEE Transactions on Evolutionary Computation*, 26(2):364–378, 2021.

[41] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.

[42] Hangyu Zhu and Yaochu Jin. Multi-objective evolutionary federated learning. *IEEE transactions on neural networks and learning systems*, 31(4):1310–1322, 2019.

[43] Zheng-yi Chai, Chuan-dong Yang, and Ya-lun Li. Communication efficiency optimization in federated learning based on multi-objective evolutionary algorithm. *Evolutionary Intelligence*, pages 1–12, 2022.

[44] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[45] Ivanoe De Falco, Antonio Della Cioppa, Tomas Koutny, Martin Ubl, Michal Krcma, Umberto Scafuri, and Ernesto Tarantino. A federated learning-inspired evolutionary algorithm: Application to glucose prediction. *Sensors*, 23(6):2957, 2023.

[46] John Geweke. Antithetic acceleration of monte carlo integration in bayesian inference. *Journal of Econometrics*, 38(1-2):73–89, 1988.

[47] Dimo Brockhoff, Anne Auger, Nikolaus Hansen, Dirk V Arnold, and Tim Hohm. Mirrored sampling and sequential selection for evolution strategies. In *Parallel Problem Solving from Nature, PPSN XI: 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I 11*, pages 11–21. Springer, 2010.

[48] Masahiro Yagisawa. Fully homomorphic encryption without bootstrapping. *Cryptology ePrint Archive*, 2015.

[49] Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic library-seal v2. 1. In *Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers 21*, pages 3–18. Springer, 2017.

[50] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, 2012.

[51] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. Innovative technology for cpu based attestation and sealing. In *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, volume 13. ACM New York, NY, USA, 2013.

[52] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[53] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[54] Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010.

[55] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs. *Unpublished manuscript*, 2018.

[56] Robert Tjarko Lange. evosax: Jax-based evolution strategies. *arXiv preprint arXiv:2212.04180*, 2022.

[57] Ninghui Jia, Zhihao Qu, Baoliu Ye, Yanyan Wang, Shihong Hu, and Song Guo. A comprehensive survey on communication-efficient federated learning in mobile edge environments. *IEEE Communications Surveys & Tutorials*, 2025.

[58] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6655–6659. IEEE, 2013.

[59] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[60] Yipeng Liu, Jiani Liu, Zhen Long, Ce Zhu, Yipeng Liu, Jiani Liu, Zhen Long, and Ce Zhu. Tensor decomposition in deep networks. *Tensor Computation for Data Analysis*, pages 241–263, 2022.

[61] Wenqi Wang, Yifan Sun, Brian Eriksson, Wenlin Wang, and Vaneet Aggarwal. Wide compression: Tensor ring nets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9329–9338, 2018.

[62] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[63] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

[64] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. *Advances in neural information processing systems*, 26, 2013.

[65] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[66] Jingfei Zhao, Ilia Shumailov, Takuma Chinen, Ilia Shumailov, and Dawn Song. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2306.03341*, 2023.

[67] Haemin Park and Diego Klabjan. Communication-efficient federated low-rank update algorithm and its connection to implicit regularization. *arXiv preprint arXiv:2409.12371*, 2024.

[68] Mingzhao Guo, Dongzhu Liu, Osvaldo Simeone, and Dingzhu Wen. Low-rank gradient compression with error feedback for mimo wireless federated learning. *arXiv preprint arXiv:2401.07496*, 2024.

[69] Sixu Hu, Linshan Jiang, and Bingsheng He. Practical hybrid gradient compression for federated learning systems. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 4147–4155, 2024.

[70] D Yao, W Pan, Y Wan, H Jin, and L Sun. Fedhm: Efficient federated learning for heterogeneous models via low-rank factorization. arxiv. *arXiv preprint arXiv:2111.14655*, 2021.

[71] Nam Hyeon-Woo, Moon Ye-Bin, and Tae-Hyun Oh. Fedpara: Low-rank hadamard product for communication-efficient federated learning. *arXiv preprint arXiv:2108.06098*, 2021.

[72] Wonyong Jeong and Sung Ju Hwang. Factorized-fl: Personalized federated learning with parameter factorization & similarity matching. *Advances in Neural Information Processing Systems*, 35:35684–35695, 2022.

[73] Muhammad Ghufran Areeb Hameed, Thuong-Hai Bui, Yookyung Park, Shafiq Joty, and Steven CH Hoi. Rosa: Random subspace adaptation for efficient fine-tuning. In *International Conference on Learning Representations (ICLR)*, 2023.

[74] Haoran Zhao, Jiayu Zhang, Qinbin Sun, Zhouchen Lin, Yang Wang, Yefeng Zheng, and Shiqiang Liu. Separate: A simple low-rank projection for gradient compression. *arXiv preprint arXiv:2309.08386*, 2023.

[75] Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving lora in privacy-preserving federated learning. *arXiv preprint arXiv:2403.12313*, 2024.

[76] Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*, 2023.

[77] Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. Asymmetry in low-rank adapters of foundation models. *arXiv preprint arXiv:2402.16842*, 2024.

[78] Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. *arXiv preprint arXiv:2402.03293*, 2024.

[79] Pengxin Guo, Shuang Zeng, Yanran Wang, Huijie Fan, Feifei Wang, and Liangqiong Qu. Selective aggregation for low-rank adaptation in federated learning. *arXiv preprint arXiv:2410.01463*, 2024.

[80] Mohammad Mahdi Rahimi, Hasnain Irshad Bhatti, Younghyun Park, Humaira Kousar, Do-Yeon Kim, and Jaekyun Moon. Evofed: leveraging evolutionary strategies for communication-efficient federated learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[81] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning (ICML)*, 2018.

[82] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[83] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations (ICLR)*, 2018.

[84] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *International Conference on Artificial Intelligence and Statistics*, pages 2021–2031. PMLR, 2020.

[85] Shiqiang Sun, Jakub Konecny, Ananda Theertha Suresh, and Brendan McMahan. Qfedavg: Quantized federated averaging. *arXiv preprint arXiv:2002.05645*, 2020.

[86] Sheikh Shams Azam, Seyyedali Hosseinalipour, Qiang Qiu, and Christopher Brinton. Recycling model updates in federated learning: Are gradient subspaces low-rank? In *International Conference on Learning Representations*, 2021.

[87] Yongjeong Oh, Yo-Seb Jeon, Mingzhe Chen, and Walid Saad. Vector quantized compressed sensing for communication-efficient federated learning. In *2022 IEEE Globecom Workshops (GC Wkshps)*, pages 365–370. IEEE, 2022.

[88] Sangjun Park and Wan Choi. Regulated subspace projection based local model update compression for communication-efficient federated learning. *IEEE Journal on Selected Areas in Communications*, 41(4):964–976, 2023.

[89] Zai Shi and Atilla Eryilmaz. Communication-efficient subspace methods for high-dimensional federated learning. In *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*, pages 543–550. IEEE, 2021.

[90] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[91] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[92] Chunyuan Li, Hang Su, Xiaowei Shen, Yizhe Li, Yiren Wang, Yiming Chen, and Lawrence Carin. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations (ICLR)*, 2018.

[93] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[94] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[95] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[96] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European conference on computer vision (ECCV)*, pages 184–199, 2018.

[97] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning (ICML)*, 2015.

[98] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, 2016.

[99] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*, 2017.

[100] Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1st edition, 2018.

[101] Terence Tao. *Topics in Random Matrix Theory*. Graduate Studies in Mathematics, Vol. 132. American Mathematical Society, 2012.

[102] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Unpublished*, 2009.

[103] Stanford University. Tiny imagenet visual recognition challenge. https://www.kaggle.com/c/tiny-imagenet, 2015.

[104] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[105] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.

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

[107] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[108] Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. Llava-med: Training a large language-and-vision assistant for biomedicine in one day. *Advances in Neural Information Processing Systems*, 36:28541–28564, 2023.

[109] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

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

[111] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*, 2023.

[112] Yuren Mao, Yuhang Ge, Yijiang Fan, Wenyi Xu, Yu Mi, Zhonghao Hu, and Yunjun Gao. A survey on lora of large language models. *Frontiers of Computer Science*, 19(7):197605, 2025.

[113] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE transactions on neural networks and learning systems*, 34(12):9587–9603, 2022.

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

[115] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International conference on machine learning*, pages 7252–7261. PMLR, 2019.

[116] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[117] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

[118] Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein. On the loss landscape of a class of deep neural networks with no bad local valleys. *arXiv preprint arXiv:1809.10749*, 2018.

[119] Dawei Li, Tian Ding, and Ruoyu Sun. On the benefit of width for neural networks: Disappearance of bad basins. *arXiv preprint arXiv:1812.11039*, 2018.

[120] Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116, 2022.

[121] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. arxiv. *Learning*, 2018.

[122] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for on-device federated learning. *arXiv preprint arXiv:1910.06378*, 2(6), 2019.

[123] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[124] C Daniel Freeman and Joan Bruna. Topology and geometry of half-rectified network optimization. *arXiv preprint arXiv:1611.01540*, 2016.

[125] Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. Repair: Renormalizing permuted activations for interpolation repair. *arXiv preprint arXiv:2211.08403*, 2022.

[126] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[127] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7252–7261. PMLR, 09–15 Jun 2019.

[128] Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

[129] N. Joseph Tatro, Pin-Yu Chen, Payel Das, Igor Melnyk, Prasanna Sattigeri, and Rongjie Lai. Optimizing mode connectivity via neuron alignment. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[130] Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries, 2023.

[131] Fuxun Yu, Weishan Zhang, Zhuwei Qin, Zirui Xu, Di Wang, Chenchen Liu, Zhi Tian, and Xiang Chen. Fed2: Feature-aligned federated learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 2066–2074, 2021.

[132] Theo Putterman, Derek Lim, Yoav Gelberg, Stefanie Jegelka, and Haggai Maron. Learning on loras: Gl-equivariant processing of low-rank weight spaces for large finetuned models. *arXiv preprint arXiv:2410.04207*, 2024.

[133] George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with svd to tie the knots. *arXiv preprint arXiv:2410.19735*, 2024.

[134] Yongle Li, Bo Liu, Sheng Huang, ZHeng ZHang, Xiaotong Yuan, and Richang Hong. Communication-efficient and personalized federated foundation model fine-tuning via tri-matrix adaptation. *arXiv preprint arXiv:2503.23869*, 2025.

[135] Tailin Zhou, Jun Zhang, and Danny HK Tsang. Mode connectivity and data heterogeneity of federated learning. *arXiv preprint arXiv:2309.16923*, 2023.

[136] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[137] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International conference on machine learning*, pages 2089–2099. PMLR, 2021.

[138] Seok-Ju Hahn, Minwoo Jeong, and Junghye Lee. Connecting low-loss subspace for personalized federated learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 505–515, 2022.

[139] George Stoica, Daniel Bolya, Jakob Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. *arXiv preprint arXiv:2305.03053*, 2023.

[140] Stefan Horoi, Albert Manuel Orozco Camacho, Eugene Belilovsky, and Guy Wolf. Harmony in diversity: Merging neural networks with canonical correlation analysis. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[141] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[142] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, 2019.

[143] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[144] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[145] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

[146] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[147] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.

[148] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[149] Tianyu Ding, Tianyi Chen, Haidong Zhu, Jiachen Jiang, Yiqi Zhong, Jinxin Zhou, Guangzhi Wang, Zhihui Zhu, Ilya Zharkov, and Luming Liang. The efficiency spectrum of large language models: An algorithmic survey. *arXiv preprint arXiv:2312.00678*, 2023.

[150] Wenzhi Fang, Dong-Jun Han, Liangqi Yuan, Seyyedali Hosseinalipour, and Christopher G Brinton. Federated sketching lora: On-device collaborative fine-tuning of large language models. *arXiv preprint arXiv:2501.19389*, 2025.

[151] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *International Conference on Learning Representations*, 2016.

[152] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*, 2021.

[153] Theo Putterman, Derek Lim, Yoav Gelberg, Stefanie Jegelka, and Haggai Maron. Learning on loras: Gl-equivariant processing of low-rank weight spaces for large finetuned models. *arXiv preprint arXiv:2410.04207*, 2024.

[154] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *arXiv preprint arXiv:2306.01708*, 2023.

[155] Yuexiang Xie, Zhen Wang, Dawei Gao, Daoyuan Chen, Liuyi Yao, Weirui Kuang, Yaliang Li, Bolin Ding, and Jingren Zhou. Federatedscope: A flexible federated learning platform for heterogeneity. *Proceedings of the VLDB Endowment*, 16(5):1059–1072, 2023.

Acknowledgment

The completion of this doctoral journey would not have been possible without the invaluable support, guidance, and encouragement from numerous individuals, to whom I owe my deepest gratitude.

First and foremost, I would like to express my sincere appreciation to my advisor, Professor Jaekyun Moon, for their exemplary mentorship, profound knowledge, and unwavering support throughout my PhD research. Their insightful feedback, patience, and trust provided the foundation that allowed me to pursue my ideas with confidence and rigor.

I am grateful to the distinguished members of my doctoral committee, whose constructive feedback, critical evaluation, and valuable suggestions significantly improved the quality of this thesis. Their intellectual input and expertise have profoundly shaped my academic growth.

My heartfelt thanks go to my colleagues and lab members at KAIST, especially Hasnain Irshad Bhatti, Humaira Kousar, Younghyun Park, and Dong-jun Han, whose collaborative spirit, stimulating discussions, and camaraderie made my research experience not only productive but truly enjoyable. Their friendship and cooperation fostered an environment that continually motivated and inspired me.

I wish to thank the administrative and support staff at KAIST for their assistance and responsiveness throughout my PhD program. Their efficiency and kindness made navigating administrative challenges smooth and manageable.

I am deeply grateful to the friends and colleagues I had the privilege of meeting during my PhD journey. Their companionship, insightful conversations, encouragement, and emotional support made this challenging path not only bearable but deeply meaningful and fulfilling.

I would like to thank my family. To my parents, I owe immeasurable thanks for their unconditional love, encouragement, and patience. Their steadfast emotional support and unwavering belief in me have been the foundation of my perseverance through every challenge. To my sisters, Reyhane, and Rahil and my only brother Amirhossein, I extend my deepest gratitude for their constant encouragement, love, and support. Their belief in me and their companionship through every stage of this journey have been truly invaluable. Their presence continues to inspire and uplift me every day.

Finally, I extend my deepest gratitude to Maria, whose unwavering support, understanding, and companionship have been invaluable sources of strength and joy throughout this journey.

This achievement is a collective effort, and I dedicate this thesis to everyone mentioned above, who made this accomplishment possible.

Mohammad Mahdi Rahimi
KAIST, 2025
June 20, 2025