DECA: DECENTRALIZING BLOCK-WISE ADAM FOR EFFICIENT LLM FULL-PARAMETER FINE-TUNING ON NON-IID DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning *large language models* (LLMs) in privacy-sensitive and resource-constrained environments remains challenging. Since training data are often distributed across multiple clients, decentralized fine-tuning offers a natural paradigm for collaborative adaptation without a central server. However, enabling *full-parameter fine-tuning* (FPFT) in this decentralized setting is difficult: FPFT provides strong adaptation capacity but incurs prohibitive resource consumption for billion-scale models. Existing decentralized LLM fine-tuning methods therefore mainly rely on parameter-efficient updates, which improve efficiency but may restrict downstream performance. Moreover, client data are typically non-IID, making decentralized optimization more vulnerable to client drift and unstable convergence. To address these challenges, we propose DECA, a resource-efficient decentralized FPFT framework for LLMs on non-IID data. DECA partitions model parameters into disjoint blocks and performs sequential block-wise Adam optimization, reducing resource consumption while preserving decentralized full-parameter adaptation. To stabilize training, DECA further introduces first- and second-order block-wise moment estimates with fresh local gradient statistics and consensus-derived discrepancy signals. We provide rigorous theoretical analysis and extensive experiments, showing that DECA achieves fast convergence, strong downstream performance, and significant resource efficiency. Our code is available at `https://anonymous.4open.science/r/DECA-3F73`.

1 INTRODUCTION

Large language models (LLMs), such as LLaMA (Dubey et al., 2024), Qwen (Yang et al., 2025), and GPT (Achiam et al., 2023), have demonstrated remarkable capabilities in language understanding, reasoning, and generation. These models, typically built on the Transformer architecture (Vaswani et al., 2017), serve as versatile foundations for a wide array of tasks. To effectively deploy these foundation models for domain-specific applications, fine-tuning remains an essential step for aligning model behavior with specialized downstream tasks.

Most state-of-the-art studies related to LLM fine-tuning, e.g., *parameter-efficient fine-tuning* (PEFT) methods (Houlsby et al., 2019; Lester et al., 2021; Hu et al., 2022) and *full-parameter fine-tuning* (FPFT) methods (Kingma & Ba, 2015; Lu & Xiao, 2015; Malladi et al., 2023; Zhao et al., 2024; Luo et al., 2024), often assume that LLMs are tuned in a centralized manner, i.e., data are stored in a central infrastructure with sufficient resources. However, in real-world scenarios, data are distributed across different devices often in a non-IID fashion, while collecting the large volume of data from the devices to a single infrastructure may result in significant communication overhead and risk of privacy leakage. To this end, *federated learning* (FL) (McMahan et al., 2017) is applied, such that each client device calculates a local model update according to its private data, and a global model is trained by aggregating these local updates at a central server. Although FL has been integrated with both PEFT (Guo et al., 2025; Zhao et al., 2023) and FPFT methods (Qin et al., 2024; Shu et al., 2025), they generally trade performance for reducing the intensive demand on communication resources.

Alternatively, the *decentralized* paradigm offers a purely distributed solution that further eliminates the need for the central server, as each client only interacts with its direct neighbors. Though many

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

state-of-the-art studies have been performed to investigate the application of the decentralized learning paradigm in training small-scale *deep neural network* (DNN) models (Lin et al., 2021; Esfandiari et al., 2021; de Vos et al., 2023; Ghiasvand et al., 2025b; Wang et al., 2025; 2026), only very few proposals exist to decentralize LLM fine-tuning, e.g., decentralized LoRA (Ghiasvand et al., 2025a; Saadati et al., 2026). Unfortunately, since PEFT techniques tune a subset of model parameters to reduce resource consumption, they potentially limit performance on downstream tasks (Zhang et al., 2024a; Luo et al., 2024; Pu et al., 2023).

In this paper, we propose **DECA**, a lightweight decentralized framework for FPFT of LLMs on non-IID data. DECA partitions model parameters into disjoint blocks and decentralizes block-wise Adam optimization, enabling clients to collaboratively fine-tune the blocks in a sequential manner with significantly reduced resource consumption. Furthermore, DECA introduces first- and second-order *block-wise moment approximations* (BMAs), which extend Adam-style moment estimation to decentralized block-wise optimization by incorporating fresh local gradient statistics and consensus-derived discrepancy signals. This allows each client to form a more globally consistent update direction within each active block, thereby mitigating client drift and stabilizing optimization under non-IID data. Our theoretical analysis shows that DECA converges under non-IID data while matching the asymptotic convergence behavior of state-of-the-art decentralized optimization methods. Our experiments on classification and generation tasks with models ranging from 1.5B to 8B parameters demonstrate that DECA consistently outperforms decentralized baselines and enables practical FPFT on resource-constrained devices. Our main contributions are summarized as follows:

- We propose DECA, a decentralized FPFT framework enabling block-wise, BMA-enhanced Adam optimization for collaborative fine-tuning of LLMs on non-IID data.
- We provide a rigorous theoretical analysis establishing state-of-the-art convergence rates under non-IID data, together with substantial improvements in memory, computation, and communication efficiency.
- We conduct extensive experiments showing that DECA consistently outperforms existing baselines across diverse downstream tasks and model scales.

The remainder of this paper is organized as follows. Sec. 2 introduces the system model. Sec. 3 presents the DECA algorithm, followed by the theoretical analysis in Sec. 4. Sec. 5 reports our experiment results, and Sec. 6 reviews related work. Finally, Sec. 7 concludes the paper.

2 System Model and Preliminaries

Consider a decentralized learning system represented by an undirected graph with weights $\mathcal{G} = \{\mathcal{N}, \mathbf{W}\}$, where $\mathcal{N} = \{1, 2, \cdots, N\}$ denotes a set of N clients, and $\mathbf{W} \in [0,1]^{N\times N}$ denotes the adjacency matrix of graph G. Let $\mathcal{D}_i$ represent the local data distribution of client $i \in \mathcal{N}$, and $D_i = |\mathcal{D}_i|$ denote the number of data samples in $\mathcal{D}_i$. It is worth noting that the data are non-IID across different clients. The communication topology is characterized by $\mathbf{W}$. We assume $\mathbf{W}$ is a symmetric doubly stochastic matrix (Lian et al., 2017; Esfandiari et al., 2021; Yu et al., 2019; Zhang et al., 2022a), such that $\sum_{i=1}^{N} w_{i,j} = \sum_{j=1}^{N} w_{i,j} = 1$ and $w_{i,j} = w_{j,i}$ for $\forall i, j \in \mathcal{N}$, where $w_{i,j} \in [0,1]$ denotes the element in the i-th row and j-th column of the matrix $\mathbf{W}$. We use $w_{i,j}$ to encode how much client i can affect client j, while $w_{i,j} = 0$ implies clients i and j are disconnected. Let $\mathcal{N}_i = \{j \in \mathcal{N} \mid w_{i,j} > 0\}$ denote the set of neighbors for client i, with cardinality $N_i = |\mathcal{N}_i|$. The objective can be formulated as follows:

$$\min_{x\in\mathbb{R}^d} \mathcal{F}(x) = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\mathbb{E}_{\zeta_i\sim\mathcal{D}_i}\left[F_i(x;\zeta_i)\right]}_{\triangleq f_i(x)} \tag{1}$$

where $x \in \mathbb{R}^d$ represents the parameters of the target global model, $f_i(x) \triangleq \mathbb{E}_{\zeta_i\sim\mathcal{D}_i}[F_i(x;\zeta_i)]$ denotes the local loss function of client i, and $\zeta_i \sim \mathcal{D}_i$ represents the randomly sampled data from $\mathcal{D}_i$. Without loss of generality, we assume $F_i(\cdot;\cdot) = F(\cdot;\cdot)$, $\forall i \in \mathcal{N}$. Unfortunately, enabling FPFT in a decentralized setting is rather challenging.

- **Communication Overhead:** In traditional decentralized optimization, each client i has to exchange its full local model $x_i \in \mathbb{R}^d$ with its neighbors. Since LLMs typically contain

billions of parameters, exchanging full model updates incurs substantial communication overhead, i.e., bit-width $\times\, d \times (N_i - 1)$, which exceeds the bandwidth capacity of clients.

- **Memory Constraints:** FPFT mandates the storage of full-parameter local updates. Furthermore, the aggregation step necessitates accommodating extremely high-dimensional parameters received from neighbors, a requirement that frequently exceeds the physical GPU memory capacity of individual clients.
- **Statistical Divergence:** The non-IID nature of $\mathcal{D}_i$ creates a mismatch between each local objective $f_i(x)$ and the global objective $\mathcal{F}(x)$. As a result, local updates may bias client models, causing client drift. Without centralized aggregation, such drift can outpace the consensus process induced by the communication topology $\mathbf{W}$, increasing the risk of unstable training or divergence.

3 Our Algorithm

3.1 Overview

We propose **DECA**, a novel decentralized optimization framework that integrates *block-wise Adam optimization* with first- and second-order *block-wise moment approximations* (BMAs). Focusing on the resource-intensive Transformer core (while treating embedding and LM head layers as static), we partition the trainable parameter space into B disjoint blocks, denoted as $x = \{x_1, x_2, \ldots, x_B\}$. Let $d_k = |x_k|$ denote the number of parameters in the k-th block. The algorithm operates over T global rounds. In each round t, we sequentially optimize the blocks from $k = 1$ to B. Specifically, for the k-th block, clients perform R steps of collaborative decentralized optimization, while holding the other blocks fixed. Formally, the sub-problem for optimizing block k is defined as:

$$\min_{x_k \in \mathbb{R}^{d_k}} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\zeta \sim \mathcal{D}_i} \left[F_i(x_1, \ldots, x_k, \ldots, x_B; \zeta) \right] \tag{2}$$

Standard *block coordinate descent* (BCD)-based methods typically use simple gradient descent or basic Adam updates (Bengio et al., 2006; Wright, 2015; Belilovsky et al., 2019; Luo et al., 2024). However, directly applying block-wise Adam to decentralized non-IID settings is challenging, as local adaptive updates may amplify client drift, while intermittently updated blocks suffer from stale moment estimates. To address this issue, BMA constructs hybrid first- and second-order estimates by combining fresh local gradient statistics with a consensus-derived discrepancy signal, thereby correcting local updates toward network-wide agreement and stabilizing decentralized block-wise optimization.

3.2 Algorithm Design

The pseudo-code of our algorithm is given in **Algorithm** 1. The optimization of each block k during round t follows an iterative process managed through an inner loop of R steps. At the beginning of the k-th block update in round t, each client i initializes its BMAs: $m_{i,k}^{[t,0]} = 0$ and $v_{i,k}^{[t,0]} = 0$. At each step $r \in \{0, 1, \ldots, R-1\}$, client i calculates its local gradient based on its local data $\mathcal{D}_i$:

$$g_{i,k}^{[t,r]} = \nabla_k F_i(x_{i,k}^{[t,r]}, \zeta), \quad \text{where } \zeta \sim \mathcal{D}_i. \tag{3}$$

Subsequently, client i updates its first and second BMAs using the hyperparameters α_1 and α_2:

$$m_{i,k}^{[t,r+\frac{1}{2}]} = \alpha_1 m_{i,k}^{[t,r]} + (1-\alpha_1) g_{i,k}^{[t,r]}, \text{ and } v_{i,k}^{[t,r+\frac{1}{2}]} = \alpha_2 v_{i,k}^{[t,r]} + (1-\alpha_2) g_{i,k}^{[t,r]} \odot g_{i,k}^{[t,r]} \tag{4}$$

Following this, we perform bias correction to obtain the estimates $\widehat{m}_{i,k}^{[t,r]}$ and $\widehat{v}_{i,k}^{[t,r]}$:

$$\widehat{m}_{i,k}^{[t,r]} = m_{i,k}^{[t,r+\frac{1}{2}]} / (1 - \alpha_1^{r+1}), \quad \widehat{v}_{i,k}^{[t,r]} = v_{i,k}^{[t,r+\frac{1}{2}]} / (1 - \alpha_2^{r+1}). \tag{5}$$

The local model parameters are then updated as follows:

$$x_{i,k}^{[t,r+\frac{1}{2}]} = x_{i,k}^{[t,r]} - \gamma \cdot \widehat{m}_{i,k}^{[t,r]} \Big/ \left(\sqrt{\widehat{v}_{i,k}^{[t,r]}} + \epsilon \right). \tag{6}$$

Algorithm 1 DECA (for each client $i \in \mathcal{N}$)

Input: Graph $\mathcal{G} = \{\mathcal{N}, \mathbf{W}\}$, number of communication rounds T, number of local updates R, granularity of model partitioning B, learning rate γ, and hyperparameters α_1, α_2, β_1 and β_2.
Output: Final global model $x^{[T]} = \frac{1}{N}\sum_{i=1}^{N} x_i^T$

Randomly initialize local model $x_i^{[0]} = \{x_{i,k}^{[0]}\}_{k=1}^{B}$
for $t = 0$ **to** $T-1$ **do**
  for $k = 1$ **to** B **do**
    $m_{i,k}^{[t,0]} = 0$, $v_{i,k}^{[t,0]} = 0$
    for $r = 0$ **to** $R-1$ **do**
      $g_{i,k}^{[t,r]} = \nabla_k F_i(x_{i,k}^{[t,r]}, \zeta)$ where $\zeta \sim \mathcal{D}_i$;
      $m_{i,k}^{[t,r+\frac{1}{2}]} = \alpha_1 m_{i,k}^{[t,r]} + (1-\alpha_1) g_{i,k}^{[t,r]}$ and $v_{i,k}^{[t,r+\frac{1}{2}]} = \alpha_2 v_{i,k}^{[t,r]} + (1-\alpha_2) g_{i,k}^{[t,r]} \odot g_{i,k}^{[t,r]}$;
      $\widehat{m}_{i,k}^{[t,r]} = m_{i,k}^{[t,r+\frac{1}{2}]}/(1-\alpha_1^{r+1})$ and $\widehat{v}_{i,k}^{[t,r]} = v_{i,k}^{[t,r+\frac{1}{2}]}/(1-\alpha_2^{r+1})$;
      $x_{i,k}^{[t,r+\frac{1}{2}]} = x_{i,k}^{[t,r]} - \gamma \frac{\widehat{m}_{i,k}^{[t,r]}}{\sqrt{\widehat{v}_{i,k}^{[t,r]}} + \epsilon}$;
      Exchange $x_{i,k}^{[t,r+\frac{1}{2}]}$ with its neighbors;
      $x_{i,k}^{[t,r+1]} = \sum_{j \in \mathcal{N}_i} w_{i,j} x_{j,k}^{[t,r+\frac{1}{2}]}$;
      $h_{i,k}^{[t,r]} = \frac{x_{i,k}^{[t,r]} - x_{i,k}^{[t,r+1]}}{\left\| x_{i,k}^{[t,r]} - x_{i,k}^{[t,r+1]} \right\|}$;
      $m_{i,k}^{[t,r+1]} = \beta_1 m_{i,k}^{[t,r+\frac{1}{2}]} + (1-\beta_1) h_{i,k}^{[t,r]}$ and $v_{i,k}^{[t,r+1]} = \beta_2 v_{i,k}^{[t,r+\frac{1}{2}]} + (1-\beta_2) h_{i,k}^{[t,r]} \odot h_{i,k}^{[t,r]}$;
    end for
    $x_{i,k}^{[t+1,0]} = x_{i,k}^{[t,R]}$;
  end for
end for

where ϵ is a small constant for numerical stability when the second moment estimate $\widehat{v}_{i,k}^{[t,r]}$ vanishes.

Client i then shares its parameters with its neighbors and performs weighted aggregation:

$$x_{i,k}^{[t,r+1]} = \sum_{j \in \mathcal{N}_i} w_{i,j} x_{j,k}^{[t,r+\frac{1}{2}]}. \tag{7}$$

To mitigate the challenges of data heterogeneity, we introduce a block-wise consensus-derived discrepancy signal to correct local optimization directions and reduce client drift. Specifically, we calculate the normalized difference between the local model and the aggregated model:

$$h_{i,k}^{t,r} = \left(x_{i,k}^{[t,r]} - x_{i,k}^{[t,r+1]} \right) \Big/ \left\| x_{i,k}^{[t,r]} - x_{i,k}^{[t,r+1]} \right\| \tag{8}$$

which is used to update both the first- and second-order BMAs for the subsequent step using the hyperparameters β_1 and β_2:

$$m_{i,k}^{[t,r+1]} = \beta_1 m_{i,k}^{[t,r+\frac{1}{2}]} + (1-\beta_1) h_{i,k}^{t,r} = \alpha_1 \beta_1 m_{i,k}^{[t,r]} + \beta_1 (1-\alpha_1) g_{i,k}^{[t,r]} + (1-\beta_1) h_{i,k}^{[t,r]} \tag{9}$$

and

$$\begin{aligned} v_{i,k}^{[t,r+1]} =& \beta_2 v_{i,k}^{[t,r+\frac{1}{2}]} + (1-\beta_2) h_{i,k}^{t,r} \odot h_{i,k}^{t,r} \\ =& \alpha_2 \beta_2 v_{i,k}^{[t,r]} + \beta_2 (1-\alpha_2)\, g_{i,k}^{[t,r]} \odot g_{i,k}^{[t,r]} + (1-\beta_2)\, h_{i,k}^{[t,r]} \odot h_{i,k}^{[t,r]}. \end{aligned} \tag{10}$$

Unlike standard Adam optimization (Kingma & Ba, 2015) or its decentralized momentum variants (Lin et al., 2021), which continuously accumulates moment estimates from local stochastic gradients, our design updates the BMAs using the intermediate moments $m_{i,k}^{[t,r+\frac{1}{2}]}$ and $v_{i,k}^{[t,r+\frac{1}{2}]}$. This design is critical for block-wise optimization, where moment estimates associated with inactive parameter blocks can become stale due to intermittent updates. Specifically, for each active block, we first incorporate the fresh gradient to reconstruct objective-aligned first- and second-order moment estimates, and then inject the consensus-derived local discrepancy $h_{i,k}^{[t,r]}$ within the same parameter

subspace. As a result, the update direction remains primarily aligned with the local optimization objective while being adaptively corrected toward network-wide consensus, thereby mitigating client drift and stabilizing block-wise optimization under non-IID data.

4 ANALYSIS

Due to space limitations, this section focuses on the theoretical convergence analysis of DECA. Additional analysis of communication complexity, memory consumption, and computational complexity is provided in Appendix B. Before presenting our main results, we first give the requisite assumptions which have been widely adopted in theoretical analysis of decentralized optimization (Yu et al., 2019; Esfandiari et al., 2021; Lin et al., 2021). Specifically, **Assumption** 4.1 imposes Lipschitz smoothness on the objective functions. **Assumption** 4.2 bounds the stochastic gradient variance and the magnitude of the block-wise gradients. **Assumption** 4.3 characterizes the spectral properties of the communication topology. **Assumption** 4.4 implies that switching from block k to block $k+1$ may increase the objective value, but such increase is bounded, and the block transitions cannot arbitrarily destabilize the overall optimization.

Assumption 4.1 (**Lipschitz smoothness**). Each local objective function $f_i(\cdot)$ is L-smooth for any $i \in \mathcal{N}$. Specifically, there exist a constant $L \geq 0$ such that

$$f_i(x) \leq f_i(y) + \nabla f_i^\top(y)(x-y) + \frac{L}{2}\|x-y\|^2,\ \forall x, y \in \mathbb{R}^d \tag{11}$$

and the overall function $f(\cdot)$ is also L-smooth such that:

$$f(x) \leq f(y) + \nabla f^\top(y)(x-y) + \frac{L}{2}\|x-y\|^2,\ \forall x, y \in \mathbb{R}^d \tag{12}$$

Assumption 4.2 (**Bounded Variance and Gradient**). There exist non-negative constants ξ and δ such that the variance of the stochastic gradients is bounded:

$$\mathbb{E}_{\zeta\sim\mathcal{D}_i}\left[\,\|\nabla_k F_i(x;\zeta) - \nabla_k f_i(x)\|^2\,\right] \leq \xi^2, \forall i \in \mathcal{N}, x \in \mathbb{R}^d, \forall k \in \{1,\ldots,B\}. \tag{13}$$

and the block-wise stochastic gradient is bounded:

$$\mathbb{E}_{\zeta\sim\mathcal{D}_i}\left[\|\nabla_k F_i(x,\zeta)\|^2\right] \leq \delta^2, \forall i \in \mathcal{N}, x \in \mathbb{R}^d, \forall k \in \{1,\ldots,B\}. \tag{14}$$

Assumption 4.3 (**Doubly Stochastic Matrix**). The adjacent matrix $\mathbf{W}$ associated with the communication graphs $\mathcal{G}$ is a doubly stochastic matrix. Its spectral gap satisfies:

$$\lambda_1(\mathbf{W}) = 1 \text{ and } \max\left\{|\lambda_2(\mathbf{W})|, |\lambda_N(\mathbf{W})|\right\} \leq \sqrt{\rho} \tag{15}$$

where $\lambda_i(\mathbf{W})$ is the i-th largest eigenvalue of $\mathbf{W}$ and $\rho \in [0,1)$ is a constant measuring the connectivity of the graph.

We define a virtual sequence $\hat{x}_k^{[t,r]}$ which has SGD style but scaled by adaptive step sizes:

$$\hat{x}_k^{[t,r]} = \begin{cases} \bar{x}_{k-1}^{[t,R]} & \text{if } r = 0, \\ \hat{x}_k^{[t,r-1]} - \eta \bar{g}_k^{[t,r-1]}/(1-\alpha_1^{r+1}) & \text{if } r \geq 1. \end{cases} \tag{16}$$

where $\bar{g}_k^{[t,r]} = \frac{1}{N}\sum_{i=1}^N \nabla_k F_i(x_{i,k}^{[t,r]})$ and $\bar{x}_k^{[t,r]}$ represents the average of $x_{i,k}^{[t,r]}$ over all clients. We also define an error sequence $e_k^{[t,r+1]} = \hat{x}_k^{[t,r+1]} - \bar{x}_k^{[t,r+1]}$.

Assumption 4.4 (**Error-controlled Block Transition**). For any block k and round t, there exists constant $C_{\text{err}} > 0$ that

$$\mathbb{E}\big[\mathcal{F}(\hat{x}_{k+1}^{[t,0]})\big] - \mathbb{E}\big[\mathcal{F}(\hat{x}_k^{[t,R]})\big] \leq C_{\text{err}}\,\mathbb{E}\big[\|e_k^{[t,R]}\|^2\big], \tag{17}$$

It is implied that the increase of the objective value along the virtual sequence, from $\hat{x}_k^{[t,R]}$ to $\hat{x}_{k+1}^{[t,0]}$, is controlled by the squared norm of error $e_k^{[t,R]}$.

Theorem 4.5 establishes the convergence guarantee of DECA under the above assumption. The average squared gradient norm is bounded by the initial optimality gap, together with additional error terms induced by stochastic gradient variance, decentralized consensus, and block-wise transitions.

Theorem 4.5. *Recall $x_i^{[0]}$ and $x_{i,k}^{[t,r]}$ denote the initial local model parameters of client i, and the parameters of block k at client i during inner step r of round t, respectively. Let $\bar{x}_k^{[t,r]} \triangleq \frac{1}{N}\sum_{i=1}^N x_{i,k}^{[t,r]}$ and $\bar{x}^{[0]} = \frac{1}{N}\sum_{i=1}^N x_i^{[0]}$. Let $\mathcal{F}^\star$ represent the optimal value of the global objective function. Define $0 < \lambda < \frac{1-\alpha_1^2\beta_1^2}{\alpha_1^2\beta_1^2}$, $C_0 = \frac{(1+\rho)\rho\left((1+\frac{1}{\lambda})(16\delta^2+8\rho)+4+2\delta^2\right)}{(1-\rho)(1-\alpha_1)^2\epsilon^2\left(1-(1+\lambda)\beta_1^2\alpha_1^2\right)}$, and $C_1 = \frac{\alpha_1^2+\delta^2(1-\alpha_1)^2}{(1-\alpha_1)^2\epsilon^2}$. Under **Assumptions** 4.1–4.4, when $0 < \gamma \leq \min\left\{\frac{(1-\beta_1)}{3L\sqrt{TBR}}, \frac{2(1-\beta_1)}{L+6RC_{err}+\sqrt{(L+6RC_{err})^2+36L^2R^2}}\right\}$, we have*

$$\frac{1}{TBR}\sum_{t=0}^{T-1}\sum_{r=0}^{R-1}\sum_{k=1}^{B}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}\left(\bar{x}_k^{[t,r]}\right)\right\|^2\right] \leq \frac{8(1-\beta_1^R)}{TBR\gamma}\left(\mathbb{E}\left[\mathcal{F}(\bar{x}^0)-\mathcal{F}^\star\right]\right) + 8C_1C_{err}(1-\beta_1^R)\gamma + \frac{4(1-\beta_1^R)L\gamma\xi^2}{N(1-\beta_1)^2} + \frac{4L^2\gamma^2(4C_0+3RC_1)}{(1-\rho)(1-\beta_1)} \tag{18}$$

In the following corollary, we characterize the convergence rate of DECA more explicitly.

Corollary 4.6. *Let learning rate $\gamma = \mathcal{O}(1/\sqrt{TBR})$. When T is sufficiently large such that $T \geq \frac{\mathcal{O}\left(L^2+R^2C_{err}^2+L^2R^2\right)}{L^2BR}$, we have $\frac{1}{TBR}\sum_{t=0}^{T-1}\sum_{r=0}^{R-1}\sum_{k=1}^{B}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\bar{x}_k^{[t,r]})\right\|^2\right] = \mathcal{O}\left(\frac{1}{\sqrt{TBR}}\right)$.*

The above theoretical results formally establish that DECA achieves a convergence rate of $\mathcal{O}(1/\sqrt{TBR})$, which matches the optimal asymptotic rates of state-of-the-art decentralized learning algorithms (e.g., (Lin et al., 2021; Lian et al., 2017; Esfandiari et al., 2021; Xu et al., 2021; Ghiasvand et al., 2025a)) when the number of blocks B is treated as a constant. Notably, this guarantee holds even under non-IID data distributions. Intuitively, this theoretical result captures the fundamental trade-off of block-wise stochastic optimization: partitioning the model into B blocks reflects the sequential traversal of all parameters. The convergence rate of $\mathcal{O}(1/\sqrt{TBR})$ matches the standard stochastic optimization rate $\mathcal{O}(1/\sqrt{TR})$ under a unified accounting of block-level updates, demonstrating that DECA converges to solutions comparable to full-model decentralized methods.

5 Experiments

In this section, we first introduce the experimental settings in Sec.5.1, and then report the results of our extensive experiments in Sec.5.2 and Sec.5.3. More details on the experimental setup and additional experiment results are provided in Appendices C and D, respectively.

5.1 Experiment Settings

We validate our framework across four widely adopted LLMs with varying parameter scales: Llama-2-7B (Touvron et al., 2023), Llama-3.1-8B-Instruct (Dubey et al., 2024), Qwen2-1.5B (Yang et al., 2024), and Qwen2.5-3B-Instruct (Yang et al., 2025). Our evaluation encompasses both classification and generation tasks. For classification, we use three standard datasets (NWGI, TFNS, and MNLI) with their pre-defined training and test splits, and report accuracy (Acc.) and F_1 score as evaluation metrics. To evaluate generation capability, we use the Alpaca dataset and assess model outputs with Vicuna (VIC.) (Chiang et al., 2023) and MT-Bench (MT.) (Zheng et al., 2024).

We consider an Erdős–Rényi (ER) graph consisting of eight clients. To simulate non-IID data distributions, we partition the data using a Dirichlet distribution parameterized by μ, (Lin et al., 2021; Takezawa et al., 2023; Ghiasvand et al., 2025a; Guo et al., 2025). A smaller μ (e.g., $\mu = 0.25$ in our case) induces more skewed client data distributions and thus stronger heterogeneity. We set the learning rate to $\gamma = 5 \times 10^{-5}$, the total number of communication rounds to $T = 4$, and the number of local update steps per block to $R = 48$. For the Adam optimizer, we adopt the following hyperparameters: $\alpha_1 = 0.9$, $\alpha_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$. For BMA, we adopt $\beta_1 = 0.9$, $\beta_2 = 0.999$ for classification tasks and $\beta_1 = 0.3$, $\beta_2 = 0.333$ for generation tasks. For model partitioning, we group every two consecutive Transformer layers into a trainable block, such that the Llama-2-7B, Llama-3.1-8B, Qwen2-1.5B, and Qwen2.5-3B models are partitioned into 32, 36, 28 and 32 blocks,

Table 1: Performance comparison in classification tasks. Results are reported as mean±std. Values in **bold** indicate the best performance among the *decentralized* methods, while underlined values denote the second best results. The last column reports the average performance across all datasets.

Methods	NWGI		TFNS		MNLI		Average	
	Acc.	F_1	*Acc.*	F_1	*Acc.*	F_1	*Acc.*	F_1
Qwen2.5-3B								
BAdam	60.44±0.26	38.00±0.22	90.98±0.21	88.23±0.47	86.95±0.20	86.93±0.18	79.46	71.05
Dec-Adapter	51.92±0.81	34.76±0.79	88.76±0.31	85.69±0.49	81.41±0.84	79.38±0.73	74.03	66.61
Dec-LoRA	52.76±0.32	34.53±0.71	87.33±1.39	82.77±1.80	80.73±0.95	80.04± 1.27	73.61	65.78
DeCAF	44.01±1.08	26.47±1.50	67.15±1.03	54.65±0.88	70.93±1.09	66.60±1.11	60.70	49.24
DECA	**54.36±0.86**	**35.54±0.72**	**89.37±0.37**	**86.78±0.52**	**85.24±0.61**	**84.56±1.10**	**76.32**	**68.96**
Llama-3.1-8B								
BAdam	59.29±0.11	40.54±0.26	91.29±0.25	88.48±0.18	89.80±0.11	89.59±0.22	80.13	72.87
Dec-Adapter	13.98±1.34	5.72±0.88	26.67±5.25	17.03±2.65	33.07±1.45	20.17±2.03	24.57	14.31
Dec-LoRA	54.57±0.22	38.21±0.21	**90.56±0.67**	**87.85±0.81**	86.85±1.40	86.52±1.62	77.33	70.86
DeCAF	45.71±2.02	28.65±0.54	65.85±3.01	54.23±2.69	74.17±2.38	69.68±2.45	61.91	50.85
DECA	**57.24±0.17**	**39.94±0.17**	90.30±0.18	87.44±0.31	**89.18±0.76**	**88.98±0.73**	**78.91**	**72.12**

Table 2: Performance comparison in generation tasks.

Methods	Qwen2-1.5B				Llama-2-7B			
	VIC.	**MT-1**	**MT-2**	**MT.**	**VIC.**	**MT-1**	**MT-2**	**MT.**
BAdam	5.94	4.98	3.38	4.18	5.98	4.55	3.14	3.84
Dec-Adapter	5.38	4.18	2.65	3.41	4.86	3.92	2.62	3.27
Dec-LoRA	5.52	4.43	2.95	3.69	5.60	4.30	2.87	3.59
DeCAF	**5.71**	4.87	3.26	4.06	**5.91**	4.31	**3.00**	3.65
DECA	5.68	**4.88**	**3.30**	**4.10**	5.86	**4.50**	2.93	**3.72**

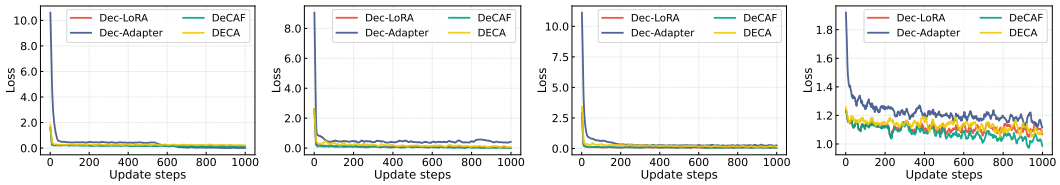
respectively. We choose **BAdam** (Luo et al., 2024) which performs FPFT in a centralized setting to provide a reference upper-bound performance. Besides, we select three state-of-the-art decentralized LLM fine-tuning methods as baselines: **Dec-Adapter** (Houlsby et al., 2019) extends PEFT methods originally implemented via adapters to decentralized scenarios. **Dec-LoRA** (Ghiasvand et al., 2025a) and **DeCAF** (Saadati et al., 2026) propose decentralized LoRA strategies.

5.2 Comparison with Baselines

We first compare DECA with existing baselines on classification tasks. As shown in Table 1, DECA achieves the best average accuracy and F_1 score on both Qwen2.5-3B and Llama-3.1-8B models among the decentralized methods and achieves performance close to that of centralized BAdam, demonstrating its consistent effectiveness across model scales. On Qwen2.5-3B, DECA outperforms the strongest baseline by 2.29% in average accuracy and 2.35% in average F_1 score, with improvements observed on all three datasets. On Llama-3.1-8B, DECA also obtains the best average results. These results indicate that DECA effectively alleviates the limitations imposed by the low-rank approximations used in prior decentralized fine-tuning methods.

Table 2 compares DECA with the decentralized PEFT baselines on generation tasks. On Qwen2-1.5B, DECA achieves the best MT-1 (4.88), MT-2 (3.30), and overall MT (4.10) scores, while remaining close to DeCAF on VIC. On Llama-2-7B, DECA obtains the highest MT-1 (4.50) and overall MT (3.72) scores, although DeCAF performs slightly better on VIC. and MT-2. Overall, DECA consistently delivers the best or near-best performance across model scales, validating the effectiveness of the proposed method for generation tasks. It is worth noting that, although DeCAF can achieve slightly better results on a few metrics by employing TSVD decomposition, it incurs substantially higher computation overhead, up to $230.52\times$ that of DECA, as shown in Appendix D.5.

To illustrate the convergence behaviors of the different algorithms on the classification and generation tasks, we plot the training loss trajectories over model update steps (which correspond to iterations of



(a) Llama-3.1-8B on NWGI (b) Llama-3.1-8B on TFNS (c) Llama-3.1-8B on MNLI (d) Qwen2-1.5B on Alpaca

Figure 1: Training loss of different baseline algorithms using Llama-3.1-8B and Qwen2-1.5B model.

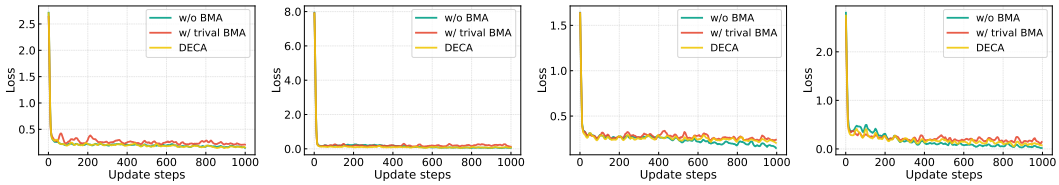
the innermost loop r) in Fig. 1. On the classification tasks, DECA achieves a substantially lower loss than Dec-Adapter and converges at a faster rate, demonstrating more efficient optimization under non-IID data. Compared with the LoRA-based decentralized PEFT baselines, Dec-LoRA and DeCAF, DECA exhibits comparable convergence rates and final loss values, indicating that it can maintain stable optimization performance while using full-parameter fine-tuning. On the generation tasks, DECA again clearly outperforms Dec-Adapter, which consistently suffers from the highest loss values and the slowest convergence. Meanwhile, DECA shows convergence behavior similar to Dec-LoRA, reaching comparable final losses with a similar convergence speed. Although DeCAF continues to reduce its loss throughout training (with much higher computation overhead), it exhibits larger fluctuations within the observed update steps. Overall, DECA achieves competitive convergence performance compared with LoRA-based decentralized algorithms.

5.3 Ablation Study

To investigate the importance of the BMA module in our algorithm, we design two ablation settings: **w/ trivial BMA** and **w/o BMA**. In the former setting, we let $m_{i,k}^{[t,r+1]} = \alpha_1 m_{i,k}^{[t,r]} + (1-\alpha_1) h_{i,k}^{[t,r]}$ and $v_{i,k}^{[t,r+1]} = \alpha_2 v_{i,k}^{[t,r]} + (1-\alpha_2)\, h_{i,k}^{[t,r]} \odot h_{i,k}^{[t,r]}$, which can be viewed as a block-wise extension of the quasi-global momentum principle in (Lin et al., 2021), while in the latter one, we ablate the entire BMA module by removing Lines 12–13 in **Algorithm** 1, which yields a trivial decentralized variant of the block-wise Adam optimization (Luo et al., 2024). Table 3 illustrates that DECA consistently achieves the best performance among ablation settings on classification tasks. Taking Qwen2.5-3B as an example, DECA reaches an average accuracy of 76.32% and an average F_1 score of 68.96%, while removing BMA leads to a clear degradation of 6.76% and 7.58%, respectively; replacing by w/ trivial BMA further reduces performance to 58.46% and 53.83%. A similar pattern is observed on Llama-3.1-8B, where DECA achieves 78.91% accuracy and a 72.12% F_1 score, consistently outperforming w/ trivial BMA and w/o BMA. Fig. 2 further compares the training loss under different ablation settings across models. All methods exhibit similar convergence speeds in the early stage of fine-tuning. However, as training proceeds, w/ trivial BMA shows noticeable oscillations in the loss trajectory, especially on Qwen2.5-3B on NWGI dataset, suggesting that overemphasizing consensus-derived information may disturb local optimization. In contrast, w/o BMA remains relatively stable, but its performance is clearly inferior to DECA, as shown in Table 3. These results indicate that DECA achieves a better balance between local gradient-driven optimization and consensus-guided correction, leading to both stable convergence and stronger downstream performance.

6 Related Work

To reduce the computational burden of LLM training, PEFT methods—such as Adapter (Houlsby et al., 2019), Prompt Tuning (Lester et al., 2021; Li & Liang, 2021), and the widely adopted LoRA (Hu et al., 2022; Lialin et al., 2024; Xia et al., 2024)—update only a fraction of parameters. While efficient, PEFT often restricts model capacity. Conversely, memory-efficient FPFT methods have emerged, leveraging techniques like on-the-fly gradient computation (LOMO) (Lv et al., 2024), zeroth-order approximations (MeZO) (Malladi et al., 2023), gradient projections (GaLore) (Zhao et al., 2024), and CPU offloading (Ren et al., 2021; Liu et al., 2024). Notably, (Luo et al., 2024) employs block coordinate descent to construct a block-wise Adam optimizer. However, this approach is strictly centralized and its theoretical convergence guarantee holds only in a deterministic setting. Extending it to decentralized stochastic optimization is highly non-trivial due to the mismatch between local block-wise updates and globally heterogeneous data distributions.



(a) Qwen2.5-3B on NWGI (b) Qwen2.5-3B on MNLI (c) Llama-3.1-8B on NWGI (d) Llama-3.1-8B on TFNS

Figure 2: Training loss of different ablation algorithms using Qwen2.5-3B and Llama-3.1-8B model.

Table 3: Ablation results for classification tasks.

Methods	NWGI		TFNS		MNLI		Average	
	Acc.	F_1	*Acc.*	F_1	*Acc.*	F_1	*Acc.*	F_1
Qwen2.5-3B								
DECA	**54.36**	**35.54**	**89.37**	**86.78**	**85.24**	**84.56**	**76.32**	**68.96**
w/ trivial BMA	30.62	21.38	76.66	73.49	68.10	66.63	58.46	53.83
w/o BMA	48.67	31.53	82.38	76.62	77.62	75.98	69.56	61.38
Llama-3.1-8B								
DECA	**57.24**	**39.94**	**90.30**	**87.44**	**89.18**	**88.98**	**78.91**	**72.12**
w/ trivial BMA	56.85	37.48	90.23	87.35	87.26	87.01	78.11	70.61
w/o BMA	53.81	37.54	86.03	83.21	78.85	76.07	72.90	65.61

To preserve privacy, FL aggregates updates via a central server. Due to communication constraints, most FL-LLM works rely on PEFT, including prompt-based FedPrompt (Zhao et al., 2023) and various LoRA adaptations like FDLoRA (Qi et al., 2024), FedSA-LoRA (Guo et al., 2025), and FFA-LoRA (Sun et al., 2024). While recent works like FedKSeed (Qin et al., 2024) and Ferret (Shu et al., 2025) attempt federated FPFT, they remain dependent on a central coordinator. Decentralized learning eliminates the server bottleneck via peer-to-peer communication. Foundational algorithms like DPSGD (Lian et al., 2017) and DMSGD (Yu et al., 2019) established convergence for IID data. To address data heterogeneity, advanced techniques such as Gradient Tracking (Pu & Nedi, 2021; Takezawa et al., 2023) and Cross-Gradient Aggregation (Esfandiari et al., 2021) have been developed. (Lin et al., 2021) introduces Quasi-Global (QG) momentum for decentralized SGD under data heterogeneity, which locally approximates the global optimization direction. Building on this principle, we design first- and second-order BMAs that integrate fresh local gradient statistics with consensus-derived discrepancy signals in a block-wise manner, thereby mitigating client drift and stabilizing Adam-style optimization on non-IID data with theoretical guarantees. While decentralized adaptation for LLMs has been explored via PEFT (e.g., Dec-LoRA (Ghiasvand et al., 2025a) and DeCAF (Saadati et al., 2026)), these methods are limited by updating only a small subset of model parameters. *To the best of our knowledge, no existing work has established a decentralized framework for FPFT, representing an unexplored frontier for achieving superior task-specific adaptation.* A detailed literature survey is available in Appendix E.

7 Conclusion

In this paper, we presented DECA, a resource-efficient decentralized FPFT framework for LLMs under non-IID data. DECA decentralizes block-wise Adam optimization by partitioning model parameters into disjoint blocks, enabling clients to collaboratively fine-tune the full model with reduced resource consumption. To mitigate client drift caused by data heterogeneity, DECA introduces first- and second-order BMAs that incorporate fresh local gradient statistics and consensus-derived discrepancy signals into Adam-style moment estimation, thereby stabilizing block-wise optimization without central coordination. We establish convergence guarantees for DECA under stochastic gradients and non-IID data, and extensive experiments on classification and generation tasks demonstrate its effectiveness across diverse LLMs.

AI USE STATEMENT

Large Language Models (LLMs) were used as writing assistance tools of this manuscript. Their use was limited to improving the presentation of the text, including polishing academic language, correcting grammatical issues, rephrasing sentences, and enhancing the coherence and readability of the manuscript.

It is important to note that the LLMs did not participate in formulating the research questions, developing the proposed methods, designing the experiments, interpreting the results, or conducting any scientific analysis. All technical contributions, methodological decisions, experimental procedures, and conclusions presented in this work were independently developed and verified by the authors. Therefore, the use of LLMs was restricted to linguistic refinement and did not contribute to the scientific or intellectual content of the study.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. In *Proc. of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP)*, pp. 7319–7328, 2021.

S. Aketi, A. Hashemi, and K. Roy. Global Update Tracking: A Decentralized Learning Algorithm for Heterogeneous Data. In *Proc. of the 37th NeurIPS*, 2024.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy Layerwise Learning Can Scale To ImageNet. In *Proc. of the 36th International Conference on Machine Learning (ICML)*, pp. 583–593, 2019.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-Wise Training of Deep Networks. In *Proc. of the 20th Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 153–160, 2006.

Stephen P. Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized Gossip Algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

Hong-You Chen, Cheng-Hao Tu, Ziwei Li, and Han-Wei Shen an Wei-Lun Chao. On the Importance and Applicability of Pre-Training for Federated Learning. In *Proc. of the 11th International Conference on Learning Representations (ICLR)*, 2023.

Shuaijun Chen, Omid Tavallaie, Niousha Nazemi, and Albert Y. Zomaya. RBLA: Rank-Based-LoRA-Aggregation for Fine-Tuning Heterogeneous Models in FLaaS. In *Proc. of the 31st International Conference on Web Service (ICWS)*, pp. 47–62, 2024.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training Deep Nets with Sublinear Memory Cost. *arXiv preprint arXiv:1604.06174*, 2016.

W. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *https://vicuna.lmsys.org (accessed 14 April 2023)*, 2(3):6, 2023.

Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, and Gauri Joshi. Heterogeneous LoRA for Federated Fine-tuning of On-Device Foundation Models. In *Proc. of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 12903–12913, 2024.

Martijn de Vos, Sadegh Farhadkhani, Rachid Guerraoui, Anne-Marie Kermarrec, Rafael Pires, and Rishi Sharma. Epidemic Learning: Boosting Decentralized Learning with Randomized Communication. In *Proc. of the 36th NeurIPS*, 2023.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. In *Proc. of the 36th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, and A. Fan. The LLaMA 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024.

Yasaman Esfandiari, Sin Yong Tan, Zhanhong Jiang, Aditya Balu, Ethan Herron, Chinmay Hegde, and Soumik Sarkar. Cross-Gradient Aggregation for Decentralized Learning from Non-IID Data. In *Proc. of the 38th International Conference on Machine Learning (ICML)*, pp. 3036–3046, 2021.

Sajjad Ghiasvand, Mahnoosh Alizadeh, and Ramtin Pedarsani. Decentralized low-rank fine-tuning of large language models. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pp. 334–345, 2025a.

Sajjad Ghiasvand, Amirhossein Reisizadeh, Mahnoosh Alizadeh, and Ramtin Pedarsani. Robust Decentralized Learning With Local Updates and Gradient Tracking. *IEEE Transactions on Networking*, 33(4):2036–2048, 2025b.

Pengxin Guo, Shuang Zeng, Yanran Wang, Huijie Fan, Feifei Wang, and Liangqiong Qu. Selective Aggregation for Low-Rank Adaptation in Federated Learning. In *Proc. of the 13th International Conference on Learning Representations (ICLR)*, 2025.

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 *Proc. of the 36th International Conference on Machine Learning (ICML)*, pp. 2790–2799, 2019.

E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *Proc. of the 10th International Conference on Learning Representations (ICLR)*, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proc. of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

Soroush Abbasi Koohpayegani, KL Navaneet, Parsa Nooralinejad, Soheil Kolouri, and Hamed Pirsiavash. NOLA: Networks as Linear Combination of Low Rank Random Basis. *arXiv preprint arXiv:2310.02556*, 2023.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. VeRA: Vector-based Random Matrix Adaptation. In *Proc. of the 12th International Conference on Learning Representations (ICLR)*, 2024.

Brian Lester, Rami Al-Rfou, and Noah Constant. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proc. of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3045–3059, 2021.

C. Li, G. Li, and P. Varshney. Decentralized Federated Learning via Mutual Knowledge Transfer. *IEEE Internet of Things Journal*, 9(2):1136–1147, 2021.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the Intrinsic Dimension of Objective Landscapes. In *Proc. of the 6th International Conference on Learning Representations (ICLR)*, 2018.

Xiang Lisa Li and Percy Liang. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP)*, pp. 4582–4597, 2021.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. ReLoRA: High-Rank Training Through Low-Rank Updates. In *Proc. of the 12th International Conference on Learning Representations (ICLR)*, 2024.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. In *Proc. of the 31st Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 5330–5340, 2017.

Tao Lin, Sai Praneeth Karimireddy, Sebastian U. Stich, and Martin Jaggi. Quasi-global Momentum: Accelerating Decentralized Deep Learning on Heterogeneous Data. In *Proc. of the 38th International Conference on Machine Learning (ICML)*, pp. 6654–6665, 2021.

Yongkang Liu, Yiqun Zhang, Qian Li, Tong Liu, Shi Feng, Daling Wang, Yifei Zhang, and Hinrich Schütze. HiFT: A Hierarchical Full Parameter Fine-Tuning Strategy. In *Proc. of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 18266–18287, 2024.

Z. Lu and L. Xiao. On the complexity analysis of randomized block-coordinate descent methods. *Mathematical Programming*, 152(1):615–642, 2015.

Q. Luo, H. Yu, and X. Li. BAdam: A Memory Efficient Full Parameter Optimization Method for Large Language Models. In *Proc. of the 38th Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 24926–24958, 2024.

Kai Lv, Yuqing Yang, Tengxiao Liu, Qipeng Guo, and Xipeng Qiu. Full Parameter Fine-tuning for Large Language Models with Limited Resources. In *Proc. of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 8187–8198, 2024.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. Fine-Tuning Language Models With Just Forward Passes. In *Proc. of the 36th Annual Conference on Neural Information Processing Systems (NeurIPS)*, pp. 53038–53075, 2023.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proc. of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1273–1282, 2017.

John Nguyen, Jianyu Wang, Kshitiz Malik, Maziar Sanjabi, and Michael G. Rabbat. Where to Begin? On the Impact of Pre-Training and Initialization in Federated Learning. In *Proc. of the 11th International Conference on Learning Representations (ICLR)*, 2023.

George Pu, Anirudh Jain, Jihan Yin, and Russell Kaplan. Empirical Analysis of The Strengths and Weaknesses of PEFT Techniques for LLMs. *arXiv preprint arXiv:2304.14999*, 2023.

S. Pu and A. Nedi. Distributed Stochastic Gradient Tracking Methods. *Mathematical Programming*, 187(1):409–457, 2021.

Jiaxing Qi, Zhongzhi Luan, Shaohan Huang, Carol Fung, Hailong Yang, and Depei Qian. FDLoRA: Personalized Federated Learning of Large Language Model via Dual LoRA Tuning. *arXiv preprint arXiv:2406.07925*, 2024.

Zhen Qin, Daoyuan Chen, Bingchen Qian, Bolin Ding, Yaliang Li, and Shuiguang Deng. Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes. In *Proc. of the 41st International Conference on Machine Learning (ICML)*, 2024.

Liangqiong Qu, Yuyin Zhou, Paul Pu Liang, Yingda Xia, Feifei Wang, Ehsan Adeli, Fei-Fei Li, and Daniel L. Rubin. Rethinking Architecture Design for Tackling Data Heterogeneity in Federated Learning. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10051–10061, 2022.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. ZeRO-Offload: Democratizing Billion-Scale Model Training. In *Proc. of the 2021 USENIX Annual Technical Conference (ATC)*, pp. 551–564, 2021.

Nastaran Saadati, Zhanhong Jiang, Joshua R. Waite, Shreyan Ganguly, Aditya Balu, Chinmay Hegde, and Soumik Sarkar. DeCAF: Decentralized Consensus-And-Factorization for Low-Rank Adaptation of Foundation Models. *Neural Networks*, 2026.

K. Scaman, F. Bach, S. Bubeck, Y. Lee, and L. Massoulié. Optimal Algorithms for Smooth and Strongly Convex Distributed Optimization in Networks. In *Proc. of the 34th International Conference on Machine Learning (ICML)*, pp. 3027–3036, 2017.

K. Scaman, F. Bach, S. Bubeck, Y. Lee, and L. Massoulié. Optimal Algorithms for Non-smooth Distributed Optimization in Networks. In *Proc. of the 32nd NIPS*, pp. 2745–2754, 2018.

Y. Shi, L. Shen, K. Wei, Y. Sun, B. Yuan, X. Wang, and D. Tao. Improving the Model Consistency of Decentralized Federated Learning. In *Proc. of the 40th ICML*, volume 202, pp. 31269–31291, 2023.

Yao Shu, Wenyang Hu, See-Kiong Ng, Bryan Kian Hsiang Low, and Fei Richard Yu. Ferret: Federated full-parameter tuning at scale for large language models. In *Proc. of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving LoRA in Privacy-preserving Federated Learning. In *Proc. of the 12th International Conference on Learning Representations (ICLR)*, 2024.

Y. Takezawa, H. Bao, K. Niwa, R. Sato, and M. Yamada. Momentum Tracking: Momentum Acceleration for Decentralized Deep Learning on Heterogeneous Data. *Trans. on Machine Learning Research*, 2023, 2023.

H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Paul Tseng. Convergence of A Block Coordinate Descent Method for Nondifferentiable Minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. In *Proc. of the 30th Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 5998–6008, 2017.

Lina Wang, Yunsheng Yuan, Chunxiao Wang, and Feng Li. PDSL: Privacy-Preserved Decentralized Stochastic Learning with Heterogeneous Data Distribution. In *Proc. of the 45th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 736–746, 2025.

Lina Wang, Yunsheng Yuan, Feng Li, and Lingjie Duan. ROSS: RObust decentralized Stochastic learning based on Shapley values. *IEEE Transactions on Networking*, 34:2911–2926, 2026.

Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. FLoRA: Federated Fine-Tuning Large Language Models with Heterogeneous Low-Rank Adaptations. In *Proc. of the 38th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Chenxing Wei, Yao Shu, Ying Tiffany He, and Fei Yu. Flexora: Flexible Low-Rank Adaptation for Large Language Models. In *Proc. of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 14643–14682, 2025.

Orion Weller, Marc Marone, Vladimir Braverman, Dawn J. Lawrie, and Benjamin Van Durme. Pretrained Models for Multilingual Federated Learning. In *Proc. of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pp. 1413–1421, 2022.

Stephen J Wright. Coordinate Descent Algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Wenhan Xia, Chengwei Qin, and Elad Hazan. Chain of LoRA: Efficient Fine-tuning of Language Models via Residual Learning. *arXiv preprint arXiv:2401.04151*, 2024.

J. Xu, W. Zhang, and F. Wang. A(DP)2SGD: Asynchronous Decentralized Parallel Stochastic Gradient Descent With Differential Privacy. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 44(11):8036–8047, 2021.

A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, et al. Qwen2 technical report. *eprint arXiv:2407.10671*, 2024.

A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Tang, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu. QWen2.5 Technical Report. *arXiv preprint*, pp. 2412.15115, 2025.

Hao Yu, Rong Jin, and Sen Yang. On the Linear Speedup Analysis of Communication Efficient Momentum SGD for Distributed Non-Convex Optimization. In *Proc. of the 36th International Conference on Machine Learning (ICML)*, pp. 7184–7193, 2019.

Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. When Scaling Meets LLM Finetuning: The Effect of Data, Model and Finetuning Method. In *Proc. of The 12th International Conference on Learning Representations (ICLR)*, 2024a.

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.

X. Zhang, X. Chen, M. Hong, S. Wu, and J. Yi. Understanding Clipping for Federated Learning: Convergence and Client-Level Differential Privacy. In *Proc. of the 39th ICML*, volume 162, pp. 26048–26067, 2022a.

Xin Zhang, Minghong Fang, Zhuqing Liu, Haibo Yang, Jia Liu, and Zhengyuan Zhu. NET-FLEET: Achieving Linear Convergence Speedup for Fully Decentralized Federated Learning with Heterogeneous Data. In *Proc. of the 23rd International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc)*, pp. 71–80, 2022b.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Diederik P. Kingma, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use Fewer Learning Rates To Gain More. In *Proc. of The 13th International Conference on Learning Representations (ICLR)*, 2025.

Zixin Zhang, Fan Qi, and Changsheng Xu. Enhancing Storage and Computational Efficiency in Federated Multimodal Learning for Large-Scale Models. In *Proc. of the 41st International Conference on Machine Learning (ICML)*, 2024b.

Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. FedPrompt: Communication-Efficient and Privacy-Preserving Prompt Tuning in Federated Learning. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient LLM training by gradient low-rank projection. In *Proc. of 41st International Conference on Machine Learning (ICML)*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Proc. of the 36th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

CONTENTS

1 Introduction **1**

2 System Model and Preliminaries **2**

3 Our Algorithm **3**

3.1 Overview 3

3.2 Algorithm Design 3

4 Analysis **5**

5 Experiments **6**

5.1 Experiment Settings 6

5.2 Comparison with Baselines 7

5.3 Ablation Study 8

6 Related Work **8**

7 Conclusion **9**

A Proof of Theorem 4.5 **17**

A.1 Preliminary Lemmas 17

A.2 Proof Details 18

B Additional Theoretical Analysis **25**

B.1 Communication Complexity 25

B.2 Memory Consumption 25

B.3 Computation Complexity 26

C More Information about Our Experiment Settings **26**

D Additional Experiment Results **27**

D.1 Different Communication Topologies 28

D.2 Different Numbers of Clients 28

D.3 Different data heterogeneity 29

D.4 Different Partitioning Granularities 30

D.5 Resource Consumption 31

D.6 Hyper-parameter Sensitivity 32

E Comprehensive Literature Survey **33**

E.1 PEFT and FPFT 33

E.2 Federated LLM Fine-Tuning 34

E.3 Decentralized Learning/Fine-Tuning 35

APPENDIX

A PROOF OF THEOREM 4.5

A.1 PRELIMINARY LEMMAS

We first reformulate $x_{i,k}^{[t,r]}$ into a block-wise sequence and define $\bar{x}_k^{[t,r]}$ as:

$$x_{i,k}^{[t,r]} = \{\theta_{i,1}^{[t+1,0]}, \ldots, \theta_{i,k-1}^{[t+1,0]}, \theta_{i,k}^{[t,r]}, \theta_{i,k+1}^{[t,0]}, \ldots, \theta_{i,B}^{[t,0]}\} \tag{19}$$
$$\bar{x}_k^{[t,r]} = \{\bar{\theta}_1^{[t+1,0]}, \ldots, \bar{\theta}_{k-1}^{[t+1,0]}, \bar{\theta}_k^{[t,r]}, \bar{\theta}_{k+1}^{[t,0]}, \ldots, \bar{\theta}_B^{[t,0]}\}$$

Before diving into the proof details, we introduce several preliminary lemmas.

Lemma A.1 (Uniform bound for the adaptive reciprocal)**.** *For every client $i \in \mathcal{N}$, block $k \in \{1, \ldots, B\}$, outer round t, and inner step r, define*

$$u_{i,k}^{[t,r]} = \frac{1}{\sqrt{\widehat{v}_{i,k}^{[t,r]}} + \epsilon},$$

where the square root and the division are understood coordinatewise. Then

$$\|u_{i,k}^{[t,r]}\|_\infty \le \frac{1}{\epsilon}.$$

Proof. By **Algorithm** 1, $v_{i,k}^{[t,0]} = 0$. Moreover,

$$v_{i,k}^{[t,r+\frac{1}{2}]} = \alpha_2 v_{i,k}^{[t,r]} + (1-\alpha_2) g_{i,k}^{[t,r]} \odot g_{i,k}^{[t,r]},$$

and

$$v_{i,k}^{[t,r+1]} = \beta_2 v_{i,k}^{[t,r+\frac{1}{2}]} + (1-\beta_2) h_{i,k}^{[t,r]} \odot h_{i,k}^{[t,r]}.$$

Since both $g_{i,k}^{[t,r]} \odot g_{i,k}^{[t,r]}$ and $h_{i,k}^{[t,r]} \odot h_{i,k}^{[t,r]}$ are coordinatewise nonnegative, induction on r yields that $v_{i,k}^{[t,r+\frac{1}{2}]}$ is coordinatewise nonnegative. Hence the bias-corrected second moment $\widehat{v}_{i,k}^{[t,r]} = \frac{v_{i,k}^{[t,r+\frac{1}{2}]}}{1-\alpha_2^{r+1}}$ is also coordinatewise nonnegative. Therefore each coordinate of $\sqrt{\widehat{v}_{i,k}^{[t,r]}} + \epsilon$ is at least ϵ, so each coordinate of $u_{i,k}^{[t,r]}$ is at most $1/\epsilon$. Taking the maximum over coordinates proves the claim. □

Lemma A.2 (Mixing identities and spectral contraction)**.** *Let*

$$\mathbf{J} = \frac{1}{N}\mathbf{1}\mathbf{1}^\top.$$

*Under **Assumption** 4.3 and the symmetry of the DECA mixing matrix in the undirected communication model, the following identities hold:*

$$\mathbf{J}\mathbf{W} = \mathbf{W}\mathbf{J} = \mathbf{J}, \qquad (\mathbf{I}-\mathbf{J})\mathbf{W} = \mathbf{W}(\mathbf{I}-\mathbf{J}) = \mathbf{W}-\mathbf{J}.$$

Moreover, for every integer $q \ge 1$,

$$\|(\mathbf{I}-\mathbf{J})\mathbf{W}^q\|_{\mathfrak{S}} \le (\sqrt{\rho})^q.$$

Proof. By **Assumption** 4.3, $\mathbf{W}\mathbf{1} = \mathbf{1}$ and $\mathbf{1}^\top\mathbf{W} = \mathbf{1}^\top$. Hence

$$\mathbf{W}\mathbf{J} = \mathbf{W}\Big(\frac{1}{N}\mathbf{1}\mathbf{1}^\top\Big) = \frac{1}{N}(\mathbf{W}\mathbf{1})\mathbf{1}^\top = \frac{1}{N}\mathbf{1}\mathbf{1}^\top = \mathbf{J},$$

and similarly,

$$\mathbf{J}\mathbf{W} = \frac{1}{N}\mathbf{1}(\mathbf{1}^\top\mathbf{W}) = \frac{1}{N}\mathbf{1}\mathbf{1}^\top = \mathbf{J}.$$

Therefore,

$$(\mathbf{I}-\mathbf{J})\mathbf{W} = \mathbf{W}-\mathbf{J} = \mathbf{W}(\mathbf{I}-\mathbf{J}).$$

Since the communication graph is undirected, $\mathbf{W}$ is symmetric. Hence, $\mathbf{W}$ admits an orthonormal eigenbasis. The vector $\mathbf{1}$ is an eigenvector associated with eigenvalue 1, while $\mathbf{J}$ is the orthogonal projector onto $\mathrm{span}\{\mathbf{1}\}$. Consequently, $(\mathbf{I}-\mathbf{J})\mathbf{W}^q$ vanishes on $\mathrm{span}\{\mathbf{1}\}$ and has eigenvalues $\lambda_j(\mathbf{W})^q$ on $\mathrm{span}\{\mathbf{1}\}^\perp$. By **Assumption** 4.3,

$$|\lambda_j(\mathbf{W})| \le \sqrt{\rho}, \qquad j = 2, \dots, N.$$

Hence

$$\|(\mathbf{I}-\mathbf{J})\mathbf{W}^q\|_{\mathfrak{S}} = \max_{2\le j\le N} |\lambda_j(\mathbf{W})|^q \le (\sqrt{\rho})^q.$$

□

Lemma A.3 (Frobenius contraction under mixing)**.** *Under **Lemma** A.2, for every matrix $\mathbf{X} \in \mathbb{R}^{N\times d}$ and every integer $q \ge 1$,*

$$\|(\mathbf{I}-\mathbf{J})\mathbf{W}^q\mathbf{X}\|_{\mathfrak{F}}^2 \le \rho^q \|\mathbf{X}\|_{\mathfrak{F}}^2.$$

Proof. For any compatible matrices $\mathbf{A}$ and $\mathbf{X}$,

$$\|\mathbf{A}\mathbf{X}\|_{\mathfrak{F}} \le \|\mathbf{A}\|_{\mathfrak{S}} \|\mathbf{X}\|_{\mathfrak{F}}.$$

Applying this inequality with $\mathbf{A} = (\mathbf{I}-\mathbf{J})\mathbf{W}^q$ and using **Lemma** A.2, we get

$$\|(\mathbf{I}-\mathbf{J})\mathbf{W}^q\mathbf{X}\|_{\mathfrak{F}} \le \|(\mathbf{I}-\mathbf{J})\mathbf{W}^q\|_{\mathfrak{S}} \|\mathbf{X}\|_{\mathfrak{F}} \le (\sqrt{\rho})^q \|\mathbf{X}\|_{\mathfrak{F}}.$$

Squaring both sides yields the claim. □

Lemma A.4 (Hadamard-product inequality)**.** *For any matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{N\times d}$,*

$$\|\mathbf{A} \odot \mathbf{B}\|_{\mathfrak{F}} \le \|\mathbf{A}\|_{\mathfrak{F}} \, \|\mathbf{B}\|_{\infty}.$$

Proof. By the definitions of the Frobenius norm and the entrywise maximum norm,

$$\|\mathbf{A} \odot \mathbf{B}\|_{\mathfrak{F}}^2 = \sum_{i=1}^{N}\sum_{j=1}^{d} |A_{ij}B_{ij}|^2 \le \sum_{i=1}^{N}\sum_{j=1}^{d} |A_{ij}|^2 \, \|\mathbf{B}\|_{\infty}^2 = \|\mathbf{A}\|_{\mathfrak{F}}^2 \|\mathbf{B}\|_{\infty}^2.$$

Taking square roots completes the proof. □

A.2 Proof Details

The proof first introduces a virtual block-wise sequence $\hat{x}_k^{[t,r]}$ that follows a centralized SGD-style recursion with adaptive step scaling, and uses it as an analyzable proxy for the averaged decentralized iterate $\bar{x}_k^{[t,r]}$. For each active block, the proof applies smoothness to establish a one-step descent inequality for $\mathbb{E}[F(\hat{x}_k^{[t,r+1]})] - \mathbb{E}[F(\hat{x}_k^{[t,r]})]$, where the descent term is proportional to $\|\nabla_k F(\bar{x}_k^{[t,r]})\|^2$ and the remaining terms capture stochastic gradient noise, consensus error, and the sequence tracking error $e_k^{[t,r]} = \hat{x}_k^{[t,r]} - \bar{x}_k^{[t,r]}$. The consensus error is then bounded using the spectral contraction of the mixing matrix, together with the uniform bound on the adaptive reciprocal and the Hadamard-product inequality to handle Adam-style scaling. Finally, the sequence tracking error and the objective increase caused by transitions between consecutive blocks are controlled by the error-controlled block-transition assumption. Summing the resulting descent inequalities over all rounds, blocks, and inner steps yields the final average squared-gradient bound and, with $\gamma = O(1/\sqrt{TBR})$, DECA has the convergence rate of $O(1/\sqrt{TBR})$.

Non-convex Single-step Update. For all $r \ge 0$, we bridge the gap between $\hat{x}_k^{[t,r]}$ and $\bar{x}_k^{[t,r]}$ with consensus error $\left\|\bar{x}_k^{[t,r]} - x_{i,k}^{[t,r]}\right\|^2$ and error sequence $e_k^{[t,r]}$:

$$\begin{aligned}
&\mathbb{E}\left[\mathcal{F}(\hat{x}_k^{[t,r+1]})\right] - \mathbb{E}\left[\mathcal{F}(\hat{x}_k^{[t,r]})\right] \\
=&\mathbb{E}\left[\left\langle \nabla_k \mathcal{F}(\hat{x}_k^{[t,r]}), \hat{x}_k^{[t,r+1]} - \hat{x}_k^{[t,r]} \right\rangle\right] + \frac{L}{2}\mathbb{E}\left[\left\|\hat{x}_k^{[t,r+1]} - \hat{x}_k^{[t,r]}\right\|^2\right] \\
=&\mathbb{E}\left[\left\langle \nabla_k \mathcal{F}(\hat{x}_k^{[t,r]}), \hat{\theta}_k^{[t,r+1]} - \hat{\theta}_k^{[t,r]} \right\rangle\right] + \frac{L}{2}\mathbb{E}\left[\left\|\hat{\theta}_k^{[t,r+1]} - \hat{\theta}_k^{[t,r]}\right\|^2\right]
\end{aligned}$$

$$
\begin{aligned}
&= -\frac{\gamma}{N(1-\alpha_1^{r+1})}\sum_{i=1}^{N}\mathbb{E}\left[\left\langle\nabla_k\mathcal{F}(\hat{x}_k^{[t,r]}),\nabla_k f_i(x_{i,k}^{[t,r]})\right\rangle\right]+\frac{L\gamma^2}{2(1-\alpha_1^{r+1})^2}\mathbb{E}\left[\left\|\bar{g}_k^{[t,r]}\right\|^2\right]\\
&\overset{①}{\leq} -\frac{\gamma}{N(1-\alpha_1^{r+1})}\sum_{i=1}^{N}\mathbb{E}\left[\left\langle\nabla_k\mathcal{F}(\hat{x}_k^{[t,r]}),\nabla_k f_i(x_{i,k}^{[t,r]})\right\rangle\right]\\
&\quad+\frac{L\gamma^2}{2(1-\alpha_1^{r+1})^2}\mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right]+\frac{L\gamma^2\xi^2}{2N(1-\alpha_1^{r+1})^2}\\
&\overset{②}{\leq} -\frac{\gamma}{(1-\alpha_1^{r+1})}\mathbb{E}\left[\left\langle\nabla_k\mathcal{F}(\hat{x}_k^{[t,r]}),\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\rangle\right]\\
&\quad+\frac{L\gamma^2}{2(1-\alpha_1^{r+1})^2}\mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right]+\frac{L\gamma^2\xi^2}{2N(1-\alpha_1^{r+1})^2}\\
&\overset{③}{\leq}\left(\frac{L\gamma^2}{2(1-\alpha_1^{r+1})^2}-\frac{\gamma}{2(1-\alpha_1^{r+1})}\right)\mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right]\\
&\quad+\frac{L\gamma^2\xi^2}{2N(1-\alpha_1^{r+1})^2}-\frac{\gamma}{2(1-\alpha_1^{r+1})}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\hat{x}_k^{[t,r]})\right\|^2\right]\\
&\quad+\frac{\gamma}{2(1-\alpha_1^{r+1})}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\hat{x}_k^{[t,r]})-\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right]\\
&\overset{④}{\leq}\left(\frac{L\gamma^2}{2(1-\alpha_1^{r+1})^2}-\frac{\gamma}{2(1-\alpha_1^{r+1})}\right)\mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right]\\
&\quad+\frac{L\gamma^2\xi^2}{2N(1-\alpha_1^{r+1})^2}-\frac{\gamma}{4(1-\alpha_1^{r+1})}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\bar{x}_k^{[t,r]})\right\|^2\right]\\
&\quad+\frac{\gamma}{2(1-\alpha_1^{r+1})}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\hat{x}_k^{[t,r]})-\nabla_k\mathcal{F}(\bar{x}_k^{[t,r]})\right\|^2\right]\\
&\quad+\frac{\gamma}{2(1-\alpha_1^{r+1})}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\hat{x}_k^{[t,r]})-\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right]\\
&\leq\left(\frac{L\gamma^2}{2(1-\alpha_1^{r+1})^2}-\frac{\gamma}{2(1-\alpha_1^{r+1})}\right)\mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right]\\
&\quad+\frac{L\gamma^2\xi^2}{2N(1-\alpha_1^{r+1})^2}-\frac{\gamma}{4(1-\alpha_1^{r+1})}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\bar{x}_k^{[t,r]})\right\|^2\right]\\
&\quad+\frac{3\gamma}{2(1-\alpha_1^{r+1})}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\hat{x}_k^{[t,r]})-\nabla_k\mathcal{F}(\bar{x}_k^{[t,r]})\right\|^2\right]\\
&\quad+\frac{\gamma}{(1-\alpha_1^{r+1})}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\bar{x}_k^{[t,r]})-\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right]\\
&\leq\left(\frac{L\gamma^2}{2(1-\alpha_1^{r+1})^2}-\frac{\gamma}{2(1-\alpha_1^{r+1})}\right)\mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right]\\
&\quad+\frac{L\gamma^2\xi^2}{2N(1-\alpha_1^{r+1})^2}-\frac{\gamma}{4(1-\alpha_1^{r+1})}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\bar{x}_k^{[t,r]})\right\|^2\right]
\end{aligned}
$$

$$
\begin{aligned}
&+\frac{3L^2\gamma}{2(1-\alpha_1^{r+1})}\mathbb{E}\left[\left\|\hat{x}_k^{[t,r]}-\bar{x}_k^{[t,r]}\right\|^2\right]\\
&+\frac{\gamma}{N(1-\alpha_1^{r+1})}\sum_{i=1}^{N}\mathbb{E}\left[\left\|\nabla_k f_i(\bar{x}_k^{[t,r]})-\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right]\\
\overset{⑤}{\leq}&\left(\frac{L\gamma^2}{2(1-\alpha_1^{r+1})^2}-\frac{\gamma}{2(1-\alpha_1^{r+1})}\right)\mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right]\\
&+\frac{L\gamma^2\xi^2}{2N(1-\alpha_1^{r+1})^2}-\frac{\gamma}{4(1-\alpha_1^{r+1})}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\bar{x}_k^{[t,r]})\right\|^2\right]\\
&+\frac{3L^2\gamma}{2(1-\alpha_1^{r+1})}\mathbb{E}\left[\left\|e_k^{[t,r]}\right\|^2\right]+\frac{L^2\gamma}{N(1-\alpha_1^{r+1})}\sum_{i=1}^{N}\mathbb{E}\left[\left\|\bar{x}_k^{[t,r]}-x_{i,k}^{[t,r]}\right\|^2\right]
\end{aligned}
\tag{20}
$$

① comes from

$$
\mathbb{E}\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla_k F_i(x_{i,k}^{[t,r]},\xi_{i,b}^{[t,r]})-\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\leq\mathbb{E}\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2+\frac{\xi^2}{N}
$$

with $f_i(x_{i,k}^{[t,r]})$ represeting an unbiased estimation on $F_i(x_{i,k}^{[t,r]},\zeta)$; ② comes from characteristic of inner-product; ③ refers to $2E\langle A,B\rangle=-A^2-B^2+(A-B)^2$; ④ is derived from $\frac{1}{2}\mathbb{E}\|A-B\|^2-\frac{1}{4}\mathbb{E}\|B\|^2+\frac{1}{2}\mathbb{E}\|A\|^2=\mathbb{E}\|A\|^2-\mathbb{E}\langle A,B\rangle+\frac{1}{4}\mathbb{E}\|B\|^2\geq 0$ where $A=\nabla_k\mathcal{F}(\bar{x}_k^{[t,r]})+\nabla_k\mathcal{F}(\hat{x}_k^{[t,r]})$ and $B=\nabla_k\mathcal{F}(\bar{x}_k^{[t,r]})$; ⑤ satisfies when $\left(\frac{L\gamma^2}{2(1-\alpha_1^{r+1})^2}-\frac{\gamma}{2(1-\alpha_1^{r+1})}\right)\leq 0$.

Rewrite $x_k^{[t,r+1]}$ and $m_i^{[t,r+1]}$ as

$$
\begin{aligned}
x_k^{[t,r+1]}&=\sum_{j\in\mathcal{N}_i}w_{ij}x_{j,k}^{t,\tau+\frac{1}{2}}=\sum_{j\in\mathcal{N}_i}w_{ij}x_{j,k}^{[t,r]}-\gamma\sum_{j\in\mathcal{N}_i}w_{ij}\frac{\hat{m}_j^{[t,r]}}{\sqrt{\hat{v}_j^{[t,r]}}+\epsilon}\\
&=\sum_{j\in\mathcal{N}_i}w_{ij}x_{j,k}^{[t,r]}-\frac{\gamma}{(1-\alpha_1^{r+1})}\sum_{j\in\mathcal{N}_i}w_{ij}\left(\alpha_1 m_j^{[t,r]}+(1-\alpha_1)g_{j,k}^{[t,r]}\right)\odot u_j^{[t,r]}\\
&=\sum_{j\in\mathcal{N}_i}w_{ij}x_{j,k}^{[t,r]}-\frac{\gamma\alpha_1}{(1-\alpha_1^{r+1})}\sum_{j\in\mathcal{N}_i}w_{ij}\left(m_j^{[t,r]}\odot u_j^{[t,r]}\right)\\
&\quad-\frac{\gamma(1-\alpha_1)}{(1-\alpha_1^{r+1})}\sum_{j\in\mathcal{N}_i}w_{ij}\left(g_{j,k}^{[t,r]}\odot u_j^{[t,r]}\right)
\end{aligned}
\tag{21}
$$

and

$$
\begin{aligned}
m_i^{[t,r+1]}=&\beta_1 m_i^{[t,r+\frac{1}{2}]}+(1-\beta_1)h_{i,k}^{[t,r]}\\
=&\beta_1\alpha_1 m_i^{[t,r]}+\beta_1(1-\alpha_1)g_{i,k}^{[t,r]}+\frac{1-\beta_1}{\left\|d_{i,k}^{[t,r]}\right\|}\left(x_{i,k}^{[t,r]}-\sum_{j\in\mathcal{N}_i}w_{ij}x_{i,k}^{[t,r+\frac{1}{2}]}\right)\\
=&\beta_1\alpha_1 m_i^{[t,r]}+\beta_1(1-\alpha_1)g_{i,k}^{[t,r]}+\frac{1-\beta_1}{\left\|d_{i,k}^{[t,r]}\right\|}\left(x_{i,k}^{[t,r]}-\sum_{j\in\mathcal{N}_i}w_{ij}x_{j,k}^{[t,r]}\right)\\
&+\frac{\gamma(1-\beta_1)\alpha_1}{(1-\alpha_1^{r+1})\left\|d_{i,k}^{[t,r]}\right\|}\sum_{j\in\mathcal{N}_i}w_{ij}\left(m_j^{[t,r]}\odot u_j^{[t,r]}\right)\\
&+\frac{\gamma(1-\beta_1)(1-\alpha_1)}{(1-\alpha_1^{r+1})\left\|d_{i,k}^{[t,r]}\right\|}\sum_{j\in\mathcal{N}_i}w_{ij}\left(g_{j,k}^{[t,r]}\odot u_j^{[t,r]}\right)
\end{aligned}
\tag{22}
$$

where $\left\|d_{i,k}^{[t,r]}\right\| = \left\|x_{i,k}^{[t,r]} - \sum_{j\in\mathcal{N}_i} w_{ij} x_{j,k}^{[t,r+\frac{1}{2}]}\right\|$. Let $\mathbf{H}^{[t,r]} \triangleq \mathbf{M}^{[t,r]} \odot \mathbf{U}^{[t,r]}$, $\mathbf{Q}_k^{[t,r]} \triangleq \mathbf{G}_k^{[t,r]} \odot \mathbf{U}^{[t,r]}$, $\Delta_k^{[t,r]} = \operatorname{diag}\left[\left\|x_{i,k}^{[t,r]} - \sum_{j\in\mathcal{N}_i} w_{ij} x_{j,k}^{[t,r+\frac{1}{2}]}\right\|^{-1}, \forall i \in \mathcal{N}\right]$, we have the matrix forms of $x_k^{[t,r+1]}$ and $m_i^{[t,r+1]}$:

$$\mathbf{X}_k^{[t,r+1]} = \mathbf{W}\mathbf{X}_k^{[t,r]} - \frac{\gamma\alpha_1}{(1-\alpha_1^{r+1})}\mathbf{W}\mathbf{H}^{[t,r]} - \frac{\gamma(1-\alpha_1)}{(1-\alpha_1^{r+1})}\mathbf{W}\mathbf{Q}_k^{[t,r]} \tag{23}$$

and

$$\begin{aligned}
\mathbf{M}^{[t,r+1]} =& \beta_1 \mathbf{M}^{[t,r+\frac{1}{2}]} + (1-\beta_1)\Delta_k^{[t,r]}\left(\mathbf{X}_k^{[t,r]} - \mathbf{X}_k^{[t,r+\frac{1}{2}]}\right) \\
=& \beta_1\alpha_1\mathbf{M}^{[t,r]} + \beta_1(1-\alpha_1)\mathbf{G}_k^{[t,r]} + (1-\beta_1)\Delta_k^{[t,r]}\left(\mathbf{X}_k^{[t,r]} - \mathbf{X}_k^{[t,r+\frac{1}{2}]}\right) \\
=& \beta_1\alpha_1\mathbf{M}^{[t,r]} + \beta_1(1-\alpha_1)\mathbf{G}_k^{[t,r]} + (1-\beta_1)\left(\mathbf{I}-\mathbf{W}\right)\Delta_k^{[t,r]}\mathbf{X}_k^{[t,r]} \\
&+ \frac{\gamma\alpha_1(1-\beta_1)}{(1-\alpha_1^{r+1})}\mathbf{W}\Delta_k^{[t,r]}\mathbf{H}^{[t,r]} + \frac{\gamma(1-\beta_1)(1-\alpha_1)}{(1-\alpha_1^{r+1})}\mathbf{W}\Delta_k^{[t,r]}\mathbf{Q}_k^{[t,r]}
\end{aligned} \tag{24}$$

Consensus Error. Let $\mathbf{J} = \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top$. We bound the Frobenius norm of $\mathbf{X}_k^{[t,r+1]} - \bar{\mathbf{X}}_k^{[t,r+1]}$:

$$\begin{aligned}
&\frac{1}{N}\mathbb{E}\left\|\mathbf{X}_k^{[t,r+1]} - \bar{\mathbf{X}}_k^{[t,r+1]}\right\|_{\mathfrak{F}}^2 \\
=&\frac{1}{N}\mathbb{E}\left\|\begin{array}{l}\mathbf{W}\left(\mathbf{X}_k^{[t,r]} - \bar{\mathbf{X}}_k^{[t,r]}\right) - (\mathbf{W}-\mathbf{W}\mathbf{J})\frac{\gamma\alpha_1}{(1-\alpha_1^{r+1})}\mathbf{H}^{[t,r]} \\ -(\mathbf{W}-\mathbf{W}\mathbf{J})\frac{\gamma(1-\alpha_1)}{(1-\alpha_1^{r+1})}\mathbf{Q}_k^{[t,r]}\end{array}\right\|_{\mathfrak{F}}^2 \\
\overset{②}{\leq}&\frac{1+\rho}{2N}\mathbb{E}\left\|\mathbf{X}_k^{[t,r]} - \bar{\mathbf{X}}_k^{[t,r]}\right\|_{\mathfrak{F}}^2 + \frac{2(1+\rho)}{N(1-\rho)}\mathbb{E}\left\|(\mathbf{I}-\mathbf{J})\,\mathbf{W}\frac{\gamma\alpha_1}{(1-\alpha_1^{r+1})}\mathbf{H}^{[t,r]}\right\|_{\mathfrak{F}}^2 \\
&+ \frac{2(1+\rho)}{N(1-\rho)}\mathbb{E}\left\|(\mathbf{I}-\mathbf{J})\,\mathbf{W}\frac{\gamma(1-\alpha_1)}{(1-\alpha_1^{r+1})}\mathbf{Q}_k^{[t,r]}\right\|_{\mathfrak{F}}^2 \\
\overset{③}{\leq}&\frac{1+\rho}{2N}\mathbb{E}\left\|\mathbf{X}_k^{[t,r]} - \bar{\mathbf{X}}_k^{[t,r]}\right\|_{\mathfrak{F}}^2 + \frac{2(1+\rho)\rho\gamma^2\alpha_1^2}{N(1-\rho)(1-\alpha_1^{r+1})^2}\mathbb{E}\left\|\mathbf{H}^{[t,r]}\right\|_{\mathfrak{F}}^2 \\
&+ \frac{2(1+\rho)\rho\gamma^2(1-\alpha_1)^2}{N(1-\rho)(1-\alpha_1^{r+1})^2}\mathbb{E}\left\|\mathbf{Q}_k^{[t,r]}\right\|_{\mathfrak{F}}^2 \\
\overset{④}{\leq}&\frac{1+\rho}{2N}\mathbb{E}\left\|\mathbf{X}_k^{[t,r]} - \bar{\mathbf{X}}_k^{[t,r]}\right\|_{\mathfrak{F}}^2 + \frac{2(1+\rho)\rho\gamma^2\alpha_1^2}{N(1-\rho)(1-\alpha_1^{r+1})^2\epsilon^2}\mathbb{E}\left\|\mathbf{M}^{[t,r]}\right\|_{\mathfrak{F}}^2 \\
&+ \frac{2(1+\rho)\delta^2\rho\gamma^2(1-\alpha_1)^2}{(1-\rho)(1-\alpha_1^{r+1})^2\epsilon^2} \\
\overset{⑤}{\leq}&\frac{1+\rho}{2N}\mathbb{E}\left\|\mathbf{X}_k^{[t,r]} - \bar{\mathbf{X}}_k^{[t,r]}\right\|_{\mathfrak{F}}^2 + \frac{4(1+\rho)\rho\gamma^2\alpha_1^2}{N(1-\rho)(1-\alpha_1^{r+1})^2\epsilon^2}\mathbb{E}\left\|\mathbf{M}^{[t,r]} - \bar{\mathbf{M}}^{[t,r]}\right\|_{\mathfrak{F}}^2 \\
&+ \frac{4(1+\rho)\rho\gamma^2\alpha_1^2 + 2(1+\rho)\delta^2\rho\gamma^2(1-\alpha_1)^2}{(1-\rho)(1-\alpha_1^{r+1})^2\epsilon^2}
\end{aligned} \tag{25}$$

where we have ① using Young's inequality, ② and ③ according to **Lemma** A.3, ④ from **Lemma** A.1 and **Lemma** A.4, and ⑤ since $\left\|h_{i,k}^{[t,r]}\right\| \leq 1$ and thus $\left\|m_{i,k}^{[t,r+1]}\right\| = \left\|\beta_1 m_{i,k}^{[t,r+\frac{1}{2}]} + (1-\beta_1)h_{i,k}^{[t,r]}\right\| \leq \beta_1\left\|m_{i,k}^{[t,r]}\right\| + \beta_1(1-\beta_1)\left\|g_{i,k}^{[t,r]}\right\| + (1-\beta_1)\left\|h_{i,k}^{[t,r]}\right\| \leq \frac{\beta_1\delta+1}{1+\beta_1},\ \forall i,k,t,r.$

Momentum Error. Using the same way to bound the consensus error, we have:

$$\frac{1}{N}\mathbb{E}\left\|\mathbf{M}_k^{[t,r+1]} - \bar{\mathbf{M}}_k^{[t,r+1]}\right\|_F^2$$

$$
\begin{aligned}
&= \frac{1}{N}\mathbb{E}\left\| \begin{array}{l} \beta_1\alpha_1\left(\mathbf{M}^{[t,r]} - \bar{\mathbf{M}}^{[t,r]}\right) + \beta_1(1-\alpha_1)\left(\mathbf{G}_k^{[t,r]} - \bar{\mathbf{G}}_k^{[t,r]}\right) \\ + (1-\beta_1)(\mathbf{I}-\mathbf{J})\mathbf{W}\left[\Delta_k^{[t,r]}\left(\mathbf{X}_k^{[t,r]} - \mathbf{X}_k^{[t,r+\frac{1}{2}]}\right)\right] \end{array} \right\|_F^2 \\
&\overset{①}{\leq} \frac{1+\lambda}{N}\beta_1^2\alpha_1^2\mathbb{E}\left\|\mathbf{M}^{[t,r]} - \bar{\mathbf{M}}^{[t,r]}\right\|_F^2 + \frac{2(1+\frac{1}{\lambda})\beta_1^2(1-\alpha_1)^2}{N}\mathbb{E}\left\|\mathbf{G}_k{}^{[t,r]} - \bar{\mathbf{G}}_k^{[t,r]}\right\|_F^2 \\
&\quad + \frac{2(1+\frac{1}{\lambda})\rho(1-\beta_1)^2}{N}\mathbb{E}\left\|\Delta_k^{[t,r]}\left(\mathbf{X}_k^{[t,r]} - \mathbf{X}_k^{[t,r+\frac{1}{2}]}\right)\right\|_F^2 \\
&\leq \frac{1+\lambda}{N}\beta_1^2\alpha_1^2\mathbb{E}\left\|\mathbf{M}^{[t,r]} - \bar{\mathbf{M}}^{[t,r]}\right\|_F^2 + 4\left(1+\frac{1}{\lambda}\right)\beta_1^2(1-\alpha_1)^2\delta^2 + 2\left(1+\frac{1}{\lambda}\right)\rho(1-\beta_1)^2 \\
&\overset{②}{\leq} \frac{1}{1-(1+\lambda)\beta_1^2\alpha_1^2}\left(4\left(1+\frac{1}{\lambda}\right)\beta_1^2(1-\alpha_1)^2\delta^2 + 2\left(1+\frac{1}{\lambda}\right)\rho(1-\beta_1)^2\right) \qquad (26)
\end{aligned}
$$

where ① comes from Young's inequality $\|A+B\|^2 \leq (1+\lambda)\|A\|^2 + (1+\frac{1}{\lambda})\|B\|^2$ where $0 < \lambda < \frac{1-\alpha_1^2\beta_1^2}{\alpha_1^2\beta_1^2}$; ② holds for $\sum_z q^z < \frac{1}{1-q}, \forall q < 1$ and $\mathbf{M}^{[t,0]} - \bar{\mathbf{M}}^{[t,0]} = 0$. Substituting Eq. (26) into Eq. (25) and letting $C_2 = 4(1+\rho)\rho\alpha_1^2 + 2(1+\rho)\delta^2\rho(1-\alpha_1)^2$ and $C_3 = 4(1+\frac{1}{\lambda})\beta_1^2(1-\alpha_1)^2\delta^2 + 2\left(1+\frac{1}{\lambda}\right)\rho(1-\beta_1)^2$, we have:

$$
\begin{aligned}
\frac{1}{N}\mathbb{E}\left\|\mathbf{X}_k^{[t,r+1]} - \bar{\mathbf{X}}_k^{[t,r+1]}\right\|_{\mathfrak{F}}^2 \leq& \frac{1+\rho}{2N}\mathbb{E}\left\|\mathbf{X}_k^{[t,r]} - \bar{\mathbf{X}}_k^{[t,r]}\right\|_{\mathfrak{F}}^2 + \frac{C_2\gamma^2}{(1-\rho)(1-\alpha_1^{r+1})^2\epsilon^2} \\
&+ \frac{4(1+\rho)\rho\gamma^2\alpha_1^2}{N(1-\rho)(1-\alpha_1^{r+1})^2\epsilon^2}\mathbb{E}\left\|\mathbf{M}^{[t,r]} - \bar{\mathbf{M}}^{[t,r]}\right\|_{\mathfrak{F}}^2 \\
\leq& \frac{1+\rho}{2N}\mathbb{E}\left\|\mathbf{X}_k^{[t,r]} - \bar{\mathbf{X}}_k^{[t,r]}\right\|_{\mathfrak{F}}^2 + \frac{C_2\gamma^2}{(1-\rho)(1-\alpha_1^{r+1})^2\epsilon^2} \\
&+ \frac{4C_3(1+\rho)\rho\gamma^2\alpha_1^2}{(1-\rho)(1-\alpha_1^{r+1})^2\epsilon^2\left(1-(1+\lambda)\beta_1^2\alpha_1^2\right)} \\
\leq& \frac{1+\rho}{2N}\mathbb{E}\left\|\mathbf{X}_k^{[t,r]} - \bar{\mathbf{X}}_k^{[t,r]}\right\|_{\mathfrak{F}}^2 + C_4\gamma^2 \leq \frac{2C_4\gamma^2}{1-\rho} \qquad (27)
\end{aligned}
$$

where $C_4 = \frac{4C_3(1+\rho)\rho\alpha_1^2 + C_2\left(1-(1+\lambda)\beta_1^2\alpha_1^2\right)}{(1-\rho)(1-\alpha_1)^2\epsilon^2\left(1-(1+\lambda)\beta_1^2\alpha_1^2\right)}$ and the last inequality comes from $\mathbf{X}_k^{[t,0]} - \bar{\mathbf{X}}_k^{[t,0]} = \mathbf{X}_{k-1}^{[t,R]} - \bar{\mathbf{X}}_{k-1}^{[t,R]}$ and $\mathbf{X}_1^{[0,0]} = \bar{\mathbf{X}}_1^{[0,0]}$.

Sequence Tracking Error. Next we show the error boundary of $e_k^{[t,r]}$. According to the definition of the error term (see Eq. (16)), we have

$$
\begin{aligned}
e_k^{[t,r+1]} =& \hat{x}_k^{[t,r+1]} - \bar{x}_k^{[t,r+1]} \\
=& e_k^{[t,r]} + \gamma\frac{1}{1-\alpha_1^{r+1}}\left[\frac{1}{N}\sum_{i=1}^N\sum_{j\in\mathcal{N}_i} w_{ij}\left(\alpha_1 m_j^{[t,r]} + (1-\alpha_1)g_{j,k}^{[t,r]}\right)\odot u_j^{[t,r]} - \bar{g}_k^{[t,r]}\right] \\
=& e_k^{[t,0]} + \sum_{\ell=0}^r\frac{\gamma}{1-\alpha_1^{\ell+1}}\left[\frac{1}{N}\sum_{i=1}^N\sum_{j\in\mathcal{N}_i} w_{ij}\left(\alpha_1 m_j^{[t,\ell]} + (1-\alpha_1)g_{j,k}^{[t,\ell]}\right)\odot u_j^{[t,\ell]} - \bar{g}_k^{[t,\ell]}\right] \qquad (28)
\end{aligned}
$$

Under **Assumption 4.4**, we have $e_k^{[t,0]} = \hat{x}_k^{[t,0]} - \bar{x}_k^{[t,0]} = 0, \forall t, k$. Hence, by taking the expectation at both sides, we obtain

$$
\begin{aligned}
&\mathbb{E}\left[\left\|e_k^{[t,r+1]}\right\|^2\right] \\
\leq& r\gamma^2\sum_{\ell=0}^r\frac{1}{(1-\alpha_1^{\ell+1})^2}\mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^N\sum_{j\in\mathcal{N}_i} w_{ij}\left(\alpha_1 m_j^{[t,\ell]} + (1-\alpha_1)g_{j,k}^{[t,\ell]}\right)\odot u_j^{[t,\ell]} - \bar{g}_k^{[t,\ell]}\right\|^2\right]
\end{aligned}
$$

$$
\begin{aligned}
&\leq 2r\gamma^2 \sum_{\ell=0}^{r} \frac{1}{(1-\alpha_1^{\ell+1})^2} \frac{\alpha_1^2 + \delta^2(1-\alpha_1)^2}{\epsilon^2} + 2r\gamma^2 \sum_{\ell=0}^{r} \frac{1}{(1-\alpha_1^{\ell+1})^2} \mathbb{E}\left[\left\|\bar{g}_k^{[t,\ell]}\right\|^2\right] \\
&\leq 2r \frac{\gamma^2\alpha_1^2 + \gamma^2\delta^2(1-\alpha_1)^2}{(1-\alpha_1)^2\epsilon^2} + \frac{3r\gamma^2}{(1-\alpha_1)^2} \sum_{\ell=0}^{r} \mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N} \nabla_k f_i\left(x_{i,k}^{[t,\ell]}\right)\right\|^2\right] \qquad (29)
\end{aligned}
$$

Finally, we substitute Eqs. (25), (26), and (29) into Eq. (20) and obtain

$$
\begin{aligned}
&\mathbb{E}\left[\mathcal{F}(\hat{x}_k^{[t,r+1]})\right] - \mathbb{E}\left[\mathcal{F}(\hat{x}_k^{[t,r]})\right] \\
&\leq \left(\frac{L\gamma^2}{2(1-\alpha_1^{r+1})^2} - \frac{\gamma}{2(1-\alpha_1^{r+1})}\right) \mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N} \nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right] \\
&\quad - \frac{\gamma}{4(1-\alpha_1^{r+1})} \mathbb{E}\left[\left\|\nabla_k \mathcal{F}(\bar{x}_k^{[t,r]})\right\|^2\right] + \frac{L\gamma^2\xi^2}{2N(1-\alpha_1^{r+1})^2} \\
&\quad + \frac{3L^2\gamma}{2(1-\alpha_1^{r+1})} \mathbb{E}\left[\left\|e_k^{[t,r]}\right\|^2\right] + \frac{L^2\gamma}{N(1-\alpha_1^{r+1})} \sum_{i=1}^{N} \mathbb{E}\left[\left\|\bar{x}_k^{[t,r]} - x_{i,k}^{[t,r]}\right\|^2\right] \\
&\leq \left(\frac{L\gamma^2}{2(1-\alpha_1)^2} - \frac{\gamma}{2(1-\alpha_1)}\right) \mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N} \nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right] - \frac{\gamma}{4(1-\alpha_1^{r+1})} \mathbb{E}\left[\left\|\nabla_k \mathcal{F}(\bar{x}_k^{[t,r]})\right\|^2\right] \\
&\quad + \frac{L\gamma^2\xi^2}{2N(1-\alpha_1)^2} + \frac{3L^2\gamma}{2(1-\alpha_1)} \mathbb{E}\left[\left\|e_k^{[t,r]}\right\|^2\right] + \frac{L^2\gamma}{(1-\alpha_1)} \mathbb{E}\left[\left\|\bar{\mathbf{X}}_k^{[t,r]} - \mathbf{X}_{i,k}^{[t,r]}\right\|^2\right] \\
&\leq \left(\frac{L\gamma^2}{2(1-\alpha_1)^2} - \frac{\gamma}{2(1-\alpha_1)}\right) \mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N} \nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right] - \frac{\gamma}{4(1-\alpha_1^{r+1})} \mathbb{E}\left[\left\|\nabla_k \mathcal{F}(\bar{x}_k^{[t,r]})\right\|^2\right] \\
&\quad + \frac{L\gamma^2\xi^2}{2N(1-\alpha_1)^2} + \frac{3L^2\gamma}{2(1-\alpha_1)} \mathbb{E}\left[\left\|e_k^{[t,r]}\right\|^2\right] + \frac{2L^2C_4\gamma^3}{(1-\rho)(1-\alpha_1)} \qquad (30)
\end{aligned}
$$

Descent Iteration. By summing over $r = 0 \to R-1$, we get

$$
\begin{aligned}
&\mathbb{E}\left[\mathcal{F}(\hat{x}_{k+1}^{[t,0]})\right] - \mathbb{E}\left[\mathcal{F}(\hat{x}_k^{[t,0]})\right] \\
=&\mathbb{E}\left[\mathcal{F}(\hat{x}_{k+1}^{[t,0]})\right] - \mathbb{E}\left[\mathcal{F}(\hat{x}_k^{[t,R]})\right] + \mathbb{E}\left[\mathcal{F}(\hat{x}_k^{[t,R]})\right] - \mathbb{E}\left[\mathcal{F}(\hat{x}_k^{[t,R-1]})\right] \\
&+ \ldots + \mathbb{E}\left[\mathcal{F}(\hat{x}_k^{[t,2]})\right] - \mathbb{E}\left[\mathcal{F}(\hat{x}_k^{[t,1]})\right] + \mathbb{E}\left[\mathcal{F}(\hat{x}_k^{[t,1]})\right] - \mathbb{E}\left[\mathcal{F}(\hat{x}_k^{[t,0]})\right] \\
\leq& C_{err}\mathbb{E}\left[\left\|e_k^{[t,R]}\right\|^2\right] + \frac{2RL^2C_4\gamma^3}{(1-\rho)(1-\alpha_1)} + \frac{RL\gamma^2\xi^2}{2N(1-\alpha_1)^2} \\
&+ \sum_{r=0}^{R-1} \left(\frac{L\gamma^2}{2(1-\alpha_1)^2} - \frac{\gamma}{2(1-\alpha_1)}\right) \mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N} \nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right] \\
&- \sum_{r=0}^{R-1} \frac{\gamma}{4(1-\alpha_1^{r+1})} \mathbb{E}\left[\left\|\nabla_k \mathcal{F}(\bar{x}_k^{[t,r]})\right\|^2\right] + \sum_{r=0}^{R-1} \frac{3L^2\gamma}{2(1-\alpha_1)} \mathbb{E}\left[\left\|e_k^{[t,r]}\right\|^2\right] \qquad (31)
\end{aligned}
$$

Furthermore, by summing over a full round block switch, we have

$$
\begin{aligned}
&\mathbb{E}\left[\mathcal{F}(\hat{x}_1^{[t+1,0]})\right] - \mathbb{E}\left[\mathcal{F}(\hat{x}_1^{[t,0]})\right] \\
=&\mathbb{E}\left[\mathcal{F}(\hat{x}_1^{[t+1,0]})\right] - \mathbb{E}\left[\mathcal{F}(\hat{x}_B^{[t,0]})\right] + \ldots + \mathbb{E}\left[\mathcal{F}(\hat{x}_2^{[t,0]})\right] - \mathbb{E}\left[\mathcal{F}(\hat{x}_1^{[t,0]})\right] \\
\leq& \left(\frac{L\gamma^2}{2(1-\alpha_1)^2} - \frac{\gamma}{2(1-\alpha_1)}\right) \sum_{k=1}^{B} \sum_{r=0}^{R-1} \mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N} \nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right] + \frac{RBL\gamma^2\xi^2}{2N(1-\alpha_1)^2}
\end{aligned}
$$

$$+\sum_{k=1}^{B}\sum_{r=0}^{R-1}\frac{3L^2\gamma}{2(1-\alpha_1)}\mathbb{E}\left[\left\|e_k^{[t,r]}\right\|^2\right]+C_{err}\sum_{k=1}^{B}\mathbb{E}\left[\left\|e_k^{[t,R]}\right\|^2\right]$$
$$-\sum_{k=1}^{B}\sum_{r=0}^{R-1}\frac{\gamma}{4(1-\alpha_1^{r+1})}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\bar{x}_k^{[t,r]})\right\|^2\right]+\frac{2RBL^2C_4\gamma^3}{(1-\rho)(1-\alpha_1)} \tag{32}$$

Moreover, summing from $t=0\to T-1$ and letting $\mathcal{F}^\star=\mathbb{E}\left[\mathcal{F}(\hat{x}_1^{[T,0]})\right]$ and $\mathbb{E}\left[\mathcal{F}(\hat{x}^0)\right]=\mathbb{E}\left[\mathcal{F}(\hat{x}_1^{[0,0]})\right]$, we obtain

$$\begin{aligned}\mathcal{F}^\star-\mathbb{E}\left[\mathcal{F}(\hat{x}^0)\right]=&\mathbb{E}\left[\mathcal{F}(\hat{x}_1^{[T,0]})\right]-\mathbb{E}\left[\mathcal{F}(\hat{x}_1^{[T-1,0]})\right]+\ldots+\mathbb{E}\left[\mathcal{F}(\hat{x}_1^{[1,0]})\right]-\mathbb{E}\left[\mathcal{F}(\hat{x}_1^{[0,0]})\right]\\
&\le\left(\frac{L\gamma^2}{2(1-\alpha_1)^2}-\frac{\gamma}{2(1-\alpha_1)}\right)\sum_{t=0}^{T-1}\sum_{k=1}^{B}\sum_{r=0}^{R-1}\mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right]\\
&\quad+\frac{TBRL\gamma^2\xi^2}{2N(1-\alpha_1)^2}+\sum_{t=0}^{T-1}\sum_{k=1}^{B}\sum_{r=0}^{R-1}\frac{3L^2\gamma}{2(1-\alpha_1)}\mathbb{E}\left[\left\|e_k^{[t,r]}\right\|^2\right]\\
&\quad+\frac{2TBRL^2C_4\gamma^3}{(1-\rho)(1-\alpha_1)}+C_{err}\sum_{t=0}^{T-1}\sum_{k=1}^{B}\mathbb{E}\left[\left\|e_k^{[t,R]}\right\|^2\right]\\
&\quad-\sum_{t=0}^{T-1}\sum_{k=1}^{B}\sum_{r=0}^{R-1}\frac{\gamma}{4(1-\alpha_1^R)}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\bar{x}_k^{[t,r]})\right\|^2\right]\end{aligned} \tag{33}$$

By substituting Eq. (29) into Eq. (33), we obtain:

$$\begin{aligned}&\mathcal{F}^\star-\mathbb{E}\left[\mathcal{F}(\hat{x}^0)\right]\\
&\le\left(\frac{L\gamma^2}{2(1-\alpha_1)^2}-\frac{\gamma}{2(1-\alpha_1)}\right)\sum_{t=0}^{T-1}\sum_{k=1}^{B}\sum_{r=0}^{R-1}\mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right]\\
&\quad+\frac{3R\gamma^2}{(1-\alpha_1)^2}\left(\frac{3L^2\gamma R}{2(1-\alpha_1)}+C_{err}\right)\sum_{t=0}^{T-1}\sum_{k=1}^{B}\sum_{r=0}^{R-1}\mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i(x_{i,k}^{[t,r]})\right\|^2\right]\\
&\quad+\frac{TBRL\gamma^2\xi^2}{2N(1-\alpha_1)^2}++\frac{2C_4TBRL^2\gamma^3}{(1-\rho)(1-\alpha_1)}+\frac{3C_1TBR^2L^2\gamma^3}{2(1-\alpha_1)}+C_1C_{err}TBR\gamma^2\\
&\quad-\sum_{t=0}^{T-1}\sum_{k=1}^{B}\sum_{r=0}^{R-1}\frac{\gamma}{4(1-\alpha_1^R)}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\bar{x}_k^{[t,r]})\right\|^2\right]\end{aligned} \tag{34}$$

To eliminate the term $\mathbb{E}\left[\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla_k f_i\left(x_{i,k}^{[t,r]}\right)\right\|^2\right]$, it suffices to choose the learning rate γ such that its coefficient is non-positive. This requires

$$\frac{L\gamma^2}{2(1-\alpha_1)^2}-\frac{\gamma}{2(1-\alpha_1)}+\frac{3R\gamma^2}{(1-\alpha_1)^2}\left(\frac{3L^2\gamma R}{2(1-\alpha_1)}+C_{err}\right)\le 0 \tag{35}$$

Hence, we have:

$$\gamma\le\frac{2(1-\alpha_1)}{L+6RC_{err}+\sqrt{(L+6RC_{err})^2+36L^2R^2}} \tag{36}$$

Dividing by T, B and R and reordering the inequality yields:

$$\frac{1}{TBR}\sum_{t=0}^{T-1}\sum_{k=1}^{B}\sum_{r=0}^{R-1}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\bar{x}_k^{t,r})\right\|^2\right]$$

$$\leq \frac{8(1-\alpha_1^R)}{TBR\gamma}\left(\mathbb{E}\left[\mathcal{F}(\hat{x}^0)-\mathcal{F}^\star\right]\right)+\frac{4(1-\alpha_1^R)L\gamma\xi^2}{N(1-\alpha_1)^2}$$
$$+\frac{16(1-\alpha_1^R)L^2C_4\gamma^2}{(1-\rho)(1-\alpha_1)}+\frac{12(1-\alpha_1^R)RC_1L^2\gamma^2}{(1-\alpha_1)}+8C_1C_{err}(1-\alpha_1^R)\gamma \tag{37}$$

Let $\gamma=\mathcal{O}\left(\frac{1}{\sqrt{TBR}}\right)$, and γ satisfies

$$0<\gamma\leq\min\left\{\frac{(1-\beta_1)}{3L\sqrt{TBR}},\frac{2(1-\beta_1)}{L+6RC_{err}+\sqrt{(L+6RC_{err})^2+36L^2R^2}}\right\} \tag{38}$$

When

$$\frac{1}{\sqrt{TBR}}\geq\max\left\{\frac{(1-\beta_1)}{3L\sqrt{TBR}},\frac{2(1-\beta_1)}{L+6RC_{err}+\sqrt{(L+6RC_{err})^2+36L^2R^2}}\right\} \tag{39}$$

such that

$$T\geq\max\left\{\frac{(1-\beta_1)^2}{9L^2BR},\frac{\mathcal{O}\left(L^2+R^2C_{err}^2+L^2R^2\right)}{L^2BR}\right\} \tag{40}$$

we obtain

$$\frac{1}{TBR}\sum_{t=0}^{T-1}\sum_{k=1}^{B}\sum_{r=0}^{R-1}\mathbb{E}\left[\left\|\nabla_k\mathcal{F}(\bar{x}_k^{t,r})\right\|^2\right]=\mathcal{O}\left(\frac{1}{\sqrt{TBR}}\right) \tag{41}$$

Remark A.5. Compared with QGM (Lin et al., 2021), which proves convergence for decentralized SGD with first-order momentum, and BAdam (Luo et al., 2024), which analyzes centralized block-wise Adam only in the deterministic case, DECA establishes convergence for decentralized block-wise Adam under stochastic gradients and non-IID data. The proof simultaneously controls stochastic gradient noise, peer-to-peer consensus error, client drift, stale block-wise moment estimates, and the objective perturbation induced by transitions between consecutive blocks. By introducing a virtual averaged block sequence and bounding its deviation from decentralized client iterates, we prove that DECA achieves an average squared-gradient convergence rate of $O(1/\sqrt{TBR})$, thereby preserving the standard stochastic non-convex convergence behavior under block-wise optimization.

B Additional Theoretical Analysis

B.1 Communication Complexity

We analyze the per-round communication complexity of each client, where d denotes the number of model parameters and $N_{\max}$ denotes the maximum node degree of the communication graph. Standard decentralized algorithms, such as D-SGD (Lian et al., 2017), synchronize full-model parameters with neighboring clients at each iteration, incurring a per-client communication cost of $\mathcal{O}(dN_{\max})$. For LLMs with billions of parameters, such full-parameter exchange can easily saturate communication links and lead to severe latency. In contrast, DECA exchanges only the active block of size d/B during each inner step, resulting in a per-round complexity of $\mathcal{O}(dRN_{\max}/B)$. Unlike decentralized PEFT methods (Ghiasvand et al., 2025a; Saadati et al., 2026), which improve communication efficiency by restricting updates to low-rank parameter subspaces, DECA reduces bandwidth cost through block-wise communication while preserving full-parameter fine-tuning capacity.

B.2 Memory Consumption

We analyze the GPU memory consumption of DECA. Let M denote the FP16 memory required to store the model parameters, i.e., $M\approx 2d$ bytes. In standard decentralized FPFT with Adam (Kingma & Ba, 2015), each client stores FP16 parameters for forward and backward propagation, together with FP32 master parameters, gradients, and first- and second-order moment estimates. This results in a total memory cost of approximately $9M$. When the consensus-derived local discrepancy is

further incorporated, an additional FP32 auxiliary vector h is required (see Line 12 in **Algorithm** 1), increasing the memory cost to approximately $11M$. In contrast, DECA maintains FP32 optimizer states only for the active block, reducing the peak memory usage to $M + 10M/B$. This analysis excludes the memory required for intermediate activations, as activation storage is inherent to backpropagation rather than optimizer-state maintenance. Nevertheless, DECA is fully compatible with gradient checkpointing (Chen et al., 2016), which trades additional computation for reduced activation memory and can further improve feasibility on memory-constrained devices.

B.3 Computation Complexity

We analyze the computational efficiency of DECA under a layer-wise partitioning strategy that groups consecutive Transformer layers into blocks. The computational cost is measured using an *atomic backward operation*, defined as the backward pass of one batch through a single Transformer module. In standard decentralized SGD or LoRA, trainable parameters span all layers, requiring full-depth backpropagation for every batch. Let R denote the number of batches processed per block (see **Algorithm** 1). A global round processes RB batches, each traversing all B blocks, resulting in RB^2 atomic operations. In contrast, DECA updates only the active block. Backpropagation proceeds only from the loss to the active block, traversing k blocks when updating the k-th block. Summing over all blocks yields a total cost of $R\sum_{k=1}^{B} k = \frac{1}{2}RB(B+1)$. Moreover, DECA reduces backward computation by skipping parameter gradient calculations for frozen intermediate layers. Instead, it performs only the vector Jacobian products required to propagate gradients to the active block, effectively halving the arithmetic cost for these layers.

C More Information about Our Experiment Settings

We select the following models to demonstrate that our algorithm scales effectively across different parameter sizes ($1.5\text{B} - 8\text{B}$) and architectures:

- **Qwen2-1.5B** is a lightweight yet capable model from the Qwen series, designed for efficient deployment under limited computational and memory budgets. Its inclusion allows us to assess whether DECA remains effective for extremely compact models, further validating its applicability to mobile, edge, and resource-constrained decentralized scenarios.
- **Qwen2.5-3B-Instruct** is a compact architecture tailored for mobile and edge applications. We include this model to explicitly validate the feasibility of our approach within resource-constrained decentralized environments.
- **Llama-2-7B** is a widely adopted open-weights model from Meta, representing a strong previous-generation baseline in the 7B parameter class. We include this model to evaluate the effectiveness of DECA on a mature and extensively benchmarked architecture, enabling direct comparison with prior decentralized and instruction-following studies.
- **Llama-3.1-8B-Instruct** is the latest open-weights offering from Meta. This model serves as the contemporary industry standard for the 8B parameter class. Its instruction-tuning allows it to act as a robust baseline for assessing chat and instruction-following capabilities.

We utilize the following datasets to evaluate the model's understanding capabilities across diverse domains. NWGI, TFNS, and MNLI are datasets for classification tasks, while Alpaca is a dataset for generation tasks.

- **NWGI (News Writer Genre Identification)** [1] is a literary analysis task where the model classifies the genre or stylistic identity of a news writer based on article text. In this work, it serves as a benchmark dataset for evaluating the model's performance on classification tasks.
- **TFNS (Twitter Financial News Sentiment)** [2] is a domain-specific sentiment analysis benchmark that tasks models with classifying financial tweets as "Bullish" (positive) or

[1] https://huggingface.co/datasets/dohonba/nwgi

[2] https://huggingface.co/datasets/zeroshot/twitter-financial-news-sentiment

"Bearish" (negative). This dataset is specifically designed to evaluate a model's grasp of financial domain knowledge and its ability to interpret sentiment within the specialized context of market-related social media discourse.

- **MNLI (Multi-Genre Natural Language Inference)** [3] is a challenging benchmark for evaluating a model's logic and reasoning capabilities. In this task, the model is presented with a premise and a hypothesis and must determine whether the premise entails, contradicts, or remains neutral toward the hypothesis.
- **Alpaca** [4] is a dataset consisting of 52,000 instruction-response pairs generated by text-davinci-003. It serves as a standard benchmark for evaluating a large language model's ability to follow diverse user instructions. In this paper, the dataset is utilized to assess the generation capabilities of the fine-tuned models.

We employ accuracy (Acc) and F_1 score to assess model performance for classification tasks. As for the generation tasks, we adopt the following evaluation metrics:

- **Vicuna (VIC.)** (Chiang et al., 2023) is a GPT-4-based framework for evaluating chatbot quality. Instead of relying on static NLP benchmarks, it uses 80 diverse, challenging questions spanning across 8 tasks to compare model responses side-by-side, with GPT-4 serving as an automatic judge, offering an efficient proxy for human assessment of open-ended chatbot responses.
- **MT-Bench (MT.)** (Zheng et al., 2024) is a modern benchmark for evaluating multi-turn conversation quality across categories like coding, reasoning, and roleplay. It employs a "LLM-as-a-Judge" approach, using a stronger model (e.g., GPT-4) to score responses on a scale of 1–10, providing a robust proxy for human evaluation.

We select three state-of-the-art decentralized LLM fine-tuning methods as baselines. Specifically, **Dec-Adapter** (Houlsby et al., 2019) extends adapter-based PEFT to decentralized settings. **Dec-LoRA** (Ghiasvand et al., 2025a) and **DeCAF** (Saadati et al., 2026) propose decentralized LoRA-based fine-tuning strategies that eliminate the central server and rely on peer-to-peer client collaboration. By benchmarking against these methods, we systematically compare our decentralized FPFT paradigm with representative decentralized PEFT approaches across different system settings.

- **Dec-Adapter** (Houlsby et al., 2019) is a decentralized extension of adapter-based PEFT. It follows the original adapter design, where lightweight trainable modules are inserted into a frozen backbone model, and adapts this paradigm to decentralized learning by allowing clients to exchange and aggregate only adapter parameters with their neighbors.
- **Dec-LoRA** (Ghiasvand et al., 2025a) proposes a decentralized LoRa method. Specifically, each client maintains a local copy of the frozen backbone and a set of trainable low-rank adapters, exchanging and aggregating only the adapter parameters with their neighbors. This approach combines the communication efficiency of LoRA with the robustness of decentralized learning.
- **DeCAF** (Saadati et al., 2026), a decentralized LoRA-based framework that replaces traditional aggregation with consensus-aware updates, using TSVD decomposition to aggregate the local LoRA matrices.

We conduct our experiments on a GPU cluster, where each GPU corresponds to one client. Specifically, we use NVIDIA RTX 5090 GPUs for the Qwen2-1.5B and Qwen2.5-3B models, and NVIDIA RTX PRO 6000 GPUs for the larger Llama-2-7B and Llama-3.1-8B models.

D Additional Experiment Results

We evaluate the different algorithms under different communication topologies and varying numbers of clients in Sec. D.1 and Sec. D.2, respectively. We then investigate the impact of different partitioning granularities on the performance of our algorithm in Sec. D.4 and report the resource

[3] https://huggingface.co/datasets/SetFit/mnli

[4] https://huggingface.co/datasets/tatsu-lab/alpaca

Table 4: Performance comparison of different algorithms under different communication topologies.

Topology	Method	NWGI		TFNS		MNLI		Average	
		Acc.	F_1	*Acc.*	F_1	*Acc.*	F_1	*Acc.*	F_1
Ring	Dec-Adapter	**56.11**	36.43	88.73	**85.84**	79.54	78.04	74.79	66.77
	Dec-LoRA	53.52	32.91	86.31	81.86	73.77	70.17	71.20	61.65
	DeCAF	43.91	28.17	67.37	56.92	72.02	67.70	61.10	50.93
	DECA	54.09	**40.11**	**89.35**	85.72	**83.31**	**83.10**	**75.58**	**69.64**
ER	Dec-Adapter	51.92	34.76	88.76	85.69	81.41	79.38	74.03	66.61
	Dec-LoRA	52.76	34.53	87.33	82.77	80.73	80.04	73.61	65.78
	DeCAF	44.01	26.47	67.15	54.65	70.93	66.60	60.70	49.24
	DECA	**54.36**	**35.54**	**89.37**	**86.78**	**85.24**	**84.56**	**76.32**	**68.96**
Bipartite	Dec-Adapter	49.47	35.70	88.34	84.74	79.48	78.64	72.43	66.36
	Dec-LoRA	52.30	32.53	83.47	75.97	77.09	74.31	70.95	60.94
	DeCAF	43.49	27.56	72.08	68.36	71.64	67.22	62.40	54.38
	DECA	**53.07**	**39.32**	**88.65**	**85.68**	**83.95**	**83.63**	**75.22**	**69.54**

consumption of the different algorithms in Sec. D.5. We finally perform hyperparameter sensitivity analysis in Sec. D.6.

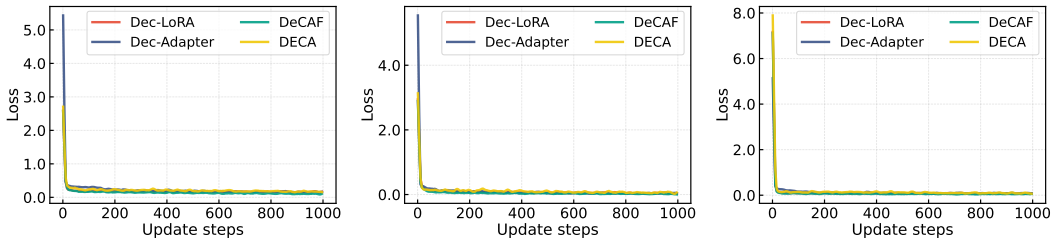
D.1 Different Communication Topologies

We employ the Qwen2.5-3B model to evaluate the classification performance of the proposed methods against baselines across ring, ER, and bipartite communication topologies, each comprising eight clients. The experimental results in Table 4 clearly demonstrate the effectiveness of DECA under the different communication topologies. Overall, DECA achieves the best average accuracy and F_1 scores across all three topologies, showing strong robustness to variations in network connectivity. In the ring topology, DECA obtains an average accuracy of 75.58% and an average F_1 score of 69.64%, outperforming Dec-Adapter by 0.79% and 2.87%, respectively. Compared with Dec-LoRA, DECA further improves the average accuracy and F_1 score by 4.38% and 7.99%, and the gains over DeCAF are even larger. Under the bipartite topology, DECA again consistently delivers the best overall performance, improving average accuracy and F_1 score by 2.79% and 3.18% over Dec-Adapter, 4.27% and 8.60% over Dec-LoRA, and 12.82% and 15.16% over DeCAF. Notably, DECA maintains highly stable performance across different topologies, with average accuracy ranging only from 75.22% to 76.32% and average F_1 score ranging from 68.96% to 69.64%. This indicates that DECA can effectively mitigate the negative impact of constrained or heterogeneous communication structures. Overall, these results confirm that DECA consistently achieves lower classification loss and stronger predictive performance than Dec-Adapter, while also surpassing decentralized PEFT baselines such as Dec-LoRA and DeCAF across different network topologies.

Furthermore, Fig. 3 presents the training loss trajectories of the different algorithms under a ring topology. DECA achieves a convergence speed and final training loss comparable to those of decentralized PEFT baselines, indicating that our block-wise full-parameter adaptation can remain stable even under sparse peer-to-peer communication. Fig. 4 further shows the training loss of DECA under different network topologies. Across all topologies, DECA exhibits steady and consistent convergence behavior, demonstrating its robustness to diverse communication structures.

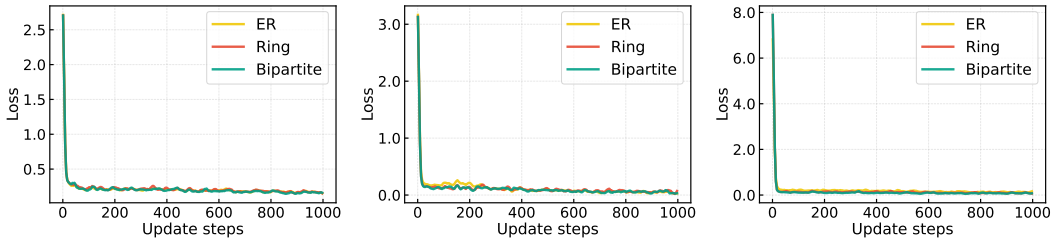
D.2 Different Numbers of Clients

We study the effect of network scale by varying the number of clients, $N \in \{8, 12, 16\}$, and applying different decentralized algorithms to fine-tune a Qwen2.5-3B model on the five classification datasets under the representative ER topology. As shown in Table 5, DECA consistently outperforms decentralized PEFT baselines across all network sizes, demonstrating its robustness to client-scale variations. In the largest setting with $N = 16$, DECA achieves an average accuracy of 70.32%, exceeding Dec-LoRA by nearly 2.25%. When $N = 12$, DECA also maintains a clear advantage over the baselines in most settings. These results confirm that DECA enables effective FPFT even as the communication network expands.



(a) NWGI dataset (b) TFNS dataset (c) MNLI dataset

Figure 3: Training loss of different algorithms under ring topology using Qwen2.5-3B model.



(a) NWGI dataset (b) TFNS dataset (c) MNLI dataset

Figure 4: Training loss of DECA on under different topologies using Qwen2.5-3B model.

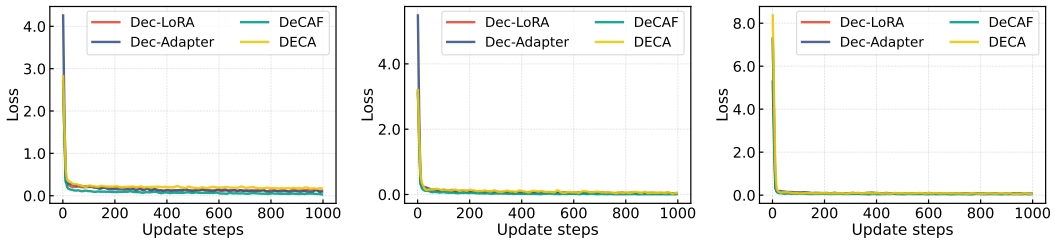
Table 5: Performance comparison of different algorithms across varying numbers of clients.

N	Algorithm	NWGI		TFNS		MNLI		Average	
		Acc.	F_1	*Acc.*	F_1	*Acc.*	F_1	*Acc.*	F_1
8	Dec-Adapter	51.92	34.76	88.76	85.69	81.41	79.38	74.03	66.61
	Dec-LoRA	52.76	34.53	87.33	82.77	80.73	80.04	73.61	65.78
	DeCAF	44.01	26.47	67.15	54.65	70.93	66.60	60.70	49.24
	DECA	**54.36**	**35.54**	**89.37**	**86.78**	**85.24**	**84.56**	**76.32**	**68.96**
12	Dec-Adapter	44.51	32.92	88.84	85.84	85.15	85.18	72.83	67.98
	Dec-LoRA	43.64	31.02	**89.80**	86.35	85.52	85.34	72.99	67.57
	DeCAF	36.66	21.14	65.95	51.18	65.44	59.61	56.02	43.98
	DECA	**46.40**	**34.04**	89.44	**86.59**	**86.85**	**86.28**	**74.23**	**68.97**
16	Dec-Adapter	46.89	30.12	84.35	81.99	78.65	**77.03**	69.96	63.05
	Dec-LoRA	46.23	29.52	82.45	79.16	75.54	73.77	68.07	60.82
	DeCAF	42.05	26.56	65.22	56.38	63.73	60.44	57.00	47.79
	DECA	**47.30**	**30.70**	**84.83**	**82.33**	**78.82**	76.69	**70.32**	**63.24**

Fig. 5 illustrates the training loss trajectories for classification tasks when $N = 12$. It is shown that, DECA achieves similar convergence rate and loss value compared to decentralized PEFT baselines. This persists regardless of the system scale, highlighting the scalability of the proposed FPFT framework. Furthermore, Fig. 6 demonstrates the training loss trajectories of DECA across the varying numbers of clients. As the number of clients grows, DECA continues to exhibit nearly unchanged convergence speed and loss levels, suggesting that it scales well and remains effective even in large-scale network settings. When more clients are involved, the heterogeneity induced by non-IID data becomes more pronounced, often causing fluctuations in the initial training stages (e.g. 16 clients). Nevertheless, DECA consistently converges to comparable loss levels across different client configurations.

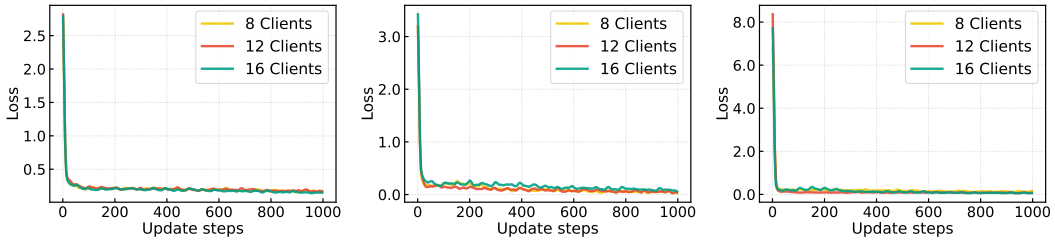
D.3 Different data heterogeneity

We further introduce a stronger heterogeneity setting with $\alpha = 0.1$, compared with $\alpha = 0.25$ in Table 1. We evaluate Dec-LoRA, DeCAF, DECA w/o BMA, and DECA using Qwen2.5-3B on



(a) NWGI dataset (b) TFNS dataset (c) MNLI dataset

Figure 5: Training loss of different algorithms with 12 clients using Qwen2.5-3B model.



(a) NWGI dataset (b) TFNS dataset (c) MNLI dataset

Figure 6: Training loss of DECA with different numbers of clients using Qwen2.5-3B model.

Table 6: Performance comparison under a stronger non-IID setting ($\alpha = 0.1$) on Qwen2.5-3B.

Algorithm	**NWGI**		**TFNS**		**MNLI**		**Average**	
	Acc.	F_1	*Acc.*	F_1	*Acc.*	F_1	*Acc.*	F_1
Dec-LoRA	37.73	23.54	65.78	66.64	68.43	61.70	57.31	50.63
DeCAF	42.62	23.87	52.78	42.81	53.24	43.48	49.55	36.72
DECA w/o BMA	41.84	18.95	66.85	59.28	60.83	53.58	56.50	43.94
DECA	**44.19**	**27.76**	**83.14**	**80.45**	**74.45**	**73.19**	**67.26**	**60.47**

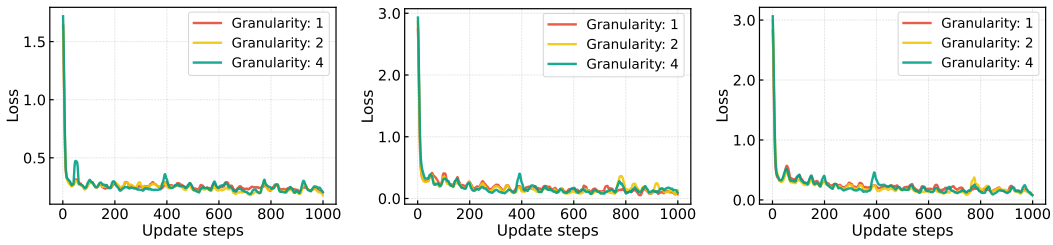
the NWGI, TFNS, and MNLI datasets. The results under $\alpha = 0.1$ are presented in Table 6. As α decreases from 0.25 to 0.1, all chosen methods experience performance degradation across the evaluated datasets, confirming that stronger data heterogeneity substantially increases the difficulty of decentralized LLM fine-tuning. Despite this overall degradation, DECA consistently achieves the best performance under the stronger non-IID setting. Specifically, DECA obtains the highest average accuracy and F1 of 67.26% and 60.47%, respectively. Compared with the strongest competing method in terms of average accuracy, Dec-LoRA, DECA improves the average accuracy by 9.95% and the average F1 by 9.84%. Moreover, DECA outperforms DECA w/o BMA by 10.76% and 16.53% in average accuracy and F1, respectively, demonstrating that the BMA mechanism remains particularly beneficial when client data distributions become highly heterogeneous. These results further validate the robustness of DECA and the effectiveness of its design for mitigating conflicting local updates under severe non-IID conditions.

D.4 Different Partitioning Granularities

We further investigate the impact of model partitioning granularity by setting the granularity to $\{1, 2, 4\}$, where the parameter controls the number of consecutive Transformer layers grouped into a single trainable block. Based on the results in Table 7, where DECA is evaluated with the Llama-3.1-8B model on the NWGI, TFNS, and MNLI classification tasks, we observe that finer-grained partitioning consistently yields better performance. Specifically, when granularity $= 1$, DECA achieves the best results across all datasets and metrics, obtaining an average accuracy of 78.81% and an average F_1 score of 71.98%. As the granularity becomes coarser, the performance gradually decreases: increasing granularity from 1 to 2 reduces the average accuracy from 78.91% to 78.10%,

Table 7: Performance of DECA under different partitioning granularities on Llama-3.1-8B model.

Granularity	NWGI		TFNS		MNLI		Average	
	Acc.	F_1	*Acc.*	F_1	*Acc.*	F_1	*Acc.*	F_1
1	57.24	39.94	90.30	87.44	89.18	88.98	78.91	72.12
2	56.68	36.73	89.94	86.78	87.68	87.49	78.10	70.33
4	54.73	36.49	88.81	85.26	86.58	86.55	76.71	69.43



(a) NWGI dataset (b) TFNS dataset (c) MNLI dataset

Figure 7: Training loss of different partitioning granularities using Llama-3.1-8B model.

while further increasing it to 4 lowers the accuracy to 76.71%. A similar monotonic degradation can also be observed in the average F_1 score, which drops from 72.12% to 70.33% and then to 69.43%. Among the three datasets, NWGI is the most sensitive to coarser partitioning, with its F_1 score decreasing from 39.94% at granularity $= 1$ to 36.49% at granularity $= 4$. These results suggest that, for DECA, a finer model partitioning strategy provides more flexible layer-wise optimization and better preserves task-specific adaptation capability, whereas overly coarse blocks may reduce the effectiveness of localized training and lead to degraded classification performance.

Fig 7 illustrates the loss curves of DECA under different block granularities across three datasets. While all settings exhibit consistent overall convergence trends, the granularity affects the optimization dynamics. Specifically, a finer granularity (e.g., granularity $= 1$) generally achieves slightly lower loss values throughout most of the training process and produces a smoother trajectory with reduced fluctuations, especially in the later stages. In contrast, a coarser granularity (e.g., granularity $= 4$) tends to exhibit more pronounced oscillations and occasional spikes, suggesting higher gradient variance and less stable updates. Overall, finer granularity improves training stability and smoothness by reducing gradient variance, while having minimal impact on convergence.

D.5 Resource Consumption

Table 8 presents a comparative analysis of computation and communication overheads across different methods and model scales. The results show that the proposed DECA framework achieves a clear efficiency advantage in end-to-end training latency by jointly reducing forward computation, backward computation, and aggregation/update overhead. Across all evaluated models, DECA consistently yields the fastest forward propagation among all methods. For example, on Llama-2-7B, DECA reduces the forward time from 323.60 ms in Dec-LoRA to 140.84 ms, achieving a reduction of approximately 56.5%. Similar gains are observed on Qwen2-1.5B, Qwen2.5-3B, and Llama-3.1-8B, where DECA reduces the forward latency by 45.3%, 43.7%, and 49.3% over Dec-LoRA, respectively.

In terms of backward propagation, aggregation and update, DECA also significantly outperforms DeCAF and Dec-LoRA across all model scales. For instance, on Llama-3.1-8B, DECA reduces the backward time from 263.99 ms in Dec-LoRA to 208.97 ms, corresponding to a reduction of about 20.8%. Although Dec-Adapter shows slightly lower backward latency, DECA compensates for this through substantially faster forward propagation, leading to a lower overall computation time for all models. Specifically, on Llama-2-7B, DECA requires only 338.67 ms in a full step, compared with 463.69 ms for Dec-Adapter, 586.99 ms for Dec-LoRA and 3458.85 ms for DeCAF, reducing the end-to-end latency by approximately 27.0%, 42.3% and 90.21%, respectively.

Table 8: Computation and communication overheads across different methods and models. This table summarizes the average forward-propagation latency (**Fwd**), backward-propagation latency (**Bkwd**), aggregation-and-update latency (**Agg & Up**), and the average number of parameters transmitted per communication step by each client (**Comm**).

Models	**Methods**	**Fwd** (ms)	**Bkwd** (ms)	**Agg & Up** (ms)	**Comm** (MB)
Qwen2-1.5B	Dec-Adapter	174.21	115.42	7.11	294.67
	DeCAF	164.79	162.02	1141.70	367.50
	Dec-LoRA	179.30	162.04	19.31	
	DECA	98.15	123.63	3.53	441.25
Qwen2.5-3B	Dec-Adapter	230.12	175.06	9.38	677.49
	DeCAF	232.44	236.66	1637.48	598.50
	Dec-LoRA	264.59	237.51	25.09	
	DECA	149.01	187.20	5.74	720.00
Llama-2-7B	Dec-Adapter	274.96	178.60	10.13	2389.04
	DeCAF	286.62	250.14	2922.09	1722.00
	Dec-LoRA	323.60	241.53	21.86	
	DECA	140.84	184.85	12.98	2100.00
Llama-3.1-8B	Dec-Adapter	258.34	197.21	10.09	2389.04
	DeCAF	249.03	274.29	3054.13	1736.00
	Dec-LoRA	287.17	263.99	21.95	
	DECA	145.51	208.97	13.30	2128.00

Regarding communication overhead, DECA introduces a moderate increase compared with LoRA-based baselines, since it transmits block-wise full-parameter updates rather than only lightweight adapter parameters. Nevertheless, the communication cost remains manageable across all model scales. More importantly, for larger models such as Llama-2-7B and Llama-3.1-8B, DECA requires less communication than Dec-Adapter, reducing the transmitted volume from 2389.04 MB to 2100.00 MB and 2128.00 MB, respectively. Therefore, DECA achieves a favorable trade-off between communication and computation: it slightly increases communication relative to PEFT-style methods, but in return substantially accelerates forward, backward, and aggregation/update stages, making decentralized full-parameter fine-tuning practical and efficient for large language models.

Fig 8 presents the computational overheads associated with different model partitioning granularities. While increasing the granularity from 1 to 4 results in only a moderate increase in backward propagation latency (e.g., rising from 208.97 ms to 256.61 ms for DECA). Fig. 9 compares the memory consumption of Llama-3.1-8B under different partitioning granularities. The results show that finer granularity is significantly more memory-efficient, with DECA requiring only roughly 46–47 GB of peak memory and 29–30 GB of average memory, which is well within the capacity of commodity GPUs (e.g., PRO 6000 with 96 GB of memory). In contrast, increasing the granularity to 4 drives the peak memory usage up to approximately 60 GB, a substantial increase that reflects the cost of storing optimizer states for larger active blocks.

D.6 Hyper-parameter Sensitivity

In this section, we conduct a sensitivity analysis of the hyperparameters in DECA. The Adam hyperparameters α_1 and α_2 control the exponential moving averages of the first- and second-order gradient moments, respectively, and are widely used with standard default settings (Kingma & Ba, 2015). Since the same Adam hyperparameters are adopted for all baselines, we do not further tune them. Instead, we focus on the BMA-specific hyperparameters β_1 and β_2, which control how strongly the consensus-derived discrepancy signal is incorporated into the first- and second-order BMAs. As these parameters determine the balance between local gradient-driven optimization and

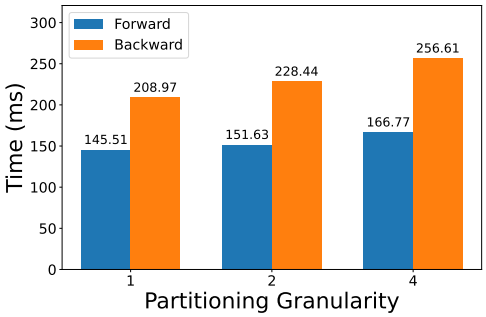


Figure 8: Forward and backward latency under different granularities using Llama-3.1-8B. We measure the average latency per single forward and backward pass over the entire training phase.

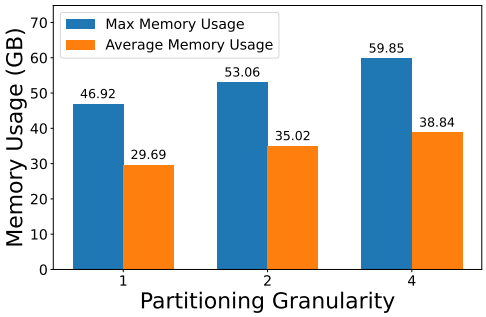


Figure 9: Maximum and average GPU memory usage under different granularities. We measure the maximum and average memory usage during the whole fine-tuning process.

Table 9: Hyperparameter sensitivity on the MNLI classification task using Qwen2.5-3B.

β_1	β_2	Acc.	F1
0.3	0.333	81.59 ± 0.16	79.18 ± 0.24
0.6	0.666	83.81 ± 0.40	82.39 ± 0.38
0.9	0.999	85.24 ± 0.61	84.56 ± 1.10

Table 10: Hyperparameter sensitivity on the Alpaca generation task using Qwen2-1.5B.

β_1	β_2	**VIC.**	**MT-1**	**MT-2**	**MT.**
0.3	0.333	5.68	4.88	3.30	4.10
0.6	0.666	5.63	4.90	3.24	4.07
0.9	0.999	5.60	4.80	3.22	4.01

consensus-guided correction, they may affect the stability and performance of DECA. Therefore, we evaluate DECA under different combinations of β_1 and β_2.

As shown in Table 9- 10, the effect of the BMA hyperparameters exhibits a task-dependent pattern, while DECA maintains stable overall performance across different configurations. For the MNLI classification task, increasing (β_1, β_2) from $(0.3, 0.333)$ to $(0.9, 0.999)$ consistently improves performance. Specifically, the accuracy increases from 81.59% to 85.24%, while the F1 score improves more substantially from 79.18% to 84.56%. These results suggest that larger momentum coefficients can benefit classification, potentially by allowing BMA to retain and exploit historical optimization information more effectively during collaborative training. In contrast, the Alpaca generation task is considerably less sensitive to the choice of (β_1, β_2). As the coefficients increase, the overall MT. score only decreases slightly from 4.10 to 4.01, with similarly small variations in VIC., MT-1, and MT-2. In particular, the performance difference between $(0.3, 0.333)$ and $(0.6, 0.666)$ is marginal, indicating that DECA remains stable over a relatively wide range of BMA hyperparameters for generation tasks. The slightly better generation performance obtained with smaller coefficients suggests that excessive momentum may provide limited additional benefit for instruction-following generation and can mildly weaken the responsiveness to recent optimization updates.

E Comprehensive Literature Survey

E.1 PEFT and FPFT

An effective strategy for LLM fine-tuning is to train a small number of model parameters, while keeping the majority of the pretrained parameters frozen (Houlsby et al., 2019). In particular, adapter tuning only finetunes the inserted small modules between layers called adapters (Houlsby et al., 2019). Prompt-tuning / prefix-tuning (Lester et al., 2021; Li & Liang, 2021) attaches additional trainable prefix tokens to the input and/or hidden layers, while remaining the base model unchanged. Another method is the Low-Rank Adaptation (LoRA) (Hu et al., 2022), which models the increment to the base model as a product of two significantly lower dimensional trainable low-rank matrices. Subsequent related research has aimed at extending its rank constraints (Lialin et al., 2024; Xia et al., 2024), further reducing the number of trainable parameters (Koohpayegani et al., 2023; Kopiczko

et al., 2024), decreasing memory usage through quantization (Dettmers et al., 2023). Presently, LoRA-based methods are commonly employed for LLM fine-tuning with limited memory resources.

To conduct full parameter fine-tuning of LLMs with limited memory, (Lv et al., 2024) proposes LOMO, which efficiently leverages the BP process to update parameters on the fly in the process of computing stochastic gradients. Consequently, LOMO helps to execute SGD for full parameter finetuning without physically storing the stochastic gradients, significantly reducing memory consumption. However, it is worth emphasizing that SGD generally converges more slowly and is often considered suboptimal compared to Adam. MeZO (Malladi et al., 2023) is to approximate SGD by using only the forward pass. The idea of MeZO derives from zeroth-order optimization, which utilizes function value difference to approximate the stochastic gradients of the trainable model parameters. Galore (Zhao et al., 2024) uses gradient low-rank projection, which largely reduces memory consumption for full parameter finetuning compared to Adam. Adam-mini (Zhang et al., 2025) proposes to apply block-wise adaptive learning rate, which reduces the memory for storing the full second moment. Another popular approach for finetuning with limited memory is to perform CPU offloads to reduce the memory consumption caused by training data and optimizers; see, e.g., (Ren et al., 2021; Liu et al., 2024). (Luo et al., 2024) adopts the well-established block coordinate descent (BCD) paradigm (Tseng, 2001; Bengio et al., 2006; Wright, 2015; Belilovsky et al., 2019) to develop a block-wise Adam optimizer for full-parameter LLM fine-tuning. However, this approach is designed for centralized training, where the data are available within a single training infrastructure, and its convergence analysis is established only for the deterministic case. Extending this paradigm to decentralized learning is highly non-trivial: under non-IID data, local block-wise updates may be biased toward heterogeneous client objectives, causing client drift and disrupting consensus. As a result, it remains unclear whether the convergence benefits of BCD can be preserved without central coordination.

E.2 Federated LLM Fine-Tuning

FL has emerged as a pivotal paradigm for training LLMs while preserving data privacy. However, the massive parameter count of modern LLMs introduces prohibitive communication and computational overheads for standard FL protocols. While recent studies suggest that fine-tuning pre-trained language models offers inherent robustness against non-IID data distributions (Qu et al., 2022; Chen et al., 2023; Nguyen et al., 2023; Weller et al., 2022), the resource constraints of edge devices render full-parameter updates often impractical. Consequently, PEFT has become the de facto standard in this domain (Hu et al., 2022; Wei et al., 2025; Lester et al., 2021). Early works utilized adapter-based (Houlsby et al., 2019; Zhang et al., 2024b) or prompt-tuning methods, such as FedPrompt (Zhao et al., 2023), which optimizes soft prompts to reduce transmission costs. Nevertheless, these methods often suffer from accuracy degradation compared to full fine-tuning, particularly under heterogeneous data distributions (Pu et al., 2023).

Among PEFT techniques, Low-Rank Adaptation (LoRA) has gained prominence due to the hypothesis that over-parameterized models reside on a low intrinsic dimension (Li et al., 2018; Aghajanyan et al., 2021). LoRA allows for the efficient merging of trainable matrices with frozen weights, introducing no inference latency. Recent research has focused on optimizing LoRA for federated settings. FDLoRA (Qi et al., 2024) introduces a dual-LoRA mechanism to decouple global knowledge sharing from local personalization. Similarly, FedSA-LoRA (Guo et al., 2025) leverages the asymmetric roles of low-rank matrices, selectively aggregating only the shared A matrices to capture global knowledge while maintaining robustness against data heterogeneity. To accommodate devices with varying capacities, "Heterogeneous LoRA" strategies have been proposed. For instance, (Cho et al., 2024) manages diverse rank allocations via zero-padding and truncation, while RBLA (Chen et al., 2024) and stacking-based methods (Wang et al., 2024) perform weighted aggregation of heterogeneous structures. FFA-LoRA (Sun et al., 2024) further reduces overhead by freezing the randomly initialized A matrices and fine-tuning only the zero-initialized B matrices. However, fixing parameters can impair the model's learning capacity, potentially leading to suboptimal performance (Zhang et al., 2023).

Despite the efficiency of PEFT, the performance gap between partial and full fine-tuning remains a critical bottleneck. Addressing this, recent works have begun to explore communication-efficient FPFT frameworks. FedKSeed (Qin et al., 2024) utilizes zeroth-order optimization to enable billion-scale training by exchanging only random seeds and scalar gradients. Similarly, Ferret (Shu et al.,

2025) leverages low-dimensional projection and shared randomness to substantially mitigate the communication overhead of full-parameter updates. These emerging methods signal a shift toward enabling high-performance FPFT within resource-constrained federated environments.

E.3 Decentralized Learning/Fine-Tuning

Recent years have witnessed significant advancements in the paradigm of decentralized learning. Foundational works such as (Lian et al., 2017) combined *Stochastic Gradient Descent* (SGD) with gossip-averaging (Boyd et al., 2006) to develop *Decentralized Parallel Stochastic Gradient Descent* (DPSGD). Subsequent studies addressed convex optimization with greater theoretical rigor: (Scaman et al., 2017) proposed the *Multi-Step Dual Accelerated* (MSDA) algorithm with a provable linear convergence rate for strongly convex smooth functions, while (Scaman et al., 2018) introduced the *Multi-Step Primal-Dual* (MSPD) algorithm for the more challenging non-smooth case. Furthermore, inspired by the success of momentum in accelerating convergence and improving generalization, (Yu et al., 2019) proposed *Decentralized Momentum Stochastic Gradient Descent* (DMSGD), extending momentum methods to decentralized topologies.

While the aforementioned studies typically assume that training data are independent and identically distributed across clients, this assumption rarely holds in real-world applications. Consequently, significant efforts have been dedicated to addressing data heterogeneity. One line of research focuses on robust optimization techniques, such as (Shi et al., 2023), which employs Sharpness Aware Minimization (SAM) to search for flat minima with uniformly low loss values across clients. Others utilize knowledge transfer, such as (Li et al., 2021), which proposes a mutual knowledge transfer mechanism where subsets of clients exchange local models to mitigate distribution shifts.

To explicitly correct for drift caused by heterogeneity, various tracking mechanisms have been introduced. Gradient tracking (Pu & Nedi, 2021) and momentum tracking (Takezawa et al., 2023) are widely adopted to estimate global gradients locally. Diverging from traditional average-gradient tracking, (Aketi et al., 2024) proposes tracking model updates by having clients store copies of their neighbors' parameters. Similarly, NET-FLEET (Zhang et al., 2022b) incorporates recursive gradient correction to approximate global stochastic gradients efficiently. More complex aggregation strategies include the *Cross Gradient Aggregation* (CGA) algorithm (Esfandiari et al., 2021), which uses quadratic programming (QP) to aggregate cross-gradients (derivatives of a local model on neighbors' data). (Lin et al., 2021) introduces Quasi-Global (QG) momentum for decentralized SGD with momentum, to approximate the global optimization direction at each client. Inspired by this idea, we construct a block-wise discrepancy signal from the gap between the local block parameters and their neighbor-averaged counterparts. Different from QG momentum, DECA incorporates this signal into both the first- and second-order moment estimates of block-wise Adam optimization. This design enables each client to preserve objective-aligned local gradient information while adaptively correcting its update direction according to decentralized consensus, thereby mitigating client drift and stabilizing decentralized block-wise optimization under non-IID data.

While prior studies primarily focus on decentralized learning for small-scale DNNs, only a few have addressed the unique challenges of decentralized LLM training. (Ghiasvand et al., 2025a) proposes Dec-LoRA, a decentralized fine-tuning framework based on LoRA; however, its performance is inherently limited by updating only a small subset of model parameters. (Saadati et al., 2026) further combines decentralized LoRA with truncated singular value decomposition (TSVD)-based matrix factorization to mitigate consensus interference, but this factorization introduces substantial overhead. *To the best of our knowledge, no existing work has established a resource-efficient and high-performance decentralized framework for FPFT, leaving decentralized FPFT an unexplored yet promising direction for achieving stronger task-specific adaptation.*