

ZorBA: Zeroth-order Federated Fine-tuning of LLMs with Heterogeneous Block Activation

Chuiyang Meng*, Ming Tang[†], and Vincent W.S. Wong*

*Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada

[†]Department of Computer Science and Engineering, Southern University of Science and Technology, China

Email: *{chuiyangmeng, vincentw}@ece.ubc.ca, [†]tangm3@sustech.edu.cn

Abstract—Federated fine-tuning of large language models (LLMs) enables collaborative tuning across distributed clients. However, due to the large size of LLMs, local updates in federated learning (FL) may incur substantial video random-access memory (VRAM) usage. Moreover, frequent model exchange may lead to significant communication overhead. To tackle these challenges, in this paper we propose ZorBA, a zeroth-order optimization-based federated fine-tuning framework with heterogeneous block activation. ZorBA leverages zeroth-order optimization to eliminate the storage of gradients at the clients by forward passes. ZorBA includes a heterogeneous block activation mechanism in which the central server allocates different subsets of transformer blocks to clients in order to accelerate the convergence rate and reduce the VRAM usage. Furthermore, ZorBA utilizes shared random seeds and the finite differences of gradients in order to reduce the communication overhead. We conduct theoretical analysis to characterize the effect of block activation decisions on the convergence rate and VRAM usage. To jointly enhance the convergence rate and reduce the VRAM usage, we formulate an optimization problem to optimize the block activation decisions. We propose an ϵ -constraint lexicographic algorithm to solve this problem. Experimental results show that ZorBA outperforms three federated fine-tuning baselines in VRAM usage by up to 62.41% and incurs a low communication overhead.

Index Terms—Large language models, federated fine-tuning, zeroth-order optimization, heterogeneous block activation

I. INTRODUCTION

Large language models (LLMs) [1]–[5] have shown remarkable performance across numerous natural language processing tasks. Fine-tuning techniques further enhance LLMs’ ability to handle specific downstream applications [6]–[8]. Meanwhile, federated learning (FL) [9]–[12] has emerged as a promising paradigm for collaborative training of machine learning models. By allowing multiple resource-constrained clients to locally train the models and aggregate their parameters without sharing raw data, FL facilitates the deployment of large-scale machine learning applications across distributed environments. By combining these two approaches, federated fine-tuning empowers LLMs to adapt to diverse and decentralized datasets across clients [13]–[19].

However, most conventional FL approaches use backpropagation (BP) to compute the first-order gradients for model updates. This leads to two critical challenges for directly applying FL to fine-tune LLMs. First, LLMs typically contain a number of transformer blocks, resulting in model sizes ranging from hundreds of millions to billions of parameters.

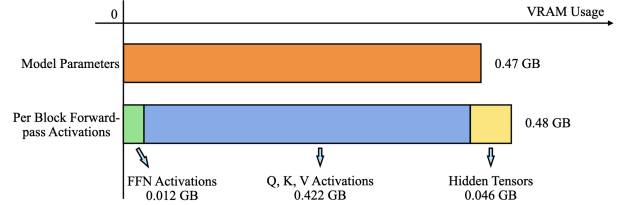


Fig. 1: VRAM usage during zeroth-order optimization with OPT-125M. The forward-pass activations per block include tensors for hidden tensors, projections for Q, K, and V in transformer blocks, and feed-forward networks (FFNs).

Storing the corresponding gradients during fine-tuning requires substantial video random-access memory (VRAM) on graphics processing units (GPUs). This requirement may exceed the VRAM capacities of resource-constrained clients, thus hindering the deployment of FL for LLM fine-tuning. Second, conventional FL cannot be used when the first-order gradients are unavailable, such as in models with non-differentiable operators or in black-box systems.

To address the aforementioned challenges, zeroth-order optimization has emerged as an effective alternative to the conventional first-order FL approaches in LLM fine-tuning [20]–[25]. In particular, zeroth-order optimization replaces BP-based first-order optimization approaches with a forward-pass-only approach (i.e., a BP-free approach). Zeroth-order optimization estimates the true gradients by using finite differences of loss function values generated by random perturbation vectors. Several recent studies have successfully integrated zeroth-order optimization into federated fine-tuning [26]–[31]. Nevertheless, the aforementioned works applied zeroth-order optimization to all blocks, leading to three main limitations. First, as shown in [21], zeroth-order optimization approaches exhibit slower convergence rates when compared with first-order approaches in high-dimensional model parameter space. This is because the perturbation vectors introduce variance on the estimated gradients, which grows with the model dimension. Second, while zeroth-order approaches eliminate the need to store the gradients at the clients, they still require storing forward-pass activations. As shown in Fig. 1, these activations constitute a significant portion of VRAM usage and increase linearly with the number of activated blocks. Third, due to frequent client-server communications in FL, the high dimension of the estimated gradients may result in considerable communication overhead.

Based on the aforementioned discussions, we focus on addressing the following question: *Is there a zeroth-order approach enabling clients to activate fewer blocks while reducing the overhead during client-server communications?*

In response, we aim to develop a federated fine-tuning framework integrating zeroth-order optimization with a selective block activation strategy across distributed clients. Achieving such a goal is challenging due to the following unexplored questions: (i) *How can this framework optimally allocate subsets of blocks to clients in order to jointly improve the convergence rate while reducing the VRAM usage across clients?* (ii) *How can we improve the convergence rate of this federated fine-tuning framework?* (iii) *Can we propose an efficient and scalable algorithm to determine the block activation decisions under multiple clients and blocks?* In this work, we make the following contributions as answers to the aforementioned questions:

- We propose ZorBA, a zeroth-order federated fine-tuning framework with heterogeneous block activation for LLMs. ZorBA reduces the VRAM usage by using zeroth-order optimization. It synchronizes the perturbation vectors across the clients and central server via shared random seeds, which prevents gradient leakage and reduces the communication overhead. ZorBA includes a heterogeneous block activation mechanism, enabling the clients to activate different subsets of blocks based on the convergence rate and VRAM constraints.
- We analyze the convergence rate of ZorBA in the non-convex setting. Through a toy example, we reveal key insights into optimizing block activation decisions. We theoretically quantify how these decisions impact the convergence rate and VRAM usage across clients.
- We formulate a multi-objective optimization problem to jointly improve the convergence rate and reduce VRAM usage across clients. To solve this problem, we propose an ϵ -constraint lexicographic algorithm by decoupling the problem into two subproblems. First, we derive a closed-form expression for the maximal least activated block across all clients. Then, we design a greedy algorithm under VRAM constraints. It activates additional blocks to minimize the number of clients whose least popularity remains at this maximum. We obtain the Pareto front and select the block activation matrix on the front in ZorBA, balancing the convergence rate and total VRAM usage.
- We conduct experiments on AG-News, SST-2, and SNLI datasets by using OPT-125M and OPT-1.3B models, respectively. We compare ZorBA with FedIT [14], FedZO [26], and DeComFL [30]. Results show that ZorBA converges faster than FedZO and DeComFL. Moreover, ZorBA reduces the communication overhead significantly and reduces the total VRAM usage by up to 62.41%.

The rest of this paper is organized as follows. Our proposed ZorBA framework is introduced in Section II. The convergence rate of ZorBA and theoretical insights are presented in Section III. The problem formulation and the proposed algorithms are

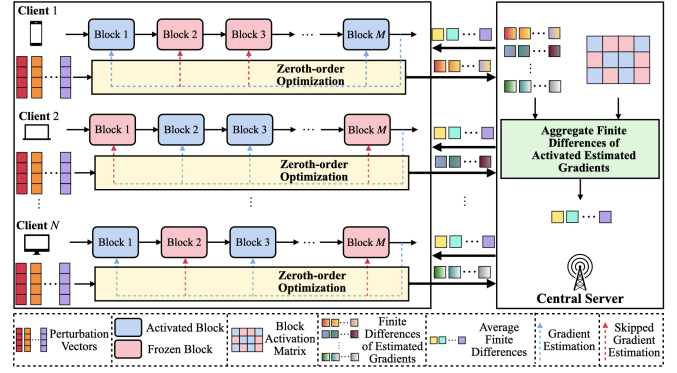


Fig. 2: Illustration of our proposed ZorBA framework.

shown in Section IV. Experimental results are presented in Section V. Conclusions are drawn in Section VI.

Notations: We use italic upper case letters, boldface upper case letters, and boldface lower case letters to denote sets, matrices, and vectors, respectively. $\mathbb{R}^{M \times N}$ denotes the set of real-valued matrices. $\mathbf{I}_{d \times d} \in \mathbb{R}^{d \times d}$ denotes the identity matrix. $\emptyset$ denotes the empty set. Mathematical operators $|\cdot|$, $\mathbb{E}[\cdot]$, $(\cdot)^\top$, $[\cdot; \cdot]$, $\text{d}(\cdot)$, $\langle \cdot, \cdot \rangle$, $[\cdot]_*$, $\|\cdot\|$, and $\|\cdot\|_{op}$ denote the cardinality of a set, expectation, transpose, column-wise concatenation, derivative, inner product, floor function, 2-norm, and operator norm of a matrix, respectively. $\sim$ denotes “distributed as” and $\mathcal{U}(\cdot)$ denotes the uniform distribution.

II. ZORBA

In this section, we present ZorBA, a federated fine-tuning framework. ZorBA (i) enables heterogeneous block activation in which each client updates different subsets of transformer blocks, (ii) incorporates shared random seeds for independent perturbation vector generation between the clients and central server in order to reduce the communication overhead, and (iii) uses zeroth-order optimization to estimate the gradients with only forward passes in order to reduce the VRAM usage.

We consider a central server and N clients. Let $\mathcal{T} = \{1, 2, \dots, T\}$ and $\mathcal{N} = \{1, 2, \dots, N\}$ denote the set of T fine-tuning rounds and the set of N clients, respectively. Let $\mathcal{M} = \{1, 2, \dots, M\}$ denote the set of M transformer blocks of the pretrained model. Note that each transformer block is a matrix of model parameters. For illustration simplicity, we flatten all M blocks of the pretrained model and concatenate them into one vector as $\mathbf{w} = [\mathbf{w}_1; \mathbf{w}_2; \dots; \mathbf{w}_M] \in \mathbb{R}^d$, where d is the model dimension. Let $\mathbf{w}_m$ denote the model parameters of the m -th block, $m \in \mathcal{M}$. We consider that each client $n \in \mathcal{N}$ has a local dataset $\mathcal{D}_n$. Let $\xi_n \sim \mathcal{D}_n$ denote a mini-batch of training samples of $\mathcal{D}_n$. Let $\mathcal{D}$ denote the dataset of all clients. The central server broadcasts $\bar{\mathbf{w}}^1$ to all clients at the beginning of the first fine-tuning round. The ZorBA framework is shown in Fig. 2.

Heterogeneous Block Activation: In practical systems, clients often have limited VRAM when fine-tuning large models. In addition, exchanging the full model update between the clients and central server may result in gradient leakage and

privacy concerns. Therefore, in ZorBA, a heterogeneous block activation mechanism is introduced. In particular, based on the VRAM capacity of each client, the central server activates a subset of blocks for each client to update. Let $\mathbf{a}_n \in \{0, 1\}^M$ denote the block activation decision vector for client n . We have $a_{m,n} = 1$ if the m -th block is activated for client n ; otherwise, $a_{m,n} = 0$. Let $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_N] \in \{0, 1\}^{M \times N}$ denote the block activation allocation decision matrix. Client n utilizes its decision vector $\mathbf{a}_n$ to determine the activated blocks of the local model. At the beginning of the t -th training round, all clients and the central server have the same model, which is obtained at the end of the previous fine-tuning round. It can be expressed in a “block form” as $\bar{\mathbf{w}}^t = [\bar{\mathbf{w}}_1^t; \bar{\mathbf{w}}_2^t; \dots; \bar{\mathbf{w}}_M^t] \in \mathbb{R}^d$. We denote the parameters of the m -th block of client n after block activation as $\hat{\mathbf{w}}_{m,n}^t = a_{m,n} \bar{\mathbf{w}}_m^t, m \in \mathcal{M}, n \in \mathcal{N}$. The activated model parameters of client n in the t -th fine-tuning round are written as $\hat{\mathbf{w}}_n^t = [\hat{\mathbf{w}}_{1,n}^t; \hat{\mathbf{w}}_{2,n}^t; \dots; \hat{\mathbf{w}}_{M,n}^t] = [a_{1,n} \bar{\mathbf{w}}_1^t; a_{2,n} \bar{\mathbf{w}}_2^t; \dots; a_{M,n} \bar{\mathbf{w}}_M^t] \in \mathbb{R}^d$.

Shared Random Seeds: Due to the large model size, exchanging parameters between clients and the central server incurs significant communication overhead. Hence, ZorBA introduces a shared random seed mechanism that replaces model exchange with seed exchange. At the beginning of fine-tuning, the central server initializes a set of P random seeds $\mathcal{S} = \{s_1, s_2, \dots, s_P\}$ and shares it with all clients. These random seeds are used to initialize random number generators which sample perturbation vectors $\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_P\}$ from a standard Gaussian distribution with zero mean and an identity matrix as the covariance matrix [19]. Then, these perturbation vectors are normalized onto the unit sphere. Each perturbation vector in set $\mathcal{V}$ will be used for gradient estimation in zeroth-order estimation. To match the block structure of $\hat{\mathbf{w}}_n^t$, each perturbation vector is written as $\mathbf{v}_p = [\mathbf{v}_{p,1}; \mathbf{v}_{p,2}; \dots; \mathbf{v}_{p,M}] \in \mathbb{R}^d$, where $\mathbf{v}_{p,m}$ is the perturbation vector for the m -th block initialized by random seed s_p . Since all clients and the central server share the same set of random seeds, they can independently generate the same perturbation vectors based on the exchanged random seeds.

Fine-tuning Process: Let $F_n(\hat{\mathbf{w}}_n^t; \xi_n)$ denote the local loss function of client $n \in \mathcal{N}$ on ξ_n with model $\hat{\mathbf{w}}_n^t$. In the t -th fine-tuning round, the central server randomly selects a subset of Q seeds from set $\mathcal{S}$ and sends them to all clients. Each client then determines the corresponding set of Q perturbation vectors, denoted by $\mathcal{Q}^t \subseteq \mathcal{V}$, for zeroth-order optimization. Client $n \in \mathcal{N}$ determines $F_n(\hat{\mathbf{w}}_n^t; \xi_n)$, and then estimates the gradient of $\hat{\mathbf{w}}_n^t$ via zeroth-order optimization by Q trials, each corresponding to a perturbation vector. Note that client n estimates only the gradients of model parameters in the activated blocks. For the frozen blocks, the corresponding gradient estimation is skipped. In each trial, client n estimates the gradient by performing two forward passes that evaluate the local loss at the current point and at a perturbed point. This is done by adding a sampled perturbation vector $\mathbf{v}_q \in \mathcal{Q}^t$, scaled by a small smoothing constant μ , to the activated model parameters $\hat{\mathbf{w}}_n^t$. The constant μ controls the finite difference of the perturbation applied for gradient estimation.

Then, client n calculates two losses as $F_n(\hat{\mathbf{w}}_n^t + \mu \mathbf{v}_q; \xi_n)$ and $F_n(\hat{\mathbf{w}}_n^t; \xi_n)$, respectively. The finite difference of the estimated gradient of client n in the direction of the perturbation vector $\mathbf{v}_q$ in the t -th fine-tuning round is given by $\rho_{\mathbf{v}_q,n}^t = \frac{F_n(\hat{\mathbf{w}}_n^t + \mu \mathbf{v}_q; \xi_n) - F_n(\hat{\mathbf{w}}_n^t; \xi_n)}{\mu}$. By averaging all Q trials, the estimated gradient of the m -th block of client n in the t -th fine-tuning round can be determined by

$$\tilde{\nabla}_{\hat{\mathbf{w}}_{m,n}^t} F_n(\hat{\mathbf{w}}_n^t; \xi_n) = \frac{1}{Q} \sum_{\mathbf{v}_q \in \mathcal{Q}^t} \rho_{\mathbf{v}_q,n}^t \mathbf{v}_{q,m}. \quad (1)$$

After finishing local updates, the clients send the model update to the central server for aggregation. Instead of directly transmitting the full estimated gradient, client n transmits the finite differences of the estimated gradients corresponding to the sampled perturbation vectors. Since both the model $\bar{\mathbf{w}}^t$ and the sampled perturbation vector subset $\mathcal{Q}^t$ are identical between the clients and central server, transmitting only the finite difference of the estimated gradient is sufficient for the server to determine the estimated gradient. Upon receiving the finite differences of the estimated gradients from all clients, the central server updates the global model based on the estimated gradients over the activated blocks. In particular, the m -th block is updated as follows:

$$\begin{aligned} \bar{\mathbf{w}}_m^{t+1} &= \sum_{n \in \mathcal{N}} \frac{a_{m,n}}{\sum_{n' \in \mathcal{N}} a_{m,n'}} \left(\bar{\mathbf{w}}_m^t - \eta^t \tilde{\nabla}_{\hat{\mathbf{w}}_{m,n}^t} F_n(\hat{\mathbf{w}}_n^t; \xi_n) \right) \\ &= \bar{\mathbf{w}}_m^t - \eta^t \left(\frac{1}{Q} \sum_{n \in \mathcal{N}} \frac{a_{m,n}}{\sum_{n' \in \mathcal{N}} a_{m,n'}} \sum_{\mathbf{v}_q \in \mathcal{Q}^t} \rho_{\mathbf{v}_q,n}^t \mathbf{v}_{q,m} \right) \\ &= \bar{\mathbf{w}}_m^t - \eta^t \sum_{\mathbf{v}_q \in \mathcal{Q}^t} \bar{\rho}_{\mathbf{v}_q}^t \mathbf{v}_{q,m}, \quad m \in \mathcal{M}, \end{aligned} \quad (2)$$

where η^t denotes the learning rate in the t -th fine-tuning round. $\bar{\rho}_{\mathbf{v}_q}^t = \frac{1}{Q} \sum_{n \in \mathcal{N}} \frac{a_{m,n}}{\sum_{n' \in \mathcal{N}} a_{m,n'}} \rho_{\mathbf{v}_q,n}^t$ denotes the average finite difference of estimated gradients in the direction of $\mathbf{v}_q$. Let $\bar{\mathcal{P}}^t = \{\bar{\rho}_{\mathbf{v}_q}^t : \mathbf{v}_q \in \mathcal{Q}^t\}$ denote the set of average finite differences for all perturbation vectors in the t -th fine-tuning round. The aggregated model is updated by $\bar{\mathbf{w}}^{t+1} = [\bar{\mathbf{w}}_1^{t+1}; \bar{\mathbf{w}}_2^{t+1}; \dots; \bar{\mathbf{w}}_M^{t+1}]$.

Then, the central server broadcasts the set of average finite differences of estimated gradients $\bar{\mathcal{P}}^t$ to all clients. By using the shared random seeds, the server no longer broadcasts high-dimensional parameters and reduces the communication overhead significantly. Since both the model $\bar{\mathbf{w}}^t$ and the sampled perturbation vector subset $\mathcal{Q}^t$ are shared, upon receiving $\bar{\mathcal{P}}^t$, each client can obtain the model for the $(t+1)$ -th fine-tuning round as $\bar{\mathbf{w}}^{t+1}$ by using eqn. (2).

VRAM Usage Model: In zeroth-order optimization, the VRAM usage consists of the storage for both model parameters and forward-pass activations of all blocks. We characterize the total VRAM usage as the sum of these two components. We define $\psi_{\max,n}$ as the VRAM capacity for client $n \in \mathcal{N}$.

The VRAM usage for model parameters is identical across clients, as they share the same pre-trained model architecture. Thus, for any client $n \in \mathcal{N}$ and fine-tuning round $t \in \mathcal{T}$, the VRAM usage for model parameters is denoted as ψ_{md} .

Let B , L , H , and K denote the batch size, input size, hidden size, and number of attention heads, respectively. The VRAM usage for forward-pass activations of each block consists of three parts: (i) hidden state activations: $\psi_{\text{hd}} = BLH$; (ii) activations of Q, K, and V modules in the transformer block: $\psi_{\text{qkv}} = 3KBLH$; (iii) activations of feed-forward network (FFN): $\psi_{\text{ffn}} = \alpha BLH$, where α is the FFN expansion ratio for the model. The VRAM usage for each block is given by $\psi_{\text{block}} = \psi_{\text{hd}} + \psi_{\text{qkv}} + \psi_{\text{ffn}} = (\alpha + 3K + 1)BLH$. ψ_{block} models the VRAM usage to store the intermediate states of the blocks at client n . We adopt an additive form since zeroth-order training evaluates multiple perturbations per round and reuses these stored states across evaluations. For client $n \in \mathcal{N}$ in each fine-tuning round, the VRAM usage of the total activations is the VRAM required to store the activations of all activated blocks, which is given by $\psi_{\text{act},n}(\mathbf{a}_n) = (\alpha + 3K + 1)BLH \sum_{m \in \mathcal{M}} a_{m,n}$. Thus, the total VRAM usage of client $n \in \mathcal{N}$ is $\psi_{\text{tot},n}(\mathbf{a}_n) = \psi_{\text{act},n}(\mathbf{a}_n) + \psi_{\text{md}}$.

III. THEORETICAL ANALYSIS OF ZORBA

In this section, we analyze how the block activation matrix affects the convergence rate of our proposed ZorBA. Without loss of generality, we conduct analysis under nonconvex loss functions. We denote the expected loss of client n with model $\mathbf{w}_n^t$ as $f_n(\mathbf{w}_n^t) = \mathbb{E}_{\xi_n \sim \mathcal{D}_n} F_n(\mathbf{w}_n^t; \xi_n)$. We denote the global loss function as $f(\mathbf{w}^t)$. We denote the gradient of $f(\mathbf{w}^t)$ as $\nabla f(\mathbf{w}^t) = \mathbb{E}_{n \in \mathcal{N}} \nabla f_n(\mathbf{w}^t)$. We first present the following assumptions which are widely used in the literature (e.g., [12], [32], [33]).

Assumption 1. *The loss function of each client $n \in \mathcal{N}$ is continuously differentiable and L -smooth. That is, for arbitrary two vectors $\mathbf{w}_n^t$ and $\tilde{\mathbf{w}}_n^t$, we have $f_n(\mathbf{w}_n^t) \leq f_n(\tilde{\mathbf{w}}_n^t) + \langle \nabla f_n(\tilde{\mathbf{w}}_n^t), \mathbf{w}_n^t - \tilde{\mathbf{w}}_n^t \rangle + \frac{L}{2} \|\mathbf{w}_n^t - \tilde{\mathbf{w}}_n^t\|^2$.*

Assumption 2. *The variance of the local stochastic gradient of each client $n \in \mathcal{N}$ is upper-bounded, i.e., $\mathbb{E}_{\xi_n \sim \mathcal{D}_n} [\|\nabla F_n(\mathbf{w}_n^t; \xi_n) - \nabla f_n(\mathbf{w}_n^t)\|^2] \leq \sigma^2$.*

Assumption 3. *The dissimilarity between the local gradient of each client $n \in \mathcal{N}$ and the gradient of the global loss function is upper-bounded, i.e., $\|\nabla f(\mathbf{w}^t) - \nabla f_n(\mathbf{w}^t)\|^2 \leq \sigma_G^2$.*

Moreover, we introduce two lemmas to facilitate our convergence analysis.

Lemma 1. *Let $\mathbb{E}_{\xi_n \sim \mathcal{D}_n} [\tilde{\nabla} F_n(\mathbf{w}^t; \xi_n)] = \tilde{\nabla} f_n(\mathbf{w}^t)$, $n \in \mathcal{N}$. The dissimilarity between the zeroth-order and first-order gradient estimation of each client $n \in \mathcal{N}$ is upper-bounded, i.e., $\|\tilde{\nabla} f_n(\mathbf{w}^t) - \nabla f_n(\mathbf{w}^t)\| \leq \frac{1}{2} \mu L(d+3)^{\frac{3}{2}}$. This result follows from the analysis in [34].*

Lemma 2. *When μ is small enough (i.e., $\mu \rightarrow 0$), the square norm of the zeroth-order gradient estimation of each client $n \in \mathcal{N}$ satisfies $\mathbb{E}_{\xi_n \sim \mathcal{D}_n} [\|\tilde{\nabla} F_n(\mathbf{w}_n^t; \xi_n)\|^2] = \frac{d+Q-1}{Q} \mathbb{E}_{\xi_n \sim \mathcal{D}_n} [\|\nabla F_n(\mathbf{w}_n^t; \xi_n)\|^2]$. This results from [21].*

We denote $\Lambda(\mathbf{A}) = \sum_{n \in \mathcal{N}} \max_{m \in \mathcal{M}} \left\{ \left(\frac{a_{m,n}}{\sum_{n' \in \mathcal{N}} a_{m,n'}} \right)^2 \right\}$. Let $\mathbf{w}^*$ denote the optimal model. We present the convergence bound of ZorBA.

Theorem 1. (Standard convergence bound of ZorBA) *Under Assumptions 1–3 and Lemmas 1–2, if the learning rate η^t is*

small such that $\eta^t = \eta \leq \frac{1}{2LdN^2}$, the standard convergence bound of our proposed ZorBA satisfies:

$$\frac{1}{T} \sum_{t=1}^T \|\nabla f(\bar{\mathbf{w}}^t)\|^2 \leq \frac{1}{\Omega(\Lambda(\mathbf{A}))T} \sum_{t=1}^T (f(\bar{\mathbf{w}}^1) - f(\mathbf{w}^*)) + \frac{\Theta_1}{\Omega(\Lambda(\mathbf{A}))} \Lambda(\mathbf{A}), \quad (3)$$

where $\Omega(\Lambda(\mathbf{A})) = \left(\frac{\eta}{2} - \frac{3L\eta^2(d+Q-1)N}{2Q} \Lambda(\mathbf{A}) \right)$ and $\Theta_1 = \frac{1}{4} \mu^2 L^2 \eta (d+3)^3 N + \left(\eta N + \frac{3L\eta^2(d+Q-1)N}{2Q} \right) \sigma_G^2 + \frac{3L\eta^2(d+Q-1)N}{2Q} \sigma^2$.

Sketch of proof. The proof starts by showing the update of one fine-tuning round. By using Assumption 1, we expand the expected loss of the average model into two terms: a first-order descent term T_1 and a second-order variance term T_2 . T_1 captures the inner product between the true gradient and the model update made with zeroth-order optimization. In particular, T_1 is upper-bounded by using the bounded stochastic-gradient variance (Assumption 2), the gradient-dissimilarity assumption (Assumption 3), and the zeroth-order estimation dissimilarity in Lemma 1, giving a bias that depends on the block-activation decisions $\mathbf{A}$. T_2 is bounded by using Lemma 2 and Assumptions 2–3. Lemma 2 draws a connection between the zeroth-order gradients to first-order ones. Assumptions 2–3 characterize the gradient variance and data heterogeneity, respectively. By summing over all fine-tuning rounds, rearranging the inequality, and choosing the proper learning rate, the convergence bound is obtained. $\square$

According to Theorem 1, the convergence bound of ZorBA consists of two terms: an optimality gap that diminishes as the number rounds T increases, and a non-diminishing bias term (i.e., $\frac{\Theta_1}{\Omega(\Lambda(\mathbf{A}))} \Lambda(\mathbf{A})$). The bias term quantifies the impact of zeroth-order gradient estimation error, data heterogeneity, and local gradient variance on the convergence.

However, to guarantee convergence of ZorBA, Theorem 1 requires a learning rate $\eta = \mathcal{O}(\frac{1}{dN^2})$. In practical systems, the model dimension d can exceed billions, and the number of clients N may be large. Thus, the required learning rate may be impractically small. To overcome this challenge, we introduce a condition and adopt a supporting lemma from [21], allowing us to derive a convergence bound independent of d .

Condition 1. *Let $G(\mathbf{w}^t) = \max_{n \in \mathcal{N}} \max_{\xi_n \sim \mathcal{D}_n} \|\nabla F_n(\mathbf{w}_n^t; \xi_n)\|$, there exists a Hessian matrix $\mathbf{H}(\mathbf{w}^t) \preceq L \mathbf{I}_{d \times d}$ such that for arbitrary $\mathbf{w} \in \mathbb{R}^d$ with $\|\mathbf{w} - \mathbf{w}^t\| \leq \eta^t d G(\mathbf{w}^t)$, we have $\nabla^2 f(\mathbf{w}) \preceq \mathbf{H}(\mathbf{w}^t)$. The effective rank of $\mathbf{H}(\mathbf{w}^t)$ satisfies $\text{erank}(\mathbf{H}(\mathbf{w}^t)) = \frac{\text{Tr}(\mathbf{H}(\mathbf{w}^t))}{\|\mathbf{H}(\mathbf{w}^t)\|_{\text{op}}} \leq \kappa$. This is from [21].*

Condition 1 shows that the Hessian of local loss can be approximated by a matrix such as its curvature is dominated by only κ effective directions. Hence, the convergence bound depends on κ instead of d .

Lemma 3. *Let $F(\mathbf{w}; \xi)$ denote the global stochastic loss such that $f(\mathbf{w}) = \mathbb{E}_{\xi \sim \mathcal{D}} [F(\mathbf{w}; \xi)]$. We define $\Sigma(\mathbf{w}) = \mathbb{E} [\nabla F(\mathbf{w}; \xi) (\nabla F(\mathbf{w}; \xi))^T] - \nabla f(\mathbf{w}) (\nabla f(\mathbf{w}))^T$. The outer*

product of the global model difference between two consecutive rounds satisfies

$$\begin{aligned} & \mathbb{E} [(\bar{\mathbf{w}}^{t+1} - \bar{\mathbf{w}}^t)(\bar{\mathbf{w}}^{t+1} - \bar{\mathbf{w}}^t)^\top] \\ &= \left(1 + \frac{d-2}{d+2}\right)(\eta^t)^2 N \left(\nabla f(\bar{\mathbf{w}}^t) (\nabla f(\bar{\mathbf{w}}^t))^\top + \frac{1}{N} \Sigma(\bar{\mathbf{w}}^t) \right) \\ &+ \frac{d}{d+2} (\eta^t)^2 N \left(\|\nabla f(\bar{\mathbf{w}}^t)\|^2 + \frac{1}{N} \text{Tr}(\Sigma(\bar{\mathbf{w}}^t)) \right) \mathbf{I}_{d \times d}. \end{aligned} \quad (4)$$

This result follows from [21].

Then, we present the dimension-free convergence bound of our proposed ZorBA.

Theorem 2. (Dimension-free convergence bound of ZorBA) *Under Assumptions 1–3, Lemma 1, Condition 1, and Lemma 3, if the learning rate η^t satisfies $\eta^t = \eta \leq \frac{d+2}{2LdN^2(\kappa+2)}$, the dimension-free convergence bound of our proposed ZorBA is:*

$$\begin{aligned} \frac{1}{T} \sum_{t=1}^T \|\nabla f(\bar{\mathbf{w}}^t)\|^2 &\leq \frac{1}{\Phi(\Lambda(\mathbf{A}))T} (f(\bar{\mathbf{w}}^1) - f(\bar{\mathbf{w}}^*)) \\ &+ \underbrace{\frac{\Theta_2}{\Phi(\Lambda(\mathbf{A}))}}_{\text{bias term}} \Lambda(\mathbf{A}), \end{aligned} \quad (5)$$

where $\Phi(\Lambda(\mathbf{A})) = \left(\frac{\eta}{2} - \frac{d\eta^2 NL(\kappa+2)}{2(d+2)} \Lambda(\mathbf{A})\right)$ and $\Theta_2 = \eta N (\sigma_G^2 + \frac{1}{4} \mu^2 L^2 (d+3)^3) + \frac{d\eta^2 L(\kappa+2)\sigma^2}{2(d+2)}$.

Sketch of proof. To remove the dependency on the model dimension d , we expand the expected loss of the average model to second order and invoke the local κ -effective-rank condition on the Hessian (Condition 1). Lemma 3 shows the outer product of model differences, which further connects the Hessian term to the gradient norm and a trace term without d . The key insight is bounding this Hessian term by leveraging the gradient covariance matrix. The resulting convergence bound has a similar structure to Theorem 1, but with a new bias term which incorporates κ and eliminates the learning rate's dependence on d . $\square$

Theorem 2 introduces a non-diminishing bias term which reflects the impact of zeroth-order gradient error, data heterogeneity, and local gradient variance while ensuring the learning rate is independent of model dimension.

To improve convergence, it is essential to design the block activation matrix $\mathbf{A}$ to minimize this bias. Although the involved constants (e.g., L , κ , σ_G , σ) are generally intractable, Lemma 4 shows that accelerating the convergence is equivalent to minimizing $\Lambda(\mathbf{A})$, regardless of these constants.

Lemma 4. (Monotonicity of bias term) *The optimal block activation matrix $\mathbf{A}$ which minimizes the bias term $\frac{\Theta_2}{\Phi(\Lambda(\mathbf{A}))} \Lambda(\mathbf{A})$ is equivalent to the minimizer of $\Lambda(\mathbf{A})$.*

Proof. The first-order derivative of the bias term $\frac{\Theta_2}{\Phi(\Lambda(\mathbf{A}))} \Lambda(\mathbf{A})$ with respect to (w.r.t.) $\Lambda(\mathbf{A})$ can be derived as

$$\frac{d \frac{\Theta_2}{\Phi(\Lambda(\mathbf{A}))} \Lambda(\mathbf{A})}{d\Lambda(\mathbf{A})} = \frac{\Theta_2 \Phi(\Lambda(\mathbf{A})) + \frac{d\eta^2 NL(\kappa+2)}{2(d+2)} \Theta_2 \Lambda(\mathbf{A})}{(\Phi(\Lambda(\mathbf{A})))^2}. \quad (6)$$

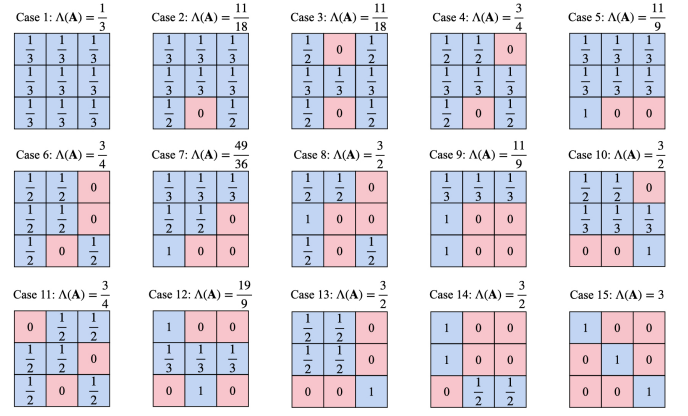


Fig. 3: An example of different block activation decisions on three clients. We consider that each client's model has three blocks. Each column denotes the blocks for each client. The blue square denotes the activated block. The pink square denotes the frozen block. The value of the block in the m -th row and n -th column corresponds to the aggregation weight, i.e., $\frac{a_{m,n}}{\sum_{n' \in \mathcal{N}} a_{m,n'}}$.

In particular, Θ_2 and $\Lambda(\mathbf{A})$ are both positive. When $\eta \leq \frac{d+2}{2LdN^2(\kappa+2)}$, $\Phi(\Lambda(\mathbf{A}))$ is non-negative. The bias term is monotonically increasing w.r.t. $\Lambda(\mathbf{A})$ and is minimized at the lower bound of $\Lambda(\mathbf{A})$. $\square$

To further demonstrate the impact of the block activation matrix $\mathbf{A}$ on the convergence rate, we present an example in Fig. 3. Based on Lemma 4, we use $\Lambda(\mathbf{A})$ to characterize the bias term and the convergence rate. We present several important observations as follows.

Observation 1. *At a large scale, increasing the number of activated blocks generally reduces Λ , thereby leading to faster convergence. The optimal convergence rate is achieved when all clients activate all the blocks, i.e., full-block fine-tuning.*

Observation 2. *At a small scale, activating more blocks does not always improve the convergence.*

Observation 3. *In some cases, activating more blocks can degrade the convergence. For example, case 5 activates one more block than case 11, but case 11 has a lower value of Λ .*

Observation 4. *The convergence rate can vary even when the total number of activated blocks remains the same. For example, both cases 8 and 9 activate the same total number of blocks. Although case 9 has a more imbalanced block allocation, it achieves a lower value of Λ .*

Based on the above observations, we provide insights into minimizing $\Lambda(\mathbf{A})$. We define $c_m(\mathbf{a}_{m,:}) = \sum_{n \in \mathcal{N}} a_{m,n}$ as the popularity of block m . It is the number of clients that activate block m . For each client n , we define its least popularity as the minimum popularity among all blocks activated by client n . The least popularity of client n satisfies $\underline{c}_n(\mathbf{A}) = \min_{m: a_{m,n}=1} c_m(\mathbf{a}_{m,:}) \geq 1$. Let $\underline{\mathbf{c}}(\mathbf{A}) = [\underline{c}_1, \underline{c}_2, \dots, \underline{c}_N] \in \mathbb{R}^N$ denote the vector of the least popularity of all clients. In particular, $\Lambda(\mathbf{A})$ satisfies $\Lambda(\mathbf{A}) = \sum_{n \in \mathcal{N}} \frac{1}{\underline{c}_n^2(\mathbf{A})}$.

To analyze the optimization of the convergence rate, we start by showing Schur-convexity and majorization.

Lemma 5. (Schur-convexity and majorization of $\Lambda(\mathbf{A})$) *We define $\mathbf{l}(\mathbf{A}) = \text{sort}^\downarrow(\underline{\mathbf{c}}(\mathbf{A}))$ as a vector of the least populari-*

ties of all clients in a descending order. If there exists another block activation matrix $\mathbf{A}'$ such that $\mathbf{l}(\mathbf{A})$ is majorized by $\mathbf{l}(\mathbf{A}')$, we have $\Lambda(\mathbf{A}) < \Lambda(\mathbf{A}')$.

Proof. We start by recalling majorization and Schur-convexity. We define two descending vectors $\mathbf{x}^\downarrow, \mathbf{y}^\downarrow \in \mathbb{R}^N$, where $x_n^\downarrow$ and $y_n^\downarrow$ denote the n -th largest value of $\mathbf{x}^\downarrow$ and $\mathbf{y}^\downarrow$, respectively. If for any $k = 1, 2, \dots, N$, $\mathbf{x}^\downarrow$ and $\mathbf{y}^\downarrow$ satisfy $\sum_{n=1}^k x_n^\downarrow \leq \sum_{n=1}^k y_n^\downarrow$ and $\sum_{n \in \mathcal{N}} x_n^\downarrow = \sum_{n \in \mathcal{N}} y_n^\downarrow$, we say $\mathbf{x}^\downarrow$ is majorized by $\mathbf{y}^\downarrow$, i.e., $\mathbf{x}^\downarrow \prec \mathbf{y}^\downarrow$. Then, we introduce Schur-convexity. We say a function $f : \mathbb{R}^N \rightarrow \mathbb{R}$ is Schur-convex if $\mathbf{x}^\downarrow \prec \mathbf{y}^\downarrow$, $f(\mathbf{x}^\downarrow) \leq f(\mathbf{y}^\downarrow)$. Due to the property of Schur-convexity, if f is a convex function, then $\sum_{n \in \mathcal{N}} f(x_n^\downarrow)$ is Schur-convex.

In our settings, $\frac{1}{c_n^2(\mathbf{A})}$ is convex and monotonically decreasing w.r.t. $c_n(\mathbf{A})$ for any $n \in \mathcal{N}$, $\Lambda(\mathbf{A}) = \sum_{n \in \mathcal{N}} \frac{1}{c_n^2(\mathbf{A})}$ is Schur-convex. Then, based on the definition of majorization, we have $\Lambda(\mathbf{A}) < \Lambda(\mathbf{A}')$ if $\mathbf{l}(\mathbf{A}) \prec \mathbf{l}(\mathbf{A}')$. $\square$

Lemma 5 provides key insight into optimizing the convergence rate, which is summarized in the following theorem.

Theorem 3. (Dominance of the least popular blocks) *The value of $\Lambda(\mathbf{A})$ depends on $\mathbf{l}(\mathbf{A})$. Minimizing $\Lambda(\mathbf{A})$ is equivalent to maximizing the least popularity of all clients (i.e., $\max_{\mathbf{A}} \min_{n \in \mathcal{N}} \{c_n(\mathbf{A})\}$) and then minimizing the number of clients which obtain this minimum.*

Proof. Since $\Lambda(\mathbf{A})$ is Schur-convex in the sorted vector of least popularities $\mathbf{l}(\mathbf{A})$, Lemma 5 shows that minimizing $\Lambda(\mathbf{A})$ is achieved by flattening $\mathbf{l}(\mathbf{A})$ as much as possible. Flattening occurs first by maximizing the minimal value across $c_n(\mathbf{A}), n \in \mathcal{N}$ (i.e., maximizing the least popularity across all clients). Then, with that minimum fixed, we reduce the number of clients which obtain this minimum. $\square$

In addition, we present the following proposition.

Proposition 1. (Λ -optimal multiplicity) $\Lambda(\mathbf{A})$ depends only on the multiset of $\mathbf{c}(\mathbf{A})$. Moreover, $\Lambda(\mathbf{A})$ is permutation invariant to $\mathbf{c}(\mathbf{A})$. That is, for any permutation π , if two block activation matrices $\mathbf{A}$ and $\mathbf{A}'$ satisfy $\mathbf{c}(\mathbf{A}') = \pi(\mathbf{c}(\mathbf{A}))$, they achieve a similar convergence rate.

Proof. The value of $\Lambda(\mathbf{A})$ depends only on the multiset of least popularities $\{c_1(\mathbf{A}), c_2(\mathbf{A}), \dots, c_N(\mathbf{A})\}$. The ordering of its elements plays no role because $\Lambda(\mathbf{A}) = \sum_{n \in \mathcal{N}} \frac{1}{c_n^2(\mathbf{A})}$ is a symmetric sum. Therefore, for any permutation π , replacing $\mathbf{c}(\mathbf{A})$ with $\pi(\mathbf{c}(\mathbf{A}))$ leaves the value of the sum unchanged. $\square$

The above analysis uncovers a counterintuitive insight, which is summarized in the following remark.

Remark 1. *While the total number of activated blocks affects the convergence rate, the factor that controls the convergence is how the least popularities are distributed across clients.*

In general, increasing the total number of activated blocks across all clients enhances the convergence rate at the cost of higher VRAM usage. This reveals a fundamental trade-off between convergence performance and VRAM usage: activating more blocks accelerates optimization but demands more VRAM resources on each client.

IV. PROBLEM FORMULATION AND ϵ -CONSTRAINT LEXICOGRAPHIC ALGORITHM

A. Problem Formulation

Based on the aforementioned theoretical analysis, we aim to jointly minimize $\Lambda(\mathbf{A})$ to improve the convergence rate of ZorBA and reduce each client's VRAM usage. We formulate the problem as follows:

$$\mathcal{P}_1 : \underset{\mathbf{A}}{\text{minimize}} \quad \Lambda(\mathbf{A}), \psi_{\text{tot},1}(\mathbf{a}_1), \psi_{\text{tot},2}(\mathbf{a}_2), \dots, \psi_{\text{tot},N}(\mathbf{a}_N) \quad (7a)$$

$$\text{subject to} \quad \sum_{m \in \mathcal{M}} a_{m,n} \geq 1, \quad n \in \mathcal{N}, \quad (7b)$$

$$\sum_{n \in \mathcal{N}} a_{m,n} \geq 1, \quad m \in \mathcal{M}, \quad (7c)$$

$$a_{m,n} \in \{0, 1\}, \quad m \in \mathcal{M}, n \in \mathcal{N}, \quad (7d)$$

where constraint (7b) ensures that each client activates and updates at least one block for fine-tuning. Constraint (7c) ensures that each block is activated and updated by at least one client. Constraint (7d) ensures that the block activation decisions are binary variables. The challenges of solving $\mathcal{P}_1$ are twofold. First, $\mathcal{P}_1$ is an NP-hard integer programming problem. Second, $\mathcal{P}_1$ contains multiple objectives which depend on the block activation decisions. To address these challenges, we propose an ϵ -constraint lexicographic algorithm to achieve a close-to-optimal solution in the following subsection.

B. ϵ -constraint Lexicographic Algorithm

We adopt the ϵ -constraint method [35] to transform the original multi-objective problem into a single-objective optimization problem, which allows us to balance the convergence performance with VRAM constraints. In particular, for each client n , we denote the VRAM usage reduction as $\epsilon_n = \tau \psi_{\text{max},n}$, where τ denotes the desired reduction ratio in VRAM usage. To explore a diverse set of trade-offs, we sample τ independently from a uniform distribution as $\tau \sim \mathcal{U}[0, 1]$. Let $\epsilon = [\epsilon_1, \epsilon_2, \dots, \epsilon_N] \in \mathbb{R}^N$ denote the resulting VRAM reduction vector for all clients. We generate E such vectors and collect them in the set $\mathcal{E}$, which is used to evaluate the candidate solutions under different VRAM reduction scenarios across clients. We solve $\mathcal{P}_1$ for each reduction vector $\epsilon \in \mathcal{E}$. Under a given ϵ , problem $\mathcal{P}_1$ is reformulated as

$$\mathcal{P}_2 : \underset{\mathbf{A}}{\text{minimize}} \quad \Lambda(\mathbf{A}) \quad (8a)$$

$$\text{subject to} \quad \psi_{\text{tot},n}(\mathbf{a}_n) = \psi_{\text{max},n} - \epsilon_n, \quad n \in \mathcal{N}, \quad (8b)$$

constraints (7b)–(7d).

Problem $\mathcal{P}_2$ is difficult to solve since matrix $\mathbf{A}$ is a binary matrix with dimension $M \times N$. For each ϵ , solving problem $\mathcal{P}_2$ incurs an exponential computation complexity of up to $\mathcal{O}(2^{MN})$, which is not scalable as M and N grow.

To reduce the complexity, we propose a lexicographic optimization algorithm. Based on Theorem 3, we equivalently transform $\mathcal{P}_2$ into two subproblems, namely $\mathcal{P}_I$ and $\mathcal{P}_{II}$.

Problem $\mathcal{P}_I$ aims to maximize the least popularity of all clients. Let γ^* denote the optimal objective value to $\mathcal{P}_I$. Problem $\mathcal{P}_{II}$ is the least popularity adjustment problem. The objective is to minimize the number of clients whose least popularity remains γ^* after additional block activation.

Least popularity maximization problem $\mathcal{P}_I$: $\mathcal{P}_I$ is formulated as follows:

$$\begin{aligned} \mathcal{P}_I : \quad & \underset{\mathbf{A}}{\text{maximize}} \quad \min_{n \in \mathcal{N}} \{c_n(\mathbf{A})\} \\ & \text{subject to} \quad \text{constraints (7b)–(7d), (8b)}. \end{aligned} \quad (9a)$$

The optimal objective value of $\mathcal{P}_I$ is given by Theorem 4.

Theorem 4. (Optimal least popularity) *Let $\mathcal{B}$ denote any non-empty subset of blocks, i.e., $\mathcal{B} \subseteq \mathcal{M}$ and $\mathcal{B} \neq \emptyset$. γ^* satisfies*

$$\gamma^* = \min_{\mathcal{B}} \left\{ \left\lfloor \frac{\sum_{n \in \mathcal{N}} \min \left\{ \frac{\psi_{\max, n} - \psi_{\min} - \epsilon_n}{(\alpha + 3K + 1)BLH}, |\mathcal{B}| \right\}}{|\mathcal{B}|} \right\rfloor \right\}. \quad (10)$$

Proof. For client $n \in \mathcal{N}$, given its VRAM capacity, the maximum block activation budget is given by $\lambda_n^* = \frac{\psi_{\max, n} - \psi_{\min} - \epsilon_n}{(\alpha + 3K + 1)BLH}$. We consider that each block in the subset $\mathcal{B} \subseteq \mathcal{M}$ is to be activated by at least γ clients. Then, the total number of activated blocks in this subset $\mathcal{B}$ is $|\mathcal{B}|\gamma$. However, client $n \in \mathcal{N}$ can only activate at most $\min\{\lambda_n^*, |\mathcal{B}|\}$ blocks. The total number of activated blocks of all clients for subset $\mathcal{B}$ is $\sum_{n \in \mathcal{N}} \min\{\lambda_n^*, |\mathcal{B}|\}$. To make γ feasible, we must have $|\mathcal{B}|\gamma \leq \sum_{n \in \mathcal{N}} \min\{\lambda_n^*, |\mathcal{B}|\}$. Note that this inequality must hold for all possible subsets $\mathcal{B}$. Hence, the optimal γ is

$$\begin{aligned} \gamma^* &= \min_{\mathcal{B}} \left\{ \gamma : |\mathcal{B}|\gamma \leq \sum_{n \in \mathcal{N}} \min\{\lambda_n^*, |\mathcal{B}|\}, \mathcal{B} \subseteq \mathcal{M}, \mathcal{B} \neq \emptyset \right\} \\ &= \min_{\mathcal{B}} \left\{ \left\lfloor \frac{\sum_{n \in \mathcal{N}} \min \left\{ \frac{\psi_{\max, n} - \psi_{\min} - \epsilon_n}{(\alpha + 3K + 1)BLH}, |\mathcal{B}| \right\}}{|\mathcal{B}|} \right\rfloor \right\}. \end{aligned} \quad (11)$$

□

After γ^* has been determined, we apply Dinic's algorithm [36] to construct the optimal block activation matrix for problem $\mathcal{P}_I$. We denote this matrix as $\tilde{\mathbf{A}} \in \{0, 1\}^{M \times N}$, which serves as the initial block activation matrix. Based on $\tilde{\mathbf{A}}$, we obtain the initial popularity of each block and initial least popularity of each client as $\tilde{c}_m, m \in \mathcal{M}$ and $\tilde{c}_n, n \in \mathcal{N}$, respectively. The initial popularity vector and initial least popularity vector are denoted as $\tilde{\mathbf{c}} \in \mathbb{R}^M$ and $\tilde{\mathbf{c}} \in \mathbb{R}^N$, respectively.

Least popularity adjustment problem $\mathcal{P}_{II}$: Note that $\tilde{\mathbf{A}}$ may not be optimal to problem $\mathcal{P}_2$. Hence, based on Theorem 3, we formulate problem $\mathcal{P}_{II}$ to activate as many additional blocks as possible on top of $\tilde{\mathbf{A}}$ such that the number of clients with the least popularity γ^* is minimized. Let $x_{m,n} \in \{0, 1\}$ denote the additional block activation decision. We set $x_{m,n} = 1$ to additionally activate the m -th block of client n when $\tilde{a}_{m,n} = 0$. Otherwise, $x_{m,n} = 0$. That is,

$$x_{m,n}\tilde{a}_{m,n} = 0, \quad m \in \mathcal{M}, n \in \mathcal{N}. \quad (12)$$

Let $\mathbf{X} \in \{0, 1\}^{M \times N}$ denote the additional block activation decision matrix. Note that the total number of blocks that each

client activates must satisfy the VRAM constraint. That is,

$$\sum_{m \in \mathcal{M}} (x_{m,n} + \tilde{a}_{m,n}) \leq \lambda_n^*, \quad n \in \mathcal{N}. \quad (13)$$

Then, given the decision matrix $\mathbf{X}$, we obtain the popularity of each block and the least popularity of each client as

$$c_m = \tilde{c}_m + \sum_{n \in \mathcal{N}} x_{m,n}, \quad m \in \mathcal{M}, \quad (14)$$

$$c_n = \min_{m: \tilde{a}_{m,n} + x_{m,n} = 1} c_m, \quad n \in \mathcal{N}. \quad (15)$$

In addition, we introduce an auxiliary indicator variable $y_n \in \{0, 1\}$ to characterize whether client n 's least popularity is still γ^* after additional block activation. In particular, if $c_n = \gamma^*$, then $y_n = 1$. Otherwise, $y_n = 0$. If client n activates the m -th block after additional block activation and its least popularity is larger than γ^* , then the popularity of the m -th block must be larger than γ^* . That is, for $x_{m,n} + \tilde{a}_{m,n} = 1$ and $y_n = 0$, c_m must satisfy $c_m \geq \gamma^* + 1, m \in \mathcal{M}, n \in \mathcal{N}$. We use the big-M method [37] to linearize this condition as

$$c_m \geq (\gamma^* + 1) - \beta_1 (1 - (x_{m,n} + \tilde{a}_{m,n})) - \beta_2 y_n, \quad m \in \mathcal{M}, n \in \mathcal{N}, \quad (16)$$

where β_1 and β_2 are two positive constants. When $\beta_1 = \beta_2 = \gamma^* + 1$, the aforementioned condition always holds. Thus, inequality (16) is equivalent to

$$c_m \geq (\gamma^* + 1) - (\gamma^* + 1) (1 - (x_{m,n} + \tilde{a}_{m,n})) - (\gamma^* + 1) y_n, \quad m \in \mathcal{M}, n \in \mathcal{N}. \quad (17)$$

In summary, problem $\mathcal{P}_{II}$ can be formulated as follows:

$$\mathcal{P}_{II} : \quad \underset{\mathbf{X}, \{y_n\}_{n \in \mathcal{N}}}{\text{minimize}} \quad \sum_{n \in \mathcal{N}} y_n \quad (18a)$$

subject to constraints (12)–(15), (17),

$$x_{m,n}, y_n \in \{0, 1\}, \quad m \in \mathcal{M}, n \in \mathcal{N}. \quad (18b)$$

To solve problem $\mathcal{P}_{II}$, we propose a greedy update algorithm. We start by introducing several terms. Let $\mathcal{L} = \{m : \tilde{c}_m = \gamma^*, m \in \mathcal{M}\}$ denote the set of blocks with its initial popularity as γ^* . Let $\mathcal{F} = \{n : \tilde{c}_n = \gamma^*, n \in \mathcal{N}\}$ denote the set of clients whose initial least popularity is γ^* . Let $\mathbf{r} \in \mathbb{R}^N$ denote the vector of remaining VRAM budgets, in which each element $r_n = \lfloor \lambda_n^* \rfloor - \sum_{m \in \mathcal{M}} \tilde{a}_{m,n}$ represents the number of additional blocks client n can activate after the initial block activation. For client $n \in \mathcal{N}$, we denote $\mathcal{W}_n = \{m \in \mathcal{L} : \tilde{a}_{m,n} = 1\}$ as the “bottleneck” set. It includes those blocks which are activated while satisfying the initial popularity.

The greedy update algorithm can be summarized as follows: We define $\text{Gain_ls}[m]$ as the number of clients whose current least popularity is γ^* and would increase to $\gamma^* + 1$ if their m -th block were activated. Among all blocks with popularity γ^* (i.e., $m \in \mathcal{L}$), we denote m^* as the “most valuable” block that maximally reduces the number of clients with least popularity γ^* when activated. We then activate the m^* -th block by assigning it to any client n with remaining VRAM budget

Algorithm 1 Greedy update algorithm for $\mathcal{P}_{\text{II}}$

```
1: Input: Initial block activation matrix  $\tilde{\mathbf{A}}$ ; initial popularity vector  $\tilde{\mathbf{c}}$ ; initial least popularity vector  $\tilde{\mathbf{g}}$ ; optimal least popularity  $\gamma^*$ ;  $\mathcal{W}_n, n \in \mathcal{N}$ ;  $\mathcal{L}$ ;  $\mathcal{F}$ ;  $\mathbf{r}$ ;  $\lambda_n^*, n \in \mathcal{N}$ ; Gain_ls = [].
2: While  $\mathcal{L} \neq \emptyset$ ,  $\mathcal{F} \neq \emptyset$ , and  $\sum_{n \in \mathcal{N}} \lambda_n^* > 0$  do
3:   Create a candidate block index set  $\mathcal{C} := \{m \in \mathcal{L} : \exists n \in \mathcal{N}, \text{ such that } \tilde{a}_{m,n} = 0 \text{ and } r_n > 0\}$ .
4:   For  $m \in \mathcal{C}$  do
5:     Gain_ls[m] :=  $|\{n \in \mathcal{F} : m \in \mathcal{W}_n\}|$ .
6:   End for
7:    $m^* := \arg \max_{m \in \mathcal{C}} \{\text{Gain\_ls}[m]\}$ .
8:   Randomly select client  $n$  such that  $r_n > 0$  and  $\tilde{a}_{m^*,n} = 0$ .
9:    $x_{m^*,n} := 1$ ,  $r_n := r_n - 1$ .
10:  Remove  $m^*$  from  $\mathcal{L}$ .
11:  For  $n \in \mathcal{N}$  and  $\tilde{a}_{m^*,n} = 1$  do
12:    Remove  $m^*$  from  $\mathcal{W}_n$ . Remove  $n$  from  $\mathcal{F}$  if  $\mathcal{W}_n = \emptyset$ .
13:  End for
14: End while
15:  $\mathbf{A} := \tilde{\mathbf{A}} + \mathbf{X}$ .
16: Output: Updated block activation matrix  $\mathbf{A}$ .
```

to activate more blocks (i.e., $r_n > 0$) and has not yet activated m^* -th block (i.e., $\tilde{a}_{m^*,n} = 0$). Thus, the popularity of the m^* -th block increases from γ^* to $\gamma^* + 1$, i.e., $c_{m^*} \leftarrow \gamma^* + 1$. Next, for every client $n \in \mathcal{N}$ whose “bottleneck” set $\mathcal{W}_n$ includes block m^* , we remove m^* from $\mathcal{W}_n$. If a client’s bottleneck set becomes empty, it is removed from the set of remaining clients $\mathcal{F}$. This procedure is repeated iteratively until the number of clients with the least popularity γ^* is minimized. We summarize this algorithm in Algorithm 1.

To solve problem $\mathcal{P}_1$, we iterate over each $\epsilon \in \mathcal{E}$. For each ϵ , we solve the corresponding subproblems $\mathcal{P}_1$ and $\mathcal{P}_{\text{II}}$, and obtain the optimal block activation matrix $\mathbf{A}_\epsilon^*$. Based on $\mathbf{A}_\epsilon^*$, we can determine the corresponding value of $\Lambda(\mathbf{A}_\epsilon^*)$ and $\psi_{\text{tot},n}(\mathbf{A}_\epsilon^*), n \in \mathcal{N}$. We denote the total usage of all clients as $\sum_{n \in \mathcal{N}} \psi_{\text{tot},n}(\mathbf{A}_\epsilon^*)$. By going over all $\epsilon \in \mathcal{E}$, we can form a Pareto front which characterizes the trade-off between Λ and total VRAM usage of all clients. We can choose the optimal block activation matrix $\mathbf{A}^*$ on the Pareto front to balance the VRAM usage and the convergence rate for ZorBA. The algorithm to solve problem $\mathcal{P}_1$ is shown in Algorithm 2.

By using Dinic’s algorithm, solving problem $\mathcal{P}_1$ incurs a computation complexity of $\mathcal{O}(MN\sqrt{M} + N \log_2 N)$. The greedy update algorithm in the second stage incurs a computation complexity of $\mathcal{O}(\min\{M, N\}MN)$. By going through $\mathcal{E}$, our proposed ϵ -constraint lexicographic algorithm incurs a total computation complexity of $\mathcal{O}(EMN(\sqrt{M} + N \log_2 N + \min\{M, N\}))$, which is significantly lower than that of directly solving problem $\mathcal{P}_1$. The overall workflow of our proposed ZorBA is shown in Algorithm 3.

V. PERFORMANCE EVALUATION

A. Simulation Setup

We conduct federated LLM fine-tuning over $N = 50$ clients and a central server. We use OPT-125M and OPT-1.3B [38] as the local models for experiments. In particular, OPT-125M and OPT-1.3B have 12 and 24 transformer blocks, respectively. We

Algorithm 2 ϵ -constraint lexicographic algorithm for $\mathcal{P}_1$

```
1: Input: VRAM capacities  $\psi_{\text{max},n}, n \in \mathcal{N}$ ; initial constraint parameter vector  $\mathcal{E}$ .
2: For  $\epsilon \in \mathcal{E}$  do
3:   Solve problem  $\mathcal{P}_1$  to determine  $\gamma^*$  by using eqn. (10).
4:   Construct the initial block activation matrix  $\tilde{\mathbf{A}}$ .
5:   Obtain the updated block activation matrix  $\mathbf{A}_\epsilon^*$  using Alg. 1.
6:   Determine  $\Lambda(\mathbf{A}_\epsilon^*)$  and  $\psi_{\text{tot},n}(\mathbf{A}_\epsilon^*), n \in \mathcal{N}$ .
7:   Record tuple  $(\epsilon, \mathbf{A}_\epsilon^*, \Lambda(\mathbf{A}_\epsilon^*), \sum_{n \in \mathcal{N}} \psi_{\text{tot},n}(\mathbf{A}_\epsilon^*))$ .
8: End for
9: Select  $\mathbf{A}^*$  on the Pareto front  $\{(\sum_{n \in \mathcal{N}} \psi_{\text{tot},n}(\mathbf{A}_\epsilon^*), \Lambda(\mathbf{A}_\epsilon^*)) : \epsilon \in \mathcal{E}\}$ .
10: Output: Optimal block activation matrix  $\mathbf{A}^*$ .
```

Algorithm 3 ZorBA Algorithm

```
1: Input: Initial model  $\bar{\mathbf{w}}^1$ ; local fine-tuning datasets  $\mathcal{D}_n, n \in \mathcal{N}$ ; initial random seeds  $\mathcal{S}$ ; learning rate  $\eta$ .
2: The central server solves problem  $\mathcal{P}_1$  to determine the block activation matrix  $\mathbf{A}^*$  by using Algorithm 2.
3: For client  $n \in \mathcal{N}$  in parallel do
4:   Activate local model parameters.
5: End for
6: For  $t \in \mathcal{T}$  do
7:   The central server selects a subset  $\mathcal{Q}^t$  from  $\mathcal{S}$ .
8:   For client  $n \in \mathcal{N}$  in parallel do
9:     Estimate the gradients  $\bar{\nabla} F_n(\bar{\mathbf{w}}_n^t; \xi_n)$  based on eqn. (1) and record the finite differences of the estimated gradients.
10:    Transmit the finite differences of the estimated gradients to the central server.
11:   End for
12:   The central server performs aggregation based on eqn. (2), obtains and broadcasts  $\bar{\mathcal{P}}^t$  back to the clients.
13:   For client  $n \in \mathcal{N}$  in parallel do
14:     Determine  $\bar{\mathbf{w}}^{t+1}$  based on  $\bar{\mathcal{P}}^t$ .
15:   End for
16: End for
17: Output: fine-tuned model  $\bar{\mathbf{w}}^{T+1}$ .
```

conduct experiments on text classification datasets, including AG-News [39], SST-2 [40], and SNLI [41]. We use the Dirichlet distribution $\text{Dir}(1.0)$ to create non-independent and identically distributed data partitioning across clients’ local datasets. To characterize the VRAM capacity, we denote the maximum VRAM of all clients as the sum of the number of model parameters and activations of all blocks. We set the minimum VRAM of all clients to the sum of the number of model parameters and activations of one block. The VRAM capacity of each client follows a uniform distribution between the minimum and maximum of VRAM. We use the number of transmitted parameters to characterize the communication overhead. In addition, we set $\alpha = 4$, $\eta = 5 \times 10^{-5}$, $\mu = 10^{-4}$, $B = 8$, $E = 1000$, $P = 4096$, $Q = 10$, and $T = 500$. We compare the performance of ZorBA with the following baseline schemes:

- FedIT [14]: Clients perform first-order BP with all blocks.
- FedZO [26]: Clients sample one perturbation vector to perform zeroth-order optimization with all blocks during each fine-tuning round. Then, clients exchange the estimated gradients with the central server.
- DeComFL [30]: Clients use a shared perturbation vector

TABLE I: Comparison of the number of fine-tuning rounds required to achieve the target average testing accuracy.

Model	Datasets	ZorBA	FedIT	FedZO	DeComFL
OPT-125M	AG-News	138	49	155	181
	SST-2	167	70	161	205
	SNLI	231	112	255	246
OPT-1.3B	AG-News	61	23	79	68
	SST-2	99	48	104	104
	SNLI	157	82	186	173

TABLE II: Comparison of the total VRAM usage of all clients per round (in GB) and total communication overhead (i.e., Commun.) of all clients to achieve the target accuracy.

Model	Datasets	Metrics	ZorBA	FedIT	FedZO	DeComFL
OPT-125M	AG-News	VRAM	47.31	118.67	95.37	95.37
		Commun.	1.24×10^5	6.13×10^{11}	1.94×10^{12}	1.36×10^5
	SST-2	VRAM	59.32	118.67	95.37	95.37
		Commun.	1.67×10^5	8.75×10^{11}	2.01×10^{12}	1.54×10^5
	SNLI	VRAM	131.39	190.74	167.44	167.44
		Commun.	2.43×10^5	1.40×10^{12}	3.19×10^{12}	1.85×10^5
OPT-1.3B	AG-News	VRAM	538.04	1431.17	1189.03	1189.03
		Commun.	7.93×10^4	2.99×10^{12}	1.03×10^{13}	9.18×10^4
	SST-2	VRAM	794.49	1431.17	1189.03	1189.03
		Commun.	1.34×10^5	6.24×10^{12}	1.35×10^{13}	1.40×10^5
	SNLI	VRAM	1583.55	2378.05	2315.90	2315.90
		Commun.	2.27×10^5	1.07×10^{13}	2.42×10^{13}	2.34×10^5

to perform zeroth-order optimization with all blocks during each fine-tuning round. Then, clients exchange the finite differences of estimated gradients with the server.

B. Experiments

1) *Comparison of the Convergence*: To show the convergence rate, we compare the number of training rounds each approach takes to achieve a target accuracy. In particular, for AG-News, SST-2, and SNLI, we set the target accuracies as 0.8, 0.8, and 0.72, respectively. Since the central server determines the block activation matrix in prior, we present the total VRAM usage of all clients per fine-tuning round, which remains consistent during fine-tuning. Furthermore, we use the number of parameters transmitted between clients and the central server to characterize the total communication overhead. Results in Table I show that our proposed ZorBA converges faster than DeComFL by up to 23.76%. ZorBA also converges faster than FedZO in most cases. These results highlight the benefit of optimizing the heterogeneous block activation matrix to accelerate the convergence.

2) *Comparison of the Total VRAM Usage and Communication Overhead*: We compare the total VRAM usage and communication overhead in Table II. Our proposed ZorBA achieves a much lower total VRAM usage, which is up to 62.41%, 54.75%, and 54.75% lower than FedIT, FedZO, and DeComFL, respectively. Note that FedZO and DeComFL incur the same total VRAM usage since they both activate all blocks. In addition, ZorBA incurs negligible communication overhead when compared with FedIT and FedZO. ZorBA also achieves a comparable total communication overhead to DeComFL.

3) *Trade-off between Λ and Total VRAM Usage*: To show the trade-off between the value of Λ and the total VRAM usage of all clients, we conduct experiments by using OPT-125M and SST-2 dataset. For better illustration, we use the fraction of the total VRAM usage of all clients (i.e., $\frac{\sum_{n \in \mathcal{N}} \psi_{\text{tot},n}}{\sum_{n \in \mathcal{N}} \psi_{\text{max},n}}$).

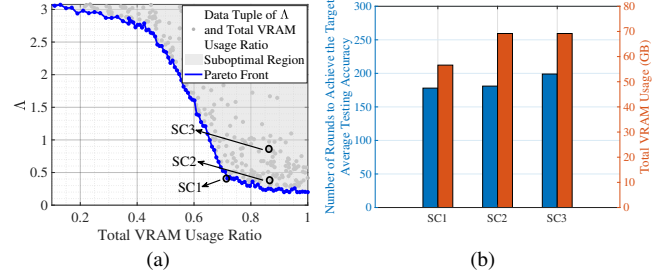


Fig. 4: (a) Pareto front of Λ versus the total VRAM usage ratio. (b) Comparison between SC1, SC2, and SC3 for the number of rounds to achieve the target average testing accuracy and the total VRAM usage.

The Pareto front is shown in Fig. 4 (a). Results show that by activating more blocks across clients, Λ generally exhibits a descending trend, which substantiates our Observation 1 in Section III. Moreover, the value of Λ has a trend of initially slow decline, followed by a sharp drop in the middle, and gradually converging to the minimum. Based on this trend, we can empirically determine the block activation matrix such that the total VRAM usage is effectively reduced without degrading the convergence rate of our proposed ZorBA.

4) *Case Study*: To further demonstrate the relationship between Λ and the convergence rate of our proposed ZorBA, we conduct three case studies. In particular, we select three scenarios from Fig. 4 (a), namely SC1, SC2, and SC3, respectively. The values of the data tuples of SC1, SC2, and SC3 are (0.71, 0.50), (0.86, 0.50), and (0.86, 0.95), respectively. The comparison of the number of fine-tuning rounds to achieve the target average testing accuracy of 0.8 and the total VRAM usage of all clients is shown in Fig. 4 (b). Results show that SC1 requires less total VRAM than SC2, indicating fewer activated blocks across all clients. Nevertheless, SC1 achieves a similar (a bit faster) convergence rate because both SC1 and SC2 yield comparable values of Λ . This supports Observation 2 and Proposition 1 in Section III. In contrast, SC3 uses more VRAM than SC1 and matches SC2 in the total number of activated blocks. However, it needs more fine-tuning rounds to reach the target accuracy. This slower convergence highlights the role of Λ , which validates Observations 3 and 4.

VI. CONCLUSION

In this work, we proposed ZorBA, a zeroth-order optimization-based federated fine-tuning framework. ZorBA leverages zeroth-order optimization and heterogeneous block activation mechanism to reduce the VRAM usage. It introduces shared random seeds to reduce the communication overhead. We theoretically analyzed the convergence rate of our proposed ZorBA and investigated the impact of block activation decisions on the convergence rate and VRAM usage. We proposed an ϵ -constraint lexicographic algorithm to optimize the block activation decisions. Experimental results show that ZorBA converges faster than zeroth-order optimization baselines, while reducing VRAM usage by up to 62.41% and incurring negligible communication overhead.

REFERENCES

- [1] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, *et al.*, “A survey of large language models,” *arXiv preprint arXiv:2303.18223*, Mar. 2023.
- [2] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, “GPT-4 technical report,” *arXiv preprint arXiv:2303.08774*, Mar. 2023.
- [3] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, *et al.*, “The Llama 3 herd of models,” *arXiv preprint arXiv:2407.21783*, Jul. 2024.
- [4] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, *et al.*, “Gemma 2: Improving open language models at a practical size,” *arXiv preprint arXiv:2408.00118*, Jul. 2024.
- [5] S. Ye, B. Ouyang, L. Zeng, T. Qian, X. Chu, J. Tang, and X. Chen, “Jupiter: Fast and resource-efficient collaborative inference of generative LLMs on edge devices,” in *Proc. IEEE INFOCOM*, London, UK, May 2025.
- [6] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, *et al.*, “Parameter-efficient fine-tuning of large-scale pre-trained language models,” *Nature Mach. Intell.*, vol. 5, no. 3, pp. 220–235, Mar. 2023.
- [7] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, *et al.*, “LoRA: Low-rank adaptation of large language models,” in *Proc. Int. Conf. Learn. Representations (ICLR)*, May 2022.
- [8] H. Liu, D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal, and C. A. Raffel, “Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning,” in *Proc. Advances Neural Info. Process. Syst. (NeurIPS)*, Dec. 2022.
- [9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, Fort Lauderdale, FL, Apr. 2017.
- [10] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated optimization in heterogeneous networks,” in *Proc. Mach. Learn. Syst. (MLSys)*, Mar. 2020.
- [11] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, “SCAFFOLD: Stochastic controlled averaging for federated learning,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jul. 2020.
- [12] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, “Tackling the objective inconsistency problem in heterogeneous federated optimization,” in *Advances Neural Info. Process. Syst. (NeurIPS)*, Dec. 2020.
- [13] Z. Qin, D. Chen, B. Qian, B. Ding, Y. Li, and S. Deng, “Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, Honolulu, HI, Jul. 2024.
- [14] J. Zhang, S. Vahidian, M. Kuo, C. Li, R. Zhang, T. Yu, G. Wang, and Y. Chen, “Towards building the federated GPT: Federated instruction tuning,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Seoul, Korea, Apr. 2024.
- [15] Z. Wang, Z. Shen, Y. He, G. Sun, H. Wang, L. Lyu, and A. Li, “FLoRA: Federated fine-tuning large language models with heterogeneous low-rank adaptations,” in *Proc. Advances Neural Info. Process. Syst. (NeurIPS)*, Vancouver, Canada, Dec. 2024.
- [16] J. Bai, D. Chen, B. Qian, L. Yao, and Y. Li, “Federated fine-tuning of large language models under heterogeneous tasks and client resources,” in *Proc. Advances Neural Info. Process. Syst. (NeurIPS)*, Vancouver, Canada, Dec. 2024.
- [17] Z. Wang, Y. Zhou, Y. Shi, and K. B. Letaief, “Federated fine-tuning for pre-trained foundation models over wireless networks,” *IEEE Trans. Wirel. Commun.*, vol. 24, no. 4, pp. 3450–3464, Apr. 2025.
- [18] W. Qiu, Y. Zhou, J. Wang, Q. Z. Sheng, and L. Cui, “FLM-TopK: Expediting federated large language model tuning by sparsifying intervalized gradients,” in *Proc. IEEE INFOCOM*, London, UK, May 2025.
- [19] K. Panchal, N. Parikh, S. Choudhary, L. Zhang, Y. Brun, and H. Guan, “Thinking forward: Memory-efficient federated finetuning of language models,” *Proc. Advances Neural Info. Process. Syst. (NeurIPS)*, Vancouver, Canada, Dec. 2024.
- [20] Y. Shu, Z. Dai, W. Sng, A. Verma, P. Jaillet, and B. K. H. Low, “Zeroth-order optimization with trajectory-informed derivative estimation,” in *Proc. Int. Conf. Learn. Representations (ICLR)*, Kigali, Rwanda, May 2023.
- [21] S. Malladi, T. Gao, E. Nichani, A. Damian, J. D. Lee, D. Chen, and S. Arora, “Fine-tuning language models with just forward passes,” in *Proc. Advances Neural Info. Process. Syst. (NeurIPS)*, New Orleans, LA, Dec. 2023.
- [22] A. Chen, Y. Zhang, J. Jia, J. Diffenderfer, J. Liu, K. Parasyris, Y. Zhang, Z. Zhang, B. Kailkhura, and S. Liu, “DeepZero: Scaling up zeroth-order optimization for deep model training,” in *Proc. Int. Conf. Learn. Representations (ICLR)*, Vienna, Austria, May 2024.
- [23] Y. Zhang, P. Li, J. Hong, J. Li, Y. Zhang, W. Zheng, P.-Y. Chen, J. D. Lee, W. Yin, M. Hong, *et al.*, “Revisiting zeroth-order optimization for memory-efficient LLM fine-tuning: A benchmark,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, Vienna, Austria, Jul. 2024.
- [24] S. Ma and H. Huang, “Revisiting zeroth-order optimization: Minimum-variance two-point estimators and directionally aligned perturbations,” in *Proc. Int. Conf. Learn. Representations (ICLR)*, Singapore, Apr. 2025.
- [25] Y. Chen, Y. Zhang, L. Cao, K. Yuan, and Z. Wen, “Enhancing zeroth-order fine-tuning for language models with low-rank structures,” in *Proc. Int. Conf. Learn. Representations (ICLR)*, Singapore, Apr. 2025.
- [26] W. Fang, Z. Yu, Y. Jiang, Y. Shi, C. N. Jones, and Y. Zhou, “Communication-efficient stochastic zeroth-order optimization for federated learning,” *IEEE Trans. Signal Process.*, vol. 70, pp. 5058–5073, Oct. 2022.
- [27] J. Chen, H. Chen, B. Gu, and H. Deng, “Fine-grained theoretical analysis of federated zeroth-order optimization,” in *Proc. Advances Neural Info. Process. Syst. (NeurIPS)*, New Orleans, LA, Dec. 2023.
- [28] Y. Qiu, U. Shanbhag, and F. Yousefian, “Zeroth-order methods for nondifferentiable, nonconvex, and hierarchical federated optimization,” in *Proc. Advances Neural Info. Process. Syst. (NeurIPS)*, New Orleans, LA, Dec. 2023.
- [29] Z. Ling, D. Chen, L. Yao, Y. Li, and Y. Shen, “On the convergence of zeroth-order federated tuning for large language models,” in *Proc. ACM SIGKDD Conf. Knowl. Discovery Data Mining (SIGKDD)*, Barcelona, Spain, Aug. 2024.
- [30] Z. Li, B. Ying, Z. Liu, C. Dong, and H. Yang, “Achieving dimension-free communication in federated learning via zeroth-order optimization,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, Vancouver, Canada, Jul. 2025.
- [31] Y. Shu, W. Hu, S.-K. Ng, B. K. H. Low, and F. R. Yu, “Ferret: Federated full-parameter tuning at scale for large language models,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, Vancouver, Canada, Jul. 2025.
- [32] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, “On the convergence of FedAvg on non-IID data,” in *Proc. Int. Conf. Learn. Representations (ICLR)*, Apr. 2020.
- [33] C. Meng, M. Tang, M. Setayesh, and V. W.S. Wong, “Tackling resource allocation for decentralized federated learning: A GNN-based approach,” *IEEE Trans. Mobile Comput.*, vol. 24, no. 10, pp. 9554–9569, Oct. 2025.
- [34] Y. Nesterov and V. Spokoiny, “Random gradient-free minimization of convex functions,” *Foundations Comput. Math.*, vol. 17, no. 2, pp. 527–566, Apr. 2017.
- [35] Y. Haimes, “On a bicriterion formulation of the problems of integrated system identification and system optimization,” *IEEE Trans. Syst. Man Cybern.*, no. 3, pp. 296–297, Jul. 1971.
- [36] Y. Dinitz, “Dinitz’s algorithm: The original version and even’s version,” in *Theoretical Computer Science: Essays in Memory of Shimon Even*, pp. 218–240, Berlin, Heidelberg, Germany: Springer, 2006.
- [37] I. Griva, S. G. Nash, and A. Sofer, *Linear and Nonlinear Optimization 2nd edition*. Philadelphia, PA, USA: SIAM, 2008.
- [38] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, *et al.*, “OPT: Open pre-trained transformer language models,” *arXiv preprint arXiv:2205.01068*, May 2022.
- [39] A. Gulli, “AG’s corpus of news articles.” http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html. Accessed: 2025-07-30.
- [40] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Seattle, WA, Oct. 2013.
- [41] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, “A large annotated corpus for learning natural language inference,” in *Proc. Annu. Meeting Assoc. Comput. Linguistics (ACL)*, Beijing, China, Jul. 2015.