FedGBA: Gradient-Boosted Adaption for Federated Fine-tuning of Foundation Model

Yifei Zhang*
yifei.zhang@nwpu.edu.cn
School of Computer Science, Northwestern Polytechnical University
Xi'an, Shanxi, China

Hao Zhu*
allenhaozhu@gmail.com
Data61/CSIRO
Canberra, Australia

Libin Wang
lbwang@nwpu.edu.cn
School of Computer Science, Northwestern Polytechnical University
Xi'an, Shanxi, China

Xiaolin Han
xiaolinh@nwpu.edu.cn
School of Computer Science, Northwestern Polytechnical University
Xi'an, Shanxi, China

Yupei Zhang
ypzhaang@nwpu.edu.cn
School of Computer Science, Northwestern Polytechnical University
Xi'an, Shanxi, China

Piotr Koniusz†
piotr.koniusz@data61.csiro.au
Data61/CSIRO
Australian National University
Canberra, Australia

Xuequn Shang†
shang@nwpu.edu.cn
School of Computer Science, Northwestern Polytechnical University
Xi'an, Shanxi, China

Abstract

The rapid advancement of large language models (LLMs) has revolutionized natural language processing, enabling breakthrough performance on various tasks. However, fine-tuning these models in a federated setting, where data is distributed across multiple participants, raises critical challenges regarding data privacy, communication, and computation efficiency. Existing approaches for federated fine-tuning of LLMs, such as FedLoRA and FLoRA, face a fundamental trade-off between expressiveness and efficiency due to the constraints on the rank of the low-rank adaptation (LoRA) matrices used to update the model. In this paper, we propose Federated Gradient Boost Adaptation (FedGBA), a novel framework that leverages the power of ensemble learning to address this trade-off. FedGBA iteratively learns a series of weak LoRA models, each focusing on correcting the errors of the previous ones, and combines them to create a strong federated model. By doing so, FedGBA achieves high expressiveness while using low-rank updates, significantly reducing the communication and computational costs compared to existing approaches. We provide a theoretical analysis of FedGBA, establishing convergence guarantees and expressiveness bounds that characterize the relationship between the number of boosting rounds, the rank of the LoRA matrices, and the approximation error. Our experimental results on a range of natural language processing tasks demonstrate that FedGBA consistently outperforms standard federated learning approaches, achieving better performance while using significantly fewer parameters.

*Both authors contributed equally to this research.
†Corresponding Author

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

CCS Concepts

• **Information systems** → *Computing platforms.*

Keywords

Federated Learning, Large Language Models, Parameter-Efficient Fine-Tuning, Gradient Boosting

ACM Reference Format:
Yifei Zhang, Hao Zhu, Libin Wang, Xiaolin Han, Yupei Zhang, Piotr Koniusz, and Xuequn Shang. 2018. FedGBA: Gradient-Boosted Adaption for Federated Fine-tuning of Foundation Model. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The rapid growth of large language models (LLMs) has opened up new possibilities for natural language processing, enabling breakthroughs in a wide range of applications [4, 6, 31]. However, developing and deploying these powerful models raises critical concerns about data privacy, security, and the responsible use of AI [23]. As LLMs are increasingly being trained on sensitive user data, it is crucial to develop methods that allow for efficient fine-tuning while safeguarding the privacy of individuals [20]. As LLMs are increasingly being trained on personal data, it is crucial to develop methods that allow for efficient fine-tuning while safeguarding the privacy of individuals [20]. This is particularly important in the context of responsible web applications, where LLMs are often integrated to enhance user experiences through features such as intelligent search, content recommendation, and natural language interfaces.

Federated learning has emerged as a promising approach for training models on decentralized data, enabling multiple parties to collaborate without directly sharing their data [27, 28]. However, fine-tuning LLMs in a federated setting presents significant challenges in terms of efficient communication, computational overhead, and accurate model aggregation [25, 39]. Recent works, such as FedLoRA [42] and FLoRA [41], have explored the integration

of low-rank adaptation (LoRA) [18] with federated learning, significantly reducing the communication and computational costs compared to full model fine-tuning while maintaining comparable performance.

Despite these advances, integrating LoRA with federated learning introduces new challenges in the aggregation stage. While LoRA reduces the size of parameter updates from individual clients, the process of aggregating these low-rank updates across multiple participants introduces additional computational complexity and potential accuracy loss. Standard aggregation methods such as FedAvg can lead to mathematical inaccuracies when applied to LoRA matrices, while more sophisticated aggregation schemes may increase computational overhead at the server [43]. This means that in practice, the efficiency gains promised by LoRA in federated settings are often diminished by the complexities of the aggregation process. This challenge becomes particularly acute as the number of participating clients grows, creating a bottleneck at the aggregation stage that can limit the scalability of federated learning with LoRA. The need to maintain accurate model updates while keeping the aggregation process efficient and scalable remains a significant obstacle in deploying LoRA-based federated learning systems [13, 15, 24].

To address this challenge, we propose Federated Gradient Boost Adaption (`FedGBA`), a novel framework that leverages the power of ensemble learning to enable efficient and accurate aggregation in federated fine-tuning of LLMs. Inspired by gradient boosting [10, 11], `FedGBA` iteratively learns and aggregates a series of simple LoRA models from clients. By decomposing the adaptation process into a sequence of lightweight updates that can be efficiently aggregated, `FedGBA` simplifies the aggregation stage while maintaining model performance. This approach significantly reduces both the computational overhead at the server and the communication costs between clients. The key contributions of our work are as follows:

i. We introduce `FedGBA`, a novel federated learning framework that leverages gradient boosting for efficient and accurate aggregation of LoRA updates. By decomposing the adaptation process into a sequence of simple updates, we simplify the aggregation step while maintaining model performance.
ii. We provide theoretical analysis of `FedGBA`, establishing convergence guarantees and aggregation error bounds that characterize how the boosting mechanism enables accurate model aggregation with reduced computational complexity. Our analysis provides insights into the trade-offs between aggregation accuracy, computational cost, and the number of boosting rounds.
iii. Through extensive experiments on a range of natural language processing tasks, we demonstrate that `FedGBA` significantly reduces server-side computational overhead while maintaining or improving model performance compared to existing federated learning approaches. Our results show that `FedGBA` scales efficiently with the number of clients while preserving the accuracy of aggregated updates.

2 Related Works

Fine-tuning on Foundation Model has become a prevailing approach for adapting these models to specific downstream tasks [7, 8, 16, 18]. The process involves training the model on a task-specific dataset, usually with a smaller learning rate compared to pre-training, to adapt its parameters to the target task. Fine-tuning has been successfully applied to a wide range of natural language processing tasks, including text classification, question answering, and natural language inference [4, 21, 30, 37]. However, fine-tuning LLMs faces several challenges. A major challenge is the computational complexity and memory requirements associated with updating billions of model parameters, which can be prohibitively expensive and time-consuming [1–3, 5, 19, 32, 34, 35, 37, 38, 46]. Additionally, the limited availability of labeled data for specific tasks poses challenges in terms of sample efficiency and generalization ability [47]. To address these challenges, various approaches have been proposed to improve the efficiency and effectiveness of LLM fine-tuning. One notable technique is LoRA [17] which freezes the pre-trained model's weights and introduces a low-rank matrix to adapt the model to new tasks. LoRA reduces the number of trainable parameters and reduces the computational burden of LLM fine-tuning. Recently, [43] provided theoretical results that characterize the expressive power of LoRA for Fully Connected Neural Networks (FCNN) and Transformer Networks (TFN), which identify the necessary rank of LoRA for adapting a frozen model to exactly match a target model. For Transformer networks, any model can be adapted to a target model of the same size with LoRA adapters of `rank_r=embedding_size/2`.

Fine-tuning on Federated Foundation Model. Recent advances in federated fine-tuning of Large Language Models (LLMs) have led to the development of FedIT [44] and FLoRA FLoRA [40], two significant contributions in this field. FedIT, introduced by Zhang et al. [44], integrates Low-Rank Adaptation (LoRA) with Federated Averaging (FedAvg) to enable efficient, privacy-preserving fine-tuning of LLMs. While FedIT significantly reduces computational and communication costs by focusing on LoRA parameters, it faces limitations due to significant aggregation noise and homogeneity constraints. FLoRA, building upon FedIT, addresses these limitations by introducing a novel stacking-based aggregation method for LoRA modules, eliminating mathematical inaccuracies present in FedIT's averaging method and supporting heterogeneous LoRA ranks across clients.

Gradient Boosting. Friedman [10, 11] is a powerful ensemble learning technique combining multiple weak learners to create a strong learner. The theoretical foundations of gradient boosting have been extensively studied, providing insights into its convergence properties, generalization ability, and robustness to overfitting. A seminal work on gradient boosting theory by Zhang and Yu [48] proved that gradient boosting achieves the optimal convergence rate for a broad class of loss functions, highlighting its theoretical optimality. Koltchinskii and Panchenko [22] further investigated theoretical properties of gradient boosting from the perspective of empirical risk minimization. They derived bounds on the generalization error of gradient boosting and showed that the technique is resilient to overfitting when the base learners are weak and the step size is chosen well. They include insights into its convergence behavior, generalization ability, and robustness, which are relevant to the theoretical analysis of our `FedGBA` framework.

3 Preliminary

Low-Rank Adaptation (LoRA) [17]. Fine-tuning large language models typically involves updating all the parameters of the pre-trained model to adapt it to a downstream task. This can be formally represented as:

$$\boldsymbol{W} = \boldsymbol{W}_0 + \Delta \boldsymbol{W}, \tag{1}$$

where $\boldsymbol{W}_0 \in \mathbb{R}^{d\times k}$ is the pre-trained weight matrix and $\Delta\boldsymbol{W}$ is the update matrix learned during fine-tuning. However, this approach has several disadvantages. Updating all the parameters of a large language model is computationally expensive, requiring significant time and resources. Storing separate copies of the fine-tuned models for each downstream task can be memory-intensive, especially when dealing with large models and multiple tasks. Fine-tuning all the parameters on a relatively small downstream dataset can lead to overfitting, especially when the target task has limited data.

Low-Rank Adaptation (LoRA) [18] addresses these issues by proposing a parameter-efficient fine-tuning technique designed specifically for adapting large language models to downstream tasks. The key idea behind LoRA is to freeze the pre-trained model's weights and inject trainable low-rank matrices into each layer of the model. By doing so, LoRA significantly reduces the number of trainable parameters while still allowing the model to adapt to specific tasks. Formally, let $\boldsymbol{W} \in \mathbb{R}^{d\times k}$ be the weight matrix of a fully connected layer in a pre-trained language model, where d is the input dimension and k is the output dimension. LoRA introduces a low-rank decomposition of the updated weight matrix:

$$\boldsymbol{W} = \boldsymbol{W}_0 + \Delta\boldsymbol{W}, \text{s.t. } \Delta\boldsymbol{W} = \boldsymbol{A}\boldsymbol{B}, \tag{2}$$

where $\boldsymbol{W}_0$ is the original pre-trained weight matrix, $\boldsymbol{A} \in \mathbb{R}^{d\times r}$ and $\boldsymbol{B} \in \mathbb{R}^{r\times k}$ are trainable matrices, and r is the rank of the adaptation. During fine-tuning, only the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are updated, while the original weights $\boldsymbol{W}_0$ remain frozen. This reduces the number of trainable parameters from dk to $(d+k)r$, which is significantly smaller when $r \ll \min(d,k)$. LoRA offers several advantages over traditional fine-tuning methods. By updating only the low-rank matrices, LoRA significantly reduces the computational cost of fine-tuning large language models. LoRA eliminates the need to store separate copies of fine-tuned models, as the original model weights remain unchanged, and only the low-rank matrices need to be stored for each task. By freezing the original model weights and learning only a small number of task-specific parameters, LoRA reduces the risk of overfitting on small downstream datasets.

Remarkably, despite its parameter efficiency, LoRA has been shown to achieve performance comparable to or even better than full fine-tuning on various natural language processing tasks [43], making it a promising technique for efficient and effective model adaptation. However, LoRA has a theoretical limitation [43]. To fully recover the performance of fine-tuning, the rank of the adaptation matrices must satisfy $r \geq$ embedding_size/2. In practice, much smaller ranks (*e.g.*, $r \in [8, 32]$) are often used to trade-off performance with efficiency. The discrepancy between the theoretical optimum and practical usage leads to a performance gap. Increasing the rank to satisfy the theoretical requirement increases memory usage and computational complexity, negating the benefits of LoRA by making it as costly as the full fine-tuning strategy.

Federated Fine-tuning with LoRA [40, 42]. It combines the parameter-efficient fine-tuning capabilities of LoRA with the privacy-preserving nature of federated learning (FL) [14]. In this setting, multiple clients collaborate to fine-tune a pre-trained language model using LoRA while keeping their local data private. The process can be described as follows:

i. *Initialization.* The server initializes a pre-trained Foundation Model $\boldsymbol{W}_0$ and sends it to the clients.
ii. *Local Low-Rank Adaptation.* Each client m initializes their local low-rank adaptation matrices $\boldsymbol{A}_m$ and $\boldsymbol{B}_m$ and trains them on their local data $\mathcal{D}_m$ to minimize the local objective:
$$\min_{\boldsymbol{A}_m, \boldsymbol{B}_m} \mathcal{L}_m(\boldsymbol{W}_0 + \boldsymbol{A}_m\boldsymbol{B}_m; \mathcal{D}_m).$$
iii. *Local Update Communication.* The clients send their updated low-rank adaptation matrices $\boldsymbol{A}_m$ and $\boldsymbol{B}_m$ (or some function of these matrices) to the server.
iv. *Server Aggregation.* The server aggregates the updates received from the clients to obtain a global update. The specific aggregation method can vary (see below for possible options).
v. *Global Model Update.* The server updates the global model using the aggregated update and sends the updated model (or the update itself) back to the clients.
vi. *Iteration.* Steps ii-v are repeated until convergence or a desired number of rounds is reached.

Federated fine-tuning with LoRA offers several advantages, such as reducing communication costs compared to sharing full model updates and preserving data privacy by keeping the data decentralized. However, it also introduces new challenges that are related to the limitations of LoRA discussed earlier.

The choice of aggregation method in step iv of the generalized framework can significantly impact the performance, communication efficiency, and accuracy of the federated fine-tuning process. Below we discuss two popular aggregation methods.

Federated Averaging (FedAvg). The server computes the average of the low-rank adaptation matrices received from the clients:

$$\bar{\boldsymbol{A}} = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{A}_m, \quad \bar{\boldsymbol{B}} = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{B}_m.$$

The global model is then updated using the averaged matrices: $\boldsymbol{W}_{avg} = \boldsymbol{W}_0 + \bar{\boldsymbol{A}}\bar{\boldsymbol{B}}$. Due to the cross interaction between $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{B}}$ in their matrix product, this averaging approach introduces mathematical inaccuracies as $\frac{1}{M}\sum_{m=1}^{M} \boldsymbol{A}_m\boldsymbol{B}_m \neq \bar{\boldsymbol{A}}\bar{\boldsymbol{B}}$.

Federated Stacking (FedStack). The server first collects the local LoRA matrices $\boldsymbol{A}_m\boldsymbol{B}_m$ and then stacks them as:

$$\bar{\boldsymbol{A}} = [\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{B}_M] \quad \text{and} \quad \bar{\boldsymbol{B}} = [\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_M]. \tag{3}$$

The $\Delta\boldsymbol{W}_{stack}$ is computed as $\Delta\boldsymbol{W}_{stack} = \bar{\boldsymbol{A}}\bar{\boldsymbol{B}}$ which results in the exact aggregation of LoRA module:

$$\Delta\boldsymbol{W}_{stack} = \bar{\boldsymbol{A}}\bar{\boldsymbol{B}} = \sum_{m=1}^{M} \boldsymbol{A}_m\boldsymbol{B}_m \neq \Delta\boldsymbol{W}_{avg}. \tag{4}$$

However, such an exact aggregation comes at the cost of increased communication overhead, as the size of stacked matrices grows linearly with the number of clients M, making it impractical for scenarios with many participants.

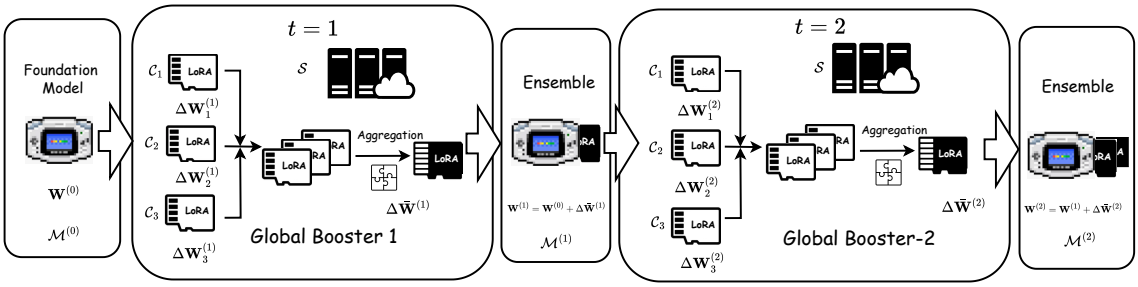


Figure 1: The pipeline of FedGBA: A global booster is aggregated via collect the local booster. Then, it merge to the FM model to ensemble the strong model. In the next iteration, new LoRA booster is then learnt.

4 Methodology

Below, we present the FedGBA method, an iterative fine-tuning framework inspired by Gradient Boosting in FL. The key stages in each GB iteration include tuning a LoRA module as a booster, merging it into the pre-trained model, and initializing a new LoRA booster to progressively refine the ensemble FFM. This process lets FedGBA apply the principle of weak learner (build strong ensemble with weak predictors) to reduce the communication cost while maintaining superior performance compared to other methods.

4.1 Gradient Boosting Perspective of LoRA

Gradient Boosting (GB) [10] is a powerful ensemble learning technique that combines multiple weak learners to create a strong learner. The key idea is to iteratively train a sequence of models, each of which corrects the mistakes of the preceding model. At each iteration, the model is trained to minimize the residual error between the current predictions and the target outputs.

Formally, let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ be a dataset with N samples, where $x_i \in \mathbb{R}^d$ is the input feature vector and $y_i \in \mathbb{R}$ is the corresponding target output. The goal of gradient boosting is to learn a function $\mathcal{M}(x)$ that maps the input features to the target outputs. The function $\mathcal{M}(x)$ is expressed as a sum of T boosters $f^{(t)}(x)$:

$$\mathcal{M}(x) = \sum_{t=1}^{T} f^{(t)}(x). \tag{5}$$

The weak learners $f^{(t)}(x)$ are typically simple models, such as decision trees or linear models that are trained to minimize the residual error. At each iteration t, the residual error r_i^t for the i-th example is computed as:

$$r_i^{(t)} = y_i - \mathcal{M}^{(t-1)}(x_i), \tag{6}$$

where $\mathcal{M}^{t-1}(x)$ is the cumulative model up to the previous iteration. The booster $f^{(t)}(x)$ is then trained to minimize the loss function $\mathcal{L}_t$ defined over the residual errors:

$$\mathcal{L}_t = \sum_{i=1}^{N} \ell\big(f^{(t)}(x_i), r_i^{(t)}\big), \tag{7}$$

where $\ell(\cdot)$ is a differentiable loss function, such as squared or absolute error loss, N is the number of samples, and $\mathcal{M}^{(t-1)}$ is the fixed model from the previous iteration whose parameters are not updated during backpropagation. After training the booster $f_m(x)$, the cumulative model $F_m(x)$ is updated as:

$$\mathcal{M}^{(t)}(x) = \mathcal{M}^{(t-1)}(x) + \eta f^{(t)}(x), \tag{8}$$

where η is a learning rate that controls the contribution of the booster to the final model. The gradient boosting algorithm iteratively repeats this process of computing residual errors, training boosters, and updating the cumulative model for a fixed number of iterations T or until a convergence criterion is met.

Low-Rank Adaptation (LoRA) [18] can be seen as a special case of gradient boosting with $T = 1$, where the pre-trained model $\boldsymbol{W}_0$ serves as the initial model $\mathcal{M}^{(0)}(x)$, and the LoRA update $\Delta\boldsymbol{W}^{(1)} = \boldsymbol{A}^{(1)}\boldsymbol{B}^{(1)}$ serves as a single booster $f^{(1)}(x)$. In this context, the LoRA update is trained to minimize the residual error between the pre-trained model's predictions and the target outputs as:

$$\mathcal{L} = \sum_{i=1}^{N} \ell\big((\boldsymbol{W}^{(0)} + \boldsymbol{A}^{(1)}\boldsymbol{B}^{(1)})x_i, y_i\big). \tag{9}$$

Generally, at the end of each gradient booster iteration t, the ensemble model $\mathcal{M}^{(t)}$ is obtained by adding the LoRA booster to the pre-trained model:

$$\boldsymbol{W}^{(t)} = \boldsymbol{W}^{(t-1)} + \eta \boldsymbol{A}^{(t)}\boldsymbol{B}^{(t)}. \tag{10}$$

After ensembling the booster $\boldsymbol{A}^{(t)}\boldsymbol{B}^{(t)}$ at iteration t, new LoRA booster matrices $\boldsymbol{A}^{(t+1)}$ and $\boldsymbol{B}^{(t+1)}$ are initialized for the next iteration $t+1$. The above scheme highlights the connection between LoRA and gradient boosting and motivates the idea of extending the gradient boosting framework to federated fine-tuning with LoRA.

4.2 Federated Gradient Boosting Adaption (FedGBA)

Let $\mathcal{D}_m$ be the local dataset resided in the m-th client. The goal of FedGBA is to operate in a federated learning setting where multiple clients collaborate to fine-tune a shared LLM $\mathcal{M}^0$ without directly exchanging their private data $\mathcal{D}_m$. The key components of the model include: 1) a pre-trained LLM $\mathcal{M}^{(0)}$ shared across all clients, 2) a central server $\mathcal{S}$ for aggregation and coordination and clients $\{\mathcal{C}_m\}_{m=1}^M$ for local training, 3) local LoRA booster on each client $\mathcal{C}_m$, and 4) a gradient boosting framework for iterative refinement.

Initialization. The server $\mathcal{S}$ distributes the pre-trained Foundational Model (FM) parameters to all M participating clients to initialise the process. A typical FM consists of multiple projection

matrices (*e.g.*, query, key, value, and output matrices) in the self-attention mechanism and weight matrices in the feedforward network. For simplicity, we denote these weight matrices in the FM as $\boldsymbol{W}^{(0)}$. At each GB iteration t, client C_m adapts these matrices $\boldsymbol{W}^{(0)}$ by initializing two low-rank matrices: $\boldsymbol{A}_m^{(t)} \in \mathbb{R}^{d\times r_m}$ and $\boldsymbol{B}_m^{(t)} \in \mathbb{R}^{r_m\times k}$. Here, r_m represents the rank used by client m, d is the input dimension, and k is the output dimension of the adapted weight matrix.

Training LoRA Boosters. Let $\mathcal{D}_m = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ be the local dataset with N samples, where $\boldsymbol{x}_i$ is the input and y_i is the target output. For each weight matrix $\boldsymbol{W}^{(t-1)}$ in layer l of FM, the corresponding LoRA matrices $\boldsymbol{A}^{(t)}$ and $\boldsymbol{B}^{(t)}$ are learned by minimizing the loss function $\mathcal{L}_t$ defined over the current model predictions and the target outputs:

$$\mathcal{L}_t = \sum_{i=1}^{N} \ell\Big(\mathcal{M}^{(t-1)}(\boldsymbol{x}_i) + f^{(t)}(\boldsymbol{x}_i), y_i\Big) + \lambda\Big(\big\|\boldsymbol{A}^{(t)}\big\|_F^2 + \big\|\boldsymbol{B}^{(t)}\big\|_F^2\Big), \tag{11}$$

where $\mathcal{M}^{(t-1)}(\cdot)$ is the fixed model from previous iteration (not involved in backpropagation), $f^{(t)}(\cdot)$ represents the current LoRA booster being trained, $\ell(\cdot)$ is a differentiable loss function (*e.g.*, cross-entropy), λ is a regularization coefficient.

In each GB iteration t, we minimize the target $\mathcal{L}_t$ for κ training steps. Since the parameters $\boldsymbol{W}^{(t-1)}$ in previous model $\mathcal{M}^{(t-1)}$ are frozen (not trainable), the LoRA booster $\left(\Delta\boldsymbol{W}^{(t)} = \boldsymbol{A}^{(t)}\boldsymbol{B}^{(t)}\right)$ focuses on learning the residual correction needed on top of the previous model's predictions.

Secure Global Booster Aggregation. After obtaining residual correction boosters from each client, the server $\mathcal{S}$ needs to aggregate these updates. For building the global booster at iteration t, two strategies can be applied:

- **Federated Stacking:** For each client's residual correction at GB iteration t, we decompose $\boldsymbol{A}_m^{(t)}$ and $\boldsymbol{B}_m^{(t)}$ into rank-1 sub-modules:

$$\boldsymbol{A}_m^{(t)} = [\boldsymbol{a}_{m,1}, \boldsymbol{a}_{m,2}, \ldots, \boldsymbol{a}_{r_m}], \boldsymbol{B}_m^{(t)} = [\boldsymbol{b}_{m,1}, \boldsymbol{b}_{m,2}, \ldots, \boldsymbol{b}_{r_m}]^\top, \tag{12}$$

  where $\boldsymbol{a}_i \in \mathbb{R}^d$ and $\boldsymbol{b}_i^\top \in \mathbb{R}^k$ are column and row vectors respectively. Then we create a pool of all rank-1 sub-modules from all clients:

$$\mathcal{A} = \{p_m \cdot \boldsymbol{a}_{m,i} | \text{ for all } m \text{ and } i\}, \mathcal{B} = \{\boldsymbol{b}_{m,i} | \text{ for all } m \text{ and } i\}, \tag{13}$$

  where $p_m = B_m / \left(\sum_{m=1}^M B_m\right)$ are scaling factors based on client batch sizes. We apply a **`random permutation`** to $\mathcal{A}$ and $\mathcal{B}$ separately and **`stack`** the permuted sub-modules to form the global correction matrices $\bar{\boldsymbol{A}}^{(t)}$ and $\bar{\boldsymbol{B}}^{(t)}$:

$$\bar{\boldsymbol{A}}^{(t)} = \left[a'_1, a'_2, \ldots, a'_{r_m\times M}\right] \bar{\boldsymbol{B}}^{(t)} = \left[b'_1, b'_2, \ldots, b'_{r_m\times M}\right]^\top, \tag{14}$$

  where a'_i and b'_i are the permuted sub-modules, $N = \|\mathcal{A}\| = \sum_{m=1}^M r_m$ is the total number of sub-modules.
- **FedAvg:** Alternatively, we can perform **`averaging`** of the residual corrections:

$$\bar{\boldsymbol{A}}^{(t)} = \sum_{m=1}^{M} p_m \boldsymbol{A}_m^{(t)} \quad \text{and} \quad \bar{\boldsymbol{B}}^{(t)} = \sum_{m=1}^{M} \boldsymbol{B}_m^{(t)}. \tag{15}$$

Merging Global Booster with the Base Model. The $\mathcal{S}$ broadcasts the global LoRA matrices $\bar{\boldsymbol{A}}_l^{(t)}$ and $\bar{\boldsymbol{B}}_l^{(t)}$ to all clients and updates the ensemble model $\mathcal{M}^{(t)}$ by merging the LoRA booster into the corresponding weight matrix $\boldsymbol{W}^{(t-1)}$ from the previous iteration in $\mathcal{M}^{(t-1)}$:

$$\boldsymbol{W}^{(t)} \leftarrow \boldsymbol{W}^{(t-1)} + \eta_t \, \bar{\boldsymbol{A}}^{(t)} \bar{\boldsymbol{B}}^{(t)}, \tag{16}$$

where η_t is a learning rate that controls the contribution of LoRA matrices. By absorbing the learning rate α_t into the LoRA matrices, we simplify the notation and make it clear that the effective update to the weight matrices is directly determined by the learned LoRAs. Thus in our case $\eta_t = 1$.

4.3 Efficient Aggregation via Sequential Updates

To address the challenges of aggregating LoRA updates in federated learning, `FedGBA` adopts a sequential update strategy using rank-1 matrices. This approach serves two critical purposes: minimizing interference between updates from different clients during aggregation, and reducing communication overhead.

Minimal Cross-Client Interference. As shown in the preliminaries section, aggregating higher-rank LoRA updates leads to mathematical inaccuracies since $\frac{1}{M}\sum_{m=1}^M \boldsymbol{A}_m\boldsymbol{B}_m \neq \bar{\boldsymbol{A}}\bar{\boldsymbol{B}}$. By using rank-1 updates, we minimize these cross-term effects as each update captures a single direction of adaptation, making the aggregation process more accurate and stable.

Communication-Efficient Updates. Rank-1 updates also provide significant communication benefits, requiring only $d+k$ parameters to be transmitted instead of $r(d+k)$ parameters in standard LoRA with rank r. This reduction in update size is particularly beneficial in federated settings where communication bandwidth is often the primary bottleneck.

`FedGBA` **offers several advantages** over other federated fine-tuning approaches:
1) **Clean Aggregation**: Rank-1 updates minimize the interference between client contributions during aggregation, leading to more accurate and stable model updates.
2) **Reduced Communication**: The rank-1 structure significantly reduces communication overhead between clients and the server, making it particularly suitable for bandwidth-constrained federated learning scenarios.

Complexity Analysis of `FedGBA`. Table 1 compares the cost of `FedGBA` and other methods in terms of local training cost, communication cost, and total cost. The local training cost α represents the computational cost for training on a single client per iteration, while the communication cost γ represents the cost of transmitting updates between clients and server. The total cost includes these components plus a fixed initialization cost β.

`FedGBA` achieves substantial reductions in both computation and communication costs through its rank-1 update strategy. Compared to methods using fixed rank R, `FedGBA` reduces the local training cost by a factor of R and the communication overhead by the same factor. This is reflected in the total cost, where `FedGBA` requires only $M\alpha T/R$ cost compared to $M\alpha T$ for standard approaches.

Table 1: Comparison of computational and communication costs. By definition, LoRA uses rank $r \triangleq R$ and FedGBA uses $r \ll R$. Here, α: comp. cost for LoRA ($r \triangleq R$) adapter; β: comp. cost for base model; M: total clients; γ: comm. cost per rank-1 matrix; T: boosting iterations.

Method	Agg. Strategy	Local Cost	Total Cost
LoRA ($r \triangleq R$)	**Centralized**	α	$M\alpha T + \beta$
FedIT ($r = R$)	**Avg.**	α	$M\alpha T + \beta + \gamma RM$
FLoRA ($r = R$)	**Stacking**	α	$M\alpha T + \beta + \gamma RM^2$
FedGBA ($r = 1$)	**Avg.**	α/R	$M\alpha T/R + \beta + \gamma M$
FedGBA ($r = 1$)	**Stacking**	α/R	$M\alpha T/R + \beta + \gamma M^2$

5 Theoretical Analysis of FedGBA

Notation and Assumptions. We consider a federated learning setting with M clients, where each client m has a local dataset $\mathcal{D}_m$. Let $\boldsymbol{W}^{(t)}$ denote the global model at communication round t. For each client m, we define local LoRA update matrices $\boldsymbol{A}_m^{(t)}$ and $\boldsymbol{B}_m^{(t)}$ at round t, and $\mathcal{E}_T = \mathbb{E}\left[\left\|\nabla_{\mathbf{W}^{(T)}}\mathcal{L}(\mathbf{W}^{(T)}) - \sum_{t=1}^{T}\bar{\mathbf{A}}^{(t)}(\bar{\mathbf{B}}^{(t)})^T\right\|_F\right]$. We make the following assumptions: 1) The global loss function $\mathcal{L}(\boldsymbol{W})$ is β-smooth and μ-strongly convex. 2) The local loss functions $\mathcal{L}_m(\boldsymbol{W})$ have L-Lipschitz continuous gradients. 3) The gradients of local loss functions are bounded: $\|\nabla\mathcal{L}_m(\boldsymbol{W})\|_F \le G$ for all m and $\boldsymbol{W}$.

Lemma 5.1 (FedGBA Gradient Approximation - Stacking). *For the stacking strategy in* FedGBA*, after T rounds of updates with K clients selected out of M total clients in each round, the update approximates the full gradient update with error:*

$$\mathcal{E}_T = O\left(\frac{1}{\sqrt{Kr_0}} + \sqrt{\frac{M-K}{KM}} + \frac{1}{\sqrt{T}}\right), \tag{17}$$

where r_0 is the LoRA rank per client. K is the number of selected clients per round. M is the total number of clients. T is the number of update rounds.

Lemma 5.2 (FedGBA Gradient Approximation - Averaging). *For the averaging strategy in* FedGBA*, after T rounds of updates with K clients selected out of M total clients in each round, the update approximates the full gradient update with error:*

$$\mathcal{E}_T = O\left(\frac{1}{\sqrt{r_0}} + \sqrt{\frac{M-K}{KM}} + \frac{1}{\sqrt{T}} + \frac{1}{\sqrt{K}}\right), \tag{18}$$

where r_0 is the LoRA rank per client, K is the number of selected clients per round, M is the total number of clients, and T is the number of update rounds.

Lemma 5.3 (Accumulated Update Bound). *For the* FedGBA *update process, the accumulated LoRA updates satisfy:*

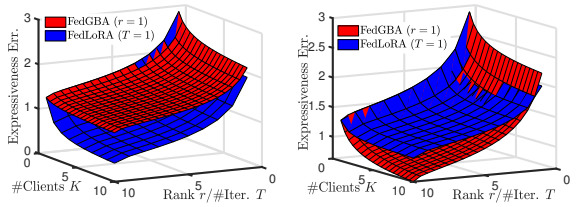
$$\|\bar{\mathbf{A}}^{(T)}\|_F, \|\bar{\mathbf{B}}^{(T)}\|_F = \begin{cases} O(T\sqrt{K}), & \text{for Stacking} \\ O(T), & \text{for Averaging} \end{cases} \tag{19}$$

where T is the number of update rounds and K is the number of selected clients per round.

Theorem 5.4 (FedGBA Convergence - Stacking). *Under the* FedGBA *update process with stacking aggregation, T update rounds, K selected clients per round out of M total clients, and rank-r_0 LoRA updates per client, we have:*

$$\mathbb{E}[\mathcal{L}(\mathbf{W}^{(T)})] - \mathcal{L}^* = O\left(\frac{1}{\sqrt{T}} + \frac{1}{r_0} + \sqrt{\frac{M-K}{KM}}\right), \tag{20}$$

where $\mathcal{L}^*$ is the optimal loss.



(a) Averaging **(b) Stacking**

Figure 2: A visualization of error bound in Th. 5.6 w.r.t. to the # clients K and LoRA rank r / GB iteration T.

Theorem 5.5 (FedGBA Convergence - Averaging). *Under the* FedGBA *update process with averaging aggregation, T update rounds, K selected clients per round out of M total clients, and rank-r_0 LoRA updates per client, we have:*

$$\mathbb{E}[\mathcal{L}(\mathbf{W}^{(T)})] - \mathcal{L}^* = O\left(\frac{1}{\sqrt{T}} + \frac{1}{r_0} + \sqrt{\frac{M-K}{KM}} + \frac{1}{\sqrt{K}}\right). \tag{21}$$

Theorem 5.6 (FedGBA Expressiveness). *Let f^* be any function in the original function class, and f_T be the function represented by the* FedGBA*-updated network after T update rounds. Then:*

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[(f_T(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2] = \begin{cases} O\left(\frac{1}{\sqrt{Kr_0}} + \frac{1}{\sqrt{T}} + \sqrt{\frac{M-K}{KM}}\right), & \text{(Stacking)} \\ O\left(\frac{1}{r_0} + \frac{1}{\sqrt{KT}} + \sqrt{\frac{M-K}{KM}}\right), & \text{(Averaging)} \end{cases} \tag{22}$$

where $\mathcal{D}$ is the data distribution, r_0 is the rank per client, K is the number of selected clients per round, M is the total number of clients, and T is the number of update rounds.

These theoretical results provide key insights into FedGBA's behavior in federated settings. The gradient approximation lemmas (Lemmas 5.1 & 5.2) reveal how aggregation error scales with key parameters, showing that stacking achieves better stability by avoiding the $\frac{1}{\sqrt{K}}$ term that appears in averaging due to cross-client interference. Lemma 5.3 bounds the growth of accumulated updates, demonstrating how stacking and averaging differ in their dependence on the number of clients. The convergence theorems (Theorems 5.4 & 5.5) characterize the optimization behavior for both aggregation strategies. Both achieve $O(\frac{1}{\sqrt{T}})$ convergence rates and $O(\frac{1}{r_0})$ rank dependency, with additional terms reflecting their distinct aggregation mechanisms. Notably, the averaging strategy incurs an extra $\frac{1}{\sqrt{K}}$ term due to client averaging effects, while stacking avoids this penalty. Theorem 5.6 and Figure 2 extend these insights to expressiveness, showing that stacking achieves $O(\frac{1}{\sqrt{Kr_0}})$ approximation error through effective client collaboration, while averaging requires more iterations $(O(\frac{1}{\sqrt{KT}}))$ to compensate for its aggregation instability. These results support three key design choices in FedGBA: 1) the effectiveness of rank-1 updates when

Table 2: MMLU and MT-bench comparison of LLaMA 7B, Tiny-LLaMA 1B on four fine tuning datasets. Results of baseline methods (*) are extracted from [40]. "Avg." means using averaging aggregation, and "Stacking" denotes the stacking aggregation.

Foundation model	*Agg. Strategy*	*Fine-tuning algorithm*	*MMLU*			*MT-bench*	
			Dolly	*Alpaca*	*Wizard*	*Wizard*	*ShareGPT*
Tiny-LLaMA	**Centralized**	*LoRA**	*27.99*	*28.03*	*29.13*	*2.34*	*2.79*
	Avg.	*FedIT**	*16.35*	*30.02*	*42.51*	*2.92*	*2.55*
	Stacking	*FLoRA**	*30.80*	*31.92*	*43.87*	*3.13*	*2.77*
	Avg.	FedGBA	*29.50*	*31.52*	*42.97*	*3.08*	*2.79*
	Stacking	FedGBA	**32.30**	**32.74**	**44.47**	**3.75**	**3.25**
LLaMA	**Centralized**	*LoRA**	**35.91**	*29.18*	*31.68*	*4.38*	*3.99*
	Avg.	*FedIT**	*29.67*	*29.41*	*33.43*	*3.07*	*3.73*
	Stacking	*FLoRA**	*30.99*	*29.85*	*34.26*	*4.21*	*3.93*
	Avg.	FedGBA	*30.50*	*30.02*	*33.97*	*3.73*	*3.97*
	Stacking	FedGBA	*32.84*	**31.96**	**35.27**	**4.91**	**4.77**

Table 3: Fine-tuning results with ViT Base and Large models on different image classification datasets. We report the accuracy (%) after 10 epochs. Avg. represents the average accuracy of each method on all datasets. The best performance is shown in bold.

Model	*Method*	*Agg. Strategy*	*Oxf. Pets*	*Sfd. Cars*	*Cifar-10*	*DTD*	*EuroSAT*	*FGVC*	*RESISC45*	*Cifar-100*	*Avg.*
ViT-Base	*LoRA*	**Centralized**	*93.19*$_{\pm 0.36}$	*45.38*$_{\pm 0.41}$	*98.78*$_{\pm 0.05}$	*74.95*$_{\pm 0.40}$	*98.44*$_{\pm 0.15}$	*25.16*$_{\pm 0.16}$	*92.70*$_{\pm 0.18}$	*92.02*$_{\pm 0.12}$	*77.58*
	FedIT	**Avg.**	*92.15*$_{\pm 0.53}$	*45.06*$_{\pm 0.28}$	*98.41*$_{\pm 0.04}$	*73.77*$_{\pm 0.37}$	*98.27*$_{\pm 0.14}$	*24.84*$_{\pm 0.33}$	*92.52*$_{\pm 0.10}$	*91.88*$_{\pm 0.14}$	*77.11*
	FLoRA	**Stacking**	*93.21*$_{\pm 0.40}$	*46.82*$_{\pm 0.85}$	***98.92***$_{\pm 0.13}$	*74.98*$_{\pm 0.73}$	***98.95***$_{\pm 0.19}$	*26.84*$_{\pm 0.43}$	*93.10*$_{\pm 0.17}$	*92.41*$_{\pm 0.15}$	*78.15*
	FedGBA	**Avg.**	*93.14*$_{\pm 0.26}$	*46.31*$_{\pm 0.24}$	*98.55*$_{\pm 0.13}$	*74.39*$_{\pm 0.35}$	*98.38*$_{\pm 0.14}$	*27.51*$_{\pm 0.84}$	*92.97*$_{\pm 0.41}$	*92.21*$_{\pm 0.16}$	*77.93*
	FedGBA	**Stacking**	***93.65***$_{\pm 0.34}$	***47.02***$_{\pm 0.45}$	*98.79*$_{\pm 0.18}$	***75.10***$_{\pm 0.64}$	*98.88*$_{\pm 0.15}$	***28.04***$_{\pm 0.42}$	***93.76***$_{\pm 0.24}$	***93.05***$_{\pm 0.17}$	**78.53**
ViT-Large	*LoRA*	**Centralized**	*94.82*$_{\pm 0.09}$	*73.25*$_{\pm 0.36}$	*99.13*$_{\pm 0.03}$	*81.79*$_{\pm 0.45}$	*98.63*$_{\pm 0.07}$	*42.32*$_{\pm 0.98}$	*94.71*$_{\pm 0.25}$	*94.87*$_{\pm 0.10}$	*84.94*
	FedIT	**Avg.**	*93.21*$_{\pm 0.34}$	*72.91*$_{\pm 0.37}$	*98.79*$_{\pm 0.02}$	*81.73*$_{\pm 0.26}$	*97.44*$_{\pm 0.08}$	*40.62*$_{\pm 0.24}$	*92.23*$_{\pm 0.21}$	*94.28*$_{\pm 0.21}$	*83.90*
	FLoRA	**Stacking**	*94.47*$_{\pm 0.44}$	*73.10*$_{\pm 0.25}$	***99.25***$_{\pm 0.03}$	*82.23*$_{\pm 0.41}$	*99.14*$_{\pm 0.10}$	*43.45*$_{\pm 0.93}$	*94.81*$_{\pm 0.07}$	*95.58*$_{\pm 0.17}$	*85.25*
	FedGBA	**Avg.**	*94.21*$_{\pm 0.21}$	*73.01*$_{\pm 0.32}$	*98.92*$_{\pm 0.07}$	*81.87*$_{\pm 0.31}$	*98.04*$_{\pm 0.18}$	*40.92*$_{\pm 0.18}$	*93.01*$_{\pm 0.41}$	*94.72*$_{\pm 0.23}$	*84.33*
	FedGBA	**Stacking**	***95.14***$_{\pm 0.04}$	***74.14***$_{\pm 0.67}$	*99.12*$_{\pm 0.05}$	***82.81***$_{\pm 0.54}$	***99.16***$_{\pm 0.08}$	***44.28***$_{\pm 0.67}$	***95.45***$_{\pm 0.27}$	***96.28***$_{\pm 0.21}$	**85.79**

combined with sufficient client participation and iterations, 2) the advantage of stacking over averaging for more stable aggregation, and 3) the importance of proper client selection in FL settings.

6 Experiment Evaluation

Baslines. In our experimental setup, we compare FedGBA against three key baselines to evaluate its performance comprehensively. First, we include FedIT, the state-of-the-art federated fine-tuning method proposed by Zhang et al. [44], which integrates LoRA with FedAvg and uses averaging as its aggregation strategy. Second, we consider FLoRA [40], an extension of FedIT that employs a stacking aggregation strategy instead of averaging, allowing us to assess the impact of different aggregation methods in federated settings. Lastly, we include Centralized LoRA, the standard LoRA method trained in a non-federated, centralized setting, which serves as a benchmark for comparing federated approaches against traditional centralized training. This diverse set of baselines enables us to evaluate FedGBA's performance against both federated and centralized approaches, as well as to understand the effectiveness of its unique features in the context of existing methods.

Settings.we configured FedGBA with a default rank of $r = 1$, while setting other FedLoRA's rank to $r = 16$. To ensure comparability, we aligned FedGBA's initial fine-tuning parameters with LoRA's configuration, including weight initialization and learning rate, as outlined by Hu et al. [18]. We maintained consistency in the number of iterations (communication rounds) $T = 20$ across FedGBA and all baseline models. Our study encompassed two categories of foundation model tasks: Large Language Models (LLMs) and Vision Transformers (ViTs), each evaluated using task-specific benchmarks. For implementing and assessing baselines, we utilized official GitHub repositories and libraries (specifically, Wang et al. [40] for LLMs and Gao et al. [12] for ViTs). We simulated identically distributed (*non-i.i.d.*) characteristics typical of federated learning as in [44] and uniformly select $M = 10$ clients in the FL training.

6.1 Instruction Fine-tuning with LLMs

Models & Datasets. In our experiments, we employ three Llama-based models of varying scales: TinyLLaMA (1.1 billion parameters) [45], Llama (7 billion parameters) [37], allowing us to evaluate FedGBA across different model capacities. Following the original LoRA paper [17], we apply LoRA modules to the self-attention layers only. For our tasks, we utilize several datasets: Databricks-dolly-15k [29], Alpaca [36], and Wizard [26] for question-answering (QA), and Wizard [26] and ShareGPT for the chat assistant task. The Dolly dataset, generated by Databricks employees, contains 15k diverse text samples [45]. Alpaca provides 52K instruction-following samples for fine-tuning LLMs [36]. The Wizard dataset, used to train WizardLM, consists of 70k instruction-output pairs featuring more complex instructions [26]. Lastly, ShareGPT is a collection of approximately 52,000 conversations scraped via the ShareGPT API, which we split into question-answering pairs for our experiments. We evaluate the federated fine-tuned models on MMLU [14] for the QA task and MT-bench [49] for the chat assistant task, respectively.

Table 2 compares the performance of different fine-tuning algorithms (Centralized LoRA, FedIT, FLoRA, and the proposed FedGBA) on two Language Model benchmarks (MMLU and MT-bench) using two foundation models (Tiny-LLaMA 1B and LLaMA 7B) across four fine-tuning datasets (Dolly, Alpaca, Wizard, and ShareGPT). The results consistently show that FedGBA, particularly with stacking aggregation, outperforms other methods across all datasets and benchmarks for both models, with the performance gain more pronounced on the larger LLaMA model. Notably, FedGBA with stacking achieves significant improvements over centralized LoRA, demonstrating its effectiveness in federated settings. For instance, on the LLaMA model with the Wizard dataset, FedGBA (stacking) achieves a score of 35.27 on MMLU, compared to 31.68 for centralized LoRA, showcasing a substantial improvement of 3.59 points. The table also highlights the impact of different aggregation strategies. Across all experiments, stacking consistently outperforms averaging for both FLoRA and FedGBA. For example, on the LLaMA model with the Dolly dataset, FedGBA with stacking achieves an MMLU score of 32.84, while averaging yields 30.50.

6.2 Image Classification Fine-tuning with ViTs

Models & Datasets. We conduct the evaluation of our method on the image classification task. We employ the Base and Large versions of the popular CV foundation model, Vision Transformer (ViT) [9]. The ViTs are pretrained on the ImageNet-21K dataset [33]. The datasets for fine-tuning include OxfordPets (37^2), CIFAR10 (10), DTD (47), EuroSAT (10) and RESISC45 (45) with small label spaces, as well as StanfordCars (196), FGVC (100) and CIFAR100 (100) with large label spaces.

Table 3 presents the results of fine-tuning Vision Transformer (ViT) models on various image classification datasets using different methods, comparing LoRA (centralized), FedIT, FLoRA, and FedGBA (with both averaging and stacking aggregation) on ViT-Base and ViT-Large models. The results consistently demonstrate FedGBA's superior performance, particularly with stacking aggregation, across most datasets for both model sizes. FedGBA often outperforms centralized LoRA, showcasing its effectiveness in federated settings without compromising performance. For instance, on the FGVC dataset with ViT-Large, FedGBA with stacking achieves 44.28% accuracy, significantly outperforming centralized LoRA (42.32%) and other federated methods. The method's success is evident across diverse datasets, highlighting its versatility and adaptability to different tasks and domains. Notably, FedGBA with stacking achieves the highest average accuracy across all datasets (78.53% for ViT-Base and 85.79% for ViT-Large), underscoring its overall effectiveness. The performance gap between FedGBA and other methods tends to widen on more challenging datasets (*e.g.*, Stanford Cars, FGVC), suggesting that FedGBA is particularly effective for complex tasks. Furthermore, the consistent improvement from ViT-Base to ViT-Large models indicates FedGBA's scalability to larger model architectures.

6.3 Understanding the Weak Learner Principle

LoRA Rank. Table 4 shows the role of rank r in FedGBA's weak learners. FedGBA with smaller ranks outperforms LoRA ($r = 16$) and FedGBA with larger ranks ($r = 32$), corroborating our discussion on weak learner complexity. The strong performance of FedGBA with low-rank adaptations suggests that combining multiple simple weak learners can effectively capture complex patterns and improve generalization. This highlights the ensemble effect in FedGBA, leading to strong performance while maintaining parameter efficiency.

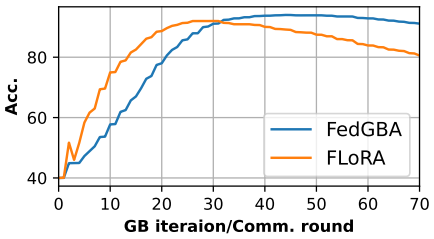


Figure 3: Comparison of FedGBA and FLoRA performance across gradient boosting iterations T in federated learning.

Table 4: Performance comparison of FedGBA of ViT-Base with different rank values (r) on image classification. Stacking aggregation is applied in FedGBA.

Method	Rank	Oxf. Pets	DTD	RESISC45	Cifar-100
LoRA*	**16**	93.19±0.36	74.95±0.40	92.70±0.18	92.02±0.12
FedGBA	**32**	92.21±0.45	74.05±0.30	92.10±0.34	91.72±0.32
FedGBA	**16**	93.01±0.40	74.98±0.73	93.10±0.17	92.31±0.31
FedGBA	**8**	93.33±0.24	74.82±0.23	93.56±0.41	92.81±0.16
FedGBA	**4**	93.43±0.21	74.89±0.31	93.52±0.21	92.89±0.12
FedGBA	**1**	**93.65**±0.34	**75.10**±0.64	**93.76**±0.24	**93.05**±0.17

The GB iteration. Figure 3 compares the performance of FedGBA and FLoRA across multiple gradient boosting (GB) iterations or communication rounds in a federated learning setting. The graph reveals several key advantages of FedGBA that align with claims made in the paper. Notably, FedGBA's performance remains stable over many iterations, while FLoRA's declines after its peak. This stability is a crucial illustration of FedGBA's robustness to overfitting, which can be directly attributed to its gradient boosting mechanism and the principle of weak learners. By employing multiple simple (weak) learners in an ensemble, FedGBA is able to continually refine its model without risk of overfitting for data since the individual booster is too simple to overfit, it is very hard to combine them in a way that the strong ensemble would overfit to training data.

7 Conclusions

We have proposed FedGBA, a novel framework for federated fine-tuning of foundation models that streamlines the aggregation of low-rank adaptations through sequential rank-1 updates. Our theoretical analysis establishes how this sequential approach minimizes cross-client interference in aggregation while maintaining strong convergence guarantees. Extensive experiments on both language and vision tasks demonstrate that FedGBA not only reduces computational and communication overhead compared to existing federated methods but also achieves better accuracy across various datasets and model sizes.

References

[1] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Maitha Alhammadi, Mazzotta Daniele, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The Falcon Series of Language Models: Towards Open Frontier Models. (2023).

[2] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*. PMLR, 2397–2430.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *ArXiv* abs/2005.14165 (2020).

[5] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. https://lmsys.org/blog/2023-03-30-vicuna/

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*. 4171–4186.

[7] Shizhe Diao, Ruijia Xu, Hongjin Su, Yilei Jiang, Yan Song, and Tong Zhang. 2021. Taming pre-trained language models with n-gram representations for low-resource domain adaptation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 3336–3349.

[8] Shizhe Diao, Tianyang Xu, Ruijia Xu, Jiawei Wang, and Tong Zhang. 2023. Mixture-of-Domain-Adapters: Decoupling and Injecting Domain Knowledge to Pre-trained Language Models Memories. *arXiv preprint arXiv:2306.05406* (2023).

[9] Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[10] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[11] Jerome H Friedman. 2002. Stochastic gradient boosting. *Computational statistics & data analysis* 38, 4 (2002), 367–378.

[12] Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. 2024. Parameter-Efficient Fine-Tuning with Discrete Fourier Transform. *arXiv preprint arXiv:2405.03003* (2024).

[13] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, et al. 2020. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518* (2020).

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

[15] Samuel Horváth, Chen-Yu Ho, Ludovit Horvath, Atal Narayan Sahu, Marco Canini, and Peter Richtárik. 2022. Natural compression for distributed deep learning. In *Mathematical and Scientific Machine Learning*. PMLR, 129–141.

[16] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*. PMLR, 2790–2799.

[17] Edward Hu, Yelong Shen, Phil Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[18] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=nZeVKeeFYf9

[19] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088* (2024).

[20] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and trends® in machine learning* 14, 1–2 (2021), 1–210.

[21] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, Vol. 1. 2.

[22] Vladimir Koltchinskii and Dmitry Panchenko. 2002. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics* 30, 1 (2002), 1–50.

[23] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems* 2 (2020), 429–450.

[24] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. 2020. Federated learning in mobile edge networks: A comprehensive survey. *IEEE communications surveys & tutorials* 22, 3 (2020), 2031–2063.

[25] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. *Advances in neural information processing systems* 33 (2020), 2351–2363.

[26] Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583* (2023).

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

[28] H Brendan McMahan, FX Yu, P Richtarik, AT Suresh, D Bacon, et al. 2016. Federated learning: Strategies for improving communication efficiency. In *Proceedings of the 29th Conference on Neural Information Processing Systems (NIPS), Barcelona, Spain*. 5–10.

[29] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. AdapterFusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247* (2020).

[30] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research* 21, 140 (2020), 1–67.

[32] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21, 1, Article 140 (jan 2020), 67 pages.

[33] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. 2021. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972* (2021).

[34] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100* (2022).

[35] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. *arXiv preprint arXiv:1906.02243* (2019).

[36] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

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

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

[39] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. 2020. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440* (2020).

[40] Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. 2024. FLoRA: Federated Fine-Tuning Large Language Models with Heterogeneous Low-Rank Adaptations. *arXiv preprint arXiv:2409.05976* (2024).

[41] Adam X Yang, Maxime Robeyns, Xi Wang, and Laurence Aitchison. 2023. Bayesian low-rank adaptation for large language models. *arXiv preprint arXiv:2308.13111* (2023).

[42] Liping Yi, Han Yu, Gang Wang, and Xiaoguang Liu. 2023. Fedlora: Model-heterogeneous personalized federated learning with lora tuning. *arXiv preprint arXiv:2310.13283* (2023).

[43] Yuchen Zeng and Kangwook Lee. 2024. The Expressive Power of Low-Rank Adaptation. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=likXVjmh3E

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

[45] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. TinyLlama: An Open-Source Small Language Model. arXiv:2401.02385 [cs.CL]

[46] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).

[47] Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2020. Revisiting few-sample BERT fine-tuning. *arXiv preprint arXiv:2006.05987* (2020).

[48] Tong Zhang and Bin Yu. 2005. Boosting with early stopping: Convergence and consistency. *The Annals of Statistics* 33, 4 (2005), 1538.

[49] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. arXiv:2306.05685 [cs.CL]

A Appendix

A.1 Proof of Lemma 5.1

Proof. We proceed in steps:

1) First, let's consider a single round t. The stacking strategy concatenates the LoRA updates from all selected clients:

$$\begin{aligned}\bar{\mathbf{A}}^{(t)} &= [\mathbf{A}_1^{(t)}, \ldots, \mathbf{A}_K^{(t)}] \\ \bar{\mathbf{B}}^{(t)} &= [\mathbf{B}_1^{(t)}, \ldots, \mathbf{B}_K^{(t)}]\end{aligned} \tag{23}$$

2) The true gradient of the global loss can be written as:

$$\nabla_{\mathbf{W}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}) = \frac{1}{M} \sum_{m=1}^{M} \nabla_{\mathbf{W}^{(t)}} \mathcal{L}_m(\mathbf{W}^{(t)}) \tag{24}$$

3) The LoRA update for a single client m approximates its local gradient. Due to the L-Lipschitz continuity of the local gradients, we can bound the approximation error:

$$\|\mathbf{A}_m^{(t)} (\mathbf{B}_m^{(t)})^T - \nabla_{\mathbf{W}^{(t)}} \mathcal{L}_m(\mathbf{W}^{(t)})\|_F \leq \frac{L}{\sqrt{r_0}} \tag{25}$$

4) Let's define the approximation error for a single client:

$$\epsilon_m^{(t)} = \nabla_{\mathbf{W}^{(t)}} \mathcal{L}_m(\mathbf{W}^{(t)}) - \mathbf{A}_m^{(t)} (\mathbf{B}_m^{(t)})^T \tag{26}$$

5) Now, we can express the difference between the true gradient and the FedGBA approximation:

$$\begin{aligned}&\nabla_{\mathbf{W}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}) - \bar{\mathbf{A}}^{(t)} (\bar{\mathbf{B}}^{(t)})^T \\ &= \frac{1}{M} \sum_{m=1}^{M} \nabla_{\mathbf{W}^{(t)}} \mathcal{L}_m(\mathbf{W}^{(t)}) - \frac{1}{K} \sum_{m \in \mathcal{S}_t} \mathbf{A}_m^{(t)} (\mathbf{B}_m^{(t)})^T \\ &= \frac{1}{M} \sum_{m=1}^{M} \epsilon_m^{(t)} + \left(\frac{1}{M} \sum_{m=1}^{M} - \frac{1}{K} \sum_{m \in \mathcal{S}_t} \right) \mathbf{A}_m^{(t)} (\mathbf{B}_m^{(t)})^T\end{aligned} \tag{27}$$

6) Taking the expectation and using the triangle inequality:

$$\begin{aligned}&\mathbb{E}\left[\left\| \nabla_{\mathbf{W}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}) - \bar{\mathbf{A}}^{(t)} (\bar{\mathbf{B}}^{(t)})^T \right\|_F \right] \\ &\leq \mathbb{E}\left[\left\| \frac{1}{M} \sum_{m=1}^{M} \epsilon_m^{(t)} \right\|_F \right] + \mathbb{E}\left[\left\| \left(\frac{1}{M} \sum_{m=1}^{M} - \frac{1}{K} \sum_{m \in \mathcal{S}_t} \right) \mathbf{A}_m^{(t)} (\mathbf{B}_m^{(t)})^T \right\|_F \right]\end{aligned} \tag{28}$$

7) For the first term, we can use the L–Lipschitz continuity bound:

$$\mathbb{E}\left[\left\| \frac{1}{M} \sum_{m=1}^{M} \epsilon_m^{(t)} \right\|_F \right] \leq \frac{L}{\sqrt{M r_0}} \tag{29}$$

8) For the second term, we can use standard results from sampling theory and the gradient bound G:

$$\mathbb{E}\left[\left\| \left(\frac{1}{M} \sum_{m=1}^{M} - \frac{1}{K} \sum_{m \in \mathcal{S}_t} \right) \mathbf{A}_m^{(t)} (\mathbf{B}_m^{(t)})^T \right\|_F \right] \leq G \sqrt{\frac{M-K}{KM}} \tag{30}$$

9) Combining these results for a single round:

$$\mathbb{E}\left[\left\| \nabla_{\mathbf{W}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}) - \bar{\mathbf{A}}^{(t)} (\bar{\mathbf{B}}^{(t)})^T \right\|_F \right] \leq \frac{L}{\sqrt{K r_0}} + G \sqrt{\frac{M-K}{KM}} \tag{31}$$

10) Finally, considering T rounds and using the linearity of expectation and Jensen's inequality:

$$\mathcal{E}_T \leq \frac{L}{\sqrt{K r_0}} + G \sqrt{\frac{M-K}{KM}} + \frac{\beta}{\sqrt{T}} \tag{32}$$

where the $\frac{\beta}{\sqrt{T}}$ term comes from the β-smoothness of the global loss function.

This completes the proof for the stacking strategy. □

A.2 Proof of Lemma 5.2

Proof. We follow a similar approach as in the stacking proof, with key differences:

1) For the averaging strategy, we have:

$$\bar{\mathbf{A}}^{(t)} = \frac{1}{K} \sum_{m \in \mathcal{S}_t} \mathbf{A}_m^{(t)}, \bar{\mathbf{B}}^{(t)} = \frac{1}{K} \sum_{m \in \mathcal{S}_t} \mathbf{B}_m^{(t)} \tag{33}$$

2) The difference between the true gradient and the FedGBA approximation now becomes:

$$\begin{aligned}&\nabla_{\mathbf{W}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}) - \bar{\mathbf{A}}^{(t)} (\bar{\mathbf{B}}^{(t)})^T \\ &= \frac{1}{M} \sum_{m=1}^{M} \nabla_{\mathbf{W}^{(t)}} \mathcal{L}_m(\mathbf{W}^{(t)}) - \frac{1}{K^2} \sum_{m,n \in \mathcal{S}_t} \mathbf{A}_m^{(t)} (\mathbf{B}_n^{(t)})^T\end{aligned} \tag{34}$$

3) We can decompose this further:

$$\begin{aligned}&= \frac{1}{M} \sum_{m=1}^{M} \epsilon_m^{(t)} + \left(\frac{1}{M} \sum_{m=1}^{M} - \frac{1}{K} \sum_{m \in \mathcal{S}_t} \right) \mathbf{A}_m^{(t)} (\mathbf{B}_m^{(t)})^T \\ &+ \frac{1}{K^2} \sum_{m \neq n \in \mathcal{S}_t} (\mathbf{A}_m^{(t)} (\mathbf{B}_m^{(t)})^T - \mathbf{A}_m^{(t)} (\mathbf{B}_n^{(t)})^T)\end{aligned} \tag{35}$$

4) Taking the expectation and using the triangle inequality:

$$\begin{aligned}&\mathbb{E}\left[\left\| \nabla_{\mathbf{W}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}) - \bar{\mathbf{A}}^{(t)} (\bar{\mathbf{B}}^{(t)})^T \right\|_F \right] \\ &\leq \mathbb{E}\left[\left\| \frac{1}{M} \sum_{m=1}^{M} \epsilon_m^{(t)} \right\|_F \right] + \mathbb{E}\left[\left\| \left(\frac{1}{M} \sum_{m=1}^{M} - \frac{1}{K} \sum_{m \in \mathcal{S}_t} \right) \mathbf{A}_m^{(t)} (\mathbf{B}_m^{(t)})^T \right\|_F \right] \\ &+ \mathbb{E}\left[\left\| \frac{1}{K^2} \sum_{m \neq n \in \mathcal{S}_t} (\mathbf{A}_m^{(t)} (\mathbf{B}_m^{(t)})^T - \mathbf{A}_m^{(t)} (\mathbf{B}_n^{(t)})^T) \right\|_F \right]\end{aligned} \tag{36}$$

5) Using the L-Lipschitz continuity and gradient bound:

$$\begin{aligned}&\mathbb{E}\left[\left\| \frac{1}{M} \sum_{m=1}^{M} \epsilon_m^{(t)} \right\|_F \right] \leq \frac{L}{\sqrt{r_0}} \mathbb{E}\left[\left\| \left(\frac{1}{M} \sum_{m=1}^{M} - \frac{1}{K} \sum_{m \in \mathcal{S}_t} \right) \mathbf{A}_m^{(t)} (\mathbf{B}_m^{(t)})^T \right\|_F \right] \\ &\leq G \sqrt{\frac{M-K}{KM}}\end{aligned} \tag{37}$$

6) For the third term, we can use the fact that $\mathbf{A}_m^{(t)}$ and $\mathbf{B}_n^{(t)}$ are independent for $m \neq n$:

$$\mathbb{E}\left[\left\| \frac{1}{K^2} \sum_{m \neq n \in \mathcal{S}_t} (\mathbf{A}_m^{(t)} (\mathbf{B}_m^{(t)})^T - \mathbf{A}_m^{(t)} (\mathbf{B}_n^{(t)})^T) \right\|_F \right] \leq \frac{G}{\sqrt{K}} \tag{38}$$

7) Combining these results for a single round:

$$\mathbb{E}\left[\left\|\nabla_{\mathbf{W}^{(t)}}\mathcal{L}(\mathbf{W}^{(t)}) - \bar{\mathbf{A}}^{(t)}(\bar{\mathbf{B}}^{(t)})^T\right\|_F\right] \leq \frac{L}{\sqrt{r_0}} + G\sqrt{\frac{M-K}{KM}} + \frac{G}{\sqrt{K}} \tag{39}$$

8) Finally, considering T rounds and using the linearity of expectation and Jensen's inequality:

$$\begin{aligned}\mathcal{E}_T &\leq \frac{1}{\sqrt{T}}\sqrt{\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla_{\mathbf{W}^{(t)}}\mathcal{L}(\mathbf{W}^{(t)}) - \bar{\mathbf{A}}^{(t)}(\bar{\mathbf{B}}^{(t)})^T\right\|_F^2\right]} \\ &\leq \frac{L}{\sqrt{r_0}} + G\sqrt{\frac{M-K}{KM}} + \frac{G}{\sqrt{K}} + \frac{\beta}{\sqrt{T}}\end{aligned} \tag{40}$$

where the $\frac{\beta}{\sqrt{T}}$ term comes from the β-smoothness of the global loss function.

This completes the proof for the averaging strategy. □

A.3 Proof of Lemma 5.3

Proof. We'll prove this for both stacking and averaging strategies:

1) Stacking Strategy: In each round t, $\bar{\mathbf{A}}^{(t)} = [\mathbf{A}_1^{(t)}, \ldots, \mathbf{A}_K^{(t)}]$

$$\|\bar{\mathbf{A}}^{(t)}\|_F^2 = \sum_{k=1}^{K}\|\mathbf{A}_k^{(t)}\|_F^2 \leq KG^2, \tag{41}$$

since $\|\mathbf{A}_k^{(t)}\|_F \leq G$ (from gradient bound). After T rounds:

$$\|\bar{\mathbf{A}}^{(T)}\|_F \leq \sum_{t=1}^{T}\|\bar{\mathbf{A}}^{(t)}\|_F \leq T\sqrt{KG^2} = O(T\sqrt{K}G) \tag{42}$$

2) Averaging Strategy: In each round t, $\bar{\mathbf{A}}^{(t)} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{A}_k^{(t)}$

$$\|\bar{\mathbf{A}}^{(t)}\|_F \leq \frac{1}{K}\sum_{k=1}^{K}\|\mathbf{A}_k^{(t)}\|_F \leq G \tag{43}$$

After T rounds:

$$\|\bar{\mathbf{A}}^{(T)}\|_F \leq \sum_{t=1}^{T}\|\bar{\mathbf{A}}^{(t)}\|_F \leq TG \tag{44}$$

The same bounds apply to $\bar{\mathbf{B}}^{(T)}$ by symmetry. □

A.4 Proof of Theorem 5.4

Proof. 1) By β-smoothness of the loss:

$$\mathcal{L}(\mathbf{W}^{(t+1)}) \leq \mathcal{L}(\mathbf{W}^{(t)}) + \langle\nabla\mathcal{L}(\mathbf{W}^{(t)}), \Delta\mathbf{W}^{(t)}\rangle + \frac{\beta}{2}\|\Delta\mathbf{W}^{(t)}\|_F^2 \tag{45}$$

2) For stacking aggregation:

$$\Delta\mathbf{W}^{(t)} = -\eta\sum_{m\in\mathcal{S}_t}\mathbf{A}_m^{(t)}(\mathbf{B}_m^{(t)})^T \tag{46}$$

3) Taking expectation:

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\mathbf{W}^{(t+1)})] \leq\ &\mathbb{E}[\mathcal{L}(\mathbf{W}^{(t)})] \\ &- \eta\mathbb{E}[\langle\nabla\mathcal{L}(\mathbf{W}^{(t)}), \sum_{m\in\mathcal{S}_t}\mathbf{A}_m^{(t)}(\mathbf{B}_m^{(t)})^T\rangle] \\ &+ \frac{\beta\eta^2}{2}\mathbb{E}[\|\sum_{m\in\mathcal{S}_t}\mathbf{A}_m^{(t)}(\mathbf{B}_m^{(t)})^T\|_F^2]\end{aligned} \tag{47}$$

4) Using Lemma 5.3 to bound the last term:

$$\mathbb{E}[\|\sum_{m\in\mathcal{S}_t}\mathbf{A}_m^{(t)}(\mathbf{B}_m^{(t)})^T\|_F^2] \leq G^2 \tag{48}$$

5) For the middle term, using Lemma 5.1:

$$\begin{aligned}&\mathbb{E}[\langle\nabla\mathcal{L}(\mathbf{W}^{(t)}), \sum_{m\in\mathcal{S}_t}\mathbf{A}_m^{(t)}(\mathbf{B}_m^{(t)})^T\rangle] \\ \geq&\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|_F^2 - (\frac{C_1}{r_0} + \sqrt{\frac{M-K}{KM}}G)\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|_F\end{aligned} \tag{49}$$

6) By strong convexity:

$$\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|_F^2 \geq 2\mu(\mathcal{L}(\mathbf{W}^{(t)}) - \mathcal{L}^*) \tag{50}$$

7) Combining these:

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\mathbf{W}^{(t+1)}) - \mathcal{L}^*] \leq\ &(1-2\mu\eta)(\mathcal{L}(\mathbf{W}^{(t)}) - \mathcal{L}^*) \\ &+ \eta(\frac{C_1}{r_0} + \sqrt{\frac{M-K}{KM}}G)\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|_F \\ &+ \frac{\beta\eta^2G^2}{2}\end{aligned} \tag{51}$$

8) Applying recursively for T iterations and using $(1-2\mu\eta)^T \leq e^{-2\mu\eta T}$:

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\mathbf{W}^{(T)}) - \mathcal{L}^*] \leq\ &e^{-2\mu\eta T}(\mathcal{L}(\mathbf{W}^{(0)}) - \mathcal{L}^*) \\ &+ (\frac{C_1}{r_0} + \sqrt{\frac{M-K}{KM}}G)\sum_{t=0}^{T-1}e^{-2\mu\eta(T-t-1)}\eta\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|_F \\ &+ \frac{\beta\eta^2G^2T}{2}\end{aligned} \tag{52}$$

9) Choosing $\eta = \frac{1}{\beta T}$ gives the final result:

$$\mathbb{E}[\mathcal{L}(\mathbf{W}^{(T)}) - \mathcal{L}^*] = O\left(\frac{1}{\sqrt{T}} + \frac{1}{r_0} + \sqrt{\frac{M-K}{KM}}\right) \tag{53}$$

□

A.5 Proof of Theorem 5.5

Proof. 1) Starting with the same β-smoothness condition:

$$\mathcal{L}(\mathbf{W}^{(t+1)}) \leq \mathcal{L}(\mathbf{W}^{(t)}) + \langle\nabla\mathcal{L}(\mathbf{W}^{(t)}), \Delta\mathbf{W}^{(t)}\rangle + \frac{\beta}{2}\|\Delta\mathbf{W}^{(t)}\|_F^2 \tag{54}$$

2) For averaging aggregation:

$$\Delta\mathbf{W}^{(t)} = -\eta\frac{1}{K}\sum_{m\in\mathcal{S}_t}\mathbf{A}_m^{(t)}(\mathbf{B}_m^{(t)})^T \tag{55}$$

3) Taking expectation:

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\mathbf{W}^{(t+1)})] \leq\ &\mathbb{E}[\mathcal{L}(\mathbf{W}^{(t)})] \\ &- \eta\mathbb{E}[\langle\nabla\mathcal{L}(\mathbf{W}^{(t)}), \frac{1}{K}\sum_{m\in\mathcal{S}_t}\mathbf{A}_m^{(t)}(\mathbf{B}_m^{(t)})^T\rangle] \\ &+ \frac{\beta\eta^2}{2}\mathbb{E}[\|\frac{1}{K}\sum_{m\in\mathcal{S}_t}\mathbf{A}_m^{(t)}(\mathbf{B}_m^{(t)})^T\|_F^2]\end{aligned} \tag{56}$$

4) Using Lemma 5.3 for averaging case:

$$\mathbb{E}[\|\frac{1}{K}\sum_{m\in\mathcal{S}_t}\mathbf{A}_m^{(t)}(\mathbf{B}_m^{(t)})^T\|_F^2] \le \frac{G^2}{K} \tag{57}$$

5) For the gradient term, using Lemma 5.2:

$$\mathbb{E}[\langle\nabla\mathcal{L}(\mathbf{W}^{(t)}), \frac{1}{K}\sum_{m\in\mathcal{S}_t}\mathbf{A}_m^{(t)}(\mathbf{B}_m^{(t)})^T\rangle]$$

$$\ge \|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|_F^2 - (\frac{C_1}{r_0} + \sqrt{\frac{M-K}{KM}}G + \frac{C_2}{\sqrt{K}})\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|_F \tag{58}$$

6) Using strong convexity as before:

$$\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|_F^2 \ge 2\mu(\mathcal{L}(\mathbf{W}^{(t)}) - \mathcal{L}^*) \tag{59}$$

7) Combining these:

$$\mathbb{E}[\mathcal{L}(\mathbf{W}^{(t+1)}) - \mathcal{L}^*] \le (1-2\mu\eta)(\mathcal{L}(\mathbf{W}^{(t)}) - \mathcal{L}^*) + \eta(\frac{C_1}{r_0} + \sqrt{\frac{M-K}{KM}}G + \frac{C_2}{\sqrt{K}})\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|_F + \frac{\beta\eta^2G^2}{2K} \tag{60}$$

8) Applying recursively for T iterations:

$$\mathbb{E}[\mathcal{L}(\mathbf{W}^{(T)}) - \mathcal{L}^*] \le e^{-2\mu\eta T}(\mathcal{L}(\mathbf{W}^{(0)}) - \mathcal{L}^*) + (\frac{C_1}{r_0} + \sqrt{\frac{M-K}{KM}}G + \frac{C_2}{\sqrt{K}})\sum_{t=0}^{T-1}e^{-2\mu\eta(T-t-1)}\eta\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|_F + \frac{\beta\eta^2G^2T}{2K} \tag{61}$$

9) Choosing $\eta = \frac{1}{\beta T}$ gives:

$$\mathbb{E}[\mathcal{L}(\mathbf{W}^{(T)}) - \mathcal{L}^*] = O\left(\frac{1}{\sqrt{T}} + \frac{1}{r_0} + \sqrt{\frac{M-K}{KM}} + \frac{1}{\sqrt{K}}\right) \tag{62}$$

□

A.6 Proof of Theorem 5.6

Proof. 1) Let $\mathbf{W}^*$ be the weights that exactly represent f^*. The approximation error can be decomposed as:

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[(f_T(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2] \le \|\mathbf{W}^* - \mathbf{W}^{(T)}\|_F^2 = \|\mathbf{W}^* - \mathbf{W}^{(0)} - \sum_{t=1}^{T}\Delta\mathbf{W}^{(t)}\|_F^2 \tag{63}$$

2) For stacking strategy:

$$\Delta\mathbf{W}^{(t)} = \sum_{m\in\mathcal{S}_t}\mathbf{A}_m^{(t)}(\mathbf{B}_m^{(t)})^T \tag{64}$$

3) Using Lemma 5.3, each update is bounded:

$$\|\Delta\mathbf{W}^{(t)}\|_F \le \frac{G}{\sqrt{K}r_0} + G\sqrt{\frac{M-K}{KM}} \tag{65}$$

4) Over T iterations, using Jensen's inequality:

$$\|\sum_{t=1}^{T}\Delta\mathbf{W}^{(t)}\|_F \le \frac{G}{\sqrt{T}}\sqrt{\sum_{t=1}^{T}\|\Delta\mathbf{W}^{(t)}\|_F^2} \tag{66}$$

5) This leads to the stacking bound:

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[(f_T(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2] = O\left(\frac{1}{\sqrt{K}r_0} + \frac{1}{\sqrt{T}} + \sqrt{\frac{M-K}{KM}}\right) \tag{67}$$

6) For averaging strategy:

$$\Delta\mathbf{W}^{(t)} = \frac{1}{K}\sum_{m\in\mathcal{S}_t}\mathbf{A}_m^{(t)}(\mathbf{B}_m^{(t)})^T \tag{68}$$

7) The averaging operation introduces additional variance:

$$\|\Delta\mathbf{W}^{(t)}\|_F \le \frac{G}{r_0} + G\sqrt{\frac{M-K}{KM}} + \frac{G}{\sqrt{KT}} \tag{69}$$

8) Leading to the averaging bound:

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[(f_T(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2] = O\left(\frac{1}{r_0} + \frac{1}{\sqrt{KT}} + \sqrt{\frac{M-K}{KM}}\right) \tag{70}$$

□

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009