Toward Quantized Model Parallelism for Graph-Augmented MLPs Based on Gradient-Free ADMM Framework

Junxiang Wang, Hongyi Li, Zheng Chai, Yongchao Wang, *Senior Member, IEEE*, Yue Cheng, and Liang Zhao, *Senior Member, IEEE*

***Abstract*— While graph neural networks (GNNs) are popular in the deep learning community, they suffer from several challenges including over-smoothing, over-squashing, and gradient vanishing. Recently, a series of models have attempted to relieve these issues by first augmenting the node features and then imposing node-wise functions based on multilayer perceptron (MLP), which are widely referred to as graph-augmented MLP (GA-MLP) models. However, while GA-MLP models enjoy deeper architectures for better accuracy, their efficiency largely deteriorates. Moreover, popular acceleration techniques such as stochastic-version or data-parallelism cannot be effectively applied due to the dependency among samples (i.e., nodes) in graphs. To address these issues, in this article, instead of data parallelism, we propose a parallel graph deep learning Alternating Direction Method of Multipliers (pdADMM-G) framework to achieve model parallelism: parameters in each layer of GA-MLP models can be updated in parallel. The extended pdADMM-G-Q algorithm reduces communication costs by introducing the quantization technique. Theoretical convergence to a (quantized) stationary point of the pdADMM-G algorithm and the pdADMM-G-Q algorithm is provided with a sublinear convergence rate $o(1/k)$, where k is the number of iterations. Extensive experiments demonstrate the convergence of two proposed algorithms. Moreover, they lead to a more massive speedup and better performance than all state-of-the-art comparison methods on nine benchmark datasets. Last but not least, the proposed pdADMM-G-Q algorithm reduces communication overheads by up to 45% without loss of performance. Our code is available at https://github.com/xianggebenben/pdADMM-G.**

***Index Terms*— Alternating direction method of multipliers (ADMM), convergence, graph neural networks (GNNs), model parallelism, quantization.**

Manuscript received 3 July 2021; revised 29 July 2022; accepted 11 November 2022. Date of publication 19 December 2022; date of current version 5 April 2024. This work was supported in part by the National Science Foundation (NSF) under Grant 1755850, Grant 1841520, Grant 2007716, Grant 2007976, Grant 1942594, and Grant 1907805; in part by the Jeffress Memorial Trust Award; in part by the Amazon Research Award; in part by the NVIDIA Graphics Processing Unit (GPU) Grant; in part by the Design Knowledge Company under Subcontract 10827.002.120.04; and in part by the National Science Foundation of China (NSFC) under Grant 62271371. *(Junxiang Wang and Hongyi Li contributed equally to this work.) (Corresponding authors: Yongchao Wang; Liang Zhao.)*

Junxiang Wang and Liang Zhao are with the Department of Computer Science and Informatics, Emory University, Atlanta, GA 30030 USA (e-mail: lzhao41@emory.edu).

Hongyi Li and Yongchao Wang are with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, Shaanxi 710071, China (e-mail: ychwang@mail.xidian.edu.cn).

Zheng Chai and Yue Cheng are with the Department of Computer Science, University of Virginia, Charlottesville, VA 22904 USA.

This article has supplementary downloadable material available at https://doi.org/10.1109/TNNLS.2022.3223879, provided by the authors.

Digital Object Identifier 10.1109/TNNLS.2022.3223879

I. Introduction

GRAPH neural networks (GNNs) have accomplished the state-of-the-art performance in various graph applications such as node classification and link prediction. This is because they handle graph-structured data via aggregating neighbor information and extending operations and definitions of the deep learning approach [1]. However, their performance has significantly been restricted via their depths due to the over-smoothing problem (i.e., the representations of different nodes in a graph tend to be similar when stacking multiple layers) [2], the over-squashing problem (i.e., the information flow among distant nodes distorts along the long-distance interactions) [3], and the gradient vanishing problem (i.e., the signals of gradients decay with the depths of GNN models) [2]. These challenges still exist even though some models such as GraphSAGE [4] have been proposed to alleviate them.

On the other hand, the graph augmented multilayer perceptron (GA-MLP) models have recently received fast-increasing attention as an alternative to deal with the aforementioned drawbacks of conventional GNNs via the augmentation of graph features. GA-MLP models augment node representations of graphs and feed them into multilayer perceptron (MLP) models. Compared with GNNs, GA-MLP models are more resistant to the over-smoothing problem [2] and therefore demonstrate outstanding performance. For example, Wu et al. [5] showed that a two-layer GA-MLP approximates the performance of the GNN models on multiple datasets.

GA-MLP models are supposed to perform better with the increase of their depths. However, similar to GNNs, GA-MLP models still suffer from the gradient vanishing problem, which is caused by the mechanism of the classic backpropagation algorithm. This is because gradient signals diminish during the transmission among deep layers. Moreover, while the models go deeper, efficiency will become an issue, especially for medium- and large-size graphs. Compared to the data such as images and texts, where identically and independently distributed [independent identically distributed (i.i.d.)] samples are assumed, efficiency issues in graph data are much more difficult to handle due to the dependency among data samples (i.e., nodes in graphs). Such dependency seriously troubles the effectiveness of using typical acceleration techniques such as sampling-based methods, and data-parallelism distributed learning in solving the efficiency issue. Therefore, parallelizing the computation along layers is a natural workaround, but the backpropagation prevents the gradients of different layers from being calculated in parallel. This is because the calculation of the gradient in one layer is dependent on its previous layers.

To handle these challenges, recently gradient-free optimization methods such as the Alternating Direction Method of Multipliers (ADMM) have been investigated to overcome the difficulties of the backpropagation algorithm. The spirit of ADMM is to decouple a neural network into layerwise subproblems such that each of them can be solved efficiently. ADMM does not require gradient calculation and therefore can avoid the gradient vanishing problem. Existing literature has shown its great potential. For example, Taylor et al. [6] and Wang et al. [7] proposed ADMM to train MLP models. Extensive experiments have demonstrated that the ADMM has outperformed most comparison methods such as gradient descent (GD).

In this article, we propose a novel parallel graph deep learning ADMM (pdADMM-G) optimization framework to train large-scale GA-MLP models, and the extended pdADMM-G-Q algorithm reduces the communication cost of the pdADMM-G algorithm by the quantization techniques. Our contributions to this article include the following.

1) We propose a novel reformulation of GA-MLP models, which splits a neural network into independent layer partitions and allow for ADMM to achieve model parallelism.
2) We propose a novel pdADMM-G framework to train a GA-MLP model. All subproblems generated by the ADMM algorithm are discussed. The extended pdADMM-G-Q algorithm reduces communication costs by introducing the quantization technique.
3) We provide the theoretical convergence guarantee of the proposed pdADMM-G algorithm and the pdADMM-G-Q algorithm. Specifically, they converge to a (quantized) stationary point of GA-MLP models when the hyperparameters are sufficiently large, and their sublinear convergence rates are $o(1/k)$.
4) We conduct extensive experiments on nine benchmark datasets to show the convergence, the massive speedup of the proposed pdADMM-G algorithm and the pdADMM-G-Q algorithm, as well as their outstanding performance when compared with all the state-of-the-art optimizers. Moreover, the proposed pdADMM-G-Q algorithm reduces communication overheads by up to 45%.

The organization of this article is shown as follows: In Section II, we summarize recent related research work to this article. In Section III, we propose the pdADMM-G algorithm and the pdADMM-G-Q algorithm to train deep GA-MLP models. Section IV details the convergence properties of the proposed pdADMM-G algorithm and the pdADMM-G-Q algorithm. Extensive experiments on nine benchmark datasets to demonstrate the convergence, speedup, communication savings, and outstanding performance of the pdADMM-G algorithm and the pdADMM-G-Q algorithm are shown in Section V, and Section VI concludes this work.

II. Related Work

This section summarizes existing literature related to this research.

A. Distributed Deep Learning

With the increase of public datasets and layers of neural networks, it is imperative to establish distributed deep learning systems for large-scale applications. Many systems have been established to satisfy such needs. Famous systems include Terngrad [8], Horovod [9], SINGA [10] Mxnet [11], TicTac [12], and Poseidon [13]. They applied some parallelism techniques to reduce computational time, and therefore improved the speedup. Existing parallelism techniques can be classified into two categories: data parallelism and model parallelism. Data parallelism focuses on distributing data across different processors and then aggregating results from them into a server. Scaling GD is one of the most common ways to reach data parallelism [14]. For example, the distributed architecture, Poseidon, is achieved by scaling GD through overlapping communication and computation over networks. The recently proposed ADMM [6], [7] is another way to achieve data parallelism. However, data parallelism suffers from the bottleneck of a neural network: for GD, the gradient should be transmitted through all processors; for ADMM, the parameters in one layer are subject to those in its previous layer. As a result, this leads to heavy communication costs and time delays. Model parallelism, however, can solve this challenge because model parallelism splits a neural network into many independent partitions. In this way, each partition can be optimized independently and reduce layer dependency. For instance, Parpas and Muir [15] proposed a parallel-in-time method from the perspective of dynamic systems; Huo et al. [16] introduced a feature replay algorithm to achieve model parallelism. Zhuang et al. [17] broke layer dependency by introducing the delayed gradient.

B. Deep Learning on Graphs

Graphs are ubiquitous structures and are popular in real-world applications. There is a surge of interest to apply deep learning techniques to graphs. For a comprehensive summary please refer to [18]. It classified existing GNN models into four categories: recurrent GNNs (RecGNNs), convolutional GNNs (ConvGNNs), graph autoencoders (GAEs), and spatial–temporal GNNs (STGNNs). RecGNNs learn node representation with recurrent neural networks via the message passing mechanisms [19], [20], [21]; ConvGNNs generalize the operations of convolution to graph data and stack multiple convolution layers to extract high-level node features [22], [23], [24]; GAEs encode node information into a latent space and reconstruct graphs from the encoded node representation [25], [26], [27]; the idea of STGNNs is to capture spatial dependency and temporal dependency simultaneously [28], [29], [30].

TABLE I
IMPORTANT NOTATIONS

Notations	Descriptions
L	Number of layers.
W_l	The weight matrix for the l-th layer.
b_l	The intercept vector for the l-th layer.
z_l	The auxiliary variable of the linear mapping for the l-th layer.
$f_l(z_l)$	The nonlinear activation function for the l-th layer.
p_l	The input for the l-th layer.
q_l	The output for the l-th layer.
X	The node representation of the graph.
A	The adjacency matrix of the graph.
y	The predefined label vector.
$R(z_L, y)$	The risk function for the L-th layer.
n_l	The number of neurons for the l-th layer.
u_l	The dual variable for the l-th layer.

III. pdADMM-G Algorithm

We propose the pdADMM-G algorithm to solve GA-MLP models in this section. Specifically, Section III-A formulates the GA-MLP model training problem, and Section III-B proposes the pdADMM-G algorithm. Section III-C extends the proposed pdADMM-G algorithm to the pdADMM-G-Q algorithm for quantization. Important notations are given in Table I.

A. Problem Formulation

Consider a graph $G = (V, E)$, where V and E are sets of nodes and edges, respectively, $|V|$ is the number of nodes, let $\Psi = \{\psi_1(A), \ldots, \psi_K(A)\}$ be a set of (usually multihop) operators $\psi_i(A):\mathbb{R}^{|V|} \to \mathbb{R}^{|V|} (i = 1, \ldots, K)$ that are functions of the adjacency matrix $A \in \{0, 1\}^{|V|\times|V|}$, and $\mathbb{R}^{|V|}$ is the domain of $\psi_i(A)$ $(i = 1, \ldots, K)$. $X_k = H\psi_k(A)$ is the augmentation of node features by the k-hop operator, where $H \in \mathbb{R}^{d\times|V|}$ is a matrix of node features, and d is the dimension of features. $X_k (k = 1, \ldots, K)$ are stacked into $X = [X_1; \ldots; X_K]$ by column. Then the GA-MLP training problem is formulated as follows [7].

Problem 1

$$\min_{W_l, b_l, z_l, p_l} R(z_L; y)$$
$$\text{s.t. } z_l = W_l p_l + b_l,\ \ p_{l+1} = f_l(z_l) (l = 1, \ldots, L-1)$$

where $p_1 = X \in \mathbb{R}^{n_0\times|V|}$ is the input of deep GA-MLP models, where $n_0 = Kd$ is the dimension of input and y is a predefined label vector. $p_l \in \mathbb{R}^{n_l\times|V|}$ is the input for the lth layer, also the output for the $(l-1)$th layer, and n_l is the number of neurons for the lth layer. $R(z_L; y)$ is a risk function for the Lth layer, which is convex and continuous; $z_l = W_l p_l + b_l$ and $p_{l+1} = f_l(z_l)$ are linear and nonlinear mappings for the lth layer, respectively, and $W_l \in \mathbb{R}^{n_l\times n_{l-1}}$ and $b_l \in \mathbb{R}^{n_l}$ are the weight matrix and the intercept vector for the lth layer, respectively.

In Problem 1, Ψ can be considered as a prepossessing step to augment node features via A, and hence it is predefined. One common choice can be $\Psi = \{I, A, A^2, \ldots, A^{K-1}\}$.

Problem 1 can be addressed by deep learning ADMM (dlADMM) [7]. However, parameters in one layer are dependent on its neighboring layers and hence cannot achieve parallelism. For example, the update of p_{l+1} on the $(l+1)$th layer needs to wait before z_l on the lth layer is updated. To address layer dependency, we relax Problem 1 to Problem 2 as follows.

Problem 2

$$\min_{\mathbf{p},\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{q}} F(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}) = R(z_L; y) + (\nu/2)\left(\sum_{l=1}^{L} \|z_l - W_l p_l - b_l\|_2^2 + \sum_{l=1}^{L-1} \|q_l - f_l(z_l)\|_2^2\right)$$
$$\text{s.t. } p_{l+1} = q_l$$

where $\mathbf{p} = \{p_l\}_{l=1}^{L}$, $\mathbf{W} = \{W_l\}_{l=1}^{L}$, $\mathbf{b} = \{b_l\}_{l=1}^{L}$, $\mathbf{z} = \{z_l\}_{l=1}^{L}$, $\mathbf{q} = \{q_l\}_{l=1}^{L-1}$, and $\nu > 0$ is a tuning parameter. As $\nu \to \infty$, Problem 2 approaches Problem 1. We reduce layer dependency by splitting the output of the lth layer and the input of the $(l+1)$th layer into two variables p_{l+1} and q_l, respectively.

B. pdADMM-G Algorithm

The high-level overview of the pdADMM-G algorithm is shown in Fig. 1. Specifically, the inputs of GA-MLP models are augmented by $H\psi_k(A)$ $(k = 1, \ldots, K)$, and then GA-MLP models are split into multiple layers, each of which can be optimized by an independent client. Therefore, layerwise training can be implemented in parallel. Moreover, the gradient vanishing problem can be avoided in this way. This is because the accumulated gradient calculated by the backpropagation algorithm is split into layerwise components.

Now we follow the ADMM routine to solve Problem 2. The augmented Lagrangian function is formulated mathematically as follows:

$$\begin{aligned} &L_\rho(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}, \mathbf{u}) \\ &= F(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}) + \sum_{l=1}^{L-1}\left(u_l^T(p_{l+1} - q_l) + (\rho/2)\|p_{l+1} - q_l\|_2^2\right) \\ &= R(z_L; y) + \phi(p_1, W_1, b_1, z_1) \\ &\quad + \sum_{l=2}^{L} \phi(p_l, W_l, b_l, z_l, q_{l-1}, u_{l-1}) \\ &\quad + (\nu/2)\sum_{l=1}^{L-1} \|q_l - f_l(z_l)\|_2^2 \end{aligned}$$

where $\phi(p_1, W_1, b_1, z_1) = (\nu/2)\|z_1 - W_1 p_1 - b_1\|_2^2$, $\phi(p_l, W_l, b_l, z_l, q_{l-1}, u_{l-1}) = (\nu/2)\|z_l - W_l p_l - b_l\|_2^2 + u_{l-1}^T(p_l - q_{l-1}) + (\rho/2)\|p_l - q_{l-1}\|_2^2$, $u_l (l = 1, \ldots, L-1)$ are dual variables, $\rho > 0$ is a parameter, and $\mathbf{u} = \{u_l\}_{l=1}^{L-1}$. The detail of the pdADMM-G algorithm is shown in Algorithm 1. Specifically, Lines 5–9 update primal variables $\mathbf{p}$, $\mathbf{W}$, $\mathbf{b}$, $\mathbf{z}$, and $\mathbf{q}$, respectively, while Line 11 updates the dual variable $\mathbf{u}$.

Due to space limit, the details of all subproblems are shown in Section A in the Supplementary Material.

Our proposed pdADMM-G algorithm can be efficient for training deep GA-MLP models via the greedy layerwise training strategy [31]. Specifically, we begin by training a

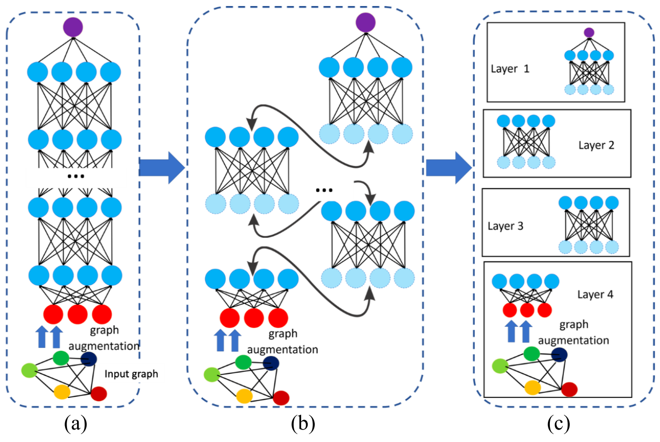

Fig. 1. Overall pdADMM-G optimization algorithm: it splits GA-MLP models into layerwise components. (a) Deep GA-MLPs. (b) GA-MLP reformulation. (c) Model parallelism.

Algorithm 1 pdADMM-G Algorithm to Solve Problem 2

```
Require: $y,\ p_1 = X,\ \rho,\ \nu$.
Ensure: $\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}$.
  Initialize $k = 0$.
  while $\mathbf{p}^k, \mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}^k$ not converged do
    $p_l^{k+1} \leftarrow \arg\min_{p_l} L_\rho(\mathbf{p}, \mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}^k, \mathbf{u}^k)$ for different $l$ in parallel.
    $W_l^{k+1} \leftarrow \arg\min_{W_l} L_\rho(\mathbf{p}^{k+1}, \mathbf{W}, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}^k, \mathbf{u}^k)$ for different $l$ in parallel.
    $b_l^{k+1} \leftarrow \arg\min_{b_l} L_\rho(\mathbf{p}^{k+1}, \mathbf{W}^{k+1}, \mathbf{b}, \mathbf{z}^k, \mathbf{q}^k, \mathbf{u}^k)$ for different $l$ in parallel.
    $z_l^{k+1} \leftarrow \arg\min_{z_l} L_\rho(\mathbf{p}^{k+1}, \mathbf{W}^{k+1}, \mathbf{b}^{k+1}, \mathbf{z}, \mathbf{q}^k, \mathbf{u}^k)$ for different $l$ in parallel.
    $q_l^{k+1} \leftarrow \arg\min_{q_l} L_\rho(\mathbf{p}^{k+1}, \mathbf{W}^{k+1}, \mathbf{b}^{k+1}, \mathbf{z}^{k+1}, \mathbf{q}, \mathbf{u}^k)$ for different $l$ in parallel.
    $r_l^k \leftarrow p_{l+1}^{k+1} - q_l^{k+1} (l = 1, \ldots, L)$ in parallel # Compute residuals.
    $u_l^{k+1} \leftarrow u_l^k + \rho(p_{l+1}^{k+1} - q_l^{k+1})$ for different $l$ in parallel.
    $k \leftarrow k + 1$.
  end while
  Output $\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}$.
```

shallow GA-MLP model. Next, more layers are increased to the GA-MLP model and their parameters are trained, then we introduce even more layers and iterate this process until the whole deep GA-MLP model is included. The pdADMM-G algorithm can achieve excellent performance as well as reduce training costs by this strategy.

Last but not least, we compare the computational costs of the proposed pdADMM-G algorithm with the state-of-the-art backpropagation algorithm, on which the GD is based. We show that they share the same level of computational costs. For the backpropagation algorithm, the most costly operation is the matrix multiplication $z_l = W_l p_l + b_l$ in the forward pass, where $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$ and $p_l \in \mathbb{R}^{n_{l-1} \times |V|}$, which requires a time complexity of $O(n_l n_{l-1} |V|)$ [32]; for the proposed pdADMM-G algorithm, the most costly operation is to compute the derivative $\nabla_{W_l} \phi$, and it also involves the matrix multiplication, and hence its time complexity is again $O(n_l n_{l-1} |V|)$. However, the proposed pdADMM-G algorithm trains the whole GA-MLP model in a model parallelism fashion [33], and therefore all computational costs can be split into different independent clients for parallel training; whereas the backpropagation algorithm is implemented sequentially, and thus it is less efficient than the proposed pdADMM-G algorithm.

C. Quantization Extension of pdADMM-G

In the proposed pdADMM-G algorithm, p_l and q_l are transmitted back and forth among layers (i.e., clients). However, the communication overheads of p_l and q_l surge for a large-scale graph G with millions of nodes. To alleviate this challenge, the quantization technique is commonly utilized to reduce communication costs by mapping continuous values into a discrete set [34]. In other words, p_l is required to fit into a countable set Δ, which is shown as follows.

Problem 3

$$\min_{\mathbf{p},\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{q}} F(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}) = R(z_L; y) + (\nu/2)\left(\sum_{l=1}^{L} \|z_l - W_l p_l - b_l\|_2^2 + \sum_{l=1}^{L-1} \|q_l - f_l(z_l)\|_2^2\right)$$
$$\text{s.t. } p_{l+1} = q_l,\ \ p_l \in \Delta = \{\delta_1, \ldots, \delta_m\}$$

where $\delta_i (i = 1, \ldots, m) \in \Delta$ are quantized values, which can be integers or low-precision values. $m = |\Delta|$ is the cardinality of Δ. To address Problem 3, we rewrite it into the following form:

$$\min_{\mathbf{p},\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{q}} R(z_L; y) + \sum_{l=2}^{L} \mathbb{I}(p_l) + (\nu/2)\left(\sum_{l=1}^{L} \|z_l - W_l p_l - b_l\|_2^2 + \sum_{l=1}^{L-1} \|q_l - f_l(z_l)\|_2^2\right)$$
$$\text{s.t. } p_{l+1} = q_l$$

where the indicator function $\mathbb{I}(p_l)$ is defined as follows: $\mathbb{I}(p_l) = 0$ if $p_l \in \Delta$, and $\mathbb{I}(p_l) = +\infty$ if $p_l \notin \Delta$. The augmented Lagrangian of Problem 3 is shown as follows:

$$\beta_\rho(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}, \mathbf{u}) = L_\rho(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}, \mathbf{u}) + \sum_{l=2}^{L} \mathbb{I}(p_l)$$

where L_ρ is the augmented Lagrangian of Problem 2. The extended pdADMM-G-Q algorithm follows the same routine as the pdADMM-G algorithm, where L_ρ is replaced with β_ρ. Due to space limit, the solutions to all subproblems generated by two proposed algorithms are shown in Section B in the Supplementary Material.

IV. Convergence Analysis

In this section, the theoretical convergence of the proposed pdADMM-G algorithm and the pdADMM-G-Q algorithm is provided. Due to space limit, we only provide sketches of proofs in this section, and their details are available in Section C in the Supplementary Material. Our problem formulations are more difficult than existing ADMM literature: the term $\|q_l - f_l(z_l)\|_2^2$ is coupled in the objective, while it is separable in the existing ADMM formulations. To address this, we impose a mild condition that $\partial f_l(z_l)$ is bounded in Assumption 1, and prove that u_l is controlled via q_l and z_l in Lemma 5 in Section C in the Supplementary Material.

First, the proper function, Lipschitz continuity, and coercivity are defined as follows.

Definition 1 (Proper Functions [35]): For a convex function $g(x) : \mathbb{R} \to \mathbb{R} \bigcup \{\pm\infty\}$, g is called proper if $\forall x \in \mathbb{R}$, $g(x) > -\infty$, and $\exists x_0 \in \mathbb{R}$ such that $g(x_0) < +\infty$.

Definition 2 (Lipschitz Continuity): A function $g(x)$ is Lipschitz continuous if there exists a constant $D > 0$ such that $\forall x_1, x_2$, the following holds:

$$\|g(x_1) - g(x_2)\| \leq D\|x_1 - x_2\|.$$

Definition 3 (Coercivity): A function $h(x)$ is coerce over the feasible set $\mathscr{F}$ means that $h(x) \to \infty$ if $x \in \mathscr{F}$ and $\|x\| \to \infty$.

Next, the definition of a quantized stationary point [34] is shown as follows.

Definition 4 (Quantized Stationary Point): The p_l is a quantized stationary point of of Problem 3 if there exists $\tau > 0$ such that

$$p_l \in \arg\min_{\delta \in \Delta} \|\delta - (p_l - \nabla_{p_l} F(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q})/\tau)\|.$$

The quantized stationary point is an extension of the stationary point in the discrete setting, and any global solution p_l to Problem 3 is a quantized stationary point to Problem 3 (see [34, Lemma 3.7]). Then the following assumption is required for convergence analysis.

Assumption 1: $f_l(z_l)$ is Lipschitz continuous with coefficient $S > 0$, $R(Z_L; y)$ is proper, and $F(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q})$ is coercive. Moreover, $\partial f_l(z_l)$ is bounded, i.e., there exists $M > 0$ such that $\|\partial f_l(z_l)\| \leq M$.

Assumption 1 is mild to satisfy: most common activation functions such as rectified linear unit (ReLU) [33] and leaky ReLU [36] satisfy Assumption 1. The risk function $R(z_l; y)$ is only required to be proper, which shows that the convergence condition of our proposed pdADMM-G is milder than that of the dlADMM, which requires $R(z_l; y)$ to be Lipschitz differentiable [7]. Due to the space limit, detailed proofs are provided in Section C in the Supplementary Material. The technical proofs follow a similar routine as dlADMM [7]. The difference consists in the fact that the dual variable u_l is controlled by q_l and z_l (see Lemma 6 in Section C in the Supplementary Material), which holds under Assumption 1, while u_l can be controlled only by z_l in the convergence proof of dlADMM. The first lemma shows that the objective keeps decreasing when ρ is sufficiently large.

Lemma 1 (Objective Decrease): For both the pdADMM-G algorithm and the pdADMM-G-Q algorithm, if $\rho > \max(4\nu S^2, (\sqrt{17} + 1)\nu/2)$, there exist $C_1 = \nu/2 - 2\nu^2 S^2/\rho > 0$ and $C_2 = \rho/2 - 2\nu^2/\rho - \nu/2 > 0$ such that it holds for any $k \in \mathbb{N}$ that

$$\begin{aligned} &L_\rho(\mathbf{p}^k,\mathbf{W}^k,\mathbf{b}^k,\mathbf{z}^k,\mathbf{q}^k,\mathbf{u}^k) - L_\rho(\mathbf{p}^{k+1}, \mathbf{W}^{k+1}, \mathbf{b}^{k+1}, \mathbf{z}^{k+1}, \mathbf{q}^{k+1}, \mathbf{u}^{k+1}) \\ &\geq \sum_{l=2}^{L} (\tau_l^{k+1}/2) \| p_l^{k+1} - p_l^k \|_2^2 + \sum_{l=1}^{L} (\theta_l^{k+1}/2) \| W_l^{k+1} - W_l^k \|_2^2 \\ &\quad + \sum_{l=1}^{L} (\nu/2) \| b_l^{k+1} - b_l^k \|_2^2 + \sum_{l=1}^{L-1} C_1 \| z_l^{k+1} - z_l^k \|_2^2 \\ &\quad + (\nu/2) \| z_L^{k+1} - z_L^k \|_2^2 + \sum_{l=1}^{L-1} C_2 \| q_l^{k+1} - q_l^k \|_2^2 \end{aligned} \tag{1}$$

$$\begin{aligned} &\beta_\rho(\mathbf{p}^k,\mathbf{W}^k,\mathbf{b}^k,\mathbf{z}^k,\mathbf{q}^k,\mathbf{u}^k) - \beta_\rho(\mathbf{p}^{k+1}, \mathbf{W}^{k+1}, \mathbf{b}^{k+1}, \mathbf{z}^{k+1}, \mathbf{q}^{k+1}, \mathbf{u}^{k+1}) \\ &\geq \sum_{l=1}^{L} (\theta_l^{k+1}/2) \| W_l^{k+1} - W_l^k \|_2^2 + \sum_{l=1}^{L} (\nu/2) \| b_l^{k+1} - b_l^k \|_2^2 \\ &\quad + \sum_{l=1}^{L-1} C_1 \| z_l^{k+1} - z_l^k \|_2^2 + (\nu/2) \| z_L^{k+1} - z_L^k \|_2^2 \\ &\quad + \sum_{l=1}^{L-1} C_2 \| q_l^{k+1} - q_l^k \|_2^2. \end{aligned} \tag{2}$$

Sketch of Proof: They can be proven via the optimality conditions of all subproblems, and Assumption 1. □

Lemma 2 shows that the objective is bounded from below when ρ is large enough, and all variables are bounded.

Lemma 2 (Bounded Objective):

1) For the pdADMM-G algorithm, if $\rho > \nu$, then $L_\rho(\mathbf{p}^k, \mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}^k, \mathbf{u}^k)$ is lower bounded. Moreover, $\mathbf{p}^k, \mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}^k$,and $\mathbf{u}^k$ are bounded, i.e., there exist $\mathbb{N}_\mathbf{p}$, $\mathbb{N}_\mathbf{W}$, $\mathbb{N}_\mathbf{b}$, $\mathbb{N}_\mathbf{z}$, $\mathbb{N}_\mathbf{q}$, and $\mathbb{N}_\mathbf{u} > 0$, such that $\|\mathbf{p}^k\| \leq \mathbb{N}_\mathbf{p}$, $\|\mathbf{W}^k\| \leq \mathbb{N}_\mathbf{W}$, $\|\mathbf{b}^k\| \leq \mathbb{N}_\mathbf{b}$, $\|\mathbf{z}^k\| \leq \mathbb{N}_\mathbf{z}$, $\|\mathbf{q}^k\| \leq \mathbb{N}_\mathbf{q}$, and $\|\mathbf{u}^k\| \leq \mathbb{N}_\mathbf{u}$.
2) For the pdADMM-G-Q algorithm, if $\rho > \nu$, then $\beta_\rho(\mathbf{p}^k, \mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}^k, \mathbf{u}^k)$ is lower bounded. Moreover, $\mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}^k$,and $\mathbf{u}^k$ are bounded, i.e., there exist $\overline{\mathbb{N}}_\mathbf{W}$, $\overline{\mathbb{N}}_\mathbf{b}$, $\overline{\mathbb{N}}_\mathbf{z}$, $\overline{\mathbb{N}}_\mathbf{q}$, and $\overline{\mathbb{N}}_\mathbf{u} > 0$, such that $\|\mathbf{W}^k\| \leq \overline{\mathbb{N}}_\mathbf{W}$, $\|\mathbf{b}^k\| \leq \overline{\mathbb{N}}_\mathbf{b}$, $\|\mathbf{z}^k\| \leq \overline{\mathbb{N}}_\mathbf{z}$, $\|\mathbf{q}^k\| \leq \overline{\mathbb{N}}_\mathbf{q}$, and $\|\mathbf{u}^k\| \leq \overline{\mathbb{N}}_\mathbf{u}$.

Sketch of Proof: We only show the sketch proof of (1) because (2) follows the same routine as (1). To prove the boundness of L_ρ, we should prove the following:

$$\begin{aligned} &L_\rho(\mathbf{p}^k, \mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}^k, \mathbf{u}^k) \\ &\quad \geq F(\mathbf{p}^k, \mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}') + ((\rho - \nu)/2) \| p_{l+1}^k - q_l^k \|_2^2 \\ &\quad > -\infty \end{aligned}$$

where $p_{l+1}^k = q_l'$. Therefore, $F(\mathbf{p}^k, \mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}')$ and $((\rho - \nu)/2)\|p_{l+1}^k - q_l^k\|_2^2$ are upper bounded by $L_\rho(\mathbf{p}^k, \mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}^k, \mathbf{u}^k)$ and hence $L_\rho(\mathbf{p}^0, \mathbf{W}^0, \mathbf{b}^0, \mathbf{z}^0, \mathbf{q}^0, \mathbf{u}^0)$ (see Lemma 1). The boundness of variables can be obtained via Assumption 1. □

Based on Lemmas 1 and 2, the following theorem ensures that the objective is convergent.

Theorem 1 (Convergent Objective):

1) For the pdADMM-G algorithm, if $\rho > \max(4\nu S^2, (\sqrt{17} + 1)\nu/2)$, then $L_\rho(\mathbf{p}^k, \mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}^k, \mathbf{u}^k)$ is convergent. Moreover, $\lim_{k\to\infty} \|\mathbf{p}^{k+1} - \mathbf{p}^k\|_2^2 = 0$,

$\lim_{k\to\infty}\|\mathbf{W}^{k+1}-\mathbf{W}^k\|_2^2=0$, $\lim_{k\to\infty}\|\mathbf{b}^{k+1}-\mathbf{b}^k\|_2^2=0$, $\lim_{k\to\infty}\|\mathbf{z}^{k+1}-\mathbf{z}^k\|_2^2=0$, $\lim_{k\to\infty}\|\mathbf{q}^{k+1}-\mathbf{q}^k\|_2^2=0$, $\lim_{k\to\infty}\|\mathbf{u}^{k+1}-\mathbf{u}^k\|_2^2=0$.

2) For the pdADMM-G-Q algorithm, if $\rho>\max(4\nu S^2,(\sqrt{17}+1)\nu/2)$, then $\beta_\rho(\mathbf{p}^k,\mathbf{W}^k,\mathbf{b}^k,\mathbf{z}^k,\mathbf{q}^k,\mathbf{u}^k)$ is convergent. Moreover, $\lim_{k\to\infty}\|\mathbf{W}^{k+1}-\mathbf{W}^k\|_2^2=0$, $\lim_{k\to\infty}\|\mathbf{b}^{k+1}-\mathbf{b}^k\|_2^2=0$, $\lim_{k\to\infty}\|\mathbf{z}^{k+1}-\mathbf{z}^k\|_2^2=0$, $\lim_{k\to\infty}\|\mathbf{q}^{k+1}-\mathbf{q}^k\|_2^2=0$, $\lim_{k\to\infty}\|\mathbf{u}^{k+1}-\mathbf{u}^k\|_2^2=0$.

Sketch of Proof: This theorem can be derived by taking the limit on both sides of Inequality (1). □

The third lemma guarantees that the subgradient of the objective is upper bounded, which is stated as follows.

Lemma 3 (Bounded Subgradient):

1) For the pdADMM-G algorithm, there exists a constant $C>0$ and $g^{k+1}\in\partial L_\rho(\mathbf{p}^{k+1},\mathbf{W}^{k+1},\mathbf{b}^{k+1},\mathbf{z}^{k+1},\mathbf{q}^{k+1},\mathbf{u}^{k+1})$ such that

$$\|g^{k+1}\|\le C(\|\mathbf{p}^{k+1}-\mathbf{p}^k\|+\|\mathbf{W}^{k+1}-\mathbf{W}^k\|+\|\mathbf{b}^{k+1}-\mathbf{b}^k\| +\|\mathbf{z}^{k+1}-\mathbf{z}^k\|+\|\mathbf{q}^{k+1}-\mathbf{q}^k\|+\|\mathbf{u}^{k+1}-\mathbf{u}^k\|).$$

2) For the pdADMM-G-Q algorithm, there exists a constant $\overline{C}>0$, $\overline{g}_{\mathbf{W}}^{k+1}\in\nabla_{\mathbf{W}^{k+1}}\beta_\rho(\mathbf{p}^{k+1},\mathbf{W}^{k+1},\mathbf{b}^{k+1},\mathbf{z}^{k+1},\mathbf{q}^{k+1},\mathbf{u}^{k+1})$, $\overline{g}_{\mathbf{b}}^{k+1}\in\nabla_{\mathbf{b}^{k+1}}\beta_\rho(\mathbf{p}^{k+1},\mathbf{W}^{k+1},\mathbf{b}^{k+1},\mathbf{z}^{k+1},\mathbf{q}^{k+1},\mathbf{u}^{k+1})$, $\overline{g}_{\mathbf{z}}^{k+1}\in\partial_{\mathbf{z}^{k+1}}\beta_\rho(\mathbf{p}^{k+1},\mathbf{W}^{k+1},\mathbf{b}^{k+1},\mathbf{z}^{k+1},\mathbf{q}^{k+1},\mathbf{u}^{k+1})$, $\overline{g}_{\mathbf{q}}^{k+1}\in\nabla_{\mathbf{q}^{k+1}}\beta_\rho(\mathbf{p}^{k+1},\mathbf{W}^{k+1},\mathbf{b}^{k+1},\mathbf{z}^{k+1},\mathbf{q}^{k+1},\mathbf{u}^{k+1})$, $\overline{g}_{\mathbf{u}}^{k+1}\in\nabla_{\mathbf{u}^{k+1}}\beta_\rho(\mathbf{p}^{k+1},\mathbf{W}^{k+1},\mathbf{b}^{k+1},\mathbf{z}^{k+1},\mathbf{q}^{k+1},\mathbf{u}^{k+1})$ such that

$$\begin{aligned}\|\overline{g}_{\mathbf{W}}^{k+1}\|&\le\overline{C}(\|\mathbf{W}^{k+1}-\mathbf{W}^k\|+\|\mathbf{b}^{k+1}-\mathbf{b}^k\|\\&\quad+\|\mathbf{z}^{k+1}-\mathbf{z}^k\|+\|\mathbf{q}^{k+1}-\mathbf{q}^k\|+\|\mathbf{u}^{k+1}-\mathbf{u}^k\|)\\ \|\overline{g}_{\mathbf{b}}^{k+1}\|&\le\overline{C}(\|\mathbf{W}^{k+1}-\mathbf{W}^k\|+\|\mathbf{b}^{k+1}-\mathbf{b}^k\|\\&\quad+\|\mathbf{z}^{k+1}-\mathbf{z}^k\|+\|\mathbf{q}^{k+1}-\mathbf{q}^k\|+\|\mathbf{u}^{k+1}-\mathbf{u}^k\|)\\ \|\overline{g}_{\mathbf{z}}^{k+1}\|&\le\overline{C}(\|\mathbf{W}^{k+1}-\mathbf{W}^k\|+\|\mathbf{b}^{k+1}-\mathbf{b}^k\|\\&\quad+\|\mathbf{z}^{k+1}-\mathbf{z}^k\|+\|\mathbf{q}^{k+1}-\mathbf{q}^k\|+\|\mathbf{u}^{k+1}-\mathbf{u}^k\|)\\ \|\overline{g}_{\mathbf{q}}^{k+1}\|&\le\overline{C}(\|\mathbf{W}^{k+1}-\mathbf{W}^k\|+\|\mathbf{b}^{k+1}-\mathbf{b}^k\|\\&\quad+\|\mathbf{z}^{k+1}-\mathbf{z}^k\|+\|\mathbf{q}^{k+1}-\mathbf{q}^k\|+\|\mathbf{u}^{k+1}-\mathbf{u}^k\|)\\ \|\overline{g}_{\mathbf{u}}^{k+1}\|&\le\overline{C}(\|\mathbf{W}^{k+1}-\mathbf{W}^k\|+\|\mathbf{b}^{k+1}-\mathbf{b}^k\|\\&\quad+\|\mathbf{z}^{k+1}-\mathbf{z}^k\|+\|\mathbf{q}^{k+1}-\mathbf{q}^k\|+\|\mathbf{u}^{k+1}-\mathbf{u}^k\|).\end{aligned}$$

Sketch of Proof: To prove this lemma, the subgradient is proven to be upper bounded by the linear combination of $\|\mathbf{p}^{k+1}-\mathbf{p}^k\|$, $\|\mathbf{W}^{k+1}-\mathbf{W}^k\|$, $\|\mathbf{b}^{k+1}-\mathbf{b}^k\|$, $\|\mathbf{z}^{k+1}-\mathbf{z}^k\|$, $\|\mathbf{q}^{k+1}-\mathbf{q}^k\|$, and $\|\mathbf{u}^{k+1}-\mathbf{u}^k\|$. □

Now based on Theorem 1 and Lemma 3, the convergence of the pdADMM-G algorithm to a stationary point is presented in the following theorem.

Theorem 2 (Convergence of the pdADMM-G Algorithm): If $\rho>\max(4\nu S^2,(\sqrt{17}+1)\nu/2)$, then for the variables $(\mathbf{p},\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{q},\mathbf{u})$ in Problem 2, starting from any $(\mathbf{p}^0,\mathbf{W}^0,\mathbf{b}^0,\mathbf{z}^0,\mathbf{q}^0,\mathbf{u}^0)$, $(\mathbf{p}^k,\mathbf{W}^k,\mathbf{b}^k,\mathbf{z}^k,\mathbf{q}^k,\mathbf{u}^k)$ has at least a limit point $(\mathbf{p}^*,\mathbf{W}^*,\mathbf{b}^*,\mathbf{z}^*,\mathbf{q}^*,\mathbf{u}^*)$, and any limit point is a stationary point of Problem 2. That is, $0\in\partial L_\rho(\mathbf{p}^*,\mathbf{W}^*,\mathbf{b}^*,\mathbf{z}^*,\mathbf{q}^*,\mathbf{u}^*)$. In other words

$$\begin{aligned}p_{l+1}^*=q_l^*,\quad \nabla_{\mathbf{p}^*}L_\rho(\mathbf{p}^*,\mathbf{W}^*,\mathbf{b}^*,\mathbf{z}^*,\mathbf{q}^*,\mathbf{u}^*)&=0\\ \nabla_{\mathbf{W}^*}L_\rho(\mathbf{p}^*,\mathbf{W}^*,\mathbf{b}^*,\mathbf{z}^*,\mathbf{q}^*,\mathbf{u}^*)&=0\\ \nabla_{\mathbf{b}^*}L_\rho(\mathbf{p}^*,\mathbf{W}^*,\mathbf{b}^*,\mathbf{z}^*,\mathbf{q}^*,\mathbf{u}^*)&=0\\ 0\in\partial_{\mathbf{z}^*}L_\rho(\mathbf{p}^*,\mathbf{W}^*,\mathbf{b}^*,\mathbf{z}^*,\mathbf{q}^*,\mathbf{u}^*)&\\ \nabla_{\mathbf{q}^*}L_\rho(\mathbf{p}^*,\mathbf{W}^*,\mathbf{b}^*,\mathbf{z}^*,\mathbf{q}^*,\mathbf{u}^*)&=0.\end{aligned}$$

Sketch of Proof: This theorem can be derived directly from Lemmas 2 and 3. □

Theorem 2 shows that our proposed pdADMM-G algorithm converges for sufficiently large ρ, which is consistent with previous literature [7]. Similarly, the convergence of the proposed pdADMM-G-Q algorithm is shown as follows.

Theorem 3 (Convergence of the pdADMM-G-Q Algorithm): If $\rho>\max(4\nu S^2,(\sqrt{17}+1)\nu/2)$, then for the variables $(\mathbf{p},\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{q},\mathbf{u})$ in Problem 3, starting from any $(\mathbf{p}^0,\mathbf{W}^0,\mathbf{b}^0,\mathbf{z}^0,\mathbf{q}^0,\mathbf{u}^0)$, $(\mathbf{p}^k,\mathbf{W}^k,\mathbf{b}^k,\mathbf{z}^k,\mathbf{q}^k,\mathbf{u}^k)$ has at least a limit point $(\mathbf{p}^*,\mathbf{W}^*,\mathbf{b}^*,\mathbf{z}^*,\mathbf{q}^*,\mathbf{u}^*)$, and any limit point $(\mathbf{W}^*,\mathbf{b}^*,\mathbf{z}^*,\mathbf{q}^*,\mathbf{u}^*)$ is a stationary point of Problem 3. Moreover, if τ_l^{k+1} is bounded, then $\mathbf{p}^*$ is a quantized stationary point of Problem 3. That is,

$$\begin{aligned}p_{l+1}^*=q_l^*,\quad \nabla_{\mathbf{W}^*}\beta_\rho(\mathbf{p}^*,\mathbf{W}^*,\mathbf{b}^*,\mathbf{z}^*,\mathbf{q}^*,\mathbf{u}^*)&=0\\ \nabla_{\mathbf{b}^*}\beta_\rho(\mathbf{p}^*,\mathbf{W}^*,\mathbf{b}^*,\mathbf{z}^*,\mathbf{q}^*,\mathbf{u}^*)&=0\\ 0\in\partial_{\mathbf{z}^*}\beta_\rho(\mathbf{p}^*,\mathbf{W}^*,\mathbf{b}^*,\mathbf{z}^*,\mathbf{q}^*,\mathbf{u}^*)&\\ \nabla_{\mathbf{q}^*}\beta_\rho(\mathbf{p}^*,\mathbf{W}^*,\mathbf{b}^*,\mathbf{z}^*,\mathbf{q}^*,\mathbf{u}^*)&=0\\ p_l^*\in\arg\min_{\delta\in\Delta}\left\|\delta-\left(p_l^*-\nabla_{p_l^*}F(\mathbf{p}^*,\mathbf{W}^*,\mathbf{b}^*,\mathbf{z}^*,\mathbf{q}^*)/\tau_l^*\right)\right\|&\end{aligned}$$

where τ_l^* is a limit point of τ_l^k.

Sketch of Proof: This theorem is proven using a similar procedure as Theorem 2, and the definition of the quantized stationary point. □

The only difference between Theorems 2 and 3 is that $\mathbf{p}^*$ is a stationary point in Problem 2 and a quantized stationary point in Problem 3. Next, the following theorem ensures the sublinear convergence rate $o(1/k)$ of the proposed pdADMM-G algorithm and the pdADMM-G-Q algorithm.

Theorem 4 (Convergence Rate):

1) For the pdADMM-G algorithm and a sequence $(\mathbf{p}^k,\mathbf{W}^k,\mathbf{b}^k,\mathbf{z}^k,\mathbf{q}^k,\mathbf{u}^k)$, define $c_k=\min_{0\le i\le k}(\sum_{l=2}^L(\tau_l^{i+1}/2)\|p_l^{i+1}-p_l^i\|_2^2+\sum_{l=1}^L(\theta_l^{i+1}/2)\|W_l^{i+1}-W_l^i\|_2^2+\sum_{l=1}^L(\nu/2)\|b_l^{i+1}-b_l^i\|_2^2+\sum_{l=1}^{L-1}C_1\|z_l^{i+1}-z_l^i\|_2^2+(\nu/2)\|z_L^{i+1}-z_L^i\|_2^2+\sum_{l=1}^{L-1}C_2\|q_l^{i+1}-q_l^i\|_2^2)$ where $C_1=\nu/2-2\nu^2S^2/\rho>0$ and $C_2=\rho/2-2\nu^2/\rho-\nu/2>0$, then the convergence rate of c_k is $o(1/k)$.
2) For the pdADMM-G-Q algorithm and a sequence $(\mathbf{W}^k,\mathbf{b}^k,\mathbf{z}^k,\mathbf{q}^k,\mathbf{u}^k)$, define $d_k=\min_{0\le i\le k}(\sum_{l=1}^L(\theta_l^{i+1}/2)\|W_l^{i+1}-W_l^i\|_2^2+\sum_{l=1}^L(\nu/2)\|b_l^{i+1}-b_l^i\|_2^2+\sum_{l=1}^{L-1}C_1\|z_l^{i+1}-z_l^i\|_2^2+(\nu/2)\|z_L^{i+1}-z_L^i\|_2^2+\sum_{l=1}^{L-1}C_2\|q_l^{i+1}-q_l^i\|_2^2)$ where $C_1=\nu/2-2\nu^2S^2/\rho>0$ and $C_2=\rho/2-2\nu^2/\rho-\nu/2>0$, then the convergence rate of d_k is $o(1/k)$.

Sketch of Proof:

1) To prove the convergence rate $o(1/k)$, c_k satisfies three conditions: (a) $c_k\ge c_{k+1}$, (b) $\sum_{k=0}^\infty c_k$ is bounded, and (c) $c_k\ge 0$.
2) d_k can be proven using a similar procedure as (1).

□

TABLE II
DATASET STATISTICS

Dataset	Node#	Edge#	Class#	Feature#	Training Set#	Validation Set#	Test Set#
Cora	2,485	10,556	7	1,433	140	500	1,000
PubMed	19,717	88,648	3	500	60	500	1,000
Citeseer	2,110	9,104	6	3,703	120	500	1,000
Amazon Computers	13,381	491,722	10	767	200	1,000	1,000
Amazon Photo	7,487	238,162	8	745	160	1,000	1,000
Coauthor CS	18,333	163,788	15	6,805	300	1,000	1,000
Coauthor Physics	34,493	495,924	5	8,415	100	1,000	1,000
Flickr	89,250	899,756	7	500	44,625	22,312	22,313
Ogbn-Arxiv	169,343	1,166,243	40	128	90,941	29,799	48,603

V. EXPERIMENTS

In this section, we evaluate the performance of the proposed pdADMM-G algorithm and the proposed pdADMM-G-Q algorithm on GA-MLP models using nine benchmark datasets. Convergence and computational overheads are demonstrated on different datasets. Speedup and test performance are compared with several state-of-the-art optimizers. All experiments were conducted on the Amazon Web Services (AWS) p2.16xlarge instance, with 16 NVIDIA K80 GPUs, 64vCPUs, a processor Intel[1] Xeon[1] CPU E5-2686 v4 at 2.30 GHz, and 732 GiB of RAM.

A. Datasets and Settings

Nine benchmark datasets were used for experimental evaluation, whose statistics are shown in Table II. Each dataset is split into a training set, a validation set, and a test set. Due to space limit, their details can be found in Section D1 in the Supplementary Material.

When it comes to experimental settings, we set $K = 4$ for the multihop operator Ψ, and defined a diagonal degree matrix D where $D_{ii} = \sum_{j=1}^{|V|} A_{ij}$, and a renormalized adjacency matrix $\tilde{A} = (D+I)^{-1/2}(A+I)(D+I)^{-1/2} \in \mathbb{R}^{|V|\times|V|}$ [1]. Moreover, we set $\Psi = \{I, \tilde{A}, \tilde{A}^2, \tilde{A}^3\}$ [2]. For all GA-MLP models, the activation function was set to ReLU. The loss function was set to the cross-entropy loss. For the pdADMM-G-Q algorithm, $\Delta = \{-1, 0, 1, \ldots, 20\}$ in Problem 3, and $\mathbf{p}$ was quantized by default.

B. Comparison Methods

GD and its variants are state-of-the-art optimizers and hence served as comparison methods. For GD-based methods, all datasets were used for training models in a full-batch fashion. All hyperparameters were chosen by maximizing the performance of validation sets. Due to space limit, hyperparameter settings of all methods are shown in Section D2 in the Supplementary Material. The following are their brief introductions.

1) *GD [37]:* The GD and its variants are the most popular deep learning optimizers. The GD updates parameters simply based on their gradients.
2) *Adaptive Learning Rate Method (Adadelta) [38]:* The Adadelta is proposed to overcome the sensitivity to hyperparameter selection.
3) *Adaptive Gradient Algorithm (Adagrad) [39]:* Adagrad is an improved version of GD: rather than fixing the learning rate during training, it adapts the learning rate to the hyperparameter.
4) *Adaptive Momentum Estimation (Adam) [40]:* Adam is the most popular optimization method for deep learning models. It estimates the first and second momentum to correct the biased gradient, and thus accelerates empirical convergence.

C. Convergence

First, to validate the convergence of two proposed algorithms, we set up a GA-MLP model with ten layers, each of which has 1000 neurons. The number of epochs was set to 100. ν and ρ were set to 0.01 and 1, respectively.

Fig. 2 demonstrates objectives and residuals of two proposed algorithms on four datasets. Overall, the objectives and residuals of the two proposed algorithms are convergent. From Fig. 2(a) and (c), the objectives of the two proposed algorithms decrease drastically at the first 50 epochs and then drop smoothly to the end. The objectives on the PubMed dataset achieve the lowest among all four datasets, whereas these on the Coauthor Computer Science (CS) dataset are the highest, which still reach near 10^5 at the 100th epoch. As for residuals, even though the residuals of the pdADMM-G-Q algorithm are higher than these of the pdADMM-G algorithm initially, they both converge sublinearly to 0, which is consistent with Theorem 2 and Theorem 3. Specifically, as shown in Fig. 2(b) and (d), the residuals on the Cora dataset decrease more slowly with fluctuation than these on other datasets, while residuals on the Amazon Computers and Amazon Photo datasets demonstrate the fastest decreasing speed at the first 40 epochs before reaching a plateau less than 10^{-6}. The residuals on the PubMed dataset accomplish the lowest values among all four datasets again with a value of less than 10^{-7}.

D. Speedup

Next, we investigate the speedup of the pdADMM-G algorithm in the large deep GA-MLP models. The running time per epoch was an average of ten epochs. ρ and ν were both set to 10^{-3}. We investigate the speedup concerning two factors: the number of layers and the number of GPUs.

For the relationship between the speedup and the number of layers, the pdADMM-G algorithm in the GA-MLP models with 4000 neurons was tested. The number of layers ranged from 8 to 17. The speedup on small datasets and large datasets are shown in Fig. 3(a) and (b), respectively. Overall, the speedup of the proposed pdADMM-G increases linearly with the number of layers. For example, the speedups on the Cora dataset and the Amazon Computers dataset rise from 3 and 3.5 gradually to 4 and 4.5, respectively. The speedup on the PubMed dataset achieves the lowest with a value of less than 3, whereas that on the Coauthor CS dataset at least doubles that on any other small dataset, with a peak of 6. Moreover, the speedup is more obvious on large datasets. For example, when the slopes of speedups are compared, the slope on the Flickr dataset is at least five times much steeper than that on the

[1]Registered trademark.

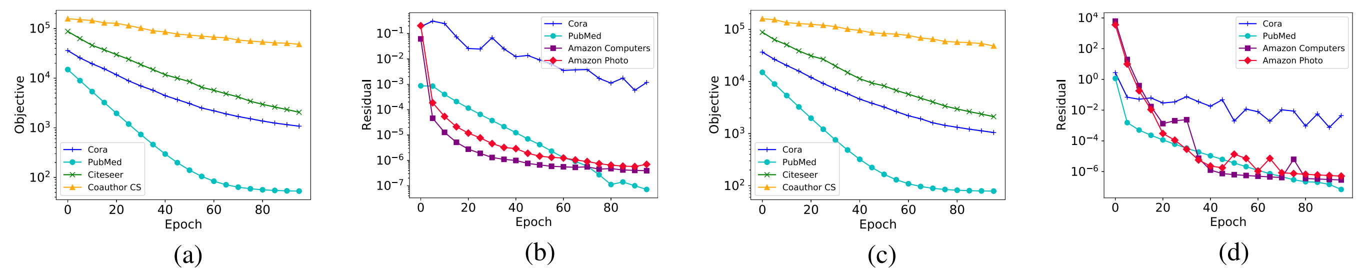

Fig. 2. Convergence curves of the pdADMM-G algorithm and the pdADMM-G-Q algorithm in four datasets: they both converge. (a) pdADMM-G objective. (b) pdADMM-G residual. (c) pdADMM-G-Q objective. (d) pdADMM-G-Q residual.

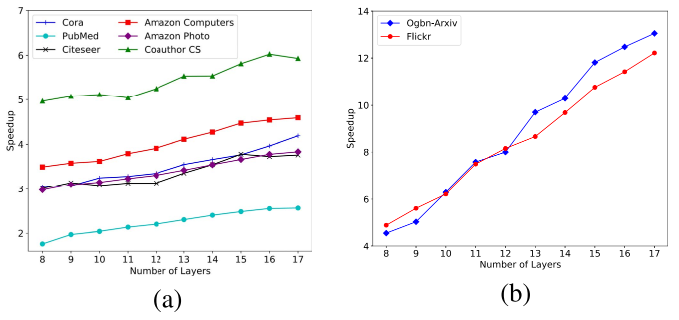

Fig. 3. Speedup of the proposed pdADMM-G on different datasets concerning the number of layers: the speedup increases linearly with the number of layers, and the slopes of speedups are higher on large datasets than those on small datasets. (a) Speedup on small datasets. (b) Speedup on large datasets.

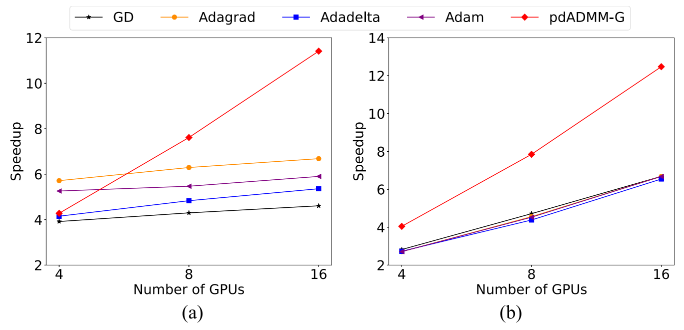

Fig. 4. Speedup of all methods on two large datasets concerning the number of GPUs: speedups of the proposed pdADMM-G are higher than these of all comparison methods. (a) Flickr. (b) Ogbn-Arxiv.

Coauthor CS dataset. The same trend is applied to the Ogbn-Arxiv dataset. This means that our proposed pdADMM-G algorithm is more suitable for large datasets.

For the relationship between the speedup and the number of GPUs, we set up a large GA-MLP model with 16 layers and 4000 neurons and kept all hyperparameters in the previous experiment. The speedup of our proposed pdADMM-G algorithm was compared with all comparison methods. Fig. 4 shows all speedups on two large datasets. The proposed pdADMM-G algorithm achieves a higher speedup than any GD-based method. For example, the speedups of eight GPUs are nearly 8 on the Flickr dataset and the Ogbn-Arxiv dataset, while the best speedups achieved via comparison methods are in the vicinity of 6 and 5 on two datasets, respectively. We also observe that while speedups of all methods scale linearly with the number of GPUs, the slopes of our proposed pdADMM-G algorithm are steeper than these of any comparison method. For example, the slope of our proposed pdADMM-G algorithm on the Flickr dataset is more than ten times steeper than that of Adam. All comparison methods show similar flat slopes, and they achieve a higher slope of the speedup on the Ogbn-Arxiv dataset than that on the Flickr dataset.

In summary, the speedup of our proposed pdADMM-G algorithm scales linearly with the number of layers and the number of GPUs. Moreover, its speedup is superior to any other comparison method significantly by more than ten times.

E. Communication Overheads

Then, it is necessary to explore how many communication overheads can be reduced using the proposed quantization technique on different quantization levels. To achieve this, we established a large GA-MLP model with ten layers, each of which consists of 1000 neurons. We set up three quantization cases: no quantization, the quantization concerning $\mathbf{p}$ only, and the quantization concerning both $\mathbf{p}$ and $\mathbf{q}$. For every quantization case, we also set up two different quantization sizes: 8 bits and 16 bits. Fig. 5 demonstrates the relationship between the test accuracy and communication overheads for different quantization cases and sizes on three datasets. Overall, communication overheads can be reduced significantly by the proposed quantization technique. The amount of reduction depends on different quantization cases and sizes. Generally speaking, the more variables are quantized and the fewer bits are compressed, then the more savings in communications can be achieved. Take the Citeseer dataset as an example, while all algorithms reach the same test accuracy above 70%, the proposed pdADMM-G (i.e., no quantization) consumes the most communication costs with the value of around 1.4×10^9 bytes. If the variable $\mathbf{p}$ is quantized using 16 bits, the communication overhead drops by 10%, and then using 8 bits saves another 5%. When variables $\mathbf{p}$ and $\mathbf{q}$ are both quantized, the communication overhead tumbles down to 1.2×10^9 bytes, which means decreases by 16.7% when it is compared with the case where only $\mathbf{p}$ is quantized. When variables are compressed to 8 bits instead of 16 bits, the communication overhead slips further to 1.1×10^9, a nearly 30% decline. The same trend is applied to the other two datasets, and they accomplish a shrink of communication overheads by 33% and 45%, respectively. This demonstrates that our proposed quantization technique is effective for reducing unnecessary communication costs without loss of performance. We also observe that the

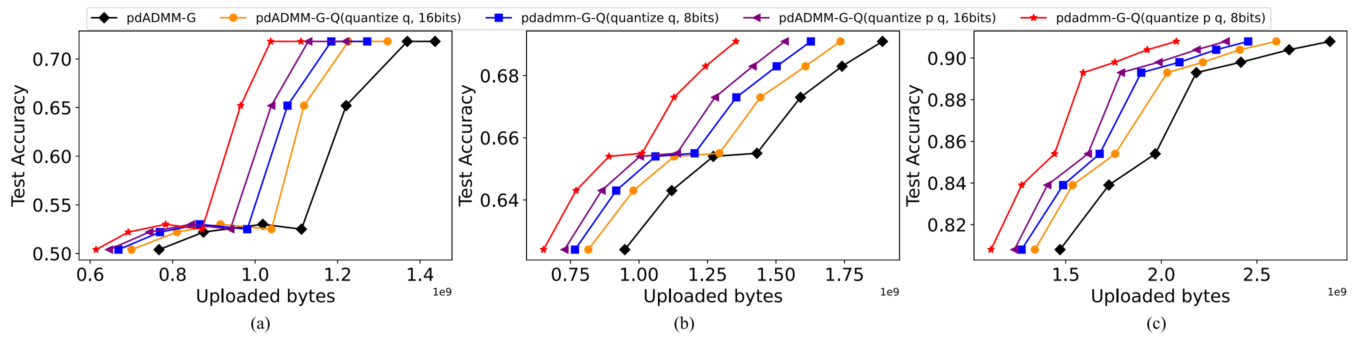

Fig. 5. Communication overheads of two proposed algorithms on three datasets: the quantization concerning both **p** and **q** using eight bits reduces the communication overheads by up to 45% without loss of performance. (a) Citeseer. (b) Amazon computers. (c) Coauthor CS.

TABLE III

TEST PERFORMANCE OF ALL METHODS WHEN THE NUMBER OF NEURONS IS 100: TWO PROPOSED ALGORITHMS OUTPERFORM ALL COMPARISON METHODS

Dataset	Cora	PubMed	Citeseer
GD	0.730±0.022	0.638± 0.080	0.637±0.040
Adadelta	0.671±0.064	0.705±0.038	0.620±0.016
Adagrad	0.726±0.025	0.753± 0.015	0.601±0.037
Adam	0.725±0.036	0.742±0.007	0.631±0.018
pdADMM-G	0.784±0.003	**0.784±0.004**	0.709±0.003
pdADMM-G-Q	**0.788±0.003**	0.782±0.003	**0.712± 0.001**
Dataset	Amazon Computers	Amazon Photo	Coauthor CS
GD	0.646±0.032	0.735± 0.169	0.884±0.010
Adadelta	0.136±0.060	0.369± 0.045	0.787±0.086
Adagrad	0.688±0.023	0.813± 0.018	0.887±0.007
Adam	0.724±0.010	0.855±0.009	0.883±0.009
pdADMM-G	**0.735±0.006**	**0.856±0.011**	**0.915±0.004**
pdADMM-G-Q	0.687± 0.054	0.832±0.010	0.914±0.003
Dataset	Coauthor Physics	Flickr	Ogbn-Arxiv
GD	0.909±0.007	0.466±0.007	0.361±0.063
Adadelta	0.915±0.014	0.461±0.008	0.523± 0.030
Adagrad	0.916±0.012	0.481±0.003	0.567±0.016
Adam	0.912±0.016	0.512 ±0.004	**0.674± 0.006**
pdADMM-G	**0.921±0.003**	**0.513±0.002**	0.647±0.002
pdADMM-G-Q	0.919±0.002	0.507±0.003	0.655±0.002

TABLE IV

TEST PERFORMANCE OF ALL METHODS WHEN THE NUMBER OF NEURONS IS 500: TWO PROPOSED ALGORITHMS OUTPERFORM ALL COMPARISON METHODS

Dataset	Cora	PubMed	Citeseer
GD	0.757±0.024	0.699±0.655	0.680±0.014
Adadelta	0.717±0.063	0.722±0.696	0.564±0.028
Adagrad	0.776±0.013	0.759±0.761	0.650 ±0.038
Adam	0.771±0.020	0.778±0.767	0.662 ±0.021
pdADMM-G	**0.786±0.005**	0.786±0.786	**0.713±0.007**
pdADMM-G-Q	**0.786±0.005**	**0.788±0.787**	0.712±0.005
Dataset	Amazon Computers	Amazon Photo	Coauthor CS
GD	0.707±0.012	0.817±0.005	0.906±0.005
Adadelta	0.243±0.063	0.380±0.069	0.880±0.011
Adagrad	**0.753±0.009**	0.866±0.007	0.911±0.003
Adam	0.739±0.022	**0.880± 0.016**	0.898±0.013
pdADMM-G	0.751±0.008	0.873±0.004	**0.920±0.002**
pdADMM-G-Q	0.748±0.004	0.865±0.007	0.919±0.003
Dataset	Coauthor Physics	Flickr	Ogbn-Arxiv
GD	0.917±0.004	0.466±0.001	0.436±0.042
Adadelta	0.917±0.004	0.462±0.001	0.584±0.031
Adagrad	0.914±0.004	0.487±0.005	0.630±0.016
Adam	0.914±0.002	**0.517±0.002**	**0.682±0.010**
pdADMM-G	**0.918±0.003**	0.515±0.002	0.655±0.001
pdADMM-G-Q	**0.918±0.002**	0.512±0.003	0.657±0.002

Coauthor CS dataset is the largest dataset among the three, and it accomplishes the greatest communication reduction.

F. Performance

Finally, we evaluate the performance of two proposed algorithms against all comparison methods on nine benchmark datasets. We set up two standard GA-MLP models with ten layers but different neurons: the first model has 100 neurons, while the second model has 500 neurons. Following the greedy layerwise training strategy [31], we first trained a two-layer GA-MLP model, and then three more layers were added to training, and finally, all ten layers were involved. The number of epochs was set to 200. We repeated all experiments five times and reported their means and the standard deviations. Due to space limit, hyperparameter settings and the performance of validation sets are shown in Section D2 and D3 in the Supplementary Material, respectively.

Table III demonstrates the performance of all methods when the number of neurons is 100. In summary, the two proposed algorithms outperform all comparison methods slightly: they occupy the best algorithms on eight datasets out of the total nine datasets. For example, they both achieve 78% test accuracy on the Cora dataset, whereas the best comparison method is GD, which only reaches 73% test accuracy, and is at least 6% lower than the two proposed algorithms. As another example, two proposed algorithms accomplish 78% test accuracy on the PubMed dataset, 4% better than that achieved by Adagrad, whose performance is the best aside from the two proposed algorithms. The Citeseer dataset shows the largest performance gap between the two proposed algorithms and all comparison methods. Two proposed algorithms reach the level of 70% test accuracy, whereas all comparison methods fall in the vicinity of 60% test accuracy. In other words, the two proposed algorithms outperform all comparison methods by more than 10%. For two proposed algorithms, the proposed pdADMM-G algorithm outperforms marginally the proposed pdADMM-G-Q algorithm due to the quantization technique. Their largest performance gap is 5%, which is achieved on the Amazon Computers dataset. The Adam is the best comparison method overall, and it serves as the best algorithm on the

Ogbn-Arxiv dataset. The Adadelta performs the worst among all comparison methods, whose performance is significantly lower than any other method on several datasets such as the Amazon Computers dataset, the Amazon Photo dataset, and the Coauthor CS dataset. Last but not least, the standard deviations of all methods remain low, and this shows that they are robust to different initializations.

Table IV shows the performance of all methods when the number of neurons is 500. In general, two proposed algorithms still reach a better performance than all comparison methods, but the gap is more narrow. For example, in Table III, the proposed pdADMM-G algorithm achieves the best on the Amazon Computers dataset. However, it is surpassed by Adagrad slightly in Table IV. We also observe that a GA-MLP model with 500 neurons performs better than that with 100 neurons, which are trained by the same algorithm. This makes sense since the wider a model is, the more expressiveness it is equipped with.

VI. Conclusion

The GA-MLP models are attractive to the deep learning community due to potential resistance to some problems of GNNs such as over-smoothing and over-squashing. In this article, we propose a novel pdADMM-G algorithm to achieve parallel training of GA-MLP models, which is accomplished by breaking the layer dependency. The extended pdADMM-G-Q algorithm reduces communication overheads by the introduction of the quantization technique. Their theoretical convergence to a (quantized) stationary point of the problem is guaranteed with a sublinear convergence rate $o(1/k)$, where k is the number of iterations. Extensive experiments verify that the two proposed algorithms not only converge in terms of objectives and residuals, and accelerate the training of deep GA-MLP models, but also stand out among all the existing state-of-the-art optimizers on nine benchmark datasets. Moreover, the pdADMM-G-Q algorithm reduces communication overheads by up to 45% without loss of performance.

References

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.

[2] L. Chen, Z. Chen, and J. Bruna, "On graph neural networks versus graph-augmented mlps," in *Proc. 9th Int. Conf. Learn. Represent.*, 2021.

[3] J. Topping, F. D. Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein, "Understanding over-squashing and bottlenecks on graphs via curvature," in *Proc. Int. Conf. Learn. Represent.*, 2022.

[4] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

[5] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6861–6871.

[6] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, "Training neural networks without gradients: A scalable admm approach," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2722–2731.

[7] J. Wang, F. Yu, X. Chen, and L. Zhao, "Admm for efficient deep learning with global convergence," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 111–119.

[8] W. Wen et al., "TernGrad: Ternary gradients to reduce communication in distributed deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1509–1519.

[9] A. Sergeev and M. Del Balso, "Horovod: Fast and easy distributed deep learning in TensorFlow," 2018, *arXiv:1802.05799*.

[10] B. C. Ooi et al., "SINGA: A distributed deep learning platform," in *Proc. 23rd ACM Int. Conf. Multimedia*, 2015, pp. 685–688.

[11] T. Chen et al., "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems," 2015, *arXiv:1512.01274*.

[12] S. H. Hashemi, S. A. Jyothi, and R. H. Campbell, "TicTac: Accelerating distributed deep learning with communication scheduling," in *Proc. 2nd SysML Conf.*, 2019, pp. 418–430.

[13] H. Zhang et al., "Poseidon: An efficient communication architecture for distributed deep learning on GPU clusters," in *Proc. Annu. Tech. Conf. (USENIX ATC)*, 2017, pp. 181–193.

[14] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 2595–2603.

[15] P. Parpas and C. Muir, "Predict globally, correct locally: Parallel-in-time optimal control of neural networks," 2019, *arXiv:1902.02542*.

[16] Z. Huo, B. Gu, and H. Huang, "Training neural networks using features replay," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6659–6668.

[17] H. Zhuang, Y. Wang, Q. Liu, and Z. Lin, "Fully decoupled neural network learning using delayed gradients," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 6013–6020, Oct. 2022.

[18] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[19] C. Gallicchio and A. Micheli, "Graph echo state networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2010, pp. 1–8.

[20] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016.

[21] H. Dai, Z. Kozareva, B. Dai, A. Smola, and L. Song, "Learning steady-states of iterative algorithms over graphs," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1106–1114.

[22] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2014.

[23] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," 2015, *arXiv:1506.05163*.

[24] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[25] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, 2016.

[26] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 1225–1234.

[27] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2609–2615.

[28] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2018, pp. 362–373.

[29] Y. Huang, Y. Weng, S. Yu, and X. Chen, "Diffusion convolutional recurrent neural network with rank influence learning for traffic forecasting," in *Proc. Int. Conf. Learn. Represent.*, Aug. 2019.

[30] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-RNN: Deep learning on spatio-temporal graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5308–5317.

[31] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 19, 2006.

[32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2022.

[33] J. Wang, Z. Chai, Y. Cheng, and L. Zhao, "Toward model parallelism for deep neural network based on gradient-free ADMM framework," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2020, pp. 591–600.

[34] T. Huang, P. Singhania, M. Sanjabi, P. Mitra, and M. Razaviyayn, "Alternating direction method of multipliers for quantization," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 208–216.

[35] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*, vol. 317. Cham, Switzerland: Springer, 2009.

[36] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," 2015, *arXiv:1505.00853*.

[37] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*. Cham, Switzerland: Springer, 2010, pp. 177–186.

[38] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," 2012, *arXiv:1212.5701*.

[39] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Feb. 2011.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc.3rd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds., San Diego, CA, USA, 2015.

Junxiang Wang received the bachelor's degree from East China Normal University, Shanghai, China, in 2012, the master's degree from George Mason University, Fairfax, VA, USA, in 2020, and the Ph.D. degree from the Department of Computer Science, Emory University, Atlanta, GA, USA, in 2022.

He is currently an Incoming Researcher with NEC Laboratories America. He has published various research papers in top-tier conferences and journals, such as ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), IEEE International Conference on Data Mining (ICDM), International World Wide Web Conference (WWW), Association for the Advancement of Artificial Intelligence (AAAI), PROCEEDINGS OF THE IEEE, and IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS). He has been invited to present his works in several optimization conferences. His research interests include data mining in social media, inverse problems on graphs, and nonconvex optimization in deep learning.

Hongyi Li received the B.E. degree in telecommunication engineering from Xidian University, Xi'an, China, in 2019, where she is currently pursuing the Ph.D. degree with the State Key Laboratory of Integrated Services Networks (ISN).

Her research interests include graph learning, optimization algorithms, and their applications in wireless communication systems.

Zheng Chai received the master's degree from Georgetown University, Washington, DC, USA, in 2018. He is currently pursuing the Ph.D. degree with the Department of Computer Science, University of Virginia, Charlottesville, VA, USA, under the supervision of Prof. Yue Cheng.

His research interests include distributed machine learning, federated learning, and high-performance computing.

Yongchao Wang (Senior Member, IEEE) received the B.E. degree in communication engineering and the M.E. and Ph.D. degrees in information and communication engineering from Xidian University, Xi'an, China, in 1998, 2004, and 2006, respectively.

From September 2008 to January 2010, he was a one-year Visiting Scholar and then a Post-Doctoral Fellow with the Department of Electronics and Communication Engineering, University of Minnesota, Minneapolis, MN, USA. From March 2016 to March 2017, he was a Visiting Scholar with Purdue University, West Lafayette, IN, USA. Since 2012, he has been a Professor with the State Key Laboratory of Integrated Services Networks (ISN), Xidian University. His research works have been applied to several real telecommunication systems.

Dr. Wang was a recipient of several awards, such as the Prize in Progress of Science and Technology from the Ministry of Education of the People's Republic of China, Shaanxi Government, and Xidian University.

Yue Cheng received the Ph.D. degree in computer science from Virginia Tech, Blacksburg, VA, USA, in 2017.

He is currently an Assistant Professor of data science and computer science with the University of Virginia (UVA), Charlottesville, VA, USA. His research interests include distributed systems, cloud and serverless computing, and high-performance computing. His current research interests focuses on the systems support for cloud- and HPC-scale data-intensive computing, such as deep learning and data analytics. His research results have been published at major conferences, such as International Conference for High Performance Computing, Networking, Storage, and Analysis (SC), International ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC), EuroSys, USENIX Annual Technical Conference (ATC), ACM Symposium on Cloud Computing (SoCC), and IEEE International Conference on Data Mining (ICDM). He has developed a set of new techniques that make stateful serverless computing efficient, scalable, and easy to program. His work on serverless data analytics has been used by tech giants, including Alibaba, Sunnyvale, CA, USA, Microsoft, Redmond, WA, USA, and Adobe, San Jose, CA, USA.

Dr. Cheng was a recipient of the NSF CAREER Award, the Meta Research Award, the Amazon Research Award, and the IEEE CS Technical Community on High Performance Computing (TCHPC) Early Career Researchers Award for excellence in high-performance computing.

Liang Zhao (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer Science, Virginia Tech, Blacksburg, VA, USA, in 2016.

He was an Assistant Professor with the Department of Information Sciences and Technology (IST) and Computer Science (CS), George Mason University, Fairfax, VA, USA. He is currently an Assistant Professor with the Department of Computer Science, Emory University, Atlanta, GA, USA. He has published more than 100 papers in top-tier conferences and journals, such as ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), IEEE International Conference on Data Mining (ICDM), IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), NeurIPS, PROCEEDINGS OF THE IEEE, ACM Transactions on Knowledge Discovery from Data (TKDD), ACM Transactions on Spatial Algorithms and Systems (TSAS), International Joint Conference on Artificial Intelligence (IJCAI), Association for the Advancement of Artificial Intelligence (AAAI), International World Wide Web Conference (WWW), ACM International Conference on Information and Knowledge Management (CIKM), SIGSPATIAL, and SDM. His research interests include data mining, artificial intelligence, and machine learning, with special interests in spatiotemporal and network data mining, deep learning on graphs, nonconvex optimization, and interpretable machine learning.

Dr. Zhao has received the NSF CAREER Award in 2020. He has also received the Outstanding Doctoral Student in the Department of Computer Science at Virginia Tech in 2017, the NSF CRII Award in 2018, the Jeffress Trust Award in 2019, the Amazon Research Award in 2020, and the Meta Research Award in 2022. He was ranked as one of the "Top 20 Rising Star in Data Mining" by Microsoft Search in 2016. He has won the Best Paper Award, the Best Poster Runner-Up, and the Best Paper Candidates in top-tier venues, such as ICDM 2021, ICDM 2019, WWW 2021, and ACM SIGSPATIAL 2022. He is recognized as a "Computing Innovative Fellow Mento" in 2021 by Computing Research Association.