Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

______________________________  ________________
Junxiang Wang  Date
The Applications of Alternating Minimization Algorithms on Deep Learning Models

By

Junxiang Wang
Doctor of Philosophy

Computer Science and Informatics

Liang Zhao, Ph.D.
Advisor

Carl Yang, Ph.D.
Committee Member

Li Xiong, Ph.D.
Committee Member

Lars Ruthotto, Ph.D.
Committee Member

Accepted:

Kimberly Jacob Arriola, Ph.D., MPH
Dean of the James T. Laney School of Graduate Studies

Date
The Applications of Alternating Minimization Algorithms on Deep Learning Models

By

Junxiang Wang
B.S., East China Normal University, Shanghai, China, 2012
M.Sc., George Mason University, VA, 2020

Advisor: Liang Zhao, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2022
Abstract

The Applications of Alternating Minimization Algorithms on Deep Learning Models
By Junxiang Wang

Gradient Descent(GD) and its variants are the most popular optimizers for training deep learning models. However, they suffer from many challenges such as gradient vanishing and backward locking, which prevent their more widespread use. To address these intrinsic drawbacks, Alternating Minimization(AM) methods have attracted attention from researchers as a potential way to train deep learning models. Their idea is to decompose a neural network into a series of linear and nonlinear equality constraints, which generate multiple subproblems and they can be minimized alternately. Their empirical evaluations demonstrate good scalability and high accuracy. They also avoid gradient vanishing problems and allow for non-differentiable activation functions, as well as allowing for complex non-smooth regularization and the constraints that are increasingly important for neural network architectures.

This dissertation discusses the applications of AM methods on deep learning models, which can be categorized into three research topics: 1). AM methods on Multi-Layer Perceptron(MLP), which includes deep learning Alternating Direction Method of Multipliers(dlADMM), and monotonous Deep Learning Alternating Minimization(mDLAM). 2). AM methods on Graph Neural Network(GNN), which includes the Invertible Validity-aware Graph Diffusion(IVGD). 3). AM methods for distributed neural network training, which includes parallel deep learning Alternating Direction Method of Multipliers(pdADMM), pdADMM-G, and pdADMM-G-Q.

For the dlADMM algorithm, parameters in each layer are updated in a backward and forward fashion. The time complexity is reduced from cubic to quadratic in(latent) feature dimensions for subproblems by iterative quadratic approximations and backtracking. Finally, we provide the convergence guarantee of the dlADMM algorithm under mild conditions with a sublinear convergence rate $o(1/k)$.

For the mDLAM algorithm, our innovative inequality-constrained formulation infinitely approximates the original problem with non-convex equality constraints, enabling our convergence proof of the proposed mDLAM algorithm regardless of the choice of hyperparameters. Our mDLAM algorithm is shown to achieve a fast linear convergence by the Nesterov acceleration technique.

For the IVGD model, we unroll AM methods into GNN models for graph source localization. Specifically, first, to inversely infer sources of graph diffusion, we propose a graph residual scenario to make existing graph diffusion models invertible with theoretical guarantees; second, we develop a novel error compensation mechanism that learns to offset the errors of the inferred sources. Finally, to ensure the validity of the inferred sources, we unroll AM methods into the proposed validity-aware layers to project inferred sources to feasible regions by encoding validity constraints. A linearization technique is proposed to strengthen the efficiency of our proposed layers. The convergence of the proposed IVGD is proven theoretically.

For the pdADMM algorithm, we achieve model parallelism by breaking layer dependency: parameters in each layer of neural networks can be updated independently
in parallel. The convergence of the proposed pdADMM to a stationary point is theoretically proven under mild conditions with a sublinear convergence rate $o(1/k)$.

For the pdADMM-G algorithm and the pdADMM-G-Q algorithm, in order to achieve model parallelism, we extend the proposed pdADMM algorithm to train the GA-MLP model, named the pdADMM-G algorithm. The extended pdADMM-G-Q algorithm reduces communication costs by introducing the quantization technique. Theoretical convergence to a (quantized) stationary point of two proposed algorithms is provided with a sublinear convergence rate $o(1/k)$.
The Applications of Alternating Minimization Algorithms on Deep Learning Models

By

Junxiang Wang
B.S., East China Normal University, Shanghai, China, 2012
M.Sc., George Mason University, VA, 2020

Advisor: Liang Zhao, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2022
Acknowledgments

Firstly, I would like to express my heartfelt gratitude to my mentor, Dr. Liang Zhao, for his support during my Ph.D study. His effective instruction and mental encouragement helped me overcome the difficulty of my research projects. I really appreciate the efforts Dr Liang Zhao made when he revised my research papers and presentation slides over and over. Without his guidance, I would never be able to publish many papers during these years.

Secondly, I appreciate all my committee members, Dr. Carl Yang, Dr. Li Xiong, and Dr. Lars Ruthotto. They provided insightful comments to the my preliminary dissertation prospectus, and the draft of the dissertation thesis. Their useful suggestions broadened my research horizon, and left me equipped with ideas of new research directions. Special thanks go to Dr. Yue Cheng, who provided valuable suggestions and hardware support from the level of system infrastructures during our research collaboration.

Thirdly, I would like to thank my research collaborators in and out of the Dr. Liang Zhao’s laboratory: Hongyi Li, Xiaojie Guo, Chen Ling, Junji Jiang, Shiyu Wang, Guangji Bai, Yuyang Gao, and Zheng Chai, to name but a few. Thank you all for your unique contributions to our papers and useful advices to improve our research ideas.

Finally, I would like to dedicate this dissertation thesis to my girlfriend, my parents, and other family members. They provided me with support and courage during this memorable journey.
Contents

1 Introduction 1
   1.1 Contributions .................................................. 7
   1.2 The Organization of the dissertation ....................... 11

2 The dlADMM Algorithm 12
   2.1 Introduction .................................................... 12
   2.2 Related Work ................................................... 15
   2.3 dlADMM ........................................................... 16
      2.3.1 Problem Transformation ................................. 16
      2.3.2 the Proposed dlADMM Algorithm ..................... 17
      2.3.3 The Quadratic Approximation and Backtracking .... 20
   2.4 Convergence Analysis ......................................... 25
      2.4.1 Assumptions ................................................. 26
      2.4.2 Key Properties ............................................. 26
   2.5 Experiments ..................................................... 29
      2.5.1 Experiment Setup ......................................... 29
      2.5.2 Experimental Results .................................... 31
   2.6 Conclusion ....................................................... 35

3 The mDLAM Algorithm 37
   3.1 Introduction ..................................................... 37
CONTENTS

3.2 Related Work .................................................. 39
3.3 Model and Algorithm ........................................ 41
  3.3.1 Inequality Approximation for Deep Learning ............ 41
  3.3.2 Alternating Optimization ................................ 42
3.4 Convergence Analysis ....................................... 46
  3.4.1 Convergence Properties .................................. 47
  3.4.2 Convergence of the Proposed mDLAM Algorithm ........ 48
  3.4.3 Discussion .................................................. 49
3.5 Experiments .................................................... 50
  3.5.1 Datasets and Parameter Settings ......................... 50
  3.5.2 Convergence ................................................ 51
  3.5.3 Performance ............................................... 52
  3.5.4 Sensitivity Analysis ...................................... 55
3.6 Conclusion ..................................................... 58

4 The pdADMM Algorithm ........................................ 59
  4.1 Introduction ................................................ 59
  4.2 Related Work .............................................. 61
  4.3 pdADMM ..................................................... 63
    4.3.1 Problem Reformulation ................................. 63
    4.3.2 Solutions to All Subproblems ......................... 65
  4.4 Convergence Analysis ...................................... 68
  4.5 Experiments ................................................ 72
    4.5.1 Datasets ............................................... 72
    4.5.2 Speedup ............................................... 73
    4.5.3 Convergence ........................................... 77
    4.5.4 Performance ........................................... 77
  4.6 Conclusion ................................................. 80
# Contents

## 5 The pdADMM-G and pdADMM-G-Q Algorithms

5.1 Introduction .................................................. 81

5.2 Related Work .................................................. 84

5.3 The pdADMM-G Algorithm .................................... 84

5.3.1 Problem Formulation ....................................... 85

5.3.2 The pdADMM-G Algorithm ................................. 86

5.3.3 Quantization Extension of pdADMM-G(pdADMM-G-Q) ... 89

5.4 Convergence Analysis ......................................... 90

5.5 Experiments .................................................... 96

5.5.1 Datasets and Settings ...................................... 97

5.5.2 Comparison Methods ...................................... 99

5.5.3 Convergence ................................................ 99

5.5.4 Speedup .................................................... 102

5.5.5 Communication Overheads ................................. 104

5.5.6 Performance ............................................... 105

5.6 Conclusion ..................................................... 108

## 6 Conclusion and Future Works

6.1 Research Tasks .................................................. 111

6.1.1 The dlADMM Algorithm ................................... 111

6.1.2 The mDLAM Algorithm ................................... 113

6.1.3 The pdADMM Algorithm ................................. 114

6.1.4 The pdADMM-G and pdADMM-G-Q Algorithms ......... 115

6.1.5 The IVGD Model .......................................... 116

6.2 Discussion ..................................................... 117

6.3 Current Publications .......................................... 119

6.3.1 Contributions of Published Papers Contributing to dissertation 119

6.3.2 Published Papers During My Ph.D. ....................... 120
List of Figures

1.1 The relationships among three research topics. .......................... 5

2.1 The overview of the proposed dlADMM algorithm. .................... 18

2.2 Convergence curves of the proposed dlADMM algorithm on two datasets when $\rho = 1$. .......................................................... 31

2.3 Divergence curves of the proposed dlADMM algorithm on two datasets when $\rho = 10^{-6}$. ....................................................... 32

2.4 Performance of the proposed dlADMM algorithm against all comparison methods on the MNIST dataset. ............................. 33

2.5 Performance of the proposed dlADMM algorithm against all comparison methods on the Fashion MNIST dataset. .................... 33

3.1 Convergence curves of the proposed mDLAM algorithm on four datasets. 52

3.2 Test accuracy of the proposed mDLAM algorithm against all comparison methods. ................................................................. 53

3.3 The relationship between the running time of the proposed mDLAM algorithm and:(a) the number of neurons;(b) the value of $\rho$. ....... 54

4.1 The overview of the proposed pdADMM algorithm. .................... 65

4.2 The relationship between the speedup of the proposed pdADMM algorithm and the number of layers on five datasets. .................... 76

4.3 Convergence curves of the proposed pdADMM algorithm. ............ 77
4.4 Training accuracy of the proposed pdADMM algorithm against all comparison methods on six datasets. 78
4.5 Test accuracy of the proposed pdADMM algorithm against all comparison methods on six datasets. 78
5.1 The overview of the proposed pdADMM-G optimization algorithm. 87
5.2 Convergence curves of the pdADMM-G algorithm and the pdADMM-G-Q algorithm on four datasets. 100
5.3 The speedup of the proposed pdADMM-G on different datasets concerning the number of layers (i.e. a weak-scaling study). 101
5.4 The speedup of all methods on two large datasets concerning the number of GPUs (i.e. a strong-scaling study). 102
5.5 Communication overheads of two proposed algorithms on three datasets. 103
List of Tables

1.1 Notations of the MLP training problem in this dissertation. 3

2.1 The relationship between running time per iteration (in second) of the proposed dIADMM algorithm and the number of neurons as well as the value of $\rho$. 34

2.2 The relationship between running time per iteration (in second) of the proposed dIADMM algorithm and the size of training samples as well as the value of $\rho$. 34

3.1 Statistics of four benchmark datasets. 50

3.2 Hyperparameter settings on four benchmark datasets. 50

3.3 The effect of $\rho$ on the test accuracy of the proposed mDLAM algorithm on four datasets. 56

3.4 The effect of the initial value of $\varepsilon$ (i.e. $\varepsilon^0$) on the test accuracy of the proposed mDLAM algorithm on four datasets. 57

4.1 The relationship between the speedup of the proposed pdADMM algorithm and the number of neurons on five datasets. 75

5.1 Important Notations. 85

5.2 Statistics of eight benchmark datasets. 97

5.3 Test performance of all methods when the number of neurons is 100. 106
5.4 Test performance of all methods when the number of neurons is 500. 107

6.1 Research tasks and status ............................................. 112

D.1 Hyperparameter settings of all methods on nine benchmark datasets
    when the number of neurons is 100. ............................. 184

D.2 Hyperparameter settings of all methods on nine benchmark datasets
    when the number of neurons is 500. ............................. 184

D.3 The validation performance of all methods when the number of neurons
    is 100. ................................................................. 185

D.4 The validation performance of all methods when the number of neurons
    is 500. ................................................................. 185
List of Algorithms

1. The proposed dlADMM algorithm ........................................ 19
2. The Backtracking Algorithm to update $a^{k+1}_l$ ........................ 21
3. The Backtracking Algorithm to update $W^{k+1}_l$ ........................ 23
4. The proposed mDLAM algorithm ........................................ 43
5. The proposed pdADMM algorithm ........................................ 64
6. The proposed pdADMM-G algorithm .................................... 88
7. The Backtracking Algorithm to Update $W^{k+1}_l$ ...................... 126
8. The Backtracking Algorithm to Update $a^{k+1}_l$ ........................ 127
Chapter 1

Introduction

Deep learning has been the focus of the machine learning community during the last decade. While conventional machine learning models such as Support Vector Machine (SVM) require some assumptions (e.g., the margin should be as large as possible), and they have limited capacity to process natural data in their raw forms, deep learning models have flexible forms and entail no requirement, and they are composed of non-linear modules that can learn multiple levels of representations automatically [59]. Because deep learning models perform surprisingly outstandingly on large-scale datasets, they require efficient optimizers to obtain a feasible solution within realistic time limits.

Gradient Descent (GD) was the most popular optimizer for training deep learning models one decade ago due to its simplicity and effectiveness. However, it has some drawbacks such as painful hyper-parameter tuning, and the possibility to plunge into saddle points [27]. To address these issues, many variants of GD approaches have been introduced and are well-known, including but not limited to GD momentum [91], AdaGrad [28], AdaDelta [118] and Adam [52]. Among them, GD momentum, AdaGrad, and AdaDelta improve GD via the first moment of the gradient, whereas Adam improves them further via the second moment of the gradient. However, they
suffer from many challenges that prevent their more widespread use: the gradient vanishing is one major challenge, which means the error signal diminishes as the gradient is backpropagated. For example, a deep Recurrent Neural Network (RNN) model easily suffers from the gradient vanishing problem because activation functions are sigmoid or tanh, and their derivatives are smaller than 1 [76]. Poor conditioning is another challenge, where a small change of input leads to a drastic change in the gradient. Even though researchers have proposed many solutions (e.g., Rectified Linear Unit (ReLU) [1], skip connection [37] and batch normalization [45]), these problems still exist because the nested structures of deep learning models amplify the effects of existing challenges such as gradient vanishing and poor conditioning. For example, a narrow Multi-Layer Perceptron (MLP) model with skip connections and ReLU may still be prone to the gradient vanishing problem.

To tackle these intrinsic drawbacks of gradient descent optimization methods, Alternating Minimization (AM) methods have started to attract attention as a potential way to solve deep learning problems. A neural network problem is reformulated as a nested function associated with multiple linear and nonlinear transformations across multi-layers. This nested structure is then decomposed into a series of linear and nonlinear equality constraints by introducing auxiliary variables and penalty hyper-parameters. The linear and nonlinear equality constraints generate multiple sub-problems, which can be minimized alternately. Many recent AM methods have focused on applying the Alternating Direction Method of Multipliers (ADMM) [93, 101], Block Coordinate Descent (BCD) [120] and Method of Auxiliary Coordinates (MAC) [9] to replace a nested neural network with a constrained problem without nesting, with empirical evaluations demonstrating good scalability in terms of the number of layers and high accuracy on the test sets. These methods also avoid gradient vanishing problems and allow for non-differentiable activation functions such as binarized neural networks [22], as well as allowing for complex non-smooth regularization and the
Table 1.1: Notations of the MLP training problem in this dissertation.

<table>
<thead>
<tr>
<th>Notations</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>The number of layers.</td>
</tr>
<tr>
<td>$W_l$</td>
<td>The weight matrix for the $l$-th layer.</td>
</tr>
<tr>
<td>$z_l$</td>
<td>The output of the linear mapping for the $l$-th layer.</td>
</tr>
<tr>
<td>$f_l(z_l)$</td>
<td>The nonlinear activation function for the $l$-th layer.</td>
</tr>
<tr>
<td>$a_l$</td>
<td>The output for the $l$-th layer.</td>
</tr>
<tr>
<td>$x$</td>
<td>The input matrix of the neural network.</td>
</tr>
<tr>
<td>$y$</td>
<td>The predefined label vector.</td>
</tr>
<tr>
<td>$R(z_L, y)$</td>
<td>The risk function.</td>
</tr>
<tr>
<td>$n_l$</td>
<td>The number of neurons for the $l$-th layer.</td>
</tr>
</tbody>
</table>

constraints that are increasingly important for deep neural architectures that are required to satisfy practical requirements such as interpretability, energy-efficiency, and cost awareness [9]. The ADMM, as a representative of AM methods, has been explored extensively for different neural network architectures. It was first used to solve the MLP problem [93], and then was extended to other architectures such as RNN [92].

This dissertation studies how AM methods can be utilized to train the MLP model, which is the most fundamental model in deep learning. Table 1.1 lists important notations utilized in this dissertation. A typical MLP model is defined by multiple linear mappings and nonlinear activation functions. A linear mapping for the $l$-th layer is composed of a weight matrix $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$, where $n_l$ is the number of neurons for the $l$-th layer; a nonlinear mapping for the $l$-th layer is defined by a continuous activation function $f_l(\bullet)$. Given an input $a_{l-1} \in \mathbb{R}^{n_{l-1}}$ from the $(l-1)$-th layer, the $l$-th layer outputs $a_l = f_l(W_l a_{l-1})$. Obviously, $a_{l-1}$ is nested in $a_l = f_l(\bullet)$. By introducing $z_l$ which serves as the output of the linear mapping for the $l$-th layer, the task of training the MLP model is formulated mathematically as follows:

Problem 1.

$$\min_{W_l, z_l, a_l} R(z_L, y) \ s.t. \ z_l = W_l a_{l-1} (l = 1, \ldots, L), \ a_l = f_l(z_l) \ (l = 1, \ldots, L - 1)$$
In Problem 1, \( a_0 = x \in \mathbb{R}^{n_0} \) is the input of the MLP model where \( n_0 \) is the number of feature dimensions, and \( y \) is a predefined label vector. \( R(z_L, y) \) is a risk function for the \( L \)-th layer, which is convex, continuous, and proper.

The goal of this dissertation is to develop optimization algorithms based on AM methods to address Problem 1. While some existing papers such as [9, 93, 120] studied this problem, they suffer from several challenges, which will be addressed in this dissertation: 1. **The lack of theoretical guarantees.** While many previous papers have studied the convergence theory of AM methods on nonconvex problems, they cannot be directly applied to neural networks. This is because a typical neural network model is a nested formulation of nonlinear functions, which causes the objective to be highly non-convex. 2. **Expensive computations of subproblems.** Some subproblems generated by AM methods are computationally expensive to address. This is because they involve matrix inversion. Computing an inverse matrix needs further sub-iterations, and its time complexity is approximate \( O(n^3) \), where \( n \) is a feature dimension [6]. 3. **Slow convergence toward solutions.** For some AM methods such as ADMM, their empirical convergence curves are long-tailed. In other words, they usually take hundreds of iterations to converge to high accuracy, even for simple examples [6]. It is often the case that they are stuck on a modest solution. 4. **The lack of investigations to parallel training neural networks.** Existing state-of-the-art algorithms are still based on GD to achieve parallel training of neural networks. However, they are subject to the backward locking problem [44]: the gradient calculations in one layer tightly depend on and have to wait for results of all previous layers, which prevents gradients of different layers from being calculated in parallel. While AM methods have great potential to address this drawback, they are rarely explored, developed, and evaluated.

In order to address all these challenges, we propose several AM methods to handle these challenges in the dissertation. They can be categorized into three research
CHAPTER 1. INTRODUCTION

Figure 1.1: The relationships among three research topics.

topics illustrated by Figure 1.1, all of which are shown below:

1. **AM methods on Multi-Layer Perceptron (MLP).** Two AM methods, the deep learning Alternating Direction Method of Multipliers (dlADMM) [101] and monotonous Deep Learning Alternating Minimization (mDLAM) [106], are proposed to train MLP models. They will be introduced in Chapters 2 and 3, respectively.

   In the dlADMM algorithm, parameters of a neural network are updated backward and then forward so that the parameter information is exchanged efficiently (i.e. address Challenge 3); the time complexity for subproblems is reduced from cubic to quadratic in (latent) feature dimensions via a dedicated algorithm that utilizes iterative quadratic approximations and backtracking (i.e. address Challenge 2). Finally, we provide the first proof of global convergence for an ADMM-based method (dlADMM) in a deep neural network problem under mild conditions (i.e. address challenge 1).

   The mDLAM algorithm approximates the original problem by an innovative formulation with inequality constraints, enabling our convergence proof of the proposed mDLAM algorithm regardless of the choice of hyperparameters (i.e. address Challenge 1). It is shown to achieve a fast linear convergence by the Nesterov acceleration technique (i.e. address Challenges 1 and 3).
2. AM methods on Graph Neural Network (GNN). We extend the applications of AM methods from the MLP to the GNN because GNN models deal with non-IID data (i.e. nodes and their connections in a graph), and the MLP can be considered as a special case of GNN models. GNN models incorporate node attributes into models and learn node representations effectively by capturing network topology and neighboring information [53]. Moreover, they can "learn" rules from the data in an end-to-end fashion. We devise GNN models via AM methods to fulfill graph learning tasks. For example, graph source localization aims to detect source nodes of a graph given their future graph diffusion patterns. It covers a wide range of real-world applications such as misinformation detection [46], Email virus localization [75], and malware detection [49].

We propose a novel GNN-based model, Invertible Validity-aware Graph Diffusion (IVGD), for graph source localization via unrolling AM methods into neural networks [104]. Specifically, first, to inversely infer sources of graph diffusion, we propose a graph residual scenario to make existing graph diffusion models invertible with theoretical guarantees; second, we develop a novel error compensation mechanism that learns to offset the errors of the inferred sources. Finally, to ensure the validity of the inferred sources, we unroll AM methods into the proposed validity-aware layers to project inferred sources to feasible regions by encoding validity constraints. A linearization technique is proposed to strengthen the efficiency of our proposed layers (i.e. address Challenge 2). The convergence of the proposed IVGD is proven theoretically (i.e. address Challenge 1). Extensive experiments on nine real-world datasets demonstrate that our proposed IVGD outperforms state-of-the-art comparison methods significantly.

3. AM methods for distributed neural network training. We extend AM methods on MLP and GNN models to realize distributed training via model parallelism (i.e. layerwise parallelism). The proposed AM methods include parallel deep
learning Alternating Direction Method of Multipliers (pdADMM) [103], the graph pdADMM(pdADMM-G) [105], and the quantized graph pdADMM(pdADMM-G-Q) [105]. They will be introduced in Chapters 4 and 5, respectively.

The pdADMM algorithm achieves layer parallelism via breaking layer dependency among variables so that parameters in each layer of neural networks can be updated independently in parallel (i.e. address Challenge 4). The convergence of the proposed pdADMM to a stationary point is theoretically proven under mild conditions (i.e. address Challenge 1).

We extend the pdADMM algorithm to train Graph Neural Networks (GNNs) such as the Graph-Augmented Multi-Layer Perceptron (GA-MLP) model, which leads to the pdADMM-G algorithm, and the pdADMM-G-Q algorithm extends the pdADMM-G algorithm by the quantization technique, in order to reduce communication overheads (i.e. address Challenge 4). 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 (i.e. address Challenge 1).

1.1 Contributions

The major proposed research contributions can be stated as follows.

The dlADMM Algorithm:

- We present a novel and efficient dlADMM algorithm to handle the MLP learning problem. The new dlADMM updates parameters in a backward and then forward fashion to speed up the convergence process.

- We propose the use of quadratic approximation and backtracking techniques to avoid the need for matrix inversion as well as reduce the computational cost for large-scale datasets. The time complexity of
1.1. CONTRIBUTIONS

subproblems in dlADMM is reduced from $O(n^3)$ to $O(n^2)$.

- **We investigate several attractive convergence properties of dlADMM.** The convergence assumptions are very mild to ensure that most deep learning applications satisfy our assumptions. dlADMM is guaranteed to converge to a stationary point globally (i.e., whatever the initialization is) when the hyperparameter is sufficiently large. We also analyze the new algorithm’s sublinear convergence rate.

- **We conduct experiments on several benchmark datasets to validate our proposed dlADMM algorithm.** The experimental results show that the proposed dlADMM algorithm performs better than most existing state-of-the-art algorithms, including GD and its variants.

The mDLAM Algorithm:

- **We propose a novel formulation for neural network optimization.** The deeply nested activation functions are disentangled into separate functions innovatively coordinated by inherently convex inequality constraints.

- **We present an accelerated optimization algorithm.** A quadratic approximation technique is utilized to avoid matrix inversion. Every subproblem has a closed-form solution. The Nesterov acceleration technique is applied to further boost convergence.

- **We investigate the convergence of the proposed mDLAM algorithm under mild conditions.** The new mDLAM algorithm is guaranteed to converge to a stationary point whatever hyperparameters we choose. Furthermore, the proposed mDLAM algorithm is shown to achieve a linear convergence rate, which is faster than existing methods.
• Extensive experiments are conducted to demonstrate the effectiveness of the proposed mDLAM algorithm. We test our proposed mDLAM algorithm on four benchmark datasets. Experimental results illustrate that our proposed mDLAM algorithm is linearly convergent on four datasets, and outperforms consistently state-of-the-art optimizers. Sensitivity analysis of the running time shows that it increases linearly with the increase of neurons and hyperparameters.

The IVGD Model:

• Design a generic end-to-end framework for source location. We develop a framework for the inverse of graph diffusion models, and learn the rules of graph diffusion models automatically. It does not require hand-crafted rules and can be used for source localization. Our framework is generic to any graph diffusion model, and the code has been released publicly.

• Develop an invertible graph diffusion model with an error compensation mechanism. We propose a new graph residual net with Lipschitz regularization to ensure the invertibility of graph diffusion models. Furthermore, we propose an error compensation mechanism to offset the errors inferred from the graph residual net.

• Propose an efficient validity-aware layer to maintain the validity of inferred sources. Our model can ensure the validity of inferred sources by automatically learning validity-aware layers. We further accelerate the optimization of the proposed layers by leveraging a linearization technique. It transforms nonconvex problems into convex problems, which have closed-form solutions. Moreover, we provide the convergence guarantees of the proposed IVGD to a feasible solution.
1.1. CONTRIBUTIONS

- **Conduct extensive experiments on nine datasets.** Extensive experiments on nine datasets have been conducted to demonstrate the effectiveness and robustness of our proposed IVGD. Our proposed IVGD outperforms all comparison methods significantly on five metrics, especially 20% on F1-Score.

**The pdADMM Algorithm:**

- **We propose a novel reformulation of the MLP training problem.** The formulation splits a neural network into independent layer partitions and allows ADMM to achieve model parallelism.

- **We present a model-parallelism version of the ADMM algorithm to train an MLP.** All parameters in each layer can be updated in parallel to speed up the training process significantly. All subproblems generated by the pdADMM algorithm are discussed in detail.

- **We investigate the convergence properties of parallel ADMM.** For the common nonlinear activation functions such as the Rectified linear unit (ReLU), we prove that the pdADMM converges to a state-of-the-art stationary point with a sublinear convergence rate $o(1/k)$.

- **We conduct extensive experiments on six benchmark datasets.** Experimental results show the massive speedup of the proposed pdADMM as well as its competitive performance with state-of-the-art optimizers.

**The pdADMM-G and pdADMM-G-Q Algorithms:**

- **We propose a novel reformulation of GA-MLP models.** It splits a neural network into independent layer partitions and allows ADMM to achieve model parallelism.

- **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.

- **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 $o(1/k)$.

- **We conduct extensive experiments to show the effectiveness of two proposed algorithms.** Experiments on nine benchmark datasets show the convergence, the massive speedup of the proposed pdADMM-G algorithm and the proposed pdADMM-G-Q algorithm, as well as their outstanding performance when compared with all state-of-the-art optimizers. Moreover, the proposed pdADMM-G-Q algorithm reduces communication overheads by up to 45%.

### 1.2 The Organization of the dissertation

The remaining dissertation is organized as follows: Chapter 2 proposes the dlADMM algorithm. Chapter 3 presents the mDLAM algorithm. The pdADMM algorithm is introduced in Chapter 4. The pdADMM-G algorithm and the pdADMM-G-Q algorithm are proposed in Chapter 5. Chapter 6 summarizes this dissertation and discusses future work.
Chapter 2

The dlADMM Algorithm

2.1 Introduction

As introduced in Chapter 1, Gradient Descent (GD) and its variants are state-of-the-art optimizers for training deep learning models. But they suffer from many limitations such as gradient vanishing and poor conditioning. Recently, the use of the Alternating Direction Method of Multipliers (ADMM) has been proposed as an alternative to GD. The ADMM splits a problem into many subproblems and coordinates them globally to obtain the solution. It has been demonstrated successfully for many machine learning applications [6]. The advantages of ADMM are numerous: it exhibits linear scaling as data is processed in parallel across cores; it does not require gradient steps and hence avoids gradient vanishing problems; it is also immune to poor conditioning [93].

Even though the performance of the ADMM seems promising, there are still several challenges at must be overcome: 1. The lack of global convergence guarantees. Even though many empirical experiments have shown that ADMM converges in deep learning applications, the underlying theory governing this convergence behavior remains mysterious. This is because a typical deep learning problem consists
of a combination of linear and nonlinear mappings, causing optimization problems
to be highly nonconvex. This means that traditional proof techniques cannot be di-
rectly applied. 2. **Slow convergence towards solutions.** Although ADMM is
a powerful optimization framework that can be applied to large-scale deep learning
applications, it usually converges slowly to high accuracy, even for simple examples
[6]. It is often the case that ADMM becomes trapped in a modest solution and hence
performs worse than GD, as the experiment described later in this chapter in Section
2.5 demonstrates. 3. **Cubic time complexity concerning feature dimensions.**
The implementation of the ADMM is very time-consuming for real-world datasets.
Experiments conducted by Taylor et al. found that ADMM required more than 7000
cores to train a neural network with just 300 neurons [93]. This computational bott-
leneck mainly originates from the matrix inversion required to update the weight
parameters. Computing an inverse matrix needs further sub-iterations, and its time
complexity is approximate $O(n^3)$, where $n$ is a feature dimension [6].

To deal with these difficulties simultaneously, in this chapter we propose a novel
optimization framework for a deep learning Alternating Direction Method of Multi-
pliers(dlADMM) algorithm. Specifically, our new dlADMM algorithm updates pa-
rameters first in a backward direction and then forwards. This update approach
propagates parameter information across the whole network and accelerates the con-
vergence process. It also avoids the operation of matrix inversion using the quadratic
approximation and backtracking techniques, reducing the time complexity from $O(n^3)$
to $O(n^2)$. Finally, to the best of our knowledge, we provide the first proof of the
global convergence of the ADMM-based method(dlADMM) in a deep neural network
problem. The assumption conditions are mild enough for many common loss func-
tions(e.g. cross-entropy loss and square loss) and activation functions(e.g. Rectified
Linear Unit(ReLU) and leaky ReLU) to satisfy. Our proposed framework and conver-
gence proof are highly flexible for Multi-Layer Perceptron(MLP) neural networks, as
well as being easily extendable to other popular network architectures such as Convolutional Neural Networks [54] and Recurrent Neural Networks [69]. Our contributions in this chapter include:

- We present a novel and efficient dlADMM algorithm to handle the MLP training problem. The new dlADMM updates parameters in a backward-forward fashion to speed up the convergence process.

- We propose the use of quadratic approximation and backtracking techniques to avoid the need for matrix inversion as well as to reduce the computational cost for large-scale datasets. The time complexity of subproblems in dlADMM is reduced from $O(n^3)$ to $O(n^2)$.

- We investigate several attractive convergence properties of dlADMM. The convergence assumptions are very mild to ensure that most deep learning applications satisfy our assumptions. dlADMM is guaranteed to converge to a stationary point globally (i.e., whatever the initialization is) when the hyperparameter is sufficiently large. We also analyze the new algorithm’s sublinear convergence rate.

- We conduct experiments on several benchmark datasets to validate our proposed dlADMM algorithm. The results show that the proposed dlADMM algorithm performs better than most existing state-of-the-art algorithms, including GD and its variants.

The rest of this chapter is organized as follows. In Section 2.2, we summarize recent research related to this topic. In Section 2.3, we present the new dlADMM algorithm, quadratic approximation, and the backtracking techniques utilized. In Section 2.4, we introduce the main convergence results for the Proposed dlADMM algorithm. The results of extensive experiments conducted to show the convergence, efficiency, and
effectiveness of our proposed new dADMM algorithm are presented in Section 2.5, and Section 2.6 concludes this chapter by summarizing the research.

2.2 Related Work

Previous literature related to this research includes optimization for deep learning models and ADMM for nonconvex problems.

**Optimization for Deep Learning Models:** The GD algorithm and its variants play a dominant role in the research conducted by the deep learning optimization community. The famous back-propagation algorithm was firstly introduced by Rumelhart et al. to train the neural network effectively [85]. Since the superior performance exhibited by AlexNet [54] in 2012, deep learning has attracted a great deal of researchers’ attention and many new optimizers based on GD have been proposed to accelerate the convergence process, including the use of Polyak momentum [80], as well as research on the Nesterov momentum and initialization by Sutskever et al. [91]. Adam is the most popular method because it is computationally efficient and requires little tuning [52]. Other well-known methods that incorporate adaptive learning rates include AdaGrad [28], RMSProp [95], and AMSGrad [82]. Recently, the Alternating Direction Method of Multipliers (ADMM) has become popular with researchers due to its excellent scalability [93]. However, even though these optimizers perform well in real-world applications, their convergence mechanisms remain mysterious. This is because convergence assumptions do not apply to deep learning problems, which often require non-differentiable activation functions such as ReLU.

**Convergence Analysis of Nonconvex ADMM:** Despite the outstanding performance of the nonconvex ADMM, its convergence theory is not well established due to the complexity of both coupled objectives and various inequality and equality constraints. Specifically, Magnusson et al. provided new convergence conditions of
ADMM for a class of nonconvex structured optimization problems [67]; Li and Pong investigated the properties of the nonconvex ADMM on the composite optimization problem [60]; Wang et al. presented mild convergence conditions of the nonconvex ADMM where the objective function can be coupled and nonsmooth [107]; Hong et al. proved that the classic ADMM converges to stationary points provided that the penalty parameter is sufficiently large [40]; Wang et al. proved the convergence of the multi-convex ADMM [100]; Liu et al. proved the convergence properties of a parallel and linearized ADMM [64]; Xie et al. proposed a deep-learning-based ADMM algorithm to study the constrained optimization problems [113]. For more work, please refer to [14, 33, 34, 94, 98, 102].

2.3 dlADMM

We present our proposed dlADMM algorithm in this section. Section 2.3.1 formulates the deep neural network problem, Section 2.3.2 introduces how the Proposed dlADMM algorithm works, and the quadratic approximation and backtracking techniques used to solve the subproblems are presented in Section 2.3.3.

2.3.1 Problem Transformation

Problem 1 is difficult to solve because of the nonlinear constraint $a_l = f_l(z_l)$. To address this, we can relax Problem 1 by adding an $\ell_2$ penalty to address Problem 2 as follows:

Problem 2.

$$
\min_{W_l,z_l,a_l} F(W, z, a) = R(z_L, y) + \left( \nu / 2 \right) \sum_{l=1}^{L-1} (\|z_l - W_l a_{l-1}\|_2^2 + \|a_l - f_l(z_l)\|_2^2)
$$

s.t. $z_L = W_L a_{L-1}$
where \( W = \{ W_l \}_{l=1}^L \), \( z = \{ z_l \}_{l=1}^L \), \( a = \{ a_l \}_{l=1}^{L-1} \) and \( \nu > 0 \) is a tuning parameter. Compared with Problem 1, Problem 2 has only a linear constraint \( z_L = W_L a_{L-1} \) and hence is easier to solve. It is straightforward to show that as \( \nu \to \infty \), the solution to Problem 2 approaches that of Problem 1.

### 2.3.2 the Proposed dlADMM Algorithm

We introduce the Proposed dlADMM algorithm to solve Problem 2 in this section. The traditional ADMM strategy for optimizing parameters is to start from the first layer and then update parameters in the following layer sequentially [93]. In this case, the parameters in the final layer are subject to the parameter update in the first layer. However, the parameters in the final layer contain important information that can be transmitted to the previous layers to speed up convergence. To achieve this, we propose our novel dlADMM framework, as shown in Figure 2.1. Specifically, the Proposed dlADMM algorithm updates parameters in two steps. In the first, the Proposed dlADMM begins updating from the \( L \)-th(final) layer and moves backward toward the first layer. The update order of parameters in the same layer is \( a_l \to z_l \to W_l \). In the second, the Proposed dlADMM reverses the update direction, beginning at the first layer and moving forward toward the \( L \)-th(final) layer. The update order of the parameters in the same layer is \( W_l \to z_l \to a_l \). The parameter information for all layers can be exchanged completely by adopting this update approach.

Now we can present our dlADMM algorithm mathematically. The augmented Lagrangian function of Problem 2 is shown in the following form:

\[
L_\rho(W, z, a, u) = R(z_L, y) + \phi(W, z, a, u)
\]

where \( \phi(W, z, a, u) = (\nu/2) \sum_{l=1}^{L-1} (\| z_l - W_l a_{l-1} \|_2^2 + \| a_l - f_l(z_l) \|_2^2) + u^T (z_L - W_L a_{L-1}) + (\rho/2) \| z_L - W_L a_{L-1} \|_2^2 \), \( u \) is a dual variable and \( \rho > 0 \) is a hyperparameter of the
Proposed dlADMM algorithm. We denote $\mathbf{W}_l^{k+1}$, $\mathbf{z}_l^{k+1}$ and $\mathbf{a}_l^{k+1}$ as the backward update of the Proposed dlADMM for the $l$-th layer in the $(k + 1)$-th iteration, while $\mathbf{W}_l^{k+1}$, $\mathbf{z}_l^{k+1}$ and $\mathbf{a}_l^{k+1}$ are denoted as the forward update of the Proposed dlADMM for the $l$-th layer in the $(k + 1)$-th iteration. Moreover, we denote

$$
\mathbf{W}_l^{k+1} = \{\{W_i^{k+1}\}_{i=1}^L\}, \quad \mathbf{z}_l^{k+1} = \{\{z_i^{k+1}\}_{i=1}^L\}, \quad \mathbf{a}_l^{k+1} = \{\{a_i^{k+1}\}_{i=1}^L\}.
$$

Then the Proposed dlADMM algorithm is shown in Algorithm 1. Specifically, Lines 5, 6, 10, 13, 15, and 16 solve six subproblems, namely, $\mathbf{a}_l^{k+1}$, $\mathbf{z}_l^{k+1}$, $\mathbf{W}_l^{k+1}$, $\mathbf{W}_l^{k+1}$, $\mathbf{z}_l^{k+1}$ and $\mathbf{a}_l^{k+1}$, respectively. Lines 21 and 22 update the residual $r^{k+1}$ and the dual variable $u^{k+1}$, respectively.
Algorithm 1: The proposed dlADMM algorithm

**Require:** \( y, a_0 = x, \rho, \nu. \)

**Ensure:** \( a_l(l = 1, \cdots, L - 1), W_l(l = 1, \cdots, L), z_l(l = 1, \cdots, L). \)

1: Initialize \( k = 0. \)
2: while \( W^{k+1}, z^{k+1}, a^{k+1} \) not converged do
3: \hspace{1em} for \( l = L \) to 1 do
4: \hspace{2em} if \( l < L \) then
5: \hspace{3em} Update \( \overline{a}_l^{k+1} \) in Equation (2.3).
6: \hspace{3em} Update \( z_l^{k+1} \) in Equation (2.4).
7: \hspace{2em} else
8: \hspace{3em} Update \( \overline{z}_L^{k+1} \) in Equation (2.5).
9: \hspace{2em} end if
10: \hspace{2em} Update \( \overline{W}_l^{k+1} \) in Equation (2.7).
11: \hspace{1em} end for
12: \hspace{1em} for \( l = 1 \) to \( L \) do
13: \hspace{2em} Update \( W_l^{k+1} \) in Equation (2.9).
14: \hspace{2em} if \( l < L \) then
15: \hspace{3em} Update \( z_l^{k+1} \) in Equation (2.10).
16: \hspace{3em} Update \( a_l^{k+1} \) in Equation (2.12).
17: \hspace{2em} else
18: \hspace{3em} Update \( z_L^{k+1} \) in Equation (2.11).
19: \hspace{3em} \( r^{k+1} \leftarrow z_L^{k+1} - W_L^{k+1} a_{L-1}^{k+1}. \)
20: \hspace{3em} \( u^{k+1} \leftarrow u^k + \rho r^{k+1}. \)
21: \hspace{2em} end if
22: \hspace{1em} end for
23: \hspace{1em} \( k \leftarrow k + 1. \)
24: end while
25: Output \( W, z, a. \)
2.3.3 The Quadratic Approximation and Backtracking

The six subproblems in Algorithm 1 are discussed in detail in this section. Most can be solved by quadratic approximation and the backtracking techniques described above, so the operation of the matrix inversion can be avoided.

1. **Update** $\alpha_{l}^{k+1}$

   The variables $\alpha_{l}^{k+1}(l = 1, \cdots, L - 1)$ are updated as follows:

   \[
   \alpha_{l}^{k+1} \leftarrow \arg \min_{\alpha_{l}} L_{\rho}(W_{l+1}^{k+1}, z_{l+1}^{k+1}, \{a_{i}^{k}\}_{i=1}^{l-1}, a_{l}, \{\alpha_{i}^{k+1}\}_{i=l+1}^{L-1}, u^{k})
   \]

   The subproblem is transformed into the following form after it is replaced by Equation (2.1).

   \[
   \alpha_{l}^{k+1} \leftarrow \arg \min_{\alpha_{l}} \phi(W_{l+1}^{k+1}, z_{l+1}^{k+1}, \{a_{i}^{k}\}_{i=1}^{l-1}, a_{l}, \{\alpha_{i}^{k+1}\}_{i=l+1}^{L-1}, u^{k})
   \] (2.2)

   Because $a_{l}$ and $W_{l+1}$ are coupled in $\phi(\bullet)$, in order to solve this problem, we must compute the inverse matrix of $W_{l+1}^{k+1}$, which involves sub-iterations and is computationally expensive [93]. In order to handle this challenge, we define $\overline{Q}_{l}(a_{l}; \tau_{l}^{k})$ as a quadratic approximation of $\phi$ at $a_{l}^{k}$, which is mathematically reformulated as follows:

   \[
   \overline{Q}_{l}(a_{l}; \tau_{l}^{k}) = \phi(W_{l+1}^{k+1}, z_{l+1}^{k+1}, \{a_{i}^{k}\}_{i=1}^{l-1}, a_{l}, \{\alpha_{i}^{k+1}\}_{i=l+1}^{L-1}, u^{k})(a_{l} - a_{l}^{k})
   \]

   where $\tau_{l}^{k+1} > 0$ is a parameter vector, $\circ$ denotes the Hadamard product (i.e. the elementwise product), and $a^{\circ b}$ denotes $a$ to the Hadamard power of $b$ and $\| \bullet \|_{1}$ is the $\ell_{1}$ norm. $\nabla a_{l}^{k}\phi$ is the gradient of $\overline{a}_{l}$ at $a_{l}^{k}$. $\overline{Q}_{l}(a_{l}; \tau_{l}^{k}) = \phi(W_{l+1}^{k+1}, z_{l+1}^{k+1}, \{a_{i}^{k+1}\}_{i=l+1}^{L-1}, u^{k})$. Rather than minimizing the original problem in Equation (2.2), we instead solve the
following problem:

$$a_{l+1}^k \leftarrow \arg \min_{a_l} Q_l(a_l; \tau_l^{k+1})$$  \hspace{1cm} (2.3)

Because $Q_l(a_l; \tau_l^{k+1})$ is a quadratic function with respect to $a_l$, the solution can be obtained by

$$a_{l+1}^k \leftarrow a_l^k - \nabla a_k^l \phi / \tau_l^{k+1}$$

given a suitable $\tau_l^{k+1}$. Now the main focus is how to choose $\tau_l^{k+1}$. Algorithm 2 shows the backtracking algorithm utilized to find a suitable $\tau_l^{k+1}$. Lines 2-5 implement a while loop until the condition $\phi(W_{l+1}^{k+1}, z_{l+1}^{k+1}, a_{l+1}^{k+1}, u^k) \leq Q_l(a_{l+1}^{k+1}; \tau_l^{k+1})$ is satisfied. As $\tau_l^{k+1}$ becomes larger and larger, $a_{l+1}^k$ is close to $a_l^k$ and $a_l^k$ satisfies the loop condition, which precludes the possibility of the infinite loop. The time complexity of Algorithm 2 is $O(n^2)$, where $n$ is the number of features or neurons.

**Algorithm 2:** The Backtracking Algorithm to update $a_{l+1}^k$

Require: $W_{l+1}^{k+1}, z_{l+1}^{k+1}, a_{l+1}^{k+1}, u^k, \rho$, some constant $\eta > 1$.

Ensure: $\tau_{l+1}^{k+1}, a_{l+1}^{k+1}$.

1: Pick up $\tilde{t}$ and $\tilde{\beta} = a_l^k - \nabla a_k^l \phi / \tilde{t}$

2: while $\phi(W_{l+1}^{k+1}, z_{l+1}^{k+1}, \{a_i^l\}_{i=1}^{l-1}, \beta, \{a_i^{k+1}\}_{i=l+1}^{L-1}, u^k) > Q_l(\tilde{\beta}; \tilde{t})$ do

3: $\tilde{t} \leftarrow \tilde{t} \eta$.

4: $\tilde{\beta} \leftarrow a_l^k - \nabla a_k^l \phi / \tilde{t}$.

5: end while

6: Output $\tau_{l+1}^{k+1} \leftarrow \tilde{t}$.

7: Output $a_{l+1}^{k+1} \leftarrow \tilde{\beta}$.

2. Update $z_{l+1}^{k+1}$

The variables $z_{l+1}^{k+1}(l = 1, \cdots, L)$ are updated as follows:

$$z_{l+1}^{k+1} \leftarrow \arg \min_{z_l} L_\rho(W_{l+1}^{k+1}, \{z_i^k\}_{i=1}^{l-1}, z_l, \{z_i^{k+1}\}_{i=l+1}^{L}, a_{l+1}^{k+1}, u^k)$$
which is equivalent to the following forms: for $\mathbf{z}_i^{k+1}(l = 1, \cdots, L - 1)$,

$$
\mathbf{z}_i^{k+1} \leftarrow \arg \min_{z_i} \phi_{\mathbf{w}_i^{k+1}}(\mathbf{W}_i^{k+1}, \{\mathbf{z}_i^{l+1}\}_{i=1}^{L}, z_i, \{\mathbf{z}_i^{k+1}\}_{i=1}^{L}, \mathbf{a}_i^{k+1}, u^k) \tag{2.4}
$$

and for $\mathbf{z}_L^{k+1}$,

$$
\mathbf{z}_L^{k+1} \leftarrow \arg \min_{z_L} \phi_{\mathbf{w}_L}(\mathbf{W}_L, \{\mathbf{z}_L^{L-1}\}, z_L, a_L, u^k) + R(z_L, y) \tag{2.5}
$$

Equation (2.4) is highly nonconvex because the nonlinear activation function $f(z_i)$ is contained in $\phi(\cdot)$. For common activation functions such as ReLU and leaky ReLU, Equation (2.4) has a closed-form solution; for other activation functions like sigmoid and hyperbolic tangent (tanh), a look-up table is recommended [93].

Equation (2.5) is a convex problem because $\phi(\cdot)$ and $R(\cdot)$ are convex with regard to $z_L$. Therefore, Equation (2.5) can be solved by Fast Iterative Soft-Thresholding Algorithm (FISTA) [2].

3. Update $\mathbf{W}_i^{k+1}$

The variables $\mathbf{W}_i^{k+1}(l = 1, \cdots, L)$ are updated as follows:

$$
\mathbf{W}_i^{k+1} \leftarrow \arg \min_{W_i} L_{\rho}(\{W_i^{l+1}\}_{i=1}^{L}, W_i, \{\mathbf{W}_i^{k+1}\}_{i=1}^{L}, \mathbf{z}_i^{k+1}, \mathbf{a}_i^{k+1}, u^k)
$$

which is equivalent to the following form:

$$
\mathbf{W}_i^{k+1} \leftarrow \arg \min_{W_i} \phi(\{W_i^{l-1}\}_{i=1}^{L}, W_i, \{\mathbf{W}_i^{k+1}\}_{i=1}^{L}, \mathbf{z}_i^{k+1}, \mathbf{a}_i^{k+1}, u^k) \tag{2.6}
$$

Due to the same challenge in updating $\mathbf{a}_i^{k+1}$, we define $\overline{P}_i(W_i; \mathbf{a}_i^{k+1})$ as a quadratic approximation of $\phi$ at $\mathbf{W}_i^{k}$. The quadratic approximation is mathematically reformu-
2.3. DLADMM

\[ P_l(W_l; \theta_l^{k+1}) = \phi(W_l^{k+1}, z_l^{k+1}, a_l^{k+1}, u^k) + (\nabla_{W_l^k} \phi)^T(\overline{W}_l^{k+1}, z_l^{k+1}, a_l^{k+1}, u^k)(W_l - W_l^k) \]

\[ + \|\theta_l^{k+1} \circ (W_l - W_l^k)^2\|_1/2 \]

where \( \theta_l^{k+1} > 0 \) is a parameter vector, which is chosen by the Algorithm 3. Instead of minimizing the Equation (2.6), we minimize the following:

\[ \overline{W}_l^{k+1} \leftarrow \arg \min_{W_l} P_l(W_l; \theta_l^{k+1}) \quad (2.7) \]

Equation (2.7) is convex and hence can be solved exactly.

**Algorithm 3:** The Backtracking Algorithm to update \( \overline{W}_l^{k+1} \)

Require: \( \overline{W}_l^{k+1}, z_l^{k+1}, a_l^{k+1}, u^k, \rho \), some constant \( \gamma > 1 \).
Ensure: \( \theta_l^{k+1}, \overline{W}_l^{k+1} \).

1. Pick up \( \alpha \) and \( \zeta = W_l^k - \nabla_{W_l^k} \phi/\alpha \).
2. while \( \phi(W_i^{k}, z_i^{k+1}, a_i^{k+1}, u^k) > P_l(\zeta; \alpha) \) do
   3. \( \alpha \leftarrow \alpha \gamma \).
   4. Solve \( \zeta \) by Equation (2.7).
5. end while
6. Output \( \theta_l^{k+1} \leftarrow \alpha \).
7. Output \( \overline{W}_l^{k+1} \leftarrow \zeta \).

4. Update \( W_l^{k+1} \)

The variables \( W_l^{k+1}(l = 1, \cdots, L) \) are updated as follows:

\[ W_l^{k+1} \leftarrow \arg \min_{W_l} L_\rho(W_i^{k+1})_{i=1}^{L-1}, W_l, (\overline{W}_i^{k+1})_{i=l+1}^{L}, z_l^{k+1}, a_l^{k+1}, u^k) \]

which is equivalent to

\[ W_l^{k+1} \leftarrow \arg \min_{W_l} \phi(W_i^{k+1})_{i=1}^{L-1}, W_l, (\overline{W}_i^{k+1})_{i=l+1}^{L}, z_l^{k+1}, a_l^{k+1}, u^k) \quad (2.8) \]
Similarly, we define $P_l(W_l; \theta_l^{k+1})$ as a quadratic approximation of $\phi$ at $W_l^{k+1}$. The quadratic approximation is then mathematically reformulated as follows [2]:

$$P_l(W_l; \theta_l^{k+1}) = \phi(W_l^{k+1}, z_{l-1}^{k+1}, a_{l-1}^{k+1}, u^k)$$
$$+ (\nabla_{W_l} \phi)^T(W_l^{k+1} - \bar{W}_l^{k+1})$$
$$+ \|\theta_l^{k+1} \circ (W_l^{k+1} - \bar{W}_l^{k+1})\|_1/2$$

where $\theta_l^{k+1} > 0$ is a parameter vector. Instead of minimizing the Equation (2.8), we minimize the following:

$$W_l^{k+1} \leftarrow \arg \min_{W_l} P_l(W_l; \theta_l^{k+1}) \quad (2.9)$$

The choice of $\theta_l^{k+1}$ is discussed in Section A.1 in the Appendix.

5. Update $z_l^{k+1}$

The variables $z_l^{k+1}(l = 1, \cdots, L)$ are updated as follows:

$$z_l^{k+1} \leftarrow \arg \min_{z_l} L_\rho(W_l^{k+1}, \{z_i^{k+1}\}_{i=1}^{l-1}, z_l, \{z_i^{k+1}\}_{i=l+1}^{L}, a_{l-1}^{k+1}, u^k)$$

which is equivalent to the following forms for $z_l(l = 1, \cdots, L - 1)$:

$$z_l^{k+1} \leftarrow \arg \min_{z_l} \phi(W_l^{k+1}, \{z_i^{k+1}\}_{i=1}^{l-1}, z_l, \{z_i^{k+1}\}_{i=l+1}^{L-1}, a_{l-1}^{k+1}, u^k) \quad (2.10)$$

and for $z_L$:

$$z_L^{k+1} \leftarrow \arg \min_{z_L} \phi(W_L^{k+1}, \{z_i^{k+1}\}_{i=1}^{L-1}, z_L, a_{L-1}^{k}, u^k) + R(z_L, y) \quad (2.11)$$

Solving Equations (2.10) and (2.11) proceeds exactly the same as solving Equations (2.4) and (2.5), respectively.

6. Update $a_l^{k+1}$
The variables $a_{l}^{k+1}(l = 1, \cdots, L - 1)$ are updated as follows:

$$a_{l}^{k+1} \leftarrow \arg \min_{a_{l}} L_{\rho}(W_{l}^{k+1}, z_{l}^{k+1}, \{a_{i}^{k}\}_{i=1}^{l-1}, a_{l}, \{\bar{a}_{i}^{k+1}\}_{i=l+1}^{L-1}, u^{k})$$

which is equivalent to the following form:

$$a_{l}^{k+1} \leftarrow \arg \min_{a_{l}} \phi(W_{l}^{k+1}, z_{l}^{k+1}, \{a_{i}^{k}\}_{i=1}^{l-1}, a_{l}, \{\bar{a}_{i}^{k+1}\}_{i=l+1}^{L-1}, u^{k})$$

$Q_{l}(a_{l}; \tau_{l}^{k+1})$ is defined as the quadratic approximation of $\phi$ at $a_{l}^{k+1}$ as follows:

$$Q_{l}(a_{l}; \tau_{l}^{k+1}) = \phi(W_{l}^{k+1}, z_{l}^{k+1}, a_{l-1}^{k+1}, u^{k}) + (\nabla_{a_{l}^{k+1}} \phi)^{T}(W_{l}^{k+1}, z_{l}^{k+1}, a_{l-1}^{k+1}, u^{k})(a_{l} - \bar{a}_{l}^{k+1})$$

$$+ \|\tau_{l}^{k+1} \circ (a_{l} - \bar{a}_{l}^{k+1})\|^{2}/2$$

and we can solve the following problem instead:

$$a_{l}^{k+1} \leftarrow \arg \min_{a_{l}} Q_{l}(a_{l}; \tau_{l}^{k+1}) \quad (2.12)$$

where $\tau_{l}^{k+1} > 0$ is a parameter vector. The solution to Equation (2.12) can be obtained by $a_{l}^{k+1} \leftarrow \bar{a}_{l}^{k+1} - \nabla_{a_{l}^{k+1}} \phi/\tau_{l}^{k+1}$. The choice of an appropriate $\tau_{l}^{k+1}$ is shown in Section A.1 in the Appendix.

### 2.4 Convergence Analysis

In this section, the theoretical convergence of the proposed dlADMM algorithm is analyzed. Before we formally present the convergence results of the Proposed dlADMM algorithms, Section 2.4.1 presents the necessary assumptions to guarantee the global convergence of dlADMM. In Section 2.4.2, we prove the global convergence of the Proposed dlADMM algorithm.
2.4. CONVERGENCE ANALYSIS

2.4.1 Assumptions

**Assumption 1** (Closed-form Solution). There exist activation functions $a_l = f_l(z_l)$ such that Equations (2.4) and (2.10) have closed form solutions

$$z_{k+1}^{l+1} = \overline{h}(W_{k+1}^{l+1}, a_{k+1}^{l+1})$$

and

$$z_{k+1}^{l} = h(W_{k+1}^{l}, a_{k+1}^{l-1}),$$

respectively, where $\overline{h}(\bullet)$ and $h(\bullet)$ are continuous functions.

This assumption can be satisfied by commonly used activation functions such as ReLU and leaky ReLU. For example, for the ReLU function $a_l = \max(z_l, 0)$, Equation (2.10) has the following solution:

$$z_{l}^{k+1} =
\begin{cases}
\min(W_{l}^{k+1}a_{l-1}^{k+1}, 0) & z_{l}^{k+1} \leq 0 \\
\max((W_{l}^{k+1}a_{l-1}^{k+1} + a_{l}^{k+1})/2, 0) & z_{l}^{k+1} \geq 0
\end{cases}$$

**Assumption 2** (Objective Function). $F(W, z, a)$ is coercive over the nonempty set $G = \{(W, z, a) : z_L - W_L a_{L-1} = 0\}$. In other words, $F(W, z, a) \to \infty$ if $(W, z, a) \in G$ and $\|(W, z, a)\| \to \infty$. Moreover, $R(z, y)$ is Lipschitz differentiable with Lipschitz constant $H \geq 0$.

The Assumption 2 is mild enough for most common loss functions to satisfy. For example, the cross-entropy and square loss are Lipschitz differentiable.

2.4.2 Key Properties

We present the main convergence result of the proposed dlADMM algorithm in this section. Specifically, as long as Assumptions 1-2 hold, then Properties 1-3 are satisfied, which are important to prove the global convergence of the proposed dlADMM algorithm. The proof details are included in Section A.3 and Section A.4 in the Appendix.
2.4. CONVERGENCE ANALYSIS

**Property 1** (Boundness). If $\rho > \sqrt{\frac{17+1}{2}}H$, then $\{W^k, z^k, a^k, u^k\}$ is bounded, and $L_\rho(W^k, z^k, a^k, u^k)$ is lower bounded.

Property 1 concludes that all variables and the value of $L_\rho$ have lower bounds. It is proven under Assumptions 1 and 2, and its proof can be found in Section A.3 in the Appendix.

**Property 2** (Sufficient Descent). If $\rho > \sqrt{\frac{17+1}{2}}H$ so that $C_1 = \rho/2 - H/2 - 2H^2/\rho > 0$ and $C_2 = \rho/2 - 2H^2/\rho > 0$, then

$$L_\rho(W^k, z^k, a^k, u^k) - L_\rho(W^{k+1}, z^{k+1}, a^{k+1}, u^{k+1})$$

$$\geq \sum_{l=1}^L (\|\bar{\theta}_l^{k+1} \circ (\bar{W}_l^{k+1} - W_l^k)\|_1/2 + \|\bar{\theta}_l^{k+1} \circ (\bar{W}_l^{k+1} - W_l^{k+1})\|_1/2$$

$$+ \|\bar{\tau}_l^{k+1} \circ (\bar{a}_l^{k+1} - a_l^k)\|_1/2 + \|\bar{\tau}_l^{k+1} \circ (\bar{a}_l^{k+1} - a_l^{k+1})\|_1/2$$

$$+ C_2 \|z_L^{k+1} - z_L^k\|^2 + C_1 \|z_L^{k+1} - z_L^{k+1}\|^2$$

(2.13)

Property 2 depicts the monotonic decrease of the objective value during iterations. The proof of Property 2 is detailed in Section A.3 in the Appendix.

**Property 3** (Subgradient Bound). There exist a bounded $C^{k+1} > 0$ and $g \in \partial L_\rho(W^{k+1}, z^{k+1}, a^{k+1})$ such that

$$\|g\| \leq C^{k+1}(\|W^{k+1} - W^{k+1}\| + \|z^{k+1} - z^{k+1}\| + \|a^{k+1} - a^{k+1}\| + \|z^{k+1} - z^k\|)$$

(2.14)

Property 3 ensures that the subgradient of the objective function is bounded by variables. The proof of Property 3 requires Property 1 and the proof is elaborated in Section A.3 in the Appendix. Now the global convergence of the Proposed dlADMM algorithm is presented. The following theorem states that Properties 1-3 are guaranteed.
Theorem 1. For any $\rho > \sqrt{\frac{17+1}{2} H}$, if Assumptions 1 and 2 are satisfied, then Properties 1-3 hold.

Proof. This theorem can be concluded by the proofs in Section A.3 in the Appendix.

The next theorem presents the global convergence of the Proposed dlADMM algorithm.

Theorem 2 (Global Convergence). If $\rho > \sqrt{\frac{17+1}{2} H}$ and $\{\theta_{l}^{k}L\}_{l=1}^{K}$, $\{\theta_{l}^{k+1}L\}_{l=1}^{K}$, and $\{\tau_{l}^{k+1}L-1\}_{l=1}^{K}$ are bounded, then for the variables $(W, z, a, u)$ in Problem 2, starting from any $(W^{0}, z^{0}, a^{0}, u^{0})$, it has at least a limit point $(W^{*}, z^{*}, a^{*}, u^{*})$, and any limit point $(W^{*}, z^{*}, a^{*}, u^{*})$ is a stationary point of Problem 2 (i.e. $0 \in \partial L_{\rho}(W^{*}, z^{*}, a^{*}, u^{*})$).

Or equivalently,

$$z_{L}^{k} = W_{L}^{k}a_{L-1}^{k}$$

$$0 \in \partial W_{L}^{*}L_{\rho}(W^{*}, z^{*}, a^{*}, u^{*})$$

$$0 \in \partial z_{L}^{*}L_{\rho}(W^{*}, z^{*}, a^{*}, u^{*})$$

$$\nabla a_{L}^{*}L_{\rho}(W^{*}, z^{*}, a^{*}, u^{*}) = 0$$

Proof. Because $(W^{k}, z^{k}, a^{k}, u^{k})$ is bounded, there exists a subsequence $(W^{s}, z^{s}, a^{s}, u^{s})$ such that $(W^{s}, z^{s}, a^{s}, u^{s}) \to (W^{*}, z^{*}, a^{*}, u^{*})$ where $(W^{*}, z^{*}, a^{*}, u^{*})$ is a limit point. By Properties 1 and 2, $L_{\rho}(W^{k}, z^{k}, a^{k}, u^{k})$ is non-increasing and lower bounded and hence converges. By Property 2, we prove that $\|W^{k+1} - W^{k}\| \to 0$, $\|a^{k+1} - a^{k}\| \to 0$, $\|W^{k+1} - W^{k}\| \to 0$, and $\|a^{k+1} - a^{k}\| \to 0$, as $k \to \infty$. Therefore $\|W^{k+1} - W^{k}\| \to 0$, and $\|a^{k+1} - a^{k}\| \to 0$, as $k \to \infty$. Moreover, from Assumption 1, we know that $z^{k+1} \to z^{k}$ and $z^{k+1} \to z^{k+1}$ as $k \to \infty$. Therefore, $z^{k+1} \to z^{k}$. We infer there exists $g^{k} \in \partial L_{\rho}(W^{k}, z^{k}, a^{k}, u^{k})$ such that $\|g^{k}\| \to 0$ as $k \to \infty$ based on Property 3. Specifically, $\|g^{s}\| \to 0$ as $s \to \infty$. According to the definition of general subgradient (Definition 8.3 in [84]), we have $0 \in \partial L_{\rho}(W^{*}, z^{*}, a^{*}, u^{*})$. In other words, the limit point $(W^{*}, z^{*}, a^{*}, u^{*})$ is a stationary point of $L_{\rho}$ defined in Equation (2.1).
Theorem 2 shows that our dlADMM algorithm converges globally for sufficiently large $\rho$, which is consistent with previous literature [51, 107]. The next theorem shows that the Proposed dlADMM converges globally with a sublinear convergence rate $o(1/k)$, where $k$ is the number of iterations.

**Theorem 3 (Convergence Rate).** For a sequence $(W^k, z^k, a^k, u^k)$, define

\[
c_k = \min_{0 \leq i \leq k} \left( \sum_{l=1}^{L} \| \theta_{l}^{i+1} \circ (W_{l}^{i} - W_{l}^{k}) \|^2 + \| a_{l}^{i+1} \circ (a_{l}^{i} - W_{l}^{i+1}) \|^2 \right) + C_2 \| z_{L}^{i+1} - z_{L}^{i} \|^2 + C_1 \| z_{L}^{i+1} - z_{L}^{i} \|^2, \]

then the convergence rate of $c_k$ is $o(1/k)$.

**Proof.** The proof of this theorem is included in Section A.4 in the Appendix. \qed

### 2.5 Experiments

In this section, we evaluate the dlADMM algorithm using benchmark datasets. The effectiveness, efficiency, and convergence properties of dlADMM are compared with state-of-the-art methods. All experiments were conducted on 64-bit Ubuntu16.04 LTS with Intel(R) Xeon processor and GTX1080Ti GPU.

#### 2.5.1 Experiment Setup

**Dataset**

In this experiment, two benchmark datasets were used for performance evaluation: MNIST [58] and Fashion MNIST [112]. The MNIST dataset has ten classes of handwritten-digit images, which was firstly introduced by Lecun et al. in 1998 [58]. It contains 55,000 training samples and 10,000 test samples with 784 features each, which is provided by the Keras library [18]. Unlike the MNIST dataset, the Fashion MNIST dataset has ten classes of assortment images on the website of Zalando, which is Europe’s largest online fashion platform [112]. The Fashion-MNIST dataset consists of 60,000 training samples and 10,000 test samples with 784 features each.
2.5. EXPERIMENTS

Experiment Settings

We set up a network architecture that contained two hidden layers with 1,000 hidden units each. The ReLU was used for the activation function for both network structures. The loss function was set as the deterministic cross-entropy loss. $\nu$ was set to $10^{-6}$. $\rho$ was initialized as $10^{-6}$ and was multiplied by 10 every 100 iterations. The number of iterations was set to 200. In the experiment, one iteration means one epoch.

Comparison Methods

Since this chapter focuses on the MLP model, GD and its variants and ADMM are state-of-the-art methods and hence served as comparison methods. For GD-based methods, the full batch dataset is used for training models. All parameters were chosen by the accuracy of the training dataset. The baselines are described as follows:

1. Gradient Descent (GD) [4]. The GD and its variants are the most popular deep learning optimizers, whose convergence has been studied extensively in the literature. The learning rate of GD was set to $10^{-6}$ for both the MNIST and Fashion MNIST datasets.

2. Adaptive gradient algorithm (Adagrad) [28]. Adagrad is an improved version of GD: rather than fixing the learning rate during iteration, it adapts the learning rate to the hyperparameter. The learning rate of Adagrad was set to $10^{-3}$ for both the MNIST and Fashion MNIST datasets.

3. Adaptive learning rate method (Adadelta) [118]. As an improved version of the Adagrad, the Adadelta is proposed to overcome the sensitivity to hyperparameter selection. The learning rate of Adadelta was set to 0.1 for both the MNIST and Fashion MNIST datasets.

4. Adaptive momentum estimation (Adam) [52]. Adam is the most popular optimization method for deep learning models. It estimates the first and second momen-
2.5. EXPERIMENTS

tum to correct the biased gradient and thus makes convergence fast. The learning rate of Adam was set to $10^{-3}$ for both the MNIST and Fashion MNIST datasets.

5. Alternating Direction Method of Multipliers (ADMM) [93]. ADMM is a powerful convex optimization method because it can split an objective function into a series of subproblems, which are coordinated to get global solutions. It is scalable to large-scale datasets and supports parallel computations. The $\rho$ of ADMM was set to 1 for both the MNIST and Fashion MNIST datasets.

2.5.2 Experimental Results

In this section, experimental results of the Proposed dlADMM algorithm are analyzed against comparison methods.

Convergence

![Figure 2.2: Convergence curves of the proposed dlADMM algorithm on two datasets when $\rho = 1$.](image)

First, we show that our proposed dlADMM algorithm converges when $\rho$ is sufficiently large and diverges when $\rho$ is small for both the MNIST dataset and the Fashion MNIST dataset.
2.5. EXPERIMENTS

Figure 2.3: Divergence curves of the proposed dlADMM algorithm on two datasets when $\rho = 10^{-6}$.

The convergence and divergence of dlADMM algorithm are shown in Figures 2.2 and 2.3 when $\rho = 1$ and $\rho = 10^{-6}$, respectively. In Figures 2.2(a) and 2.3(a), the X axis and Y axis denote the number of iterations and the logarithm of objective value, respectively. In Figures, 2.2(b) and 2.3(b), the X-axis and Y-axis denote the number of iterations and the logarithm of the residual, respectively. Figure 2.2, both the objective value and the residual decreased monotonically for the MNIST dataset and the Fashion-MNIST dataset, which validates our theoretical guarantees in Theorem 2. Moreover, Figure 2.3 illustrates that both the objective value and the residual diverge when $\rho = 10^{-6}$. The curves fluctuated drastically on the objective value. Even though there was a decreasing trend for the residual, it still fluctuated irregularly and failed to converge.

Performance

Figure 2.4 and Figure 2.5 show the curves of the training accuracy and test accuracy of our proposed dlADMM algorithm and baselines, respectively. Overall, both the training accuracy and the test accuracy of our proposed dlADMM algorithm outperformed most baselines for both the MNIST dataset and the Fashion MNIST dataset.
2.5. EXPERIMENTS

Specifically, the curves of our dlADMM algorithm soared to 0.8 at the early stage and then raised steadily towards 0.9 or more. The curves of the most GD-related methods, GD, Adadelta, and Adagrad, moved more slowly than our proposed dlADMM algorithm. The curves of the ADMM also rocketed to around 0.8 but decreased slightly later on. Only the state-of-the-art Adam performed better than dlADMM slightly.

Scalability Analysis

In this subsection, the relationship between running time per iteration of our proposed dlADMM algorithm and three potential factors, namely, the value of $\rho$, the size of
2.5. EXPERIMENTS

### MNIST dataset: From 200 to 1,000 neurons

<table>
<thead>
<tr>
<th>(\rho)</th>
<th>neurons</th>
<th>200</th>
<th>400</th>
<th>600</th>
<th>800</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10^{-6})</td>
<td></td>
<td>1.9025</td>
<td>2.7750</td>
<td>3.6615</td>
<td>4.5709</td>
<td>5.7988</td>
</tr>
<tr>
<td>(10^{-5})</td>
<td></td>
<td>2.8775</td>
<td>4.6197</td>
<td>6.3620</td>
<td>8.2563</td>
<td>10.0323</td>
</tr>
<tr>
<td>(10^{-4})</td>
<td></td>
<td>2.2761</td>
<td>3.9745</td>
<td>5.8645</td>
<td>7.6656</td>
<td>9.9221</td>
</tr>
<tr>
<td>(10^{-3})</td>
<td></td>
<td>2.4361</td>
<td>4.3284</td>
<td>6.5651</td>
<td>8.7357</td>
<td>11.3736</td>
</tr>
<tr>
<td>(10^{-2})</td>
<td></td>
<td>2.7912</td>
<td>5.1383</td>
<td>7.8249</td>
<td>10.0300</td>
<td>13.4485</td>
</tr>
</tbody>
</table>

### Fashion MNIST dataset: From 200 to 1,000 neurons

<table>
<thead>
<tr>
<th>(\rho)</th>
<th>neurons</th>
<th>200</th>
<th>400</th>
<th>600</th>
<th>800</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10^{-6})</td>
<td></td>
<td>2.0069</td>
<td>2.8694</td>
<td>4.0506</td>
<td>5.1438</td>
<td>6.7406</td>
</tr>
<tr>
<td>(10^{-5})</td>
<td></td>
<td>3.3445</td>
<td>5.4190</td>
<td>7.3785</td>
<td>9.0813</td>
<td>11.0531</td>
</tr>
<tr>
<td>(10^{-4})</td>
<td></td>
<td>2.4974</td>
<td>4.3729</td>
<td>6.4257</td>
<td>8.3520</td>
<td>10.0728</td>
</tr>
<tr>
<td>(10^{-3})</td>
<td></td>
<td>2.7108</td>
<td>4.7236</td>
<td>7.1507</td>
<td>9.4534</td>
<td>12.3326</td>
</tr>
<tr>
<td>(10^{-2})</td>
<td></td>
<td>2.9577</td>
<td>5.4173</td>
<td>8.2518</td>
<td>10.0945</td>
<td>14.3465</td>
</tr>
</tbody>
</table>

Table 2.1: The relationship between running time per iteration (in second) of the proposed dlADMM algorithm and the number of neurons as well as the value of \(\rho\).

### MNIST dataset: From 11,000 to 55,000 training samples

<table>
<thead>
<tr>
<th>(\rho)</th>
<th>size</th>
<th>11,000</th>
<th>22,000</th>
<th>33,000</th>
<th>44,000</th>
<th>55,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10^{-6})</td>
<td></td>
<td>1.0670</td>
<td>2.0682</td>
<td>3.3089</td>
<td>4.6546</td>
<td>5.7709</td>
</tr>
<tr>
<td>(10^{-5})</td>
<td></td>
<td>2.3981</td>
<td>3.9086</td>
<td>6.2175</td>
<td>7.9188</td>
<td>10.2741</td>
</tr>
<tr>
<td>(10^{-4})</td>
<td></td>
<td>2.1290</td>
<td>3.7891</td>
<td>5.6843</td>
<td>7.7625</td>
<td>9.8843</td>
</tr>
<tr>
<td>(10^{-3})</td>
<td></td>
<td>2.1295</td>
<td>4.1939</td>
<td>6.5039</td>
<td>8.8835</td>
<td>11.3368</td>
</tr>
<tr>
<td>(10^{-2})</td>
<td></td>
<td>2.6154</td>
<td>4.9638</td>
<td>7.6606</td>
<td>10.4580</td>
<td>13.4021</td>
</tr>
</tbody>
</table>

### Fashion MNIST dataset: From 12,000 to 60,000 training samples

<table>
<thead>
<tr>
<th>(\rho)</th>
<th>size</th>
<th>12,000</th>
<th>24,000</th>
<th>36,000</th>
<th>48,000</th>
<th>60,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10^{-6})</td>
<td></td>
<td>1.2163</td>
<td>2.3376</td>
<td>3.7053</td>
<td>5.1491</td>
<td>6.7298</td>
</tr>
<tr>
<td>(10^{-5})</td>
<td></td>
<td>2.5772</td>
<td>4.3417</td>
<td>6.6681</td>
<td>8.3763</td>
<td>11.0292</td>
</tr>
<tr>
<td>(10^{-4})</td>
<td></td>
<td>2.3216</td>
<td>4.1163</td>
<td>6.2355</td>
<td>8.3819</td>
<td>10.7120</td>
</tr>
<tr>
<td>(10^{-3})</td>
<td></td>
<td>2.3149</td>
<td>4.5250</td>
<td>6.9834</td>
<td>9.5853</td>
<td>12.3232</td>
</tr>
<tr>
<td>(10^{-2})</td>
<td></td>
<td>2.7381</td>
<td>5.3373</td>
<td>8.1585</td>
<td>11.1992</td>
<td>14.2487</td>
</tr>
</tbody>
</table>

Table 2.2: The relationship between running time per iteration (in second) of the proposed dlADMM algorithm and the size of training samples as well as the value of \(\rho\).
training samples, and the number of neurons was explored. The running time was calculated by the average of 200 iterations.

Firstly, when the training size was fixed, the computational result for the MNIST dataset and Fashion MNIST dataset is shown in Table 2.1. The number of neurons for each layer ranged from 200 to 1,000, with an increase of 200 each time. The value of $\rho$ ranged from $10^{-6}$ to $10^{-2}$, multiplying by 10 each time. Generally, the running time increased as the number of neurons and the value of $\rho$ became larger. However, there were a few exceptions: for example, when there were 200 neurons for the MNIST dataset, and $\rho$ increased from $10^{-5}$ to $10^{-4}$, the running time per iteration dropped from 2.8778 seconds to 2.2761 seconds.

Secondly, we fixed the number of neurons for each layer as 1,000. The relationship between running time per iteration, the training size, and the value of $\rho$ is shown in Table 2.2. The value of $\rho$ ranged from $10^{-6}$ to $10^{-2}$, multiplying by 10 each time. The training size of the MNIST dataset ranged from 11,000 to 55,000, with an increase of 11,000 each time. The training size of the Fashion MNIST dataset ranged from 12,000 to 60,000, with an increase of 12,000 each time. Similar to Table 2.2, the running time increased generally as the training sample and the value of $\rho$ became larger and some exceptions exist.

### 2.6 Conclusion

The ADMM is a good alternative to GD for deep learning problems. In this chapter, we propose a novel dlADMM algorithm to address some previously mentioned challenges. Firstly, the Proposed dlADMM updates parameters from backward to forward to transmit parameter information more efficiently. The time complexity is successfully reduced from $O(n^3)$ to $O(n^2)$ by iterative quadratic approximations and backtracking. Finally, the Proposed dlADMM is guaranteed to converge to a station-
ary solution under mild conditions. Experiments on benchmark datasets demonstrate that our proposed dLADMM algorithm outperformed most of the comparison methods.
Chapter 3

The mDLAM Algorithm

3.1 Introduction

In Chapter 2, we proposed a novel ADMM-based algorithm to train the MLP training problem efficiently with convergence guarantees. We extend this idea to Alternating Minimization (AM) methods in this chapter. However, as an emerging domain, AM for deep model optimization suffers from several unsolved challenges including 1. **Convergence properties are sensitive to penalty parameters.** Even though the dlADMM algorithm is guaranteed convergence in the MLP problem, such convergence guarantee is dependent on the choice of penalty hyperparameters: the convergence cannot be guaranteed anymore when penalty hyperparameters are small. 2. **Slow convergence rate.** To the best of our knowledge, almost all existing AM methods can only achieve a sublinear convergence rate. For example, the convergence rate of the ADMM and the BCD is proven to be $O(1/k)$, where $k$ is the number of iterations [101, 100, 120]. Therefore, there is still a lack of a theoretical framework that can achieve a faster convergence rate.

To simultaneously address these technical problems, we propose a new formulation of the neural network problem, along with a novel monotonous Deep Learning Alter-
3.1. INTRODUCTION

nating Minimization (mDLAM) algorithm. Specifically, we, for the first time, transform the original neural network optimization problem into an inequality-constrained problem that can infinitely approximate the original one. Applying this innovation to an inequality-constraint-based transformation not only ensures the convexity and hence easily ensures global minima of all subproblems but also prevents the output of a nonlinear function from changing much and reduces sensitivity to the input. Moreover, our proposed mDLAM algorithm can achieve a linear convergence rate theoretically, and the choice of hyperparameters does not affect the convergence of our mDLAM algorithm theoretically. Extensive experiments on four benchmark datasets show the convergence, effectiveness, and efficiency of the proposed mDLAM algorithm. Our contributions in this chapter include:

- We propose a novel formulation for neural network optimization. The deeply nested activation functions are disentangled into separate functions innovatively coordinated by inherently convex inequality constraints.

- We present an efficient optimization algorithm. A quadratic approximation technique is utilized to avoid matrix inversion. Every subproblem has a closed-form solution. The Nesterov acceleration technique is applied to further boost convergence.

- We investigate the convergence of the proposed mDLAM algorithm under mild conditions. The new mDLAM algorithm is guaranteed to converge to a stationary point whatever hyperparameters we choose. Furthermore, the proposed mDLAM algorithm is shown to achieve a linear convergence rate, which is faster than existing methods.

- Extensive experiments have been conducted to demonstrate the effectiveness of the proposed mDLAM algorithm. We test our proposed mDLAM algorithm on four benchmark datasets. Experimental results illustrate that our proposed
mDLAM algorithm is linearly convergent on four datasets, and outperforms consistently state-of-the-art optimizers. Sensitivity analysis of the running time shows that it increases linearly with the increase of neurons and hyperparameters.

The rest of this chapter is organized as follows: In Section 3.2, we summarize recent related research work to this chapter. In Section 3.3, we formulate the MLP training problem and present the proposed mDLAM algorithm to train the MLP model. Section 3.4 details convergence properties of the proposed mDLAM algorithm. Extensive experiments on benchmark datasets are shown in Section 3.5, and Section 3.6 concludes this work.

3.2 Related Work

All existing works on deep learning optimization methods fall into two major categories: GD methods and AM methods, which are shown as follows:

**GD Methods:** The renaissance of GD can be traced back to 1951 when Robbins and Monro published the first paper [83]. The famous back-propagation algorithm was introduced by Rumelhart et al. [85]. Many variants of GD methods have since been presented, including the use of Polyak momentum, which accelerates the convergence of iterative methods [80], and research by Sutskever et al., who highlighted the importance of Nesterov momentum and initialization [91]. During the last decade, many well-known GD methods which are incorporated with adaptive learning rates have been proposed by the deep learning community, which include but are not limited to AdaGrad [28], RMSProp [95], Adam [52], AMSGrad [82], Adabelief [128] and Adabound [65].

**Applications of AM Methods for Deep Learning:** Many recent works have applied AM algorithms to specific deep learning applications. For example, Taylor et
3.2. RELATED WORK

al. presented the ADMM to solve an MLP training problem via transforming it into an equality-constrained problem, where many subproblems split by ADMM can be solved efficiently [93], Zhang et al. handled Very Deep Supervised Hashing (VDSH) problems by utilizing an ADMM algorithm to overcome issues related to vanishing gradients and poor computational efficiency [125]. Zhang and Bastiaan trained a deep neural network by utilizing ADMM with a graph [121] and Askari et al. introduced a new framework for MLP models and optimize the objective using BCD methods [71]. Li et al. proposed an ADMM algorithm to achieve distributed learning of Graph Convolutional Network (GCN) via community detection [61]. Qiao et al. proposed an inertial proximal AM method to train MLP models [81].

Convergence of AM Methods for Deep Learning: Aside from applications, the other branch of works mathematically proves the convergence of the proposed AM approaches. For instance, Carreira and Wang proposed a method involving the use of auxiliary coordinates to replace a nested neural network with a constrained problem without nesting [9]. Lau et al. proposed a BCD optimization framework and proved the convergence via the Kurdyka-Lojasiewicz (KL) property [57], while Choromanska et al. proposed a BCD algorithm for training deep MLP models based on the concept of co-activation memory [19], and a BCD algorithm with $R$-linear convergence was proposed by Zhang and Brand to train Tikhonov regularized deep neural networks [124]. Jagatap and Hegde introduced a new family of AM methods and prove their convergence to a global minimum [47]. Yu et al. proved the convergence of the proposed ADMM for RNN models [92]. However, to the best of our knowledge, there is a lack of a flexible framework that allows for different activation functions and guarantees a linear convergence rate.
3.3 Model and Algorithm

3.3.1 Inequality Approximation for Deep Learning

In this section, we propose our problem transformation to address Problem 1. The equality constraint $a_l = f_l(z_l)$ is the most challenging one to handle here because common activation functions such as sigmoid [31] are nonlinear. This makes them nonconvex constraints and hence it is difficult to obtain the optimal solution when solving the $z_l$-subproblem [93]. Moreover, there is no guarantee for AM methods to solve the nonlinear equality constrained Problem 1 [99]. To deal with these two challenges, the following assumption is required for problem transformation:

Assumption 3. $f_l(z_l)$ ($l = 1, \ldots, n$) are quasilinear.

The quasilinearity is defined in Section B.1 in the Appendix. Assumption 3 is so mild that most of the widely used nonlinear activation functions satisfy it, including tanh [117], sigmoid [31], and the Rectified Linear Unit (ReLU) [66]. Then we innovatively transform the original nonconvex constraints into inequality constraints, which can be an infinite approximation of Problem 1. To do this, we introduce a tolerance $\varepsilon > 0$ and reformulate Problem 1 to the following:

$$\min_{W_l, z_l, a_l} R(z_L, y)$$

s.t. $z_l = W_l a_{l-1} (l = 1, \ldots, L)$, $f_l(z_l) - \varepsilon \leq a_l \leq f_l(z_l) + \varepsilon (l = 1, \ldots, L-1)$.

For the linear constraint $z_l = W_l a_{l-1}$, this can be transformed into a penalty term in the objective function to minimize the difference between $z_l$ and $W_l a_{l-1}$. The formulation is shown as follows:
Problem 3.

\[
\min_{W_l, z_l, a_l} F(W, z, a) = R(z_L, y) + \sum_{l=1}^{L} \phi(a_{l-1}, W_l, z_l),
\]
\[
s.t. f_l(z_l) - \varepsilon \leq a_l \leq f_l(z_l) + \varepsilon (l = 1, \ldots, L - 1).
\]

The penalty term is defined as \( \phi(a_{l-1}, W_l, z_l) = \frac{\rho}{2} \| z_l - a_l \|_2^2 \), where \( \rho > 0 \) a penalty parameter. \( W = \{W_i\}_{i=1}^{L}, z = \{z_i\}_{i=1}^{L}, a = \{a_i\}_{i=1}^{L-1} \). As \( \rho \to \infty \) and \( \varepsilon \to 0 \), Problem 3 approaches Problem 1.

The introduction of \( \varepsilon \) is to project the nonconvex constraints to \( \varepsilon \)-balls, thus transforming the nonconvex Problem 1 into Problem 3. Even though Problem 3 is still nonconvex because \( f_l(z_l) \) can be nonconvex(e.g. tanh and smooth sigmoid), it is convex concerning one variable when others are fixed(i.e. multi-convex), which is much easier to solve by AM [115]. For example, Problem 3 is convex with regard to \( z \) when \( W \), and \( a \) are fixed.

### 3.3.2 Alternating Optimization

We present the mDLAM algorithm to solve Problem 3 in this section. A potential challenge to solving Problem 3 is a slow theoretical convergence rate. For example, the convergence rate of the proposed dlADMM algorithm in Chapter 2 to solve Problem 3 is sublinear \( o(1/k) \), where \( k \) is the number of iterations [101]. To address this challenge, we apply the famous Nesterov acceleration technique to boost the convergence of our proposed mDLAM algorithm, and we prove its linear convergence theoretically in the next section.

Algorithm 4 shows our proposed mDLAM algorithm. To simplify the notation, \( W^{k+1}_{\leq l} = \{W_i^{k+1}\}_{i=1}^{l}, \{W_i^k\}_{i=l+1}^L \), \( z^{k+1}_{\leq l} = \{z_i^{k+1}\}_{i=1}^{l}, \{z_i^k\}_{i=l+1}^L \) and \( a^{k+1}_{\leq l} = \{a_i^{k+1}\}_{i=1}^{l}, \{a_i^k\}_{i=l+1}^L \). In Algorithm 4, Lines 6, 10, and 21 apply the Nesterov acceleration technique and update \( W_l, z_l \) and \( a_l \), respectively. the proposed mDLAM
Algorithm 4: The proposed mDLAM algorithm

Require: $y$, $a_0 = x$.
Ensure: $a_l, W_l, z_l (l = 1, \ldots, L)$.

1: Initialize $\rho, k = 0, s^0 = 0$.
2: repeat
3: $s^{k+1} \leftarrow \frac{1 + \sqrt{1 + 4(s^k)^2}}{2}$
4: $\omega^k \leftarrow \frac{s^k - 1}{s^k + 1}$
5: for $l = 1$ to $L$ do
6: $W_i^{k+1} \leftarrow W_i^k + (W_i^k - W_i^{k-1})\omega^k$ and update $W_i^{k+1}$ in Equation (3.3).
7: if $F(W_i^{k+1}, z_i^{k+1}, a_i^{k+1}) \geq F(W_i^{k+1}, z_i^{k+1}, a_i^{k+1})$ then
8: $W_i^{k+1} \leftarrow W_i^k$ and update $W_i^{k+1}$ in Equation (3.3).
9: end if
10: $z_i^{k+1} \leftarrow z_i^k + (z_i^k - z_i^{k-1})\omega^k$
11: if $l = L$ then
12: Update $z_L^{k+1}$ in Equation (3.5).
13: if $F(W_i^{k+1}, z_i^{k+1}, a_i^{k+1}) \geq F(W_i^{k+1}, z_i^{k+1}, a_i^{k+1})$ then
14: $z_L^{k+1} \leftarrow z_L^k$ and update $z_L^{k+1}$ in Equation (3.5).
15: end if
16: else
17: Update $z_i^{k+1}$ in Equation (3.4).
18: if $F(W_i^{k+1}, z_i^{k+1}, a_i^{k+1}) \geq F(W_i^{k+1}, z_i^{k+1}, a_i^{k+1})$ then
19: $z_i^{k+1} \leftarrow z_i^k$ and update $z_i^{k+1}$ in Equation (3.4).
20: end if
21: $a_i^{k+1} \leftarrow a_i^k + (a_i^k - a_i^{k-1})\omega^k$ and update $a_i^{k+1}$ in Equation (3.6).
22: if $F(W_i^{k+1}, z_i^{k+1}, a_i^{k+1}) \geq F(W_i^{k+1}, z_i^{k+1}, a_i^{k+1})$ then
23: $a_i^{k+1} \leftarrow a_i^k$ and update $a_i^{k+1}$ in Equation (3.6).
24: end if
25: end if
26: end for
27: $k \leftarrow k + 1$.
28: until convergence.
29: Output $a_l, W_l, z_l$. 

algorithm guarantees the decrease of objective $F$: for example, if the updated $W_i^{k+1}$ in Line 7 of Algorithm 4 increases the value of $F$, i.e. $F(W_{\leq i}^{k+1}, z_{\leq i-1}^{k+1}, a_{\leq i-1}^{k+1}) \geq F(W_{\leq i-1}^{k+1}, z_{\leq i-1}^{k+1}, a_{\leq i-1}^{k+1})$, then $W_i^{k+1}$ is updated again by setting $W_i^{k+1} = W_i^k$ in Line 8 of Algorithm 4, which ensures the decline of $F$. The same procedure is applied in Lines 13-15, Lines 18-20, and Lines 22-24 in Algorithm 4, respectively.

Next, all subproblems are shown as follows:

1. **Update $W_i$**

   The variables $W_i(l = 1, \cdots, L)$ are updated as follows:

   \[
   W_i^{k+1} \leftarrow \arg \min_{W_i} \phi(a_{l-1}^{k+1}, W_i, z_i^k).
   \]  

   (3.1)

Because $W_i$ and $a_{l-1}$ are coupled in $\phi(\bullet)$, solving $W_i$ requires an inversion operation of $a_{l-1}^{k+1}$, which is computationally expensive. Motivated by the proposed dlADMM algorithm in Chapter 2, we define $P_i^{k+1}(W_i; \theta_i^{k+1})$ as a quadratic approximation of $\phi$ at $W_i^k$ as follows:

\[
P_i^{k+1}(W_i; \theta_i^{k+1}) = \phi(a_{l-1}^{k+1}, W_i^{k+1}, z_i^k) + (\nabla_{W_i} \phi)(W_i - W_i^{k+1})^T + \frac{\theta_i^{k+1}}{2} \|W_i - W_i^{k+1}\|^2,
\]

where $\theta_i^{k+1} > 0$ is a scalar parameter, which can be chosen by the backtracking algorithm [101] to meet the following condition

\[
P_i^{k+1}(W_i^{k+1}; \theta_i^{k+1}) \geq \phi(a_{l-1}^{k+1}, W_i^{k+1}, z_i^k).
\]  

(3.2)

Rather than minimizing Equation (3.1), we instead minimize the following:

\[
W_i^{k+1} \leftarrow \arg \min_{W_i} P_i^{k+1}(W_i; \theta_i^{k+1}).
\]  

(3.3)
2. Update $z_l$

The variables $z_l(l = 1, \cdots, L)$ are updated as follows:

\[
\begin{align*}
    z_{l}^{k+1} & \leftarrow \arg \min_{z_l} \phi(a_{l-1}^{k+1}, W_{l}^{k+1}, z_l), \text{ s.t. } f_l(z_l) - \varepsilon \leq a_l \leq f_l(z_l) + \varepsilon \ (l < L). \\
    z_{L}^{k+1} & \leftarrow \arg \min_{z_L} \phi(a_{L-1}^{k+1}, W_{L}^{k+1}, z_L) + R(z_L, y).
\end{align*}
\]

Similar to updating $W_l$, we define $V_{l}^{k+1}(z_l)$ as follows:

\[
V_{l}^{k+1}(z_l) = \phi(a_{l-1}^{k+1}, W_{l}^{k+1}, z_l^{k+1}) + (\nabla_{z_l^{k+1}} \phi)(z_l^{k+1}) - z_l^{k+1} + \frac{\rho}{2} \| z_l - z_l^{k+1} \|^2.
\]

Hence, we solve the following problems:

\[
\begin{align*}
z_{l}^{k+1} & \leftarrow \arg \min_{z_l} V_{l}^{k+1}(z_l), \text{ s.t. } f_l(z_l) - \varepsilon \leq a_l \leq f_l(z_l) + \varepsilon \ (l < L). \quad (3.4) \\
z_{L}^{k+1} & \leftarrow \arg \min_{z_L} V_{L}^{k+1}(z_L) + R(z_L, y). \quad (3.5)
\end{align*}
\]

As for $z_l(l = 1, \cdots, l - 1)$, the solution is

\[
z_{l}^{k+1} \leftarrow \min(\max(B_{1}^{k+1}, z_{l}^{k+1} - \nabla \phi_{z_{l}^{k+1}} / \rho), B_{2}^{k+1}),
\]

where $B_{1}^{k+1}$ and $B_{2}^{k+1}$ represent the lower bound and the upper bound of the set $\{z_l | f_l(z_l) - \varepsilon \leq a_l^k \leq f_l(z_l) + \varepsilon\}$. Equation (3.5) is easy to solve using the Fast Iterative Soft Thresholding Algorithm(FISTA) [2].

3. Update $a_l$

The variables $a_l(l = 1, \cdots, L - 1)$ are updated as follows:

\[
a_{l}^{k+1} \leftarrow \arg \min_{a_l} \phi(a_l, W_{l+1}^{k}, z_{l+1}^{k}), \text{ s.t. } f_l(z_l^{k+1}) - \varepsilon \leq a_l \leq f_l(z_l^{k+1}) + \varepsilon.
\]
3.4. CONVERGENCE ANALYSIS

Similar to updating $W^{k+1}_l$, $Q^{k+1}_l(a_l; \tau^{k+1}_l)$ is defined as

$$Q^{k+1}_l(a_l; \tau^{k+1}_l) = \phi(a^{k+1}_l, W^k_{l+1}, z^k_{l+1}) + (\nabla a^{k+1}_l \phi)^T (a_l - a^{k+1}_l) + \frac{\tau^{k+1}_l}{2} \|a_l - a^{k+1}_l\|^2,$$

and this allows us to solve the following problem instead:

$$a^{k+1}_l \leftarrow \arg \min_{a_l} Q^{k+1}_l(a_l; \tau^{k+1}_l), \text{ s.t. } f_l(z^{k+1}_l) - \varepsilon \leq a_l \leq f_l(z^{k+1}_l) + \varepsilon \quad (3.6)$$

where $\tau^{k+1}_l > 0$ is a scalar parameter, which can be chosen by the backtracking algorithm [101] to meet the following condition:

$$Q^{k+1}_l(a^{k+1}_l; \tau^{k+1}_l) \geq \phi(a^{k+1}_l, W^k_{l+1}, z^k_{l+1}).$$

The solution can be obtained by

$$a^{k+1}_l \leftarrow \min(\max(f_l(z^{k+1}_l) - \varepsilon, a^{k+1}_l - \nabla a^{k+1}_l \phi/\tau^{k+1}_l), f_l(z^{k+1}_l) + \varepsilon).$$

3.4 Convergence Analysis

In this section, the convergence of the proposed algorithm is analyzed. Due to space limit, all proofs are detailed in Section B.2 and Section B.3 in the appendix. The following mild assumption is required for the convergence analysis of the proposed mDLAM algorithm:

**Assumption 4.** $F(W, z, a)$ is coercive over the domain $\{(W, z, a) | f_l(z_l) - \varepsilon \leq a_l \leq f_l(z_l) + \varepsilon \ (l = 1, \cdots, L - 1)\}.$

The coercivity is defined in Section B.1 in the Appendix. Assumption 4 is also mild such that common loss functions such as the least square loss and the cross-entropy loss satisfy it [101].
3.4. CONVERGENCE ANALYSIS

3.4.1 Convergence Properties

Firstly, the following preliminary lemma is useful to prove the convergence properties of the proposed mDLAM algorithm.

**Lemma 1.** In Algorithm 4, there exist $\alpha^k_l, \gamma^k_l, \delta^k_l > 0$ such that for $\forall k \in \mathbb{N}, W^k_l, z^k_l (l = 1, 2, \cdots, L)$, and $a^k_l (l = 1, 2, \cdots, L - 1)$, it holds that

\[
F(W^{k+1}_{\leq l-1}, z^{k+1}_{\leq l-1}, a^{k+1}_{\leq l-1}) - F(W^k_{\leq l}, z^k_{\leq l}, a^k_{\leq l}) \geq \frac{\alpha^k_{l-1}}{2} \|W^{k+1}_l - W^k_l\|_2^2, \quad (3.7)
\]
\[
F(W^{k+1}_{\leq l}, z^{k+1}_{\leq l}, a^{k+1}_{\leq l}) - F(W^k_{\leq l}, z^k_{\leq l}, a^k_{\leq l}) \geq \frac{\gamma^k_{l-1}}{2} \|z^{k+1}_l - z^k_l\|_2^2, \quad (3.8)
\]
\[
F(W^{k+1}_{\leq l}, z^{k+1}_{\leq l}, a^{k+1}_{\leq l}) - F(W^k_{\leq l}, z^k_{\leq l}, a^k_{\leq l}) \geq \frac{\delta^k_{l-1}}{2} \|a^{k+1}_l - a^k_l\|_2^2. \quad (3.9)
\]

It shows that the objective decreases when all variables are updated. Based on Assumption 4 and Lemma 1, three convergence properties hold, which are shown in the following:

**Lemma 2 (Objective Decrease).** In Algorithm 4, it holds that for any $k \in \mathbb{N}$, $F(W^k, z^k, a^k) \geq F(W^{k+1}, z^{k+1}, a^{k+1})$. Moreover, $F$ is convergent. That is, $F(W^k, z^k, a^k) \to F^*$ as $k \to \infty$, where $F^*$ is the convergent value of $F$.

This lemma guarantees the decrease and hence convergence of the objective.

**Lemma 3 (Bounded Objective and Variables).** In Algorithm 4, it holds that for any $k \in \mathbb{N}$

(a). $F(W^k, z^k, a^k)$ is upper bounded. Moreover, $\lim_{k \to \infty} W^{k+1} - W^k = 0$, $\lim_{k \to \infty} z^{k+1} - z^k = 0$, and $\lim_{k \to \infty} a^{k+1} - a^k = 0$.

(b). $(W^k, z^k, a^k)$ is bounded. That is, there exist scalars $M_W, M_z$ and $M_a$ such that $\|W^k\| \leq M_W, \|z^k\| \leq M_z$ and $\|a^k\| \leq M_a$.

This lemma ensures that the objective and all variables are bounded in the proposed mDLAM algorithm. Moreover, the gap between the same variables in the neighboring iterations (e.g. $W^{k+1}$ and $W^k$) is convergent to 0.
3.4. CONVERGENCE ANALYSIS

Lemma 4 (Subgradient Bound). In Algorithm 4, there exist $C_2 = \max(\rho M_a, \rho M_a^2 + \theta_1^{k+1}, \rho M_a^2 + \theta_2^{k+1}, \ldots, \rho M_a^2 + \theta_L^{k+1})$, and $g_1^{k+1} \in \partial W_{k+1}F$ such that for any $k \in \mathbb{N}$

$$\|g_1^{k+1}\| \leq C_2(\|W^{k+1} - W^k\| + \|z^{k+1} - z^k\| + \|W^k - W^{k-1}\|).$$

The above lemma states that the subgradient of the objective is bounded by its variables. This suggests that the subgradient is convergent to 0, and thus proves its convergence to a stationary point.

3.4.2 Convergence of the Proposed mDLAM Algorithm

Next we discuss the convergence of the proposed mDLAM algorithm. The first theorem guarantees that the proposed mDLAM algorithm converges to a stationary point.

Theorem 4 (Convergence to a Stationary Point). In Algorithm 4, for $W$ in Problem 3, for any $\rho > 0$ and $\varepsilon > 0$, starting from any $W^0$, any limit point $W^*$ is a stationary point of Problem 3. That is, $0 \in \partial W^*F$.

As stated in Theorem 4, the convergence always holds no matter how $W$ is initialized, and whatever hyperparameters $\rho$ and $\varepsilon$ are chosen. It is better than the proposed dlADMM algorithm in Chapter 2, which requires the hyperparameter to be sufficiently large.

Theorem 5 (Linear Convergence Rate). In Algorithm 4, if $F$ is locally strongly convex, then for any $\rho$, there exist $\varepsilon > 0$, $k_1 \in \mathbb{N}$ and $0 < C_1 < 1$ such that it holds for $k > k_1$ that

$$F(W^{k+1}, z^{k+1}, a^{k+1}) - F^* \leq C_1(F(W^{k-1}, z^{k-1}, a^{k-1}) - F^*).$$
Theorem 5 shows that the proposed mDLAM algorithm converges linearly for sufficiently large iterations. Common loss functions like the square loss or the cross-entropy loss are locally strongly convex [115], which make $F$ locally strongly convex. Therefore, Theorem 5 covers a wide range of loss functions. Compared with existing AM methods (e.g. dlADMM in Chapter 2) with a sublinear $o(1/k)$ convergence rate, the proposed mDLAM algorithm achieves a theoretically better linear convergence rate.

3.4.3 Discussion

We discuss convergence conditions of the proposed mDLAM algorithm compared with GD-type methods and the proposed dlADMM algorithm in Chapter 2. The comparison demonstrates that our convergence conditions are more general than others.

1. **mDLAM versus GD**

   One influential work by Ghadimi et al. [30] guaranteed that the GD converges to a stationary point, which is similar to our convergence results. While the GD requires the objective function to be Lipschitz differentiable, bounded from below [30], our mDLAM allows for non-smooth functions such as ReLU. Therefore, our convergence conditions are milder than GD.

2. **mDLAM versus dlADMM**

   Assumptions of our proposed mDLAM are milder than those of the proposed dlADMM in Chapter 2: the mDLAM requires activation functions to be quasilinear, which includes sigmoid, tanh, ReLU, and leaky ReLU, while the proposed dlADMM assumes that activation functions make subproblems solvable, which only includes ReLU and leaky ReLU. The such difference originates from different ways of addressing nonlinear activations: the proposed dlADMM treats them as $L_2$ penalties. For tanh and sigmoid, subproblems are difficult to solve and may refer to lookup tables [101]. However, the mDLAM relaxes them via inequality constraints, and subprob-
### 3.5 Experiments

In this section, we evaluate the proposed mDLAM algorithm on four benchmark datasets. Convergence and efficiency are demonstrated. The performance of the proposed mDLAM algorithm is compared with several state-of-the-art optimizers. All experiments were conducted on a 64-bit machine with Intel(R) Xeon(R) Silver 4110 CPU and 64GB RAM.

#### 3.5.1 Datasets and Parameter Settings

An important application of the MLP model is node classification on a graph based on augmented node features [15]. Specifically, given an adjacency matrix $A$ and a node feature matrix $H$ of a graph, we let the $k$-th augmented feature $X^k = HA^k (k = 0, 1, \ldots, K)$ have closed-form solutions.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Node#</th>
<th>Edge#</th>
<th>Class#</th>
<th>Feature#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2708</td>
<td>5429</td>
<td>7</td>
<td>1433</td>
</tr>
<tr>
<td>Pubmed</td>
<td>19717</td>
<td>44338</td>
<td>3</td>
<td>500</td>
</tr>
<tr>
<td>Citeseer</td>
<td>3327</td>
<td>4732</td>
<td>6</td>
<td>3703</td>
</tr>
<tr>
<td>Coauthor CS</td>
<td>18333</td>
<td>81894</td>
<td>15</td>
<td>6805</td>
</tr>
</tbody>
</table>

Table 3.1: Statistics of four benchmark datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Hyper-parameters</th>
<th>Cora</th>
<th>Pubmed</th>
<th>Citeseer</th>
<th>Coauthor CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>mDLAM</td>
<td>$\rho$</td>
<td>$1 \times 10^{-3}$</td>
<td>0.01</td>
<td>$5 \times 10^{-3}$</td>
<td>$1 \times 10^{-3}$</td>
</tr>
<tr>
<td>GD</td>
<td>$\alpha$</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>$5 \times 10^{-3}$</td>
</tr>
<tr>
<td>Adadelta</td>
<td>$\alpha$</td>
<td>0.01</td>
<td>0.1</td>
<td>0.01</td>
<td>0.05</td>
</tr>
<tr>
<td>Adagrad</td>
<td>$\alpha$</td>
<td>$5 \times 10^{-3}$</td>
<td>$5 \times 10^{-3}$</td>
<td>0.01</td>
<td>$5 \times 10^{-3}$</td>
</tr>
<tr>
<td>Adam</td>
<td>$\alpha$</td>
<td>$1 \times 10^{-3}$</td>
<td>$5 \times 10^{-4}$</td>
<td>$1 \times 10^{-3}$</td>
<td>$1 \times 10^{-3}$</td>
</tr>
<tr>
<td>dlADMM</td>
<td>$\rho$</td>
<td>$1 \times 10^{-6}$</td>
<td>$1 \times 10^{-6}$</td>
<td>$1 \times 10^{-6}$</td>
<td>$1 \times 10^{-6}$</td>
</tr>
</tbody>
</table>

Table 3.2: Hyperparameter settings on four benchmark datasets.
0, 1, · · · , 4), which encodes information of graph topology via $A^k$, and then concatenate them into the input $X = [X_0, \cdots, X_4]$ [15]. The MLP model is used to predict the node class based on the input $X$. We set up an architecture of three layers, each of which has 100 neurons. The activation function was set to ReLU. The number of epochs was set to 200. We test our model on four benchmark datasets: Cora [86], Pubmed [86], Citeseer [86] and Coauthor CS [89], whose statistics are shown in Table 3.1.

Gradient Descent (GD) [4], Adaptive learning rate method (Adadelta) [118], Adaptive gradient algorithm (Adagrad) [28], Adaptive momentum estimation (Adam) [52], and the proposed dlADMM in Chapter 2 are state-of-the-art methods and hence were served as comparison methods. The full batch dataset was used for training models. All parameters were chosen by maximizing the accuracy of training datasets. Table 3.2 shows hyperparameters of all methods: for the proposed mDLAM algorithm, $\rho$ controls quadratic terms in Problem 3; $\alpha$ is a learning rate in the comparison methods except for the proposed dlADMM. $\rho$ controls a linear constraint in the proposed dlADMM algorithm. The other hyperparameter $\varepsilon$ is chosen adaptively as follows: $\varepsilon^{k+1} = \max(\varepsilon^k/2, 0.001)$ with $\varepsilon^0 = 100$. This makes inequality constraints relaxed at the early stage (i.e. $\varepsilon^k$ is large and hence constraints are easy to satisfy) and then tightens them as the mDLAM iterates.

### 3.5.2 Convergence

Firstly, we investigate the convergence of the proposed mDLAM algorithm on four benchmark datasets using the hyperparameters summarized in Table 3.2. The relationship between the objective and the number of epochs is shown in Figure 3.1. Overall, the objectives on the four datasets all decrease monotonically, which demonstrates the convergence of the proposed mDLAM algorithm. Nevertheless, objective curves vary in tendency: the curves on the Cora and Pubmed datasets drop drasti-
3.5. EXPERIMENTS

Figure 3.1: Convergence curves of the proposed mDLAM algorithm on four datasets. Figure 3.1: Convergence curves of the proposed mDLAM algorithm on four datasets. Initially at the beginning and then reach the plateau when the epoch is around 75, while the curves on the other two datasets keep a downward tendency for the entire 200 epochs. Moreover, the objective on the Pubmed dataset is the lowest at the end of the training, while the objective on the Citeseer dataset is in the vicinity of 80, at least 60% higher than the objectives on the remaining datasets. It is easy to observe that all curves decline linearly when the epoch is higher than 100. This validates the linear convergence rate of our proposed mDLAM algorithm (i.e. Theorem 5).

3.5.3 Performance

Next, the performance of the proposed mDLAM algorithm is compared against five state-of-the-art methods, as is illustrated in Figure 3.2. The X-axis and Y-axis represent epoch and test accuracy, respectively. Overall, the proposed mDLAM algorithm is superior to all other algorithms on four datasets, which has not only the highest test
3.5. EXPERIMENTS

(a). Cora.

(b). Pubmed.

(c). Citeseer.

(d). Coauthor CS.

Figure 3.2: Test accuracy of the proposed mDLAM algorithm against all comparison methods.
3.5. EXPERIMENTS

(a). Running time versus the number of neurons.

(b). Running time versus the value of $\rho$.

Figure 3.3: The relationship between the running time of the proposed mDLAM algorithm and: (a) the number of neurons; (b) the value of $\rho$.

accuracy but also the fastest convergence speed. For example, the proposed mDLAM achieves 70% test accuracy on the Cora dataset when the epoch is 100, while GD only attains 60%, and the Adadelta reaches the plateau of around 40%; As another example, the test accuracy of the proposed mDLAM on the Coauthor CS dataset is over 80% at the 25-th epoch, whereas most comparison methods such as Adam and GD reach half of its accuracy (i.e. 40%). The Adadelta algorithm performs the worst among all comparison methods: it converges to a low test accuracy at the early stage, which is usually half of the accuracy accomplished by the proposed mDLAM algorithm. The other four comparison methods except Adagrad are on par with mDLAM in some cases: for example, the curves of dLADMM and GD are marginally behind that of mDLAM on the Coauthor CS dataset, and the performance of Adam almost reaches that of mDLAM on the Cora dataset. It is interesting to observe that curves of some methods decline at the end of 200 epochs such as the Adagrad on the Pubmed dataset and the Adam on the Coauthor CS dataset.
3.5.4 Sensitivity Analysis

We explore concerning factors of the running time and the test accuracy in this section.

Running Time

Moreover, it is important to explore the running time of the proposed mDLAM concerning two factors: the number of neurons and the value of $\rho$. The running time was averaged by 200 epochs. Figure 3.3(a) depicts the relationship between the running time and the number of neurons on four datasets, where the number of neurons ranges from 100 to 1,000. The running times on all datasets are below 1 second per epoch when the number of neurons is 100, and increase linearly with the number of neurons in general. However, the rates of increase vary on different datasets: the curve on the Coauthor CS dataset has the sharpest slope, which reaches seven seconds per epoch when 1000 neurons are applied, while the curve on the Pubmed dataset climbs slowly, which never surpasses 1 second. The curves on the Cora and the Citeseer datasets demonstrate a steady increase.

To investigate the relationship between the running time per epoch and the value of $\rho$, we change $\rho$ from $10^{-6}$ to 1 while fixing others. Similar to Figure 3.3(a), the running time per epoch demonstrates a linear increase concerning the value of $\rho$ in general, as shown in Figure 3.3(b). Specifically, the curve on the Coauthor CS dataset is still the highest in slope, whereas the slope on the Pubmed dataset is the lowest. Moreover, the effect of the value of $\rho$ is less obvious than the number of neurons. For example, in Figure 3.3(b) when $\rho$ is enlarged from $10^{-6}$ to $10^{-2}$, the running time on the Coauthor CS dataset merely ascends from around 0.65 to 1.6, while the increment of the running time on other datasets is less than 0.2. Moreover, a larger $\rho$ may reduce the running time. For instance, when $\rho$ increases from $10^{-2}$ to 1, the running time on the Coauthor CS dataset drops slightly from 1.6 seconds to 1.5 seconds per
3.5. EXPERIMENTS

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Epoch</th>
<th>40</th>
<th>80</th>
<th>120</th>
<th>160</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>$\rho = 1 \times 10^{-4}$</td>
<td>0.677</td>
<td>0.695</td>
<td>0.695</td>
<td>0.693</td>
<td>0.692</td>
</tr>
<tr>
<td></td>
<td>$\rho = 1 \times 10^{-3}$</td>
<td>0.664</td>
<td>0.701</td>
<td>0.721</td>
<td>0.737</td>
<td>0.742</td>
</tr>
<tr>
<td></td>
<td>$\rho = 1 \times 10^{-2}$</td>
<td>0.562</td>
<td>0.581</td>
<td>0.604</td>
<td>0.623</td>
<td>0.638</td>
</tr>
<tr>
<td>Pubmed</td>
<td>$\rho = 1 \times 10^{-4}$</td>
<td>0.471</td>
<td>0.407</td>
<td>0.407</td>
<td>0.407</td>
<td>0.407</td>
</tr>
<tr>
<td></td>
<td>$\rho = 1 \times 10^{-3}$</td>
<td>0.663</td>
<td>0.645</td>
<td>0.640</td>
<td>0.650</td>
<td>0.649</td>
</tr>
<tr>
<td></td>
<td>$\rho = 1 \times 10^{-2}$</td>
<td>0.743</td>
<td>0.758</td>
<td>0.762</td>
<td>0.768</td>
<td>0.773</td>
</tr>
<tr>
<td>Citeseer</td>
<td>$\rho = 1 \times 10^{-4}$</td>
<td>0.528</td>
<td>0.529</td>
<td>0.530</td>
<td>0.531</td>
<td>0.535</td>
</tr>
<tr>
<td></td>
<td>$\rho = 1 \times 10^{-3}$</td>
<td>0.651</td>
<td>0.665</td>
<td>0.664</td>
<td>0.664</td>
<td>0.666</td>
</tr>
<tr>
<td></td>
<td>$\rho = 1 \times 10^{-2}$</td>
<td>0.631</td>
<td>0.638</td>
<td>0.642</td>
<td>0.648</td>
<td>0.653</td>
</tr>
<tr>
<td>Coauthor CS</td>
<td>$\rho = 1 \times 10^{-4}$</td>
<td>0.843</td>
<td>0.881</td>
<td>0.888</td>
<td>0.896</td>
<td>0.894</td>
</tr>
<tr>
<td></td>
<td>$\rho = 1 \times 10^{-3}$</td>
<td>0.780</td>
<td>0.807</td>
<td>0.825</td>
<td>0.839</td>
<td>0.835</td>
</tr>
<tr>
<td></td>
<td>$\rho = 1 \times 10^{-2}$</td>
<td>0.688</td>
<td>0.719</td>
<td>0.724</td>
<td>0.737</td>
<td>0.738</td>
</tr>
</tbody>
</table>

Table 3.3: The effect of $\rho$ on the test accuracy of the proposed mDLAM algorithm on four datasets.

The running times on the Cora and the Citeseer datasets climb steadily.

**Test Accuracy**

Finally, we investigate the effects of hyperparameters on test accuracy, namely, the value of $\rho$ and $\varepsilon$. Because $\varepsilon$ is dynamically set, we test its initial value $\varepsilon^0$. Table 3.3 demonstrates the relationship between test accuracy and $\rho$ on four datasets. $\rho$ was chosen from $\{1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}\}$. Overall, the choice of $\rho$ has a significant effect on the test accuracy. For example, when $\rho$ is changed from $1 \times 10^{-4}$ to $1 \times 10^{-3}$ on the Pubmed dataset, the performance has improved by approximately 60%, and the gain of performance is even roughly 90% if it is modified to $1 \times 10^{-2}$. On other datasets, the change of $\rho$ affects test accuracy by around 20%. For instance, the test accuracy on the Cora dataset and the Coauthor CS dataset can be improved to 0.74.
and 0.89 if we set $\rho = 1 \times 10^{-3}$ and $\rho = 1 \times 10^{-4}$, respectively. The test accuracy on the Citeseer dataset is relatively robust to the change of $\rho$. As $\rho$ varies from $1 \times 10^{-3}$ to $1 \times 10^{-2}$, the test accuracy remains stable. The test accuracy generally increases as the proposed mDLAM algorithm iterates. However, there are some exceptions: for example, the test accuracy has dropped slightly from 0.66 to 0.65 when $\rho = 1 \times 10^{-3}$ on the Pubmed dataset. Table 3.4 shows the relationship between test accuracy and the initial value of $\varepsilon$ (i.e. $\varepsilon^0$) on four datasets. $\varepsilon^0$ was chosen from $\{1, 10, 100\}$. The test accuracy is resistant to the change of $\varepsilon^0$. For example, the test accuracy on the Coauthor CS dataset is in the vicinity of 0.89 no matter which $\varepsilon$ is chosen. Moreover, the larger a $\varepsilon^0$ is the faster convergence speed the proposed mDLAM algorithm gains. For instance, when $\varepsilon = 100$, the test accuracy is 0.08 better than that in the case where $\varepsilon = 1$ on the Citeseer dataset. Compared with Tables 3.3 and 3.4, the effect of

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Epoch</th>
<th>$\varepsilon^0 = 1$</th>
<th>$\varepsilon^0 = 10$</th>
<th>$\varepsilon^0 = 100$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>40</td>
<td>0.620</td>
<td>0.646</td>
<td>0.664</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.679</td>
<td>0.689</td>
<td>0.701</td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>0.712</td>
<td>0.718</td>
<td>0.721</td>
</tr>
<tr>
<td></td>
<td>160</td>
<td>0.735</td>
<td>0.741</td>
<td>0.737</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.743</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pubmed</td>
<td>40</td>
<td>0.717</td>
<td>0.731</td>
<td>0.743</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.744</td>
<td>0.753</td>
<td>0.758</td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>0.756</td>
<td>0.759</td>
<td>0.762</td>
</tr>
<tr>
<td></td>
<td>160</td>
<td>0.759</td>
<td>0.762</td>
<td>0.765</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Citeseer</td>
<td>40</td>
<td>0.564</td>
<td>0.584</td>
<td>0.640</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.615</td>
<td>0.626</td>
<td>0.656</td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>0.638</td>
<td>0.643</td>
<td>0.664</td>
</tr>
<tr>
<td></td>
<td>160</td>
<td>0.653</td>
<td>0.657</td>
<td>0.663</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coauthor CS</td>
<td>40</td>
<td>0.834</td>
<td>0.852</td>
<td>0.843</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.875</td>
<td>0.866</td>
<td>0.881</td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>0.887</td>
<td>0.892</td>
<td>0.888</td>
</tr>
<tr>
<td></td>
<td>160</td>
<td>0.893</td>
<td>0.896</td>
<td>0.894</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4: The effect of the initial value of $\varepsilon$ (i.e. $\varepsilon^0$) on the test accuracy of the proposed mDLAM algorithm on four datasets.
3.6 Conclusion

In this chapter, we propose a novel formulation of the original neural network problem and a novel monotonous Deep Learning Alternating Minimization (mDLAM) algorithm. Specifically, the nonlinear constraint is projected into a convex set so that all subproblems are solvable. The Nesterov acceleration technique is applied to boost the convergence of the proposed mDLAM algorithm. Furthermore, a mild assumption is established to prove the convergence of our mDLAM algorithm. Our mDLAM algorithm can achieve a linear convergence rate, which is theoretically better than existing AM methods. The effectiveness of the proposed mDLAM algorithm is demonstrated via the outstanding performance on four benchmark datasets compared with state-of-the-art optimizers.
Chapter 4

The pdADMM Algorithm

4.1 Introduction

Due to wide applications and significant success in various applications, the training of Deep Neural Network (DNN) models has gained ever-increasing attention from the machine learning community. In recent years, the constant improvement of DNNs’ performance is accompanied by a fast increase in models’ complexity and size, which indicates a clear trend toward larger and deeper networks. Such a trend leads to severe challenges for large models to fit into a single computing unit (e.g., GPU), and raises urgent demands for partitioning the model into different computing devices to parallel training. However, the inherent bottleneck from GD which prevents the gradients of different layers from being calculated in parallel. This is because in GD gradient calculations of one layer tightly depend on and have to wait for the calculated results of all previous layers, which prevents the gradients of different layers from being calculated in parallel.

To work around the drawback of gradient-based methods, alternating minimization methods have caught fast increasing attention in recent years. Amongst them for deep learning optimization, ADMM-based methods are deemed to have great po-
tential for parallelism of deep neural network training, due to their inherent nature, which can break an objective into multiple subproblems, each of which can be solved in parallel [6].

Despite the potential, a parallel algorithm based on ADMM for deep neural network training has rarely been explored, developed, and evaluated until now, due to the layer dependency among subproblems of ADMM. Even though the ADMM reduces the layer dependency compared with GD, one subproblem of ADMM is dependent on its previous subproblem. Therefore, existing ADMM-based optimizers still update parameters sequentially.

To handle the difficulties of layer dependency, in this chapter we propose a novel parallel deep learning Alternating Direction Method of Multipliers(pdADMM) optimization framework to train large-scale neural networks. Our contributions in this chapter include:

- We propose a novel reformulation of the Multi-Layer Perceptron(MLP) neural network problem, which splits a neural network into independent layer partitions and allows for ADMM to achieve model parallelism.

- We present a model-parallelism version of the ADMM algorithm to train an MLP neural network. All parameters in each layer can be updated in parallel to speed up the training process significantly. All subproblems generated by the pdADMM algorithm are discussed in detail.

- We investigated the convergence properties of parallel ADMM in the common nonlinear activation functions such as the Rectified linear unit(ReLU), and we prove that the pdADMM converges to a state-of-the-art stationary point with a sublinear convergence rate $o(1/k)$.

- We conduct extensive experiments on six benchmark datasets to show the massive speedup of the proposed pdADMM as well as its competitive performance
with state-of-the-art optimizers.

The organization of this chapter is shown as follows: In Section 4.2, we summarize recent related research work to this chapter. In Section 4.3, we formulate the novel pdADMM algorithm to train a MLP neural network. In Section 4.4, the convergence guarantee of pdADMM to a stationary point is provided. Extensive experiments on benchmark datasets to demonstrate the convergence, speedup and comparable performance of pdADMM are shown in Section 4.5, and Section 4.6 concludes this work.

4.2 Related Work

**Distributed ADMM:** ADMM is one of the commonly applied techniques in distributed optimization. Overall, the previous works on distributed ADMM can be classified into two categories: synchronous problems and asynchronous problems. Synchronous problems usually require workers to optimize parameters in time before the master update the consensus variable, while asynchronous problems allow some workers to delay parameter updates. Most literature focused on the application of the distributed ADMM to synchronous problems. For example, Mota et al. utilized the distributed ADMM for the congestion control problem [70]; Makhdoumi and Ozdaglar studied the convergence properties of the distributed ADMM on the network communication problem. For more work, please refer to [10, 11, 90, 116, 126].

On the other hand, a handful of papers investigated how asynchronous problems can be addressed by distributed ADMM. For instance, Zhang et al., Wei et al., Chang et al and Hong proved the convergence of the distributed ADMM on asynchronous problems [12, 13, 39, 123, 108]. Kumar et al. discussed the application of the ADMM on multi-agent problems over heterogeneous networks [56]. However, there still lacks a general framework for ADMM to train deep neural networks in a distributed fash-
4.2. RELATED WORK

Distributed Deep Learning: With the increased volume of data and layers of neural networks, there is a need to design distributed systems to train a deep neural network for large-scale applications. Most recent papers have proposed gradient-based distributed systems to train neural networks: For example, Wen et al. proposed Terngrad to accelerate distributed deep learning in data parallelism [109]; Sergeev et al. presented an open-source library Horovod to reduce communication overhead [88]. Other systems include SINGA [77], Mxnet [16], TicTac [36] and Poseidon [122].

Data and Model Parallelism: Data parallelism focuses on distributing data across different processors, which can be implemented in parallel. Scaling GD is one of the most common ways to reach data parallelism [129]. For example, the distributed architecture, Poseidon, is achieved by scaling GD through overlapping communication and computation over networks. The recently proposed ADMM [93, 101] is another way of data parallelism: each subproblem generated by ADMM can be solved in parallel. 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 these 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 in parallel and hence reduce time delay. For instance, Parpas and Muir proposed a parallel-in-time method from the perspective of dynamic systems [79]; Huo et al. introduced a feature replay algorithm to achieve model parallelism [43]. Zhuang et al. broke layer dependency by introducing the delayed gradient [127]. However, to the best of our knowledge, there still lacks an exploration of how to achieve model parallelism via ADMM.
4.3 pdADMM

We propose the pdADMM algorithm in this section. Specifically, Section 4.3.1 introduces the existing deep learning ADMM method, reformulates the problem, and presents the pdADMM algorithm in detail. Section 4.3.2 discusses all subproblems generated by pdADMM and the strategy to train a large-scale deep neural network via pdADMM.

4.3.1 Problem Reformulation

Problem 1 has been addressed by the proposed dlADMM and mDLAM in Chapters 2 and 3, respectively. However, parameters in one layer are dependent on its neighboring layers and hence can not achieve parallelism. For example, the update of $a_{l+1}$ on the $(l + 1)$-th layer needs to wait before $z_l$ on the $l$-th layer is updated. In order to address layer dependency, we relax Problem 1 to Problem 4 as follows:

**Problem 4.**

$$
\min_{p, W, z, q} F(p, W, z, q) = R(z_L, y) + \frac{\nu}{2}\left(\sum_{l=1}^{L} \|z_l - W_lp_l\|_2^2 + \sum_{l=1}^{L-1} \|q_l - f_l(z_l)\|_2^2\right)
$$

s.t. $p_{l+1} = q_l$.

where $p_l$ and $q_l$ are the input and the output of the $i$-th layer, respectively, $p = \{p_l\}_{l=1}^{L}$, $W = \{W_l\}_{l=1}^{L}$, $z = \{z_l\}_{l=1}^{L}$, $q = \{q_l\}_{l=1}^{L-1}$, and $\nu > 0$ is a tuning parameter.

We reduce layer dependency by splitting the output of the $l$-th layer and the input of the $(l + 1)$-th layer into two variables $p_{l+1}$ and $q_l$, respectively. As $\nu \to \infty$, Problem 4 approaches Problem 1.

The high-level overview of the pdADMM algorithm is shown in Figure 4.1. Specifically, by breaking the whole neural network into multiple layers, each of which can be optimized by an independent worker. Therefore, layerwise training can be imple-
mented 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 4, the augmented Lagrangian function is formulated mathematically as follows:

\[
L_\rho(p, W, z, q, u) = F(p, W, z, q) + \sum_{l=1}^{L-1} (u_l^T(p_{l+1} - q_l) + (\rho/2)\|p_{l+1} - q_l\|_2^2) \\
= R(z_L, y) + \phi(p_1, W_1, z_1) + \sum_{l=2}^{L} \phi(p_l, W_l, z_l, q_{l-1}, u_{l-1}) \\
+ (\nu/2) \sum_{l=1}^{L-1} \|q_l - f_l(z_l)\|_2^2.
\]

where \(\phi(p_1, W_1, z_1) = (\nu/2)\|z_1 - W_1 p_1\|_2^2, \phi(p_l, W_l, z_l, q_{l-1}, u_{l-1}) = (\nu/2)\|z_l - W_l p_l\|_2^2 + u_{l-1}^T(p_l - q_{l-1}) + (\rho/2)\|p_l - q_{l-1}\|_2^2, u_l(l = 1, \ldots, L - 1)\) are dual variables, \(\rho > 0\) is a parameter, and \(u = \{u_l\}_{l=1}^{L-1}\). The detail of the pdADMM is shown in Algorithm 5. Specifically, Lines 5-8 update primal variables \(p, W, z\) and \(q\), respectively, while Line 11 updates the dual variable \(u\). the discussion on how to solve subproblems generated by pdADMM is detailed in the next section.

**Algorithm 5:** The proposed pdADMM algorithm

**Require:** \(y, p_1 = x, \rho, \nu\).

**Ensure:** \(p, W, z, q\).

Initialize \(k = 0\).

while \(p^k, W^k, z^k, q^k\) not converged do

Update \(p_{l+1}^{k+1}\) of different \(l\) by Equation (4.1) in parallel.
Update \(W_{l+1}^{k+1}\) of different \(l\) by Equation (4.2) in parallel.
Update \(z_{l+1}^{k+1}\) of different \(l\) by Equations (4.3) and (4.4) in parallel.
Update \(q_{l+1}^{k+1}\) of different \(l\) by Equation (4.5) in parallel.
\(r_{l}^{k} \leftarrow p_{l+1}^{k+1} - q_{l+1}^{k+1}(l = 1, \ldots, L)\) in parallel # Compute residuals.
Update \(u_{l}^{k+1}\) of different \(l\) by Equation (4.6) in parallel.
\(k \leftarrow k + 1\).

end while

Output \(p, W, z, q\).
4.3.2 Solutions to All Subproblems

In this section, we discuss how to solve all subproblems generated by pdADMM in detail.

1. Update $p^{k+1}$

The variable $p^{k+1}$ is updated as follows:

$$p^{k+1}_l \leftarrow \arg \min_{p_l} L_{\rho}(p, W^k, z^k, q^k, u^k) = \phi(p_l; W_l^k, z_l^k, q_{l-1}^k, u_{l-1}^k)$$

Because $W_l$ and $p_l$ are coupled in $\phi$, solving $p_l$ should require the time-consuming operation of matrix inversion of $W_l$. To handle this, we apply similar quadratic approximation techniques as used in dlADMM [101] as follows:

$$p^{k+1}_l \leftarrow \arg \min_{p_l} U_l(p_l; \tau_l^{k+1}) \quad (4.1)$$
where $U_l(p_l; \tau_l^{k+1}) = \phi(p_l^k, W_l^k, z_l^k, q_{l-1}^k, u_{l-1}^k) + (\nabla p_l^k \phi(p_l^k, W_l^k, z_l^k, q_{l-1}^k, u_{l-1}^k)(p_l - p_l^k) + (\tau_l^{k+1}/2)\|p_l - p_l^k\|^2_2$, and $\tau_l^{k+1} > 0$ is a parameter. $\tau_l^{k+1}$ should satisfy $\phi(p_l^{k+1}, W_l^k, z_l^k, q_{l-1}^k, u_{l-1}^k) \leq U_l(p_l^{k+1}; \tau_l^{k+1})$. The solution to Equation (4.1) is: $p_l^{k+1} \leftarrow p_l^k - \nabla p_l^k \phi(p_l^k, W_l^k, z_l^k, q_{l-1}^k, u_{l-1}^k)/\tau_l^{k+1}$.

2. Update $W_l^{k+1}$

The variable $W_l^{k+1}$ is updated as follows:

$$W_l^{k+1} \leftarrow \arg\min_{W_l} L_{p_l}(p_l^{k+1}, W_l^k, z_l^k, q_l^k, u_l^k) = \arg\min_{W_l} \phi(p_l^{k+1}, W_l^1, z_l^1)$$

$$= \arg\min_{W_l} \begin{cases} \phi(p_l^{k+1}, W_l^1, z_l^1) & l = 1 \\ \phi(p_l^{k+1}, W_l, z_l^k, q_{l-1}^k, u_{l-1}^k) & 1 < l \leq L \end{cases}$$

Similar to updating $p_l$, the following subproblem should be solved instead:

$$W_l^{k+1} \leftarrow \arg\min_{W_l} V_l(W_l; \theta_l^{k+1})$$

where

$$V_l(W_l; \theta_l^{k+1}) = \phi(p_l^{k+1}, W_l^1, z_l^1) + \nabla W_l^1 \phi^T(p_l^{k+1}, W_l^1, z_l^1)(W_l - W_l^1)$$

$$+ (\theta_l^{k+1}/2)\|W_l - W_l^1\|^2_2$$

and $\theta_l^{k+1}$ is a parameter, which should satisfy $\phi(p_l^{k+1}, W_l^{k+1}, z_l^k) \leq V(W_l^{k+1}; \theta_l^{k+1})$ and $\phi(p_l^{k+1}, W_l^{k+1}, z_l^1, q_{l-1}^k, u_{l-1}^k) \leq V(W_l^{k+1}; \theta_l^{k+1})$. The solution to Equation
4.3. PDADMM

(4.2) is shown as follows:

\[
W_{l+1}^k \leftarrow W_l^k - \begin{cases} \\
\nabla W_1^k \phi(p_1^{k+1}, W_1^k, z_1^k)/\theta_l^{k+1} & l = 1 \\
\nabla W_l^k \phi(p_l^{k+1}, W_l^k, z_l^k, q_{l-1}^k, u_{l-1}^k)/\theta_l^{k+1} & 1 < l \leq L 
\end{cases}
\]

3. Update \( z^{k+1} \)

The variable \( z^{k+1} \) is updated as follows:

\[
z_l^{k+1} \leftarrow \arg \min_{z_l} \{ (\nu/2) \| z_l - W_l^{k+1} p_l^{k+1} \|^2_2 + (\nu/2) \| q_l^k - f_l(z_l) \|^2_2 + (\nu/2) \| z_l - z_l^k \|^2_2 \} \]  
\[ (l < L) \quad \text{(4.3)} \]

\[
z_L^{k+1} \leftarrow \arg \min_{z_L} R(z_L, y) + (\nu/2) \| z_L - W_L^{k+1} p_L^{k+1} \|^2_2 \]  
\[ \text{where a quadratic term } (\nu/2) \| z_l - z_l^k \|^2_2 \text{ is added in Equation (4.3) to control } z_l^{k+1} \text{ to close to } z_l^k. \text{ Equation (4.4) is convex, which can be solved by Fast Iterative Soft Thresholding Algorithm (FISTA) [2].} \]

For Equation (4.3), nonsmooth activations usually lead to closed-form solutions [101, 99]. For example, for ReLU \( f_l(z_l) = \max(z_l, 0) \), the solution to Equation (4.3) is shown as follows:

\[
z_l^{k+1} = \begin{cases} \\
\min((W_l^{k+1} p_l^{k+1} + z_l^k)/2, 0) & z_l^{k+1} \leq 0 \\
\max((W_l^{k+1} p_l^{k+1} + q_l^k + z_l^k)/3, 0) & z_l^{k+1} \geq 0 
\end{cases}
\]

For smooth activations such as tanh and sigmoid, a lookup-table is recommended [101].

4. Update \( q^{k+1} \)
4.4. CONVERGENCE ANALYSIS

The variable $q^{k+1}_l$ is updated as follows:

$$q^{k+1}_l \leftarrow \arg \min_{q_l} L_\rho(p^{k+1}_l, W^{k+1}_l, z^{k+1}_l, q^k, u^k) = \arg \min_{q_l} \phi(p^{k+1}_{l+1}, W^{k+1}_{l+1}, z^{k+1}_{l+1}, q_l, u^k).$$

(4.5)

Equation (4.5) has a closed-form solution as follows:

$$q^{k+1}_l \leftarrow (\rho p^{k+1}_{l+1} + u^k_l + \nu f_l(z^{k+1}_l))/\left(\rho + \nu\right)$$

5. Update $u^{k+1}_l$

The variable $u^{k+1}_l$ is updated as follows:

$$u^{k+1}_l \leftarrow u^k_l + \rho (p^{k+1}_{l+1} - q^{k+1}_l)$$

(4.6)

Finally, Our proposed pdADMM can be efficient for training a deep MLP neural network. To achieve this, we begin by training a swallow neural network with the first few layers of the deep neural network, then more layers are added for training step by step until finally all layers are involved in the training process. The pdADMM can achieve good performance as well as reduce training costs with this strategy.

4.4 Convergence Analysis

In this section, the theoretical convergence of the proposed pdADMM algorithm. Firstly, the Lipschitz continuity and coercivity are defined as follows:

**Definition 1.** (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 2.** (Coercivity) A function $h(x)$ is coercive over the feasible set $\mathcal{F}$ means that $h(x) \to \infty$ if $x \in \mathcal{F}$ and $\|x\| \to \infty$.

Then the following assumption is required for convergence analysis.

**Assumption 5.** $f_l(z_l)$ is Lipschitz continuous with coefficient $S > 0$, and $F$ 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 5 is mild to satisfy: most common activation functions such as ReLU and leaky ReLU satisfy Assumption 5. No assumption is needed on the risk function $R(z_L, y)$, which shows that the convergence condition of our proposed pdADMM is milder than that of the dlADMM, which requires $R(z_L, y)$ to be Lipschitz differentiable [101]. Due to space limit, the detailed proofs are provided in Section C.1 and Section C.2 in the Appendix. The technical proofs follow a similar routine as dlADMM [101]. The difference consists in the fact that the dual variable $u_l$ is controlled by $q_l$ and $z_l$ (Lemma 21 in Section C.1 in the Appendix), which holds under Assumption 5, 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 $\rho$ is sufficiently large.

**Lemma 5** (Decreasing Objective). 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
4.4. CONVERGENCE ANALYSIS

$k \in \mathbb{N}$ that

$$L_{\rho}(p^k, W^k, z^k, q^k, u^k) - L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})$$

$$\geq \sum_{t=2}^{L} (\tau_{t+1}^{k+1}/2)\|p_{t+1}^{k+1} - p_t^k\|_2^2 + \sum_{t=1}^{L} (\theta_{t+1}^{k+1}/2)\|W_t^{k+1} - W_t^k\|_2^2$$

$$+ \sum_{t=1}^{L-1} C_1\|\tau_{t+1}^{k+1} - \tau_t^k\|_2^2 + (\nu/2)\|\tau_{L+1}^{k+1} - \tau_L^k\|_2^2 + \sum_{t=1}^{L-1} C_2\|\tau_{t+1}^{k+1} - \tau_t^k\|_2^2$$

(4.7)

The second Lemma illustrates that the objective is bounded from below when $\rho$ is large enough, and all variables are bounded.

**Lemma 6** (Bounded Objective). If $\rho > \nu$, then $L_{\rho}(p^k, W^k, z^k, q^k, u^k)$ is lower bounded. Moreover, $p^k, W^k, z^k, q^k,$ and $u^k$ are bounded, i.e. there exist $N_p, N_w, N_z, N_q,$ and $N_u > 0$, such that $\|p^k\| \leq N_p$, $\|W^k\| \leq N_w$, $\|z^k\| \leq N_z$, $\|q^k\| \leq N_q$, and $\|u^k\| \leq N_u$.

Based on Lemmas 5 and 6, the following theorem ensures that the objective is convergent.

**Theorem 6** (Convergent Objective). If $\rho > \max(4\nu^2, (\sqrt{17} + 1)\nu/2)$, then $L_{\rho}(p^k, W^k, z^k, q^k, u^k)$ is convergent. Moreover, $\lim_{k \to \infty} \|p_{k+1}^k - p_k^k\|_2^2 = 0$, $\lim_{k \to \infty} \|W_{k+1}^k - W_k^k\|_2^2 = 0$, $\lim_{k \to \infty} \|z_{k+1}^k - z_k^k\|_2^2 = 0$, $\lim_{k \to \infty} \|q_{k+1}^k - q_k^k\|_2^2 = 0$, and $\lim_{k \to \infty} \|u_{k+1}^k - u_k^k\|_2^2 = 0$.

**Proof.** From Lemmas 5 and 6, we know that $L_{\rho}(p^k, W^k, z^k, q^k, u^k)$ is convergent because a monotone bounded sequence converges. Moreover, we take the limit on both sides of Inequality (4.7) to obtain

$$0 = \lim_{k \to \infty} L_{\rho}(p^k, W^k, z^k, q^k, u^k) - \lim_{k \to \infty} L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})$$

$$\geq \lim_{k \to \infty} \left( \sum_{t=2}^{L} (\tau_{t+1}^{k+1}/2)\|p_{t+1}^{k+1} - p_t^k\|_2^2 + \sum_{t=1}^{L} (\theta_{t+1}^{k+1}/2)\|W_t^{k+1} - W_t^k\|_2^2 \right.$$

$$\left. + \sum_{t=1}^{L-1} C_1\|\tau_{t+1}^{k+1} - \tau_t^k\|_2^2 + (\nu/2)\|\tau_{L+1}^{k+1} - \tau_L^k\|_2^2 + \sum_{t=1}^{L-1} C_2\|\tau_{t+1}^{k+1} - \tau_t^k\|_2^2 \right) \geq 0$$
Because $L_{\rho}(p^k, W^k, z^k, q^k, u^k)$ is convergent, then $\lim_{k \to \infty} \|p^{k+1} - p^k\|^2 = 0,$ $\lim_{k \to \infty} \|W^{k+1} - W^k\|^2 = 0,$ $\lim_{k \to \infty} \|z^{k+1} - z^k\|^2 = 0,$ and $\lim_{k \to \infty} \|q^{k+1} - q^k\|^2 = 0.$ $\lim_{k \to \infty} \|u^{k+1} - u^k\|^2 = 0$ is derived from Lemma 21 in Section C.1 in the Appendix.

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

**Lemma 7** (Bounded Subgradient). There exists a constant $C > 0$ and $g^{k+1} \in \partial L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})$ such that

$$
\|g^{k+1}\| \leq C(\|p^{k+1} - p^k\| + \|W^{k+1} - W^k\| + \|z^{k+1} - z^k\| + \|q^{k+1} - q^k\| + \|u^{k+1} - u^k\|)
$$

Now based on Theorem 6, and Lemma 7, the convergence of the pdADMM algorithm to a stationary point is presented in the following theorem.

**Theorem 7** (Convergence to a stationary Point). If $\rho > \max(4\nu S^2, (\sqrt{17} + 1)\nu/2)$, then for the variables $(p, W, z, q, u)$ in Problem 4, starting from any initialization $(p^0, W^0, z^0, q^0, u^0)$, $(p^k, W^k, z^k, q^k, u^k)$ has at least a limit point $(p^*, W^*, z^*, q^*, u^*)$, and any limit point is a stationary point of Problem 4 (i.e. $0 \in \partial L_{\rho}(p^*, W^*, z^*, q^*, u^*)$).

In other words,

$$
p^*_l + 1 = q^*_l, \ n_p L_{\rho}(p^*, W^*, z^*, q^*, u^*) = 0, \ n_w L_{\rho}(p^*, W^*, z^*, q^*, u^*) = 0,
0 \in \partial z^* L_{\rho}(p^*, W^*, z^*, q^*, u^*), \ n_q L_{\rho}(p^*, W^*, z^*, q^*, u^*) = 0.
$$

**Proof.** From Lemma 6, $(p^k, W^k, z^k, q^k, u^k)$ has at least a limit point $(p^*, W^*, z^*, q^*, u^*)$ because a bounded sequence has at least a limit point. From Lemma 7 and Theorem 6, $\|g^{k+1}\| \to 0$ as $k \to \infty$. According to the definition of general subgradient (Defintion
Theorem 8 (Convergence Rate). For a sequence \((p^k, W^k, z^k, q^k, u^k)\), define 
\[
c_k = \min_{0 \leq i \leq k} \left( \sum_{l=2}^{L} (\tau_{i}^{l+1}/2) \|p_{i}^{l+1} - p_{i}^{l}\|^2_2 + \sum_{i=1}^{L} (\theta_{i}^{l+1}/2) \|W_{i}^{l+1} - W_{i}^{l}\|^2_2 + \sum_{i=1}^{L-1} C_1 \|z_{i}^{l+1} - z_{i}^{l}\|^2_2 + (\nu/2) \|z_{L}^{l+1} - z_{L}^{l}\|^2_2 + \sum_{i=1}^{L-1} C_2 \|q_{i}^{l+1} - q_{i}^{l}\|^2_2 \right)
\]
where 
\[
C_1 = \nu/2 - 2\nu^2 S^2 / \rho > 0 \quad \text{and} \quad C_2 = \rho/2 - 2\nu^2 / \rho - \nu/2 > 0,
\]
then the convergence rate of \(c_k\) is \(o(1/k)\).

4.5 Experiments

In this section, we evaluate the performance of the proposed pdADMM using six benchmark datasets. Speedup, convergence, and accuracy performance are compared with several state-of-the-art optimizers. All experiments were conducted on a 64-bit machine with Intel Xeon(R) silver 4114 Processor and 48GB RAM.

4.5.1 Datasets

In this experiment, six benchmark datasets were used for performance evaluation:

1. MNIST [58]. The MNIST dataset has ten classes of handwritten-digit images, which was firstly introduced by Lecun et al. in 1998 [58]. It contains 55,000 training samples and 10,000 test samples with 196 features each, which is provided by the Keras library [17].

2. Fashion MNIST [112]. The Fashion MNIST dataset has ten classes of assortment images on the website of Zalando, which is Europe’s largest online fashion platform [112]. The Fashion-MNIST dataset consists of 60,000 training samples and 10,000
test samples with 196 features each.

3. Kuzushiji-MNIST (kMNIST) [20]. The kMNIST dataset has ten classes, each of which is a character to represent each of the 10 rows of Hiragana. The kMNIST dataset consists of 60,000 training samples and 10,000 test samples with 196 features each.

4. Street View House Numbers (SVHN) [72]. The SVHN dataset is obtained from house numbers in Google Street View images. It consists of ten classes of digits. In our experiments, we use three classes '0', '1' and '2'. The number of training data and test data are 24,446 and 9,248, respectively, with 768 features each.

5. CIFAR10 [55]. CIFAR10 is a collection of color images with 10 different classes. In our experiments, we use two classes '0' and '6'. The number of training data and test data are 12,000 and 2,000, respectively, with 768 features each.

6. CIFAR100 [55]. CIFAR100 is similar to CIFAR10 except that CIFAR100 has 100 classes. In our experiments, we use two classes '0' and '2'. The number of training data and test data are 5,000 and 1,000, respectively, with 768 features each.

### 4.5.2 Speedup

<table>
<thead>
<tr>
<th>Neurons#</th>
<th>Serial pdADMM(sec)</th>
<th>pdADMM(sec)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500</td>
<td>237.66</td>
<td>26.78</td>
<td>8.87</td>
</tr>
<tr>
<td>1600</td>
<td>348.70</td>
<td>31.78</td>
<td>10.97</td>
</tr>
<tr>
<td>1700</td>
<td>390.51</td>
<td>35.79</td>
<td>10.91</td>
</tr>
<tr>
<td>1800</td>
<td>475.60</td>
<td>41.37</td>
<td>11.50</td>
</tr>
<tr>
<td>1900</td>
<td>465.57</td>
<td>45.87</td>
<td>10.15</td>
</tr>
<tr>
<td>2000</td>
<td>570.90</td>
<td>50.70</td>
<td>11.26</td>
</tr>
<tr>
<td>2000</td>
<td>570.90</td>
<td>50.70</td>
<td>11.26</td>
</tr>
</tbody>
</table>
### 4.5. EXPERIMENTS

<table>
<thead>
<tr>
<th>Neurons#</th>
<th>Serial pdADMM (sec)</th>
<th>pdADMM (sec)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>2100</td>
<td>570</td>
<td>54.91</td>
<td>10.38</td>
</tr>
<tr>
<td>2200</td>
<td>678.83</td>
<td>63.59</td>
<td>10.68</td>
</tr>
<tr>
<td>2300</td>
<td>710.3</td>
<td>70.36</td>
<td>10.10</td>
</tr>
<tr>
<td>2400</td>
<td>766.82</td>
<td>62.5</td>
<td>12.27</td>
</tr>
</tbody>
</table>

**Fashion MNIST dataset**

<table>
<thead>
<tr>
<th>Neurons#</th>
<th>Serial pdADMM (sec)</th>
<th>pdADMM (sec)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500</td>
<td>358.68</td>
<td>32.65</td>
<td>10.99</td>
</tr>
<tr>
<td>1600</td>
<td>407.71</td>
<td>37.90</td>
<td>10.76</td>
</tr>
<tr>
<td>1700</td>
<td>476.79</td>
<td>44.75</td>
<td>10.65</td>
</tr>
<tr>
<td>1800</td>
<td>539.51</td>
<td>50.50</td>
<td>10.68</td>
</tr>
<tr>
<td>1900</td>
<td>599.42</td>
<td>53.88</td>
<td>11.13</td>
</tr>
<tr>
<td>2000</td>
<td>645.87</td>
<td>58.68</td>
<td>11.01</td>
</tr>
<tr>
<td>2100</td>
<td>740.39</td>
<td>67.91</td>
<td>10.90</td>
</tr>
<tr>
<td>2200</td>
<td>818.58</td>
<td>74.17</td>
<td>11.03</td>
</tr>
</tbody>
</table>

**kMNIST dataset**

<table>
<thead>
<tr>
<th>Neurons#</th>
<th>Serial pdADMM (sec)</th>
<th>pdADMM (sec)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500</td>
<td>354.85</td>
<td>32.65</td>
<td>10.87</td>
</tr>
<tr>
<td>1600</td>
<td>407.73</td>
<td>37.11</td>
<td>10.99</td>
</tr>
<tr>
<td>1700</td>
<td>472.46</td>
<td>42.58</td>
<td>11.10</td>
</tr>
<tr>
<td>1800</td>
<td>539.52</td>
<td>48.78</td>
<td>11.06</td>
</tr>
<tr>
<td>1900</td>
<td>596.84</td>
<td>55.56</td>
<td>10.74</td>
</tr>
<tr>
<td>2000</td>
<td>660.58</td>
<td>56.10</td>
<td>11.78</td>
</tr>
<tr>
<td>2100</td>
<td>737.78</td>
<td>66.95</td>
<td>11.02</td>
</tr>
<tr>
<td>2200</td>
<td>806.74</td>
<td>76.16</td>
<td>10.59</td>
</tr>
</tbody>
</table>

**CIFAR10 dataset**

<table>
<thead>
<tr>
<th>Neurons#</th>
<th>Serial pdADMM (sec)</th>
<th>pdADMM (sec)</th>
<th>Speedup</th>
</tr>
</thead>
</table>
In this experiment, we investigate the speedup of the proposed pdADMM algorithm concerning the number of layers and the number of neurons on large-scale deep learning models.
4.5. EXPERIMENTS

Figure 4.2: The relationship between the speedup of the proposed pdADMM algorithm and the number of layers on five datasets.

neural networks. The activation function was set to ReLU. The loss function was the cross-entropy loss. The running time per epoch was the average of 10 epochs. \( \rho \) and \( \nu \) were both set to \( 10^{-4} \).

Firstly, we investigated the relationship between speedup and the number of layers. We set up an MLP neural network with a different number of hidden layers, which ranges from 11 to 19. The number of neurons in each layer was fixed to 2,400. The SVHN dataset was not tried due to memory issues. Figure 4.2 shows that the speedup increases linearly with the number of layers. Specifically, the speedup reached 11 when 19 hidden layers were trained. This indicates that the deeper a neural network is, the more speedup our proposed pdADMM can gain.

Secondly, the relationship between speedup and the number of neurons was studied. Specifically, we test our proposed pdADMM algorithm on an MLP neural network with 19 hidden layers. The number of neurons in each layer ranges from 1,500 to 2,400. The speedup was shown in Table 4.1 on the MNIST and Fashion MNIST
4.5. EXPERIMENTS

datasets. Specifically, the speedup remains stable at around 10 no matter how many neurons were trained. This concludes that the speedup of the proposed pdADMM is independent of the number of neurons.

### 4.5.3 Convergence

To validate the convergence of the proposed pdADMM, we set up an MLP neural network with 9 hidden layers, each of which has 500 neurons. The ReLU was used for the activation function for both network structures. The loss function was set as the cross-entropy loss. The number of the epoch was set to 100. \( \nu \) and \( \rho \) were both set to 0.1. As shown in Figure 4.3, the objective keeps decreasing monotonically on all six datasets, and the residual converges sublinearly to 0, which is consistent with Theorems 7 and 8.

### 4.5.4 Performance

**Experimental Settings**

To evaluate the performance of the proposed pdADMM algorithm, we used the same architecture as the previous section. \( \nu \) and \( \rho \) were set to \( 10^{-4} \) in order to maximize the performance of training data. The number of the epoch was set to 100. In this
4.5. EXPERIMENTS

Figure 4.4: Training accuracy of the proposed pdADMM algorithm against all comparison methods on six datasets.

Figure 4.5: Test accuracy of the proposed pdADMM algorithm against all comparison methods on six datasets.
experiment, the full batch was used for training. As suggested by the training strategy in Section 4.3.2, we firstly trained an MLP neural network with five hidden layers, and then all layers were involved in training.

**Comparison Methods**

GD and its variants are state-of-the-art methods and hence served as comparison methods. For GD-based methods, the full batch dataset is used for training models. All parameters were chosen by maximizing the accuracy of training datasets. The baselines are described as follows:

1. Gradient Descent (GD) [4]. The GD and its variants are the most popular deep learning optimizers, whose convergence has been studied extensively in the literature. The learning rate of GD was set to 0.01.

2. Adaptive learning rate method (Adadelta) [118]. The Adadelta is proposed to overcome the sensitivity to hyperparameter selection. The learning rate of Adadelta was set to 1.

3. Adaptive gradient algorithm (Adagrad) [28]. Adagrad is an improved version of GD: rather than fixing the learning rate during training, it adapts the learning rate to the hyperparameter. The learning rate of Adagrad was set to 0.1.

4. Adaptive momentum estimation (Adam) [52]. 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 makes convergence fast. The learning rate for Adam was set to 0.001.

5. Deep learning Alternating Direction Method of Multipliers (dlADMM) [101]. The dlADMM is an improvement of the previous ADMM implementation [93]. It is guaranteed to converge to a stationary point with a rate of $o(1/k)$. $\rho$ and $\nu$ were both set to $10^{-6}$. 
4.6 Conclusion

The ADMM is considered to be a good alternative to GD for training deep neural networks. In this chapter, we propose a novel parallel deep learning Alternating Direction Method of Multipliers(pdADMM) to achieve layer parallelism. It is guaranteed to converge to a stationary solution under mild conditions. Experiments on benchmark datasets demonstrate that our proposed pdADMM not only leads to a huge speedup for deep MLPs, but also outperforms others on six benchmark datasets.
Chapter 5

The pdADMM-G and pdADMM-G-Q Algorithms

5.1 Introduction

In Chapter 4, we demonstrate the effectiveness of the proposed pdADMM algorithm. In this Chapter, we extend our proposed pdADMM to graph applications, which can be possibly achieved by Graph Neural Networks (GNNs). They have accomplished 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 [53]. 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) [15], and the over-squashing problem (i.e. the information flow among distant nodes distorts along the long-distance interactions) [96]. These challenges still exist even though some models such as GraphSAGE [35] have been proposed to alleviate them.

On the other hand, the Graph Augmented Multi-Layer 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 Multi-Layer Perceptron (MLP) models. Compared with GNNs, GA-MLP models are more resistant to the over-smoothing problem [15] and therefore demonstrate outstanding performance. For example, Wu et al. showed that a two-layer GA-MLP approximates the performance of the GNN models on multiple datasets [110].

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 (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, we extend the proposed pdADMM algorithm to train large-scale GA-MLP models, named the graph pdADMM (pdADMM-G) algorithm, and the quantized graph pdADMM (pdADMM-G-Q) algorithm reduces communication costs of the pdADMM-G algorithm by the quantization technique. Our contributions to this chapter include:

- **We propose a novel reformulation of GA-MLP models.** It splits a neural
network into independent layer partitions and allow for ADMM to achieve model parallelism.

- **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.

- **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 $o(1/k)$.

- **We conduct extensive experiments to show the effectiveness of two proposed algorithms.** Experiments on nine benchmark datasets 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 state-of-the-art optimizers. Moreover, the pdADMM-G-Q algorithm reduces communication overheads by up to 45%.

The organization of this chapter is shown as follows: In Section 5.2, we summarize recent related research work to this chapter. In Section 5.3, we propose the pdADMM-G algorithm and the pdADMM-G-Q algorithm to train deep GA-MLP models. Section 5.4 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 5.5, and Section 5.6 concludes this work.
5.2 Related Work

This section summarizes existing literature related to this chapter. One branch of related research is distributed deep learning, which has been discussed in Section 4.2 in Chapter 4. The other branch of related papers are deep learning on graph, which is shown as follows:

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 [111]. It classified existing GNN models into four categories: Recurrent Graph Neural Networks (RecGNNs), Convolutional Graph Neural Networks (ConvGNNs), Graph Autoencoders (GAEs), and Spatial-Temporal Graph Neural Networks (STGNNs).

RecGNNs learn node representation with recurrent neural networks via the message passing mechanisms [23, 29, 63]; ConvGNNs generalize the operations of convolution to graph data and stack multiple convolution layers to extract high-level node features [7, 25, 38]; GAEs encode node information into a latent space and reconstruct graphs from the encoded node representation [8, 78, 97]; the idea of STGNNs is to capture spatial dependency and temporal dependency simultaneously [48, 62, 87].

5.3 The pdADMM-G Algorithm

We propose the pdADMM-G algorithm to solve GA-MLP models in this section. Specifically, Section 5.3.1 formulates the GA-MLP model training problem, and Section 5.3.2 proposes the pdADMM-G algorithm. Section 5.3.3 extends the proposed pdADMM-G algorithm to the pdADMM-G-Q algorithm for quantization.
5.3. THE PDADMM-G ALGORITHM

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

Table 5.1: Important Notations

5.3.1 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 \(\Psi = \{\psi_1(A), \cdots, \psi_K(A)\}\) be a set of (usually multi-hop) operators \(\psi_i(A) : \mathbb{R}^{|V|} \rightarrow \mathbb{R}^{|V|} (i = 1, \cdots, K)\) that are functions of the adjacency matrix \(A \in \{0,1\}^{|V|\times|V|}\), and \(\mathbb{R}^{|V|}\) is the domain of \(\psi_i(A)\) \((i = 1, \cdots, K)\). \(x_k = H\psi_k(A)\) 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. \(x_k (k = 1, \cdots, K)\) are stacked into \(x = [x_1; \cdots; x_K]\) by column. Then the GA-MLP training problem is formulated as follows [15]:

**Problem 5.**

\[
\min_{W_l,z_l,p_l} R(z_L, y), \quad \text{s.t.} \quad z_l = W_l p_l, \quad p_{l+1} = f_l(z_l) (l = 1, \cdots, L - 1),
\]

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 of the \(l\)-th layer, also the output for the \((l - 1)\)-th layer, and \(n_l\) is the number of neurons for the \(l\)-th layer. \(R(z_L, y)\) is a risk function of the \(L\)-th layer, which is convex and
5.3. THE PDADMM-G ALGORITHM

continuous; $z_l = W_l p_l$ and $p_{l+1} = f_l(z_l)$ are linear and nonlinear mappings of the $l$-th layer, respectively, and $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$ is the weight matrix of the $l$-th layer.

In Problem 5, $\Psi$ can be considered as a prepossessing step to augment node features via $A$, and hence it is predefined. One common choice can be $\Psi = \{I, A, A^2, \ldots, A^{K-1}\}$.

We apply the pdADMM algorithm in Chapter 4 to Problem 5 to realize model parallelism training. To achieve this, we relax Problem 5 to Problem 6 as follows:

Problem 6.

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

s.t. $p_{l+1} = q_l$.

where $p = \{p_l\}_{l=1}^{L}$, $W = \{W_l\}_{l=1}^{L}$, $z = \{z_l\}_{l=1}^{L}$, $q = \{q_l\}_{l=1}^{L-1}$, and $\nu > 0$ is a tuning parameter. We reduce layer dependency by splitting the output of the $l$-th layer and the input of the $(l+1)$-th layer into two variables $p_{l+1}$ and $q_l$, respectively. As $\nu \to \infty$, Problem 6 approaches Problem 5.

5.3.2 The pdADMM-G Algorithm

The high-level overview of the pdADMM-G algorithm is shown in Figure 5.1. 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 6. The augmented Lagrangian
5.3. THE PDADMM-G ALGORITHM

Figure 5.1: The overview of the proposed pdADMM-G optimization algorithm.

The function is formulated mathematically as follows:

\[ L_\rho(p, W, z, q, u) = F(p, W, z) + \sum_{t=1}^{L-1} (u_t^T(p_{t+1} - q_t) + (\rho/2)\|p_{t+1} - q_t\|^2_2) \]

\[ = R(z_L, y) + \phi(p_1, W_1, z_1) + \sum_{t=2}^{L} \phi(p_t, W_t, z_t, q_{t-1}, u_{t-1}) + (\nu/2) \sum_{l=1}^{L-1} \|q_l - f_l(z_l)\|^2_2, \]

where \( \phi(p_1, W_1, z_1) = (\nu/2)\|z_1 - W_1p_1\|^2_2, \phi(p_t, W_t, z_t, q_{t-1}, u_{t-1}) = (\nu/2)\|z_t - W_tp_t\|^2_2 + u_{t-1}^T(p_t - q_{t-1}) + (\rho/2)\|p_t - q_{t-1}\|^2_2, \)

\( u_t(l = 1, \cdots, L - 1) \) are dual variables, \( \rho > 0 \) is a parameter, and \( U = \{u_l\}_{l=1}^{L-1} \). The details of the pdADMM-G algorithm are shown in Algorithm 6. Specifically, Lines 5-8 update primal variables \( p, W, z \) and \( q \), respectively, while Line 10 updates the dual variable \( u \). The details of all subproblems are shown in Section 4.3 in Chapter 4.

Our proposed pdADMM-G algorithm can be efficient for training deep GA-MLP
5.3. THE PDADMM-G ALGORITHM

Algorithm 6: The proposed pdADMM-G algorithm

Require: $y$, $p_1 = X$, $\rho$, $\nu$.
Ensure: $p$, $W$, $z$, $q$.

Initialize $k = 0$.

while $p^k$, $W^k$, $z^k$, $q^k$ not converged do
    $p_{l}^{k+1} \leftarrow \arg \min_{p_l} L_{\rho}(p, W^k, z^k, q^k, u^k)$ for different $l$ in parallel.
    $W_{l}^{k+1} \leftarrow \arg \min_{W_l} L_{\rho}(p^{k+1}, W, z^k, q^k, u^k)$ for different $l$ in parallel.
    $z_{l}^{k+1} \leftarrow \arg \min_{z_l} L_{\rho}(p^{k+1}, W^k, z, q^k, u^k)$ for different $l$ in parallel.
    $q_{l}^{k+1} \leftarrow \arg \min_{q_l} L_{\rho}(p^{k+1}, W^k, z^k, q, u^k)$ for different $l$ in parallel.
    $r_{l}^{k} \leftarrow p_{l}^{k+1} - q_{l}^{k+1}$ ($l = 1, \ldots, L$) in parallel # Compute residuals.
    $u_{l}^{k+1} \leftarrow u_{l}^{k} + \rho(p_{l}^{k+1} - q_{l}^{k+1})$ for different $l$ in parallel.
    $k \leftarrow k + 1$.
end while

Output $p$, $W$, $z$, $q$.

models via the greedy layerwise training strategy [3]. Specifically, we begin by training a 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 gradient descent 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 $z_t = W_t p_t$ in the forward pass, where $W_t \in \mathbb{R}^{n_l \times n_{l-1}}$ and $p_t \in \mathbb{R}^{n_{l-1} \times |V|}$, which requires a time complexity of $O(n_l n_{l-1} |V|)$ [21]; for the proposed pdADMM-G algorithm, the most costly operation is to compute the derivative $\nabla_{W_t} \phi$, 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 [103], 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
5.3. THE PDADMM-G ALGORITHM

5.3.3 Quantization Extension of pdADMM-G (pdADMM-G-Q)

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 [42]. In other words, \( p_l \) is required to fit into a countable set \( \Delta \), which is shown as follows:

**Problem 7.**

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

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 \( \Delta \). To address Problem 7, we rewrite it into the following form:

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

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 7 is shown as follows:

\[
\beta_\rho(p, W, z, q, u) = L_\rho(p, W, z, q, u) + \sum_{l=2}^{L} \mathbb{I}(p_l),
\]
where $L_{\rho}$ is the augmented Lagrangian of Problem 6. The extended pdADMM-G-Q algorithm follows the same routine as the pdADMM-G algorithm, where $L_{\rho}$ is replaced with $\beta_{\rho}$. Obviously, the only difference between the pdADMM-G-Q algorithm and the pdADMM-G algorithm is the $p_l$-subproblem, which is outlined in the following:

\[ p_{l}^{k+1} \leftarrow \arg\min_{p_l} U_l(p_l; \tau_{l}^{k+1}) + \mathbb{I}(p_l), \quad (5.1) \]

where $U_l$ follows Equation (4.1). The solution to Equation (5.1) is [42]:

\[ p_{l}^{k+1} \leftarrow \arg\min_{\delta \in \Delta} \| \delta - (p_{k}^{l} - \nabla p_{l}^{k} \phi(p_{l}^{k}, W_{l}^{k}, b_{l}^{k}, z_{l}^{k}, q_{l-1}^{k}, u_{l-1}^{k}) + \tau_{l}^{k+1}) \| \cdot \]

For the pdADMM-G-Q algorithm, the variable $p$ is only required to be quantized (i.e. $p_l \in \Delta$) when the $p_l$-subproblem is solved (i.e. Equation (5.1)), and the variable $q$ can be any real number when it is updated. However, $q$ is guaranteed to fit into $\Delta$ by the linear constraint $p_{l+1} = q_l$. This design is convenient for the convergence analysis, which is detailed in the next section. One variant of the pdADMM-G-Q algorithm is to quantize $p$ and $q$ (i.e. $p_l, q_l \in \Delta$) when they are updated. In this case, the solution to $q_l$ subproblem is

\[ q_{l}^{k+1} \leftarrow \arg\min_{\delta \in \Delta} \| \delta - (\rho p_{l+1}^{k+1} + u_{l}^{k} + \nu f_{l}(z_{l}^{k+1})/(\rho + \nu) \| \cdot \]

### 5.4 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 D.1 in the Appendix. 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 6, and prove that \( u_l \) is controlled via \( q_l \) and \( z_l \) in Lemma 25 in Section D.1 in the Appendix.

Firstly, the proper function, Lipschitz continuity, and coercivity are defined as follows:

**Definition 3 (Proper Functions).** [84]. For a convex function \( g(x): \mathbb{R} \to \mathbb{R} \cup \{\pm \infty\} \), \( g \) 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 4.** (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 5.** (Coercivity) A function \( h(x) \) is coercive over the feasible set \( \mathcal{F} \) means that \( h(x) \to \infty \) if \( x \in \mathcal{F} \) and \( \|x\| \to \infty \).

Next, the definition of a quantized stationary point [42] is shown as follows:

**Definition 6.** (Quantized Stationary Point) The \( p_l \) is a quantized stationary point of of Problem 7 if there exists \( \tau > 0 \) such that

\[
p_l \in \arg\min_{\delta \in \Delta} \|\delta - (p_l - \nabla_p F(p, W, z, q)/\tau)\|.
\]

The quantized stationary point is an extension of the stationary point in the discrete setting, and any global solution \( p_l \) to Problem 7 is a quantized stationary point to Problem 7 (Lemma 3.7 in [42]). Then the following assumption is required for convergence analysis.

**Assumption 6.** \( f_l(z_l) \) is Lipschitz continuous with coefficient \( S > 0 \), \( R(z_L, y) \) is proper, and \( F(p, W, z, q) \) is coercive. Moreover, \( \partial f_l(z_l) \) is bounded, i.e. there exists \( M > 0 \) such that \( \|\partial f_l(z_l)\| \leq M \).
5.4. CONVERGENCE ANALYSIS

Assumption 6 is mild to satisfy: most common activation functions such as Rectified Linear Unit (ReLU) [103] and leaky ReLU [114] satisfy Assumption 6. 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 [101]. Due to the space limit, detailed proofs are provided in Section D.1 in the Appendix. The technical proofs follow a similar routine as dlADMM [101]. The difference consists in the fact that the dual variable $u_l$ is controlled by $q_l$ and $z_l$ (Lemma 26 in Section D.1 in the Appendix), which holds under Assumption 6, 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 $\rho$ is sufficiently large.

Lemma 8 (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

$$
L_\rho(p^k, W^k, z^k, q^k, u^k) - L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})
\geq \sum_{l=2}^L (\tau_l^{k+1}/2)\|p_l^{k+1} - p_l^k\|^2_2 + \sum_{l=1}^L (\theta_l^{k+1}/2)\|W_l^{k+1} - W_l^k\|^2_2 + \sum_{l=1}^{L-1} C_1\|z_l^{k+1} - z_l^k\|^2_2
+ (\nu/2)\|z_L^{k+1} - z_L^k\|^2_2 + \sum_{l=1}^{L-1} C_2\|q_l^{k+1} - q_l^k\|^2_2.
$$

(5.2)

$\beta_\rho(p^k, W^k, z^k, q^k, u^k) - \beta_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})$

$$
\geq \sum_{l=1}^L (\theta_l^{k+1}/2)\|W_l^{k+1} - W_l^k\|^2_2 + \sum_{l=1}^{L-1} C_1\|z_l^{k+1} - z_l^k\|^2_2 + (\nu/2)\|z_L^{k+1} - z_L^k\|^2_2
+ \sum_{l=1}^{L-1} C_2\|q_l^{k+1} - q_l^k\|^2_2.
$$

(5.3)

Sketch of Proof: They can be proven via the optimality conditions of all subproblems, and Assumption 6. □

Lemma 9 shows that the objective is bounded from below when $\rho$ is large enough, and all variables are bounded.
5.4. CONVERGENCE ANALYSIS

**Lemma 9** (Bounded Objective). (1). For the pdADMM-G algorithm, if $\rho > \nu$, then $L_\rho(p^k, W^k, z^k, q^k, u^k)$ is lower bounded. Moreover, $p^k, W^k, z^k, q^k,$ and $u^k$ are bounded, i.e. there exist $N_p, N_W, N_z, N_q,$ and $N_u > 0$, such that $\|p^k\| \leq N_p, \|W^k\| \leq N_W, \|z^k\| \leq N_z, \|q^k\| \leq N_q,$ and $\|u^k\| \leq N_u$.

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

**Sketch of Proof:** We only show the sketch proof of (1) because (2) follows the same routine as (1). In order to prove the boundness of $L_\rho$, we should prove the following:

\[
L_\rho(p^k, W^k, z^k, q^k, u^k)
\geq F(p^k, W^k, z^k, q^k) + ((\rho - \nu)/2)\|p^k_{l+1} - q^k\|_2^2
\]

\[> -\infty,
\]

where $p^k_{l+1} = q^k$. Therefore, $F(p^k, W^k, z^k, q^k)$ and $((\rho - \nu)/2)\|p^k_{l+1} - q^k\|_2^2$ are upper bounded by $L_\rho(p^k, W^k, z^k, q^k, u^k)$ and hence $L_\rho(p^0, W^0, z^0, q^0, u^0)$(Lemma 8). The boundness of variables can be obtained via Assumption 6. □

Based on Lemmas 8 and 9, the following theorem ensures that the objective is convergent.

**Theorem 9** (Convergent Objective). (1). For the pdADMM-G algorithm, if $\rho > \max(4\nu S^2, (\sqrt{17} + 1)\nu/2)$, then $L_\rho(p^k, W^k, z^k, q^k, u^k)$ is convergent. Moreover,

\[
\lim_{k \to \infty} \|p^{k+1} - p^k\|_2^2 = 0, \quad \lim_{k \to \infty} \|W^{k+1} - W^k\|_2^2 = 0, \quad \lim_{k \to \infty} \|z^{k+1} - z^k\|_2^2 = 0,
\]

\[
\lim_{k \to \infty} \|q^{k+1} - q^k\|_2^2 = 0, \quad \lim_{k \to \infty} \|u^{k+1} - u^k\|_2^2 = 0.
\]

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

\[
\lim_{k \to \infty} \|z^{k+1} - z^k\|_2^2 = 0, \quad \lim_{k \to \infty} \|q^{k+1} - q^k\|_2^2 = 0, \quad \lim_{k \to \infty} \|u^{k+1} - u^k\|_2^2 = 0.
\]
5.4. CONVERGENCE ANALYSIS

Sketch of Proof: This theorem can be derived by taking the limit on both sides of Inequality (5.2).

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

**Lemma 10 (Bounded Subgradient).** (1). For the pdADMM-G algorithm, there exists a constant $C > 0$ and $g^{k+1} \in \partial L(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})$ such that

\[ \|g^{k+1}\| \leq C(\|p^{k+1} - p^k\| + \|W^{k+1} - W^k\| + \|z^{k+1} - z^k\| + \|q^{k+1} - q^k\| + \|u^{k+1} - u^k\|). \]

(2). For the pdADMM-G-Q algorithm, there exists a constant $\overline{C} > 0$, $g_{W}^{k+1} \in \nabla W^{k+1} \beta_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})$, $g_{z}^{k+1} \in \partial z^{k+1} \beta_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})$, $g_{q}^{k+1} \in \nabla q^{k+1} \beta_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})$, $g_{u}^{k+1} \in \nabla u^{k+1} \beta_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})$ such that

\[ \|g_{W}^{k+1}\| \leq \overline{C}(\|W^{k+1} - W^k\| + \|z^{k+1} - z^k\| + \|q^{k+1} - q^k\| + \|u^{k+1} - u^k\|), \]
\[ \|g_{z}^{k+1}\| \leq \overline{C}(\|W^{k+1} - W^k\| + \|z^{k+1} - z^k\| + \|q^{k+1} - q^k\| + \|u^{k+1} - u^k\|), \]
\[ \|g_{q}^{k+1}\| \leq \overline{C}(\|W^{k+1} - W^k\| + \|z^{k+1} - z^k\| + \|q^{k+1} - q^k\| + \|u^{k+1} - u^k\|), \]
\[ \|g_{u}^{k+1}\| \leq \overline{C}(\|W^{k+1} - W^k\| + \|z^{k+1} - z^k\| + \|q^{k+1} - q^k\| + \|u^{k+1} - u^k\|). \]

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

Now based on Theorem 9, and Lemma 10, the convergence of the pdADMM-G algorithm to a stationary point is presented in the following theorem.

**Theorem 10 (Convergence of the pdADMM-G algorithm).** If $\rho > \max(4\nu S^2, \sqrt{17} + \ldots$
5.4. CONVERGENCE ANALYSIS

1) \( \nu / 2 \), then for the variables \((p, W, z, q, u)\) in Problem 6, starting from any \((p^0, W^0, z^0, q^0, u^0)\), \((p^k, W^k, z^k, q^k, u^k)\) has at least a limit point \((p^*, W^*, z^*, q^*, u^*)\), and any limit point is a stationary point of Problem 6. That is,

\[
0 \in \partial L_\rho(p^*, W^*, z^*, q^*, u^*),
\]

Sketch of Proof: This theorem can be derived directly from Lemma 9 and Lemma 10. □

Theorem 10 shows that our proposed pdADMM-G algorithm converges for sufficiently large \( \rho \), which is consistent with previous literature [101]. Similarly, the convergence of the proposed pdADMM-G-Q algorithm is shown as follows:

Theorem 11 (Convergence of the pdADMM-G-Q algorithm). If \( \rho > \max(4 \nu S^2, (\sqrt{17} + 1) \nu / 2) \), then for the variables \((p, W, z, q, u)\) in Problem 7, starting from any \((p^0, W^0, z^0, q^0, u^0)\), \((p^k, W^k, z^k, q^k, u^k)\) has at least a limit point \((p^*, W^*, z^*, q^*, u^*)\), and any limit point \((W^*, z^*, q^*, u^*)\) is a stationary point of Problem 7. Moreover, if \( \tau^k_{l+1} \) is bounded, then \( p^* \) is a quantized stationary point of Problem 7. That is

\[
p^*_{l+1} = q^*_{l}, \quad \nabla_{p^*} L_\rho(p^*, W^*, z^*, q^*, u^*) = 0, \quad 0 \in \partial_{z^*} L_\rho(p^*, W^*, z^*, q^*, u^*) = 0, \quad 0 \in \partial_{q^*} L_\rho(p^*, W^*, z^*, q^*, u^*),
\]

\[
\nabla_{q^*} \beta_\rho(p^*, W^*, z^*, q^*, u^*) = 0, \quad p^*_{l+1} = \arg \min_{\delta \in \Delta} \| \delta - (p^*_{l} - \nabla_{p^*} F(p^*, W^*, z^*, q^*) / \tau^*_l) \|.
\]

where \( \tau^*_{l} \) is a limit point of \( \tau^k_{l} \).

Sketch of Proof: This theorem is proven using a similar procedure as Theorem 10, and the definition of the quantized stationary point. □

The only difference between Theorems 10 and 11 is that \( p^* \) is a stationary point in Problem 6 and a quantized stationary point in Problem 7. Next, the following
5.5. EXPERIMENTS

Theorem 12 (Convergence Rate). (1). For the pdADMM-G algorithm and a sequence $(p^k, W^k, z^k, q^k, u^k)$, define $c_k = \min_{0 \leq i \leq k}(\sum_{l=2}^{L_i}(\tau_i^{i+1}/2)||p_i^{i+1} - p_i^i||_2^2$ $+ \sum_{l=1}^{L_i-1}(\tau_i^{i+1}/2)||W_i^{i+1} - W_i^i||_2^2 + \sum_{l=1}^{L_i-1} C_1||z_i^{i+1} - z_i^i||_2^2 + (\nu/2)||z_L^{i+1} - z_L^i||_2^2 + \sum_{l=1}^{L_i-1} C_2||q_i^{i+1} - q_i^i||_2^2$) 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 $(W^k, z^k, q^k, u^k)$, define $d_k = \min_{0 \leq i \leq k}(\sum_{l=1}^{L_i}(\theta_i^{i+1}/2)||W_i^{i+1} - W_i^i||_2^2 + \sum_{l=1}^{L_i-1} C_1||z_i^{i+1} - z_i^i||_2^2 + (\nu/2)||z_L^{i+1} - z_L^i||_2^2 + \sum_{l=1}^{L_i-1} C_2||q_i^{i+1} - q_i^i||_2^2$) 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). In order to prove the convergence rate $o(1/k)$, $c_k$ satisfies three conditions: (a) $c_k \geq c_{k+1}$, (b) $\sum_{k=0}^{\infty} c_k$ is bounded, and (c) $c_k \geq 0$.

(2). $d_k$ can be proven using a similar procedure as (1). □

5.5 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® Xeon(R) CPU E5-2686 v4 @ 2.30GHz, and 732GiB of RAM.
## 5.5. EXPERIMENTS

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Node#</th>
<th>Edge#</th>
<th>Class#</th>
<th>Feature#</th>
<th>Training Set#</th>
<th>Validation Set#</th>
<th>Test Set#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2485</td>
<td>10556</td>
<td>7</td>
<td>1433</td>
<td>140</td>
<td>500</td>
<td>1000</td>
</tr>
<tr>
<td>PubMed</td>
<td>19717</td>
<td>88648</td>
<td>3</td>
<td>500</td>
<td>60</td>
<td>500</td>
<td>1000</td>
</tr>
<tr>
<td>Citeseer</td>
<td>2110</td>
<td>9104</td>
<td>6</td>
<td>3703</td>
<td>120</td>
<td>500</td>
<td>1000</td>
</tr>
<tr>
<td>Amazon Computers</td>
<td>13381</td>
<td>491722</td>
<td>10</td>
<td>767</td>
<td>200</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Amazon Photo</td>
<td>7487</td>
<td>238162</td>
<td>8</td>
<td>745</td>
<td>160</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Coauthor CS</td>
<td>18333</td>
<td>163788</td>
<td>15</td>
<td>6805</td>
<td>300</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Coauthor Physics</td>
<td>34493</td>
<td>495924</td>
<td>5</td>
<td>8415</td>
<td>100</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Flickr</td>
<td>89250</td>
<td>899756</td>
<td>7</td>
<td>500</td>
<td>44625</td>
<td>22312</td>
<td>22313</td>
</tr>
<tr>
<td>Ogbn-Arxiv</td>
<td>169343</td>
<td>1166243</td>
<td>40</td>
<td>128</td>
<td>90941</td>
<td>29799</td>
<td>48603</td>
</tr>
</tbody>
</table>

Table 5.2: Statistics of eight benchmark datasets.

### 5.5.1 Datasets and Settings

Nine benchmark datasets were used for experimental evaluation, whose statistics are shown in Table 5.2. Each dataset is split into a training set, a validation set, and a test set. The details are below:

1. Cora [86]. The Cora dataset consists of 2708 scientific publications classified into one of seven classes. The citation network consists of 5429 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1433 unique words.

2. PubMed [86]. PubMed comprises 30M+ citations for biomedical literature that have been collected from sources such as MEDLINE, life science journals, and published online e-books. It also includes links to text content from PubMed Central and other publishers’ websites.

3. Citeseer [86]. The Citeseer dataset was collected from the Tagged.com social network website. It contains 5.6 million users and 858 million links between them. Each user has 4 features and is manually labeled as “spammer” or “not spammer”. Each
5.5. EXPERIMENTS

link represents an action between two users and includes a timestamp and a type. The network contains 7 anonymized types of links. The original task on the dataset is to identify (i.e., classify) the spammer users based on their relational and non-relational features.

4. Amazon Computers and Amazon Photo [68]. Amazon Computers and Amazon Photo are segments of the Amazon co-purchase graph, where nodes represent goods, edges indicate that two goods are frequently bought together, node features are bag-of-words encoded product reviews, and class labels are given by the product category.

5. Coauthor CS and Coauthor Physics [89]. Coauthor CS and Coauthor Physics are co-authorship graphs based on the Microsoft Academic Graph from the KDD Cup 2016 challenge 3. Here, nodes are authors, that are connected by an edge if they co-authored a paper; node features represent paper keywords for each author’s papers, and class labels indicate the most active fields of study for each author.

6. Flickr [119]. In Flickr, one node in the graph represents one image uploaded to Flickr. If two images share some common properties (e.g., same geographic location, same gallery, comments by the same user, etc.), there is an edge between the nodes of these two images. Node features are bag-of-word representation of the images and labels are classes of images.

7. Ogbn-Arxiv [41]. The Ogbn-Arxiv dataset is a directed graph, representing the citation network between all Computer Science (CS) ARXIV papers indexed by MAG. Each node is an ARXIV paper and each directed edge indicates that one paper cites another one. Each paper comes with a 128-dimensional feature vector obtained by averaging the embeddings of words in its title and abstract. In addition, all papers are also associated with the year that the corresponding paper was published.

When it comes to experimental settings, we set $K = 4$ for the multi-hop operator $\Psi$, and defined a diagonal degree matrix $D$ where $D_{ii} = \sum_{j=1}^{V} A_{ij}$, and a renormalized adjacency matrix $\tilde{A} = (D + I)^{-1/2}(A + I)(D + I)^{-1/2} \in \mathbb{R}^{V \times V}$[53]. Moreover, we
5.5. EXPERIMENTS

set $\Psi = \{I, \bar{A}, \bar{A}^2, \bar{A}^3\}$ [15]. 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, $\Delta = \{-1, 0, 1, \cdots, 20\}$ in Problem 7, and $p$ was quantized by default.

5.5.2 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 D.2.1 in the Appendix. The following are their brief introductions:

1. Gradient Descent (GD) [4]. 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) [118]. The Adadelta is proposed to overcome the sensitivity to hyperparameter selection.

3. Adaptive gradient algorithm (Adagrad) [28]. 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) [52]. Adam is the most popular optimization method for deep learning models. It estimates the first and second momentum in order to correct the biased gradient, and thus accelerates empirical convergence.

5.5.3 Convergence

Firstly, in order to validate the convergence of two proposed algorithms, we set up a GA-MLP model with 10 layers, each of which has 1000 neurons. The number of epochs was set to 100. $\nu$ and $\rho$ were set to 0.01 and 1, respectively.

Figure 5.2 demonstrates objectives and residuals of two proposed algorithms on
5.5. EXPERIMENTS

Figure 5.2: Convergence curves of the pdADMM-G algorithm and the pdADMM-G-Q algorithm on four datasets.
5.5. EXPERIMENTS

(a). The speedup on small datasets.

(b). The speedup on large datasets.

Figure 5.3: The speedup of the proposed pdADMM-G on different datasets concerning the number of layers (i.e. a weak-scaling study).

four datasets. Overall, the objectives and residuals of the two proposed algorithms are convergent. From Figure 5.2(a) and Figure 5.2(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 CS dataset are the highest, which still reach near $10^5$ at the 100-th 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 10 and Theorem 11. Specifically, as shown in Figure 5.2(b) and Figure 5.2(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 $10^{-7}$. 
5.5. EXPERIMENTS

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 10 epochs. $\rho$ and $\nu$ were both set to $10^{-3}$. We investigate the speedup concerning two factors: the number of layers (i.e., weak-scaling study) and the number of GPUs (i.e., strong-scaling study).

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 Figure 5.3(a) and Figure 5.3(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

Figure 5.4: The speedup of all methods on two large datasets concerning the number of GPUs (i.e., a strong-scaling study).

5.5.4 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 10 epochs. $\rho$ and $\nu$ were both set to $10^{-3}$. We investigate the speedup concerning two factors: the number of layers (i.e., weak-scaling study) and the number of GPUs (i.e., strong-scaling study).

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 Figure 5.3(a) and Figure 5.3(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
5.5. EXPERIMENTS

that on the 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. Figure 5.4 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 8 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 10 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 10 times.
5.5.5 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 10 layers, each of which consists of 1000 neurons. We set up three quantization cases: no quantization, the quantization concerning $p$ only, and the quantization concerning both $p$ and $q$. For every quantization case, we also set up two different quantization sizes: 8 bits and 16 bits. Figure 5.5 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 $1.4 \times 10^9$ bytes. If the variable $p$ is quantized using 16 bits, the communication overhead drops by 10%, and then using 8 bits saves another 5%. When variables $p$ and $q$ are both quantized, the communication overhead tumbles down to $1.2 \times 10^9$ bytes, which means decreases by 16.7% when it is compared with the case where only $p$ is quantized. When variables are compressed to 8 bits instead of 16 bits, the communication overhead slips further to $1.1 \times 10^9$, 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 Coauthor CS dataset is the largest dataset among the three, and it accomplishes the greatest communication reduction.
5.5.6 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 10 layers but different neurons: the first model has 100 neurons, while the second model has 500 neurons. Following the greedy layerwise training strategy [3], we firstly trained a two-layer GA-MLP model, and then three more layers were added to training, and finally, all 10 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 D.2.1 and D.2.2 in the Appendix, respectively.

Table 5.3 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 compar-
### Table 5.3: Test performance of all methods when the number of neurons is 100.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cora</th>
<th>PubMed</th>
<th>Citeseer</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>0.730±0.022</td>
<td>0.638±0.080</td>
<td>0.637±0.040</td>
</tr>
<tr>
<td>Adadelta</td>
<td>0.671±0.064</td>
<td>0.705±0.038</td>
<td>0.620±0.016</td>
</tr>
<tr>
<td>Adagrad</td>
<td>0.726±0.025</td>
<td>0.753±0.015</td>
<td>0.601±0.037</td>
</tr>
<tr>
<td>Adam</td>
<td>0.725±0.036</td>
<td>0.742±0.007</td>
<td>0.631±0.018</td>
</tr>
<tr>
<td>pdADMM-G</td>
<td>0.784±0.003</td>
<td><strong>0.784±0.004</strong></td>
<td>0.709±0.003</td>
</tr>
<tr>
<td>pdADMM-G-Q</td>
<td><strong>0.788±0.003</strong></td>
<td>0.782±0.003</td>
<td><strong>0.712±0.001</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Amazon Computers</th>
<th>Amazon Photo</th>
<th>Coauthor CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>0.646±0.032</td>
<td>0.735±0.169</td>
<td>0.884±0.010</td>
</tr>
<tr>
<td>Adadelta</td>
<td>0.136±0.060</td>
<td>0.369±0.045</td>
<td>0.787±0.086</td>
</tr>
<tr>
<td>Adagrad</td>
<td>0.688±0.023</td>
<td>0.813±0.018</td>
<td>0.887±0.007</td>
</tr>
<tr>
<td>Adam</td>
<td>0.724±0.010</td>
<td>0.855±0.009</td>
<td>0.883±0.009</td>
</tr>
<tr>
<td>pdADMM-G</td>
<td><strong>0.735±0.006</strong></td>
<td><strong>0.856±0.011</strong></td>
<td><strong>0.915±0.004</strong></td>
</tr>
<tr>
<td>pdADMM-G-Q</td>
<td>0.687±0.054</td>
<td>0.832±0.010</td>
<td>0.914±0.003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Coauthor Physics</th>
<th>Flickr</th>
<th>Ogbn-Arxiv</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>0.909±0.007</td>
<td>0.466±0.007</td>
<td>0.361±0.063</td>
</tr>
<tr>
<td>Adadelta</td>
<td>0.915±0.014</td>
<td>0.461±0.008</td>
<td>0.523±0.030</td>
</tr>
<tr>
<td>Adagrad</td>
<td>0.916±0.012</td>
<td>0.481±0.003</td>
<td>0.567±0.016</td>
</tr>
<tr>
<td>Adam</td>
<td>0.912±0.016</td>
<td>0.512±0.004</td>
<td><strong>0.674±0.006</strong></td>
</tr>
<tr>
<td>pdADMM-G</td>
<td><strong>0.921±0.003</strong></td>
<td><strong>0.513±0.002</strong></td>
<td>0.647±0.002</td>
</tr>
<tr>
<td>pdADMM-G-Q</td>
<td>0.919±0.002</td>
<td>0.507±0.003</td>
<td>0.655±0.002</td>
</tr>
</tbody>
</table>
### 5.5. EXPERIMENTS

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cora</th>
<th>PubMed</th>
<th>Citeseer</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>0.757±0.024</td>
<td>0.699±0.655</td>
<td>0.680±0.014</td>
</tr>
<tr>
<td>Adadelta</td>
<td>0.717±0.063</td>
<td>0.722±0.696</td>
<td>0.564±0.028</td>
</tr>
<tr>
<td>Adagrad</td>
<td>0.776±0.013</td>
<td>0.759±0.761</td>
<td>0.650±0.038</td>
</tr>
<tr>
<td>Adam</td>
<td>0.771±0.020</td>
<td>0.778±0.767</td>
<td>0.662±0.021</td>
</tr>
<tr>
<td>pdADMM-G</td>
<td>0.786±0.005</td>
<td>0.786±0.786</td>
<td>0.713±0.007</td>
</tr>
<tr>
<td>pdADMM-G-Q</td>
<td>0.788±0.787</td>
<td>0.788±0.787</td>
<td>0.712±0.005</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Amazon Computers</th>
<th>Amazon Photo</th>
<th>Coauthor CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>0.707±0.012</td>
<td>0.817±0.005</td>
<td>0.906±0.005</td>
</tr>
<tr>
<td>Adadelta</td>
<td>0.243±0.063</td>
<td>0.380±0.069</td>
<td>0.880±0.011</td>
</tr>
<tr>
<td>Adagrad</td>
<td>0.753±0.009</td>
<td>0.866±0.007</td>
<td>0.911±0.003</td>
</tr>
<tr>
<td>Adam</td>
<td>0.739±0.022</td>
<td>0.880±0.016</td>
<td>0.898±0.013</td>
</tr>
<tr>
<td>pdADMM-G</td>
<td>0.751±0.008</td>
<td>0.873±0.004</td>
<td>0.920±0.002</td>
</tr>
<tr>
<td>pdADMM-G-Q</td>
<td>0.748±0.004</td>
<td>0.865±0.007</td>
<td>0.919±0.003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Coauthor Physics</th>
<th>Flickr</th>
<th>Ogbn-Arxiv</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>0.917±0.004</td>
<td>0.466±0.001</td>
<td>0.436±0.042</td>
</tr>
<tr>
<td>Adadelta</td>
<td>0.917±0.004</td>
<td>0.462±0.001</td>
<td>0.584±0.031</td>
</tr>
<tr>
<td>Adagrad</td>
<td>0.914±0.004</td>
<td>0.487±0.005</td>
<td>0.630±0.016</td>
</tr>
<tr>
<td>Adam</td>
<td>0.914±0.002</td>
<td>0.517±0.002</td>
<td>0.682±0.010</td>
</tr>
<tr>
<td>pdADMM-G</td>
<td>0.918±0.003</td>
<td>0.515±0.002</td>
<td>0.655±0.001</td>
</tr>
<tr>
<td>pdADMM-G-Q</td>
<td>0.918±0.002</td>
<td>0.512±0.003</td>
<td>0.657±0.002</td>
</tr>
</tbody>
</table>

Table 5.4: Test performance of all methods when the number of neurons is 500.

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 5.4 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 5.3, the proposed pdADMM-G algorithm achieves the best on the Amazon Computers
5.6. CONCLUSION

Dataset. However, it is surpassed by Adagrad slightly in Table 5.4. 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.

5.6 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 chapter, 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.
Chapter 6

Conclusion and Future Works

The goal of this dissertation is to develop AM methods to train deep learning models as an alternative to GD. We focus on three research topics: In the topic of AM methods on Multi-Layer Perceptron (MLP), we devise two algorithms: deep learning Alternating Direction Method of Multipliers (dlADMM) and monotonous Deep Learning Alternating Minimization (mDLAM). In the topic of AM methods on Graph Neural Network (GNN), we propose the Invertible Validity-aware Graph Diffusion (IVGD) for graph source localization. In the topic of AM methods for distributed neural network training, we propose three algorithms: parallel deep learning Alternating Direction Method of Multipliers (pdADMM), graph pdADMM (pdADMM-G), and quantized graph pdADMM (pdADMM-G-Q).

For the dlADMM algorithm, the parameters in each layer are updated backward and then forward so that the parameter information in each layer is exchanged efficiently. The time complexity is reduced from cubic to quadratic in (latent) feature dimensions via a dedicated algorithm design for subproblems that enhances them by utilizing iterative quadratic approximations and backtracking. Finally, we provide the first proof of global convergence for an ADMM-based method (dlADMM) in a deep neural network problem under mild conditions. Experiments on benchmark
datasets demonstrated that our proposed dlADMM algorithm outperforms most of the comparison methods. Future work will further extend this method from Multi-Layer Perceptron(MLP) models to other architectures such as Graph Convolutional Network(GCN) models.

For the mDLAM algorithm, our innovative inequality-constrained formulation infinitely approximates the original problem with non-convex equality constraints, enabling our convergence proof of the proposed mDLAM algorithm regardless of the choice of hyperparameters. Our proposed mDLAM algorithm is shown to achieve a fast linear convergence by the Nesterov acceleration technique. Extensive experiments on multiple benchmark datasets demonstrate the convergence, effectiveness, and efficiency of the proposed mDLAM algorithm.

For the IVGD model, we make a graph diffusion model invertible by restricting its Lipschitz constant for the residual GNNs, and thus an approximate estimation of source localization can be obtained by its inversion, and then a compensation module is presented to reduce the introduced errors with skip connection. Moreover, we leverage the unrolled optimization technique to integrate validity constraints into the model, where each layer is encoded by a constrained optimization problem. To combat efficiency and scalability problems, a linearization technique is used to transform problems into solvable ones, which can be efficiently solved by closed-form solutions. Finally, the convergence of the proposed IVGD to a feasible solution is proven theoretically. Extensive experiments on nine datasets show that our proposed IVGD outperforms all comparison methods significantly on five metrics, especially 20% on F1-Score.

For the pdADMM algorithm, we achieve model parallelism training by breaking layer dependency among variables: parameters in each layer of neural networks can be updated independently in parallel. The convergence of the proposed pdADMM to a stationary point is theoretically proven under mild conditions. The convergence
rate of the pdADMM is proven to be $o(1/k)$, where $k$ is the number of iterations. Extensive experiments on six benchmark datasets demonstrated that our proposed pdADMM can lead to more than a 10 times speedup for training large-scale deep neural networks, and outperformed most of the comparison methods.

For the pdADMM-G algorithm and the pdADMM-G-Q algorithm, they are extended from the proposed pdADMM algorithm to train GA-MLP models in parallel: the pdADMM-G algorithm breaks layer dependency similar to the pdADMM algorithm. 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.

6.1 Research Tasks

The major research tasks are described as follows. The current status of these tasks is listed in Table 6.1.

6.1.1 The dlADMM Algorithm

- **Definition of the MLP training problem and problem transformation (A1).** We mathematically formulate the MLP training problem, and then transform the nonlinear constraints of this problem to $\ell_2$ penalties in the objective.
### 6.1. RESEARCH TASKS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Research Work A</strong></td>
<td>The dLADMM Algorithm</td>
<td>Completed</td>
</tr>
<tr>
<td>A1</td>
<td>Problem definition and transformation</td>
<td>Completed</td>
</tr>
<tr>
<td>A2</td>
<td>Proposal of the dLADMM algorithm</td>
<td>Completed</td>
</tr>
<tr>
<td>A3</td>
<td>Discussion of solutions to all subproblems</td>
<td>Completed</td>
</tr>
<tr>
<td>A4</td>
<td>Convergence proof of the proposed dLADMM algorithm</td>
<td>Completed</td>
</tr>
<tr>
<td>A5</td>
<td>Validation on the benchmark datasets</td>
<td>Completed</td>
</tr>
<tr>
<td><strong>Research Work B</strong></td>
<td>The mDLAM Algorithm</td>
<td>Completed</td>
</tr>
<tr>
<td>B1</td>
<td>Problem transformation and relaxation</td>
<td>Completed</td>
</tr>
<tr>
<td>B2</td>
<td>Proposal of the mDLAM algorithm</td>
<td>Completed</td>
</tr>
<tr>
<td>B3</td>
<td>Convergence proof of the proposed DLAM algorithm</td>
<td>Completed</td>
</tr>
<tr>
<td>B4</td>
<td>Validation on the benchmark datasets</td>
<td>Completed</td>
</tr>
<tr>
<td><strong>Research Work C</strong></td>
<td>The pdADMM Algorithm</td>
<td>Completed</td>
</tr>
<tr>
<td>C1</td>
<td>Problem definition and transformation</td>
<td>Completed</td>
</tr>
<tr>
<td>C2</td>
<td>Proposal of the pdADMM algorithm</td>
<td>Completed</td>
</tr>
<tr>
<td>C3</td>
<td>Convergence proof of the proposed pdADMM algorithm</td>
<td>Completed</td>
</tr>
<tr>
<td>C4</td>
<td>Validation on the benchmark datasets</td>
<td>Completed</td>
</tr>
<tr>
<td><strong>Research Work D</strong></td>
<td>The pdADMM-G-Q Algorithm and the pdADMM-G-Q Algorithm</td>
<td>Completed</td>
</tr>
<tr>
<td>D1</td>
<td>Problem definition and transformation</td>
<td>Completed</td>
</tr>
<tr>
<td>D2</td>
<td>Proposal of the pdADMM-G algorithm and the pdADMM-G-Q algorithm</td>
<td>Completed</td>
</tr>
<tr>
<td>D3</td>
<td>Convergence proofs of two proposed algorithms</td>
<td>Completed</td>
</tr>
<tr>
<td>D4</td>
<td>Validation on the benchmark datasets</td>
<td>Completed</td>
</tr>
<tr>
<td><strong>Research Work E</strong></td>
<td>The IVGD Model</td>
<td>Completed</td>
</tr>
<tr>
<td>E1</td>
<td>Definition of the graph source localization problem</td>
<td>Completed</td>
</tr>
<tr>
<td>E2</td>
<td>Proposal of invertible graph residual network</td>
<td>Completed</td>
</tr>
<tr>
<td>E3</td>
<td>Proposal of graph validity-aware layers</td>
<td>Completed</td>
</tr>
<tr>
<td>E4</td>
<td>Convergence proof of the proposed IVGD model</td>
<td>Completed</td>
</tr>
<tr>
<td>E5</td>
<td>Validation on the real-world datasets</td>
<td>Completed</td>
</tr>
<tr>
<td>F</td>
<td>Dissertation writing and revision</td>
<td>Completed</td>
</tr>
</tbody>
</table>

*Table 6.1: Research tasks and status*
• **Proposal of the dlADMM algorithm (A2).** We present a novel and efficient dlADMM algorithm to handle the MLP training problem. The proposed dlADMM updates parameters in a backward-forward fashion, in order to accelerate the convergence process.

• **Discussion of solutions to all subproblems (A3).** We discuss solutions to all subproblems generated by the proposed dlADMM algorithm. For some subproblems, we propose quadratic approximation and backtracking techniques to avoid matrix inversion as well as reduce computational costs for large-scale datasets. The time complexity of subproblems in the proposed dlADMM is reduced from $O(n^3)$ to $O(n^2)$.

• **Convergence proof of the proposed dlADMM algorithm (A4).** We investigate several convergence properties of the proposed dlADMM. The convergence assumptions are very mild to ensure that most deep learning applications satisfy our assumptions. The proposed dlADMM is guaranteed to converge to a stationary point sublinearly whatever the initialization is when the hyperparameter is sufficiently large.

• **Validation on the benchmark datasets (A5).** We conduct experiments on several benchmark datasets to validate our proposed dlADMM algorithm. Experimental results show that the proposed dlADMM algorithm performs better than most GD-based optimizers. Moreover, we also demonstrate the convergence and the efficiency of the proposed dlADMM algorithm.

### 6.1.2 The mDLAM Algorithm

• **Definition of the MLP training problem and problem relaxation (B1).** We transform the original MLP training problem into an inequality-constrained problem that can infinitely approximate the original one. Applying this inno-
6.1. RESEARCH TASKS

- **Proposal of the mDLAM algorithm**(B2). We present an efficient mDLAM algorithm. The Nesterov acceleration technique is applied to further boost convergence. The quadratic approximation technique is utilized to avoid matrix inversion, and all subproblems have closed-form solutions.

- **Convergence proof of the proposed DLAM algorithm**(B3). We investigate several convergence properties of the proposed mDLAM algorithm under mild conditions. It is guaranteed to converge to a stationary point with linear convergence whatever hyperparameters are initialized. The convergence rate and the hyperparameter conditions of the proposed mDLAM algorithm are better than those of the previously proposed dIADMM algorithm.

- **Validation on the benchmark datasets**(B4). Extensive experiments on four benchmark datasets have been conducted to demonstrate the convergence, effectiveness, and scalability of the proposed mDLAM algorithm. It achieves a linear convergence as expected, and outperforms consistently state-of-the-art optimizers.

### 6.1.3 The pdADMM Algorithm

- **Definition of the MLP training problem and problem transformation**(C1). We propose a novel transformation of the MLP training problem by breaking the layer dependency, which splits a neural network into independent layer partitions and allows for the ADMM to achieve parallel training.

- **Proposal of the pdADMM algorithm**(C2). We propose a novel pdADMM algorithm to train the deep MLP model in parallel. All parameters in each layer
can be updated independently. All subproblems generated by the proposed pdADMM algorithm have closed-form solutions.

- **Convergence proof of the proposed pdADMM algorithm (C3).** Similar to the previously proposed dlADMM algorithm, we investigate the convergence properties of the proposed pdADMM algorithm under convergence assumptions, which are milder than these of the previously proposed dlADMM algorithm. We prove that it converges to a stationary point with sublinear convergence when the hyperparameters are sufficiently large.

- **Validation on the benchmark datasets (C4).** We conduct extensive experiments on six benchmark datasets to show the massive speedup of the proposed pdADMM as well as its competitive performance with state-of-the-art optimizers. The speedup of the proposed pdADMM algorithm increases linearly with the number of layers.

### 6.1.4 The pdADMM-G and pdADMM-G-Q Algorithms

- **Definition of the GA-MLP training problem and problem transformation (D1).** We mathematically formulate the GA-MLP training problem, and then break layer dependency to achieve parallel training, which is similar to the previously proposed pdADMM algorithm.

- **Proposal of the pdADMM-G algorithm and the pdADMM-G-Q algorithm (D2).** We extend the previously proposed pdADMM algorithm to train the GA-MLP model, named the pdADMM-G algorithm. The extended pdADMM-G-Q algorithm reduces communication costs by introducing the quantization technique.

- **Convergence proofs of two proposed algorithms (D3).** We provide the theoretical convergence guarantees of two proposed algorithms. Specifically, we
investigate their several convergence properties and prove that they converge to a (quantized) stationary point of GA-MLP models sublinearly when the hyperparameters are sufficiently large.

- **Validation on the benchmark datasets (D4).** We conduct extensive experiments on nine benchmark datasets to show the convergence, the massive speedup of two proposed algorithms, as well as their outstanding performance. Moreover, the proposed pdADMM-G-Q algorithm reduces communication overheads significantly by up to 45%.

### 6.1.5 The IVGD Model

- **Definition of the graph source localization problem (E1).** We define the graph source localization problem where the prior knowledge of the graph diffusion model is available. Our goal is to design a graph source localization model constrained by graph validity constraints via utilizing such prior knowledge.

- **Proposal of invertible graph residual network (E2).** We propose a new graph residual network with Lipschitz regularization to ensure the invertibility of graph diffusion models. Furthermore, we propose an error compensation mechanism to offset the errors inferred from the graph residual network.

- **Proposal of graph validity-aware layers (E3).** We ensure the validity of inferred sources by automatically learning validity-aware layers. We further accelerate the optimization of the proposed layers by leveraging a linearization technique. It transforms nonconvex problems into convex problems, which have closed-form solutions.

- **Convergence proof of the proposed IVGD model (E4).** We prove the convergence of the proposed IVGD model for linear validity constraints. It con-
• **Validation on the real-world datasets (E5).** Extensive experiments on nine datasets have been conducted to demonstrate the effectiveness and robustness of our proposed IVGD. Our proposed IVGD outperforms all comparison methods significantly on five metrics.

### 6.2 Discussion

Finally, we summarize some recent training methods in parallel to our publications. These training methods can be either serial training methods or parallel training methods. Serial training methods are classified into two categories: 1). Stochastic Approximation (SA). This type of method trains a neural network in a stochastic manner. For example, Newman et al. proposed an algorithm called slimTrain, which automatically adapted regularization parameters for linear weights [74]. 2). Sample Average Approximation (SAA). SAA methods update the parameters of neural networks in a full-batch fashion, where our proposed AM methods fall. As an example, Newman et al. proposed the GNvpro by extending the idea of variable projection in the nonlinear least-square problem to train Deep Neural Network (DNN) training [73]. The main difference between our proposed AM methods and theirs consists in the separability of neural networks: slimTrain and GNvpro split a neural network into a nonlinear feature extractor followed by an affine mapping, while our proposed AM methods have a finer granularity of decomposition: they decompose a neural network into layerwise training modules. Such difference leads to several advantages: firstly, our proposed AM methods are proven to converge for non-differentiable activations (e.g. ReLU) [101, 106], while no such convergence guarantees are provided for slimTrain and GNvpro; secondly, they can avoid gradient vanishing/exploding prob-
lems for any neural network architecture [93, 92], whereas slimTrain and GNvpro require delicate designs of neural networks (e.g. hyperbolic neural networks); thirdly, these training modules usually have analytic solutions and hence facilitate efficient training, and our proposed AM methods converge to modest accuracy within tens of iterations, which are sufficient for real-world applications [6].

As for parallel training methods, they can be categorized into data parallelism and model parallelism. We have summarized their recent development in Chapters 4 and 5. One important related work is an MGRIT-based method proposed by Gunther et al. [32], as their method is also layerwise-based. Specifically, they substitute a parallel nonlinear multigrid iteration applied to the layer domain for the backpropagation algorithm. While their mechanism is different from our proposed AM methods, they share a common characteristic: they both forsake model precision for speedup: the MGRIT-based method employs a relaxation scheme to approximate the true solution on the fine grid; our proposed AM methods relax original training problems via imposing constraints as penalties on the objectives. One potential future direction is to combine their idea with ours to accelerate model training further.

It is important to compare our proposed AM methods with Block Coordinate Descent (BCD) methods. This is because they are important components of SAA methods and share some commonalities with our AM methods, which have been investigated recently to train neural networks [19, 120, 124]. Please refer to Chapter 3 for recent development. While they have achieved excellent performance on many datasets, their usage is restricted by the challenges of the ill-conditioning and coupled variables [73, 74]. For our proposed AM methods in this dissertation, they solve subproblems approximately: we replace each subproblem with a quadratic approximation, and optimal learning parameters are selected via iterative backtracking, and therefore, we can choose accurate directions toward solutions.
6.3 Current Publications

6.3.1 Contributions of Published Papers Contributing to dissertation

The following is a list of published papers that contribute to the dissertation and author contributions:

1. Junxiang Wang, Fuxun Yu, Xiang Chen, and Liang Zhao. ADMM for Efficient Deep Learning with Global Convergence. in Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining(KDD 2019), research track(acceptance rate: 14.2%), Alaska, USA, Aug 2019. Junxiang Wang proposed the dlADMM algorithm, proved the convergence of the dlADMM algorithm, led the experimental design and analysis and drafted the manuscript. Fuxun Yu participated in the experimental design, and Xiang Chen and Liang Zhao provided support and manuscript editing.

2. Junxiang Wang, Hongyi Li, and Liang Zhao. Accelerated Gradient-free Neural Network Training by Multi-convex Alternating Optimization. Neurocomputing,(impact factor: 5.719), 2022. Junxiang Wang proposed the mDLAM algorithm, proved the convergence of the mDLAM algorithm, led the experimental design and analysis and drafted the manuscript. Hongyi Li participated in the experimental design, and Liang Zhao provided support and manuscript editing.

3. Junxiang Wang, Zheng Chai, Yue Cheng, Liang Zhao. Toward Model Parallelism for Deep Neural Network based on Gradient-free ADMM Framework. in Proceedings of the IEEE International Conference on Data Mining(ICDM 2020), regular paper(acceptance rate: 9.8%), Sorrento, Italy, Nov 2020. Junxiang Wang proposed the pdADMM algorithm, proved the convergence of
the pdADMM algorithm, led the experimental design and analysis and drafted the manuscript. Zheng Chai participated in the experimental design, and Yue Cheng and Liang Zhao provided support and manuscript editing.


Junxiang Wang proposed the pdADMM-G algorithm and pdADMM-G-Q algorithm, proved the convergence of the two proposed algorithms, participated in the experimental design and drafted the manuscript. Hongyi Li led the experimental design and analysis. Zheng Chai participated in the experimental design. Yongchao Wang, Yue Cheng, and Liang Zhao provided support and manuscript editing.


Junxiang Wang proposed the IVGD framework, proved the convergence of the proposed IVGD, led the experimental design and analysis and drafted the manuscript. Junji Jiang participated in the experimental design. Liang Zhao provided support and manuscript editing.

6.3.2 Published Papers During My Ph.D.

Conference Papers

6.3. CURRENT PUBLICATIONS

International Conference on Data Mining (ICDM 2022), short paper (acceptance rate: 20%), Orlando, FL, USA, Nov 2022.

2. Chen Ling, Junji Jiang, Junxiang Wang, and Liang Zhao. SL-VAE: Variational Autoencoder for Source Localization. in Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2022), research track (acceptance rate: 15.0%), Washington D.C, USA, Aug 2022.


6.3. CURRENT PUBLICATIONS

Seattle, USA, Dec 2018.


Journal Papers


3. Junxiang Wang and Liang Zhao. Nonconvex Generalization of Alternating Di-
6.3. CURRENT PUBLICATIONS


6.3.3 Papers Published Before Ph.D.


6.3.4 Submitted and In-preparation Papers


2. Junxiang Wang, Junji Jiang, and Liang Zhao. An Invertible Bi-Lipschitz Surrogate Model for Black-box Graph Inverse Problems. NeurIPS 2022(Submission)

3. Hongyi Li, Junxiang Wang, Yongchao Wang, Yue Cheng, and Liang Zhao. Community-based Layerwise Distributed Training of Graph Convolutional Networks. IEEE Transactions on Neural Networks and Learning Systems(TNNLS) (Submission)

4. Chen Ling, Junji Jiang, Junxiang Wang, Renhao Xue, and Zhao Liang. DeepIM: Deep Graph Representation Learning and Optimization for Influence Maximization. WSDM 2023(In Preparation)

5. Junxiang Wang, Hongyi Li, Liang Zhao. A Convergent ADMM Framework for Efficient Neural Network Training. IEEE Transactions on Knowledge and Data Engineering(TKDE)(In Preparation)

6. Yuyang Gao, Junxiang Wang, Wei Wang, Xin Deng, Hamed Zamani, Xiaohan Yan, Yan Guo, Ahmed Awadallah, Yanfang Ye, and Liang Zhao. Asynchronous Semi-supervised Representation Learning for Email Heterogeneous Networks. CIKM 2022(Submission)
6.4 Future Research Directions

6.4.1 Parallel Training of Graph Neural Networks on Large-Scale Graphs

The GCN is one of the most popular graph learning models: it demonstrates the ability to express nodes and graphs effectively via the message-passing mechanism. Despite its effectiveness, it is challenging to train large-scale GCN models. This is because the space complexity and the time complexity to handle graphs are quadratic with respect to their nodes. Therefore, a graph with millions or billions of nodes may not fit in memory. Potential future work is to devise AM algorithms to address parallel training on large-scale graphs.

6.4.2 Stochastic AM Algorithms for Large-Scale Datasets

On one hand, the full-batch GD is popular in the small machine learning problems, they suffer from memory issues on the large-scale datasets, which is also potentially a challenge of SSA methods, including our proposed AM methods. On the other hand, however, stochastic algorithms such as Stochastic Gradient Descent (SGD) are more prevalent for large-scale datasets [5]. This motivates us to extend our proposed AM methods to stochastic formats by some variance reduction techniques such as SVRG [50] and SAGA [24]. We will investigate their theoretical convergence rates as well as verify their performance on large-scale datasets.
Appendix A

Appendix of the dlADMM Algorithm

A.1 Algorithms to Update $W_{l}^{k+1}$ and $a_{l}^{k+1}$

The algorithms to update $W_{l}^{k+1}$ and $a_{l}^{k+1}$ are described in the Algorithms 7 and 8, respectively.

---

**Algorithm 7:** The Backtracking Algorithm to Update $W_{l}^{k+1}$

| Require: $W_{l-1}^{k+1}$, $z_{i-1}^{k+1}$, $a_{i-1}^{k+1}$, $u^{k}$, $\rho$, some constant $\gamma > 1$. |
| Ensure: $\theta_{l}^{k+1}, W_{l}^{k+1}$. |
| 1: Pick up $\alpha$ and $\zeta = \overline{W_{l}}^{k+1} - \nabla_{W_{l}^{k+1}} \phi/\alpha$. |
| 2: while $\phi(\{W_{i}^{k+1}\}_{i=1}^{l-1}, \zeta, \{W_{i}^{k+1}\}_{i=l+1}^{L}, z_{i}^{k+1}, a_{i}^{k+1}, u^{k}) > P_{l}(\zeta; \alpha)$ do |
| 3: $\alpha \leftarrow \alpha \gamma$. |
| 4: Solve $\zeta$ by Equation (2.9). |
| 5: end while |
| 6: Output $\theta_{l}^{k+1} \leftarrow \alpha$. |
| 7: Output $W_{l}^{k+1} \leftarrow \zeta$. |

---

A.2 Preliminary Lemmas

The following several lemmas are preliminary results.
Algorithm 8: The Backtracking Algorithm to Update $a^{k+1}_i$

Require: $W^{k+1}_i$, $z^{k+1}_i$, $a^{k+1}_{i-1}$, $u^k$, $\rho$, some constant $\eta > 1$.
Ensure: $t^{k+1}_i$, $a^{k+1}_i$.

1: Pick up $t$ and $\beta = \frac{a^{k+1}_i}{t} - \nabla a^{k+1}_i \phi / t$.
2: while $\phi(W^{k+1}_{i+1}, z^{k+1}_{i+1}, \{a^{k+1}_{i-1}\}_{i=1}^{L-1}, \beta, \{a^{k+1}_{i-1}\}_{i=i+1}^{L-1}, u^k) > Q_l(\beta; t)$ do
3: $t \leftarrow t^\eta$.
4: $\beta \leftarrow \frac{a^{k+1}_i}{t} - \nabla a^{k+1}_i \phi / t$.
5: end while
6: Output $t^{k+1}_i \leftarrow t$.
7: Output $a^{k+1}_i \leftarrow \beta$.

Lemma 11. $\nabla z^k_L R(z^k_L; y) + u^k = 0$ for all $k \in \mathbb{N}$.

Proof. The optimality condition of $z^k_L$ in Equation (2.11) gives rise to

$$\nabla z^k_L R(z^k_L; y) + \rho (z^k_L - W^k_L a^k_{L-1}) + u^{k-1} = 0$$

Because $u^k = u^{k-1} + \rho (z^k_L - W^k_L a^k_{L-1})$, then we have $\nabla z^k_L R(z^k_L; y) + u^k = 0$. \hfill \Box

Lemma 12. It holds that $\forall z_{L,1}, z_{L,2} \in \mathbb{R}^{n_L}$,

$$R(z_{L,1}; y) \leq R(z_{L,2}; y) + \nabla z_{L,2} R^T(z_{L,2}; y)(z_{L,1} - z_{L,2}) + (H/2)\|z_{L,1} - z_{L,2}\|^2$$

$$- R(z_{L,1}; y) \leq -R(z_{L,2}; y) - \nabla z_{L,2} R^T(z_{L,2}; y)(z_{L,1} - z_{L,2}) + (H/2)\|z_{L,1} - z_{L,2}\|^2$$

Proof. Because $R(z_L; y)$ is Lipschitz differentiable by Assumption 2, so is $-R(z_L; y)$. Therefore, this lemma is proven exactly as same as Lemma 2.1 in [2]. \hfill \Box
Lemma 13. It holds that for $\forall k \in \mathbb{N}$,

$$L_\rho(\overline{W}^{k+1}_{l+1}, \overline{z}^{k+1}_{l+1}, \overline{a}^{k+1}_{l+1}, u^k) - L_\rho(\overline{W}^{k+1}_l, \overline{z}^{k+1}_l, \overline{a}^{k+1}_l, u^k) \geq \|\tau^{k+1}_l \circ (\overline{a}^{k+1}_l - a^k_l)^{o2}\|_2/2 (l = 1, \ldots, L - 1)$$

(A.1)

$$L_\rho(\overline{W}^{k+1}_{l+1}, \overline{z}^{k+1}_{l+1}, \overline{a}^{k+1}_{l+1}, u^k) \geq L_\rho(\overline{W}^{k+1}_l, \overline{z}^{k+1}_l, \overline{a}^{k+1}_l, u^k) (l = 1, \ldots, L - 1)$$

(A.2)

$$L_\rho(W^k, z^k, a^k, u^k) - L_\rho(W^k, \overline{z}^{k+1}_L, a^k, u^k) \geq (\rho/2)\|\overline{z}^{k+1}_L - z^k_L\|_2^2$$

(A.3)

Proof. Essentially, all inequalities can be obtained by applying optimality conditions of updating $\overline{a}^{k+1}_l$, $\overline{z}^{k+1}_l$, $\overline{W}^{k+1}_l$, $W^{k+1}_l$, $\overline{z}^{k+1}_l$ and $a^{k+1}_l$, respectively. We only prove Inequality (A.3), (A.5), and (A.6). This is because Inequalities (A.1), (A.4) and (A.7) follow the routine of Inequality (A.5), and Inequality (A.2) follows the routine of Inequality (A.6).

Firstly, we focus on Inequality (A.3).

$$L_\rho(W^k, z^k, a^k, u^k) - L_\rho(W^k, \overline{z}^{k+1}_L, a^k, u^k)$$

$$= R(z^k_L; y) + (u^k)^T (z^k_L - W^k_L a^k_{l-1}) + (\rho/2)\|z^k_L - W^k_L a^k_{l-1}\|_2^2$$

$$- R(\overline{z}^{k+1}_L; y) - (u^k)^T (\overline{z}^{k+1}_L - W^k_L a^k_{l-1}) - (\rho/2)\|\overline{z}^{k+1}_L - W^k_L a^k_{l-1}\|_2^2$$

$$= R(z^k_L; y) - R(\overline{z}^{k+1}_L; y) + (u^k)^T (z^k_L - \overline{z}^{k+1}_L) + (\rho/2)\|\overline{z}^{k+1}_L - z^k_L\|_2^2$$

$$+ \rho(\overline{z}^{k+1}_L - W^k_L a^k_{l-1})^T (z^k_L - \overline{z}^{k+1}_L)$$

(A.8)
where the second equality follows from the cosine rule $\|z_L^k - W_L^k a_{L-1}^k\|^2 = \|\tilde{z}_L^{k+1} - z_L^k\|^2 + (\tilde{z}_L^{k+1} - W_L^k a_{L-1}^k)^T (z_L^k - \tilde{z}_L^{k+1})$.

According to the optimality condition of Equation (2.5), we have $\nabla_{\tilde{z}_L^{k+1}} R(z_L^k; y) + u^k + \rho(z_L^k - W_L^k a_{L-1}^k) = 0$. Because $R(z_L; y)$ is convex and differentiable with regard to $z_L$, its subgradient is also its gradient. According to the definition of subgradient, we have

$$R(z_L^k; y) \geq R(\tilde{z}_L^{k+1}; y) + \nabla_{\tilde{z}_L^{k+1}} R^T (\tilde{z}_L^{k+1}; y) (z_L^k - \tilde{z}_L^{k+1}) = R(\tilde{z}_L^{k+1}; y) - (u^k + \rho(z_L^k - W_L^k a_{L-1}^k))^T (z_L^k - \tilde{z}_L^{k+1}) \quad (A.9)$$

We introduce Equation (A.9) into Equation (A.8) to obtain Equation (A.3).

Secondly, we focus on Inequality (A.5). The choice of $\theta_t^{k+1}$ in Algorithm 7 shows that

$$\phi(W_t^{k+1}; z_{t-1}^{k+1}, a_{t-1}^{k+1}, u^k) \leq P_t(W_t^{k+1}; \theta_t^{k+1}). \quad (A.10)$$

Moreover, the optimality condition of $W_t^{k+1}$ shows that

$$(\nabla_{W_t^{k+1}} \phi)^T (W_t^{k+1}; z_{t-1}^{k+1}, a_{t-1}^{k+1}, u^k) + \theta_t^{k+1} \circ (W_t - W_t^k) = 0 \quad (A.11)$$

Therefore, we have

$$L_\rho(W_t^{k+1}; z_{t-1}^{k+1}, a_{t-1}^{k+1}, u^k) - L_\rho(W_t^{k+1}; z_{t-1}^{k+1}, a_{t-1}^{k+1}, u^k)$$

$$= \phi(W_t^{k+1}; z_{t-1}^{k+1}, a_{t-1}^{k+1}, u^k) - \phi(W_t^{k+1}; z_{t-1}^{k+1}, a_{t-1}^{k+1}, u^k) \quad \text{(Definition of } L_\rho)$$

$$\geq \phi(W_t^{k+1}; z_{t-1}^{k+1}, a_{t-1}^{k+1}, u^k) - P_t(W_t^{k+1}; \theta_t^{k+1}) \quad \text{(Equation (A.10))}$$

$$= -\nabla_{W_t^{k+1}} \phi^T (W_t^{k+1} - W_t^{k+1}) - \|\theta_t^{k+1} \circ (W_t^{k+1} - W_t^{k+1})\|_2/2$$

$$= \|\theta_t^{k+1} \circ (W_t^{k+1} - W_t^{k+1})\|_2/2 \quad \text{(Equation (A.11))}.$$
Finally, we focus on Equation (A.6). This follows directly from the optimality of $z_l^{k+1}$ in Equation (2.10).

Lemma 14. If $\rho > \frac{\sqrt{T} \pi^{k+1}}{2} H$ so that $C_1 = \rho/2 - H/2 - 2H^2/\rho > 0$, then it holds that

$$
L_\rho(W^{k+1}, z_{L-1}^{k+1}, a^{k+1}, u^{k}) - L_\rho(W^{k+1}, z^{k+1}, a^{k+1}, u^{k+1}) \\
\geq C_1 \|z_L^{k+1} - z_{L}^{k+1}\|^2_2 - (2H^2/\rho)\|z_L^{k+1} - z_{L}^{k}\|^2_2.
$$

(A.12)

Proof.

$$
L_\rho(W^{k+1}, z_{L-1}^{k+1}, a^{k+1}, u^{k}) - L_\rho(W^{k+1}, z^{k+1}, a^{k+1}, u^{k+1}) \\
= R(z_L^{k+1}; y) - R(z_L^{k+1}; y) + (u^{k+1})^T(z_L^{k+1} - z_{L}^{k+1}) + (\rho/2)\|z_L^{k+1} - z_{L}^{k+1}\|^2_2 \\
- (1/\rho)\|u^{k+1} - u^{k}\|^2_2 \\
= R(z_L^{k+1}; y) - R(z_L^{k+1}; y) + \nabla z_L^{k+1}R(z_L^{k+1}; y)^T(z_L^{k+1} - z_{L}^{k+1}) + (\rho/2)\|z_L^{k+1} - z_{L}^{k+1}\|^2_2 \\
- (1/\rho)\|u^{k+1} - u^{k}\|^2_2 (\text{Lemma 11}) \\
\geq (-H/2)\|z_L^{k+1} - z_{L}^{k+1}\|^2_2 + (\rho/2)\|z_L^{k+1} - z_{L}^{k+1}\|^2_2 - (1/\rho)\|\nabla z_{L}^{k+1}R(z_L^{k+1}; y) \\
- \nabla z_{L}^{k}R(z_L^{k}; y)\|^2_2 (R(z_L^{k}; y) \text{ is Lipschitz differentiable, Lemmas 11 and 12}) \\
\geq (-H/2)\|z_L^{k+1} - z_{L}^{k+1}\|^2_2 + (\rho/2)\|z_L^{k+1} - z_{L}^{k+1}\|^2_2 - (H^2/\rho)\|z_L^{k+1} - z_{L}^{k}\|^2_2 \\
(\text{Assumption 2}) \\
\geq (-H/2)\|z_L^{k+1} - z_{L}^{k+1}\|^2_2 + (\rho/2)\|z_L^{k+1} - z_{L}^{k+1}\|^2_2 - (2H^2/\rho)\|z_L^{k+1} - z_{L}^{k+1}\|^2_2 \\
- (2H^2/\rho)\|z_L^{k+1} - z_{L}^{k}\|^2_2 (\text{Mean Inequality}) \\
= C_1 \|z_L^{k+1} - z_{L}^{k+1}\|^2_2 - (2H^2/\rho)\|z_L^{k+1} - z_{L}^{k}\|^2_2.
$$

We choose $\rho > \frac{\sqrt{T} \pi^{k+1}}{2} H$ to make $C_1 > 0$. □
A.3 Proof of Theorem 1

Proving Theorem 1 is equal to proving jointly Theorem 13, 14, and 15, which are elaborated in the following.

**Theorem 13.** Given that Assumptions 1 and 2 hold, the dlADMM satisfies Property 1.

**Proof.** There exists $z'_L$ such that $z'_L - W_L^k a_{L-1}^k \equiv 0$. By Assumption 2, we have

$$F(W^k, \{z^k_l\}_{l=1}^{L-1}, z'_L, a^k) \geq \min_S S > -\infty$$

where $S = \{F(W, z, a) : z_L - W_L a_{L-1} = 0\}$. Then we have

$$L_\rho(W^k, z^k, a^k, u^k) = F(W^k, z^k, a^k) + (u^k)^T(z^k_L - W_L^k a^k_{L-1}) + (\rho/2)\|z^k_L - W_L^k a^k_{L-1}\|^2$$

$$= F(W^k, z^k, a^k) + (u^k)^T(z^k_L - z'_L) + (\rho/2)\|z^k_L - W_L^k a^k_{L-1}\|^2$$

$$= F(W^k, z^k, a^k) + \nabla z_L^k R^T(z_L^k; y)(z'_L - z^k_L) + (\rho/2)\|z^k_L - W_L^k a^k_{L-1}\|^2$$

$$(z'_L - W_L^k a^k_{L-1} = 0)$$

$$= F(W^k, z^k, a^k) + \nabla z_L^k R^T(z_L^k; y)(z_L^k - z_L^k) + (\rho/2)\|z^k_L - W_L^k a^k_{L-1}\|^2$$

$$(\text{Lemma 11})$$

$$= (\nu/2) \sum_{l=1}^{L-1}(\|z_l^k - W_l^k a_{l-1}^k\|^2 + \|a_l^k - f(z_l^k)\|^2) + R(z_L^k; y)$$

$$+ \nabla z_L^k R^T(z_L^k; y)(z_L^k - z_L^k) + (\rho/2)\|z^k_L - W_L^k a^k_{L-1}\|^2$$

$$(\text{The definition of } F)$$

$$\geq (\nu/2) \sum_{l=1}^{L-1}(\|z_l^k - W_l^k a_{l-1}^k\|^2 + \|a_l^k - f(z_l^k)\|^2) + R(z_L^k; y)$$

$$+ (\rho - H/2)\|z_L^k - W_L^k a^k_{L-1}\|^2$$

$$(\text{Lemmas 11 and 12, } R(z_L; y) \text{ is Lipschitz differentiable})$$

$$> -\infty$$

It concludes from Lemma 13 and Lemma 14 that $L_\rho(W^k, z^k, a^k, u^k)$ is upper bounded by $L_\rho(W^0, z^0, a^0, u^0)$ and so are $(\nu/2) \sum_{l=1}^{L-1}(\|z_l^k - W_l^k a_{l-1}^k\|^2 + \|a_l^k - f(z_l^k)\|^2)$ and
A.3. PROOF OF THEOREM 1

\[ \| z_L^k - W_L^k a_{L-1} \|^2. \]

By Assumption 2, \((W^k, z^k, a^k)\) is bounded. By Lemma 11, it is obvious that \(u^k\) is bounded as well. \(\square\)

**Theorem 14.** Given that Assumptions 1 and 2 hold, the dlADMM satisfies Property 2.

**Proof.** This follows directly from Lemma 13 and Lemma 14. \(\square\)

**Theorem 15.** Given that Assumptions 1 and 2 hold, the dlADMM satisfies Property 3.

**Proof.** We know that

\[ \partial L_\rho(W^{k+1}, z^{k+1}, a^{k+1}, u^{k+1}) = (\partial W^{k+1}_L, \partial z^{k+1}_L, \nabla a^{k+1}_L, \nabla u^{k+1}_L) \]

where \(\partial W^{k+1}_L = \{\partial W^{k+1}_{L-1}\}_l^{L-1}, \partial z^{k+1}_L = \{\partial z^{k+1}_{L-1}\}_l^{L-1}, \) and \(\nabla a^{k+1}_L = \{\nabla a^{k+1}_{L-1}\}_l^{L-1}. \)

To prove Property 3, we need to give an upper bound of \(\partial W^{k+1}_L, \partial z^{k+1}_L, \nabla a^{k+1}_L\) and \(\nabla u^{k+1}_L\) by a linear combination of \(\|W^{k+1} - W^{k+1}\|, \|z^{k+1} - z^{k+1}\|, \|a^{k+1} - a^{k+1}\|\) and \(\|z^{k+1}_l - z^{k+1}_l\|\).
A.3. PROOF OF THEOREM 1

For $W_i^{k+1}(l < L)$,

$$\partial_{W_i^{k+1}} L_\rho = \nabla_{W_i^{k+1}} \phi(W_i^{k+1}, z^{k+1}, a_i^{k+1}, u_i^{k+1})$$

$$= \nu(W_i^{k+1}a_i^{k+1} - z_i^{k+1})(a_i^{k+1})^T$$

$$= \nabla_{W_i^{k+1}} \phi(W_i^{k+1}, z_i^{k+1}, a_i^{k+1}, u_i)$$

$$= \nabla_{W_i^{k+1}} \phi(W_i^{k+1}, z_i^{k+1}, a_i^{k+1}, u_i) + \theta_i^{k+1} \circ (W_i^{k+1} - \overline{W_i^{k+1}})$$

$$- \theta_i^{k+1} \circ (W_i^{k+1} - \overline{W_i^{k+1}}) - \nabla_{W_i^{k+1}} \phi(W_i^{k+1}, \overline{z_i^{k+1}}, a_i^{k+1}, u_i)$$

Because

$$\| - \theta_i^{k+1} \circ (W_i^{k+1} - \overline{W_i^{k+1}}) + \nu(W_i^{k+1}a_i^{k+1} - z_i^{k+1})(a_i^{k+1})^T - \nu(\overline{W_i^{k+1}}a_i^{k+1} - z_i^{k+1})(a_i^{k+1})^T\|$$

$$= \| - \theta_i^{k+1} \circ (W_i^{k+1} - \overline{W_i^{k+1}}) + \nu(W_i^{k+1} - \overline{W_i^{k+1}})(a_i^{k+1}) - \nu(z_i^{k+1} - \overline{z_i^{k+1}})(a_i^{k+1})^T\|$$

$$\leq \|\theta_i^{k+1} \circ (W_i^{k+1} - \overline{W_i^{k+1}})\| + \nu\|W_i^{k+1} - \overline{W_i^{k+1}}\||a_i^{k+1})(a_i^{k+1})^T\|$$

$$(\text{Triangle Inequality})$$

$$\leq \|\theta_i^{k+1} \circ (W_i^{k+1} - \overline{W_i^{k+1}})\| + \nu\|W_i^{k+1} - \overline{W_i^{k+1}}\|\,a_i^{k+1}(a_i^{k+1})^T\|$$

$$(\text{Cauchy-Schwarz inequality})$$

Because $a_i^{k+1}$ is bounded by Property 1 and Equation (A.11) holds, $\|\partial_{W_i^{k+1}} L_\rho\|$ can be upper bounded by a linear combination of $\|W_i^{k+1} - \overline{W_i^{k+1}}\|$ and $\|z_i^{k+1} - \overline{z_i^{k+1}}\|$. 
A.3. PROOF OF THEOREM 1

For $W_L^{k+1}$,

$$\partial_{W_L^{k+1}}L_\rho = \nabla_{W_L^{k+1}}(W_L^{k+1}, z_L^{k+1}, a_L^{k+1}, u^{k+1})$$

$$= \nabla_{W_L^{k+1}}(W_L^{k+1}, z_L^{k+1}, a_L^{k+1}, u^{k}) + \theta_L^{k+1} \circ (W_L^{k+1} - W_L^{k+1})$$

$$- \theta_L^{k+1} \circ (W_L^{k+1} - W_L^{k+1}) - \nabla_{W_L^{k+1}}(W_L^{k+1}, z_L^{k+1}, a_L^{k+1}, u^{k})$$

$$+ \nabla_{W_L^{k+1}}(W_L^{k+1}, z_L^{k+1}, a_L^{k+1}, u^{k})$$

$$= \nabla_{W_L^{k+1}}(W_L^{k+1}, z_L^{k+1}, a_L^{k+1}, u^{k}) + \theta_L^{k+1} \circ (W_L^{k+1} - W_L^{k+1})$$

$$- \theta_L^{k+1} \circ (W_L^{k+1} - W_L^{k+1}) + \rho(W_L^{k+1}a_L^{k+1} - z_L^{k+1} - u^{k+1}/\rho)(a_L^{k+1})^T$$

$$- \rho(W_L^{k+1}a_L^{k+1} - z_L^{k+1} - u^{k}/\rho)(a_L^{k+1})^T$$

Because

$$|| - \theta_L^{k+1} \circ (W_L^{k+1} - W_L^{k+1}) + \rho(W_L^{k+1}a_L^{k+1} - z_L^{k+1} - u^{k+1}/\rho)(a_L^{k+1})^T$$

$$- \rho(W_L^{k+1}a_L^{k+1} - z_L^{k+1} - u^{k}/\rho)(a_L^{k+1})^T||$$

$$= || - \theta_L^{k+1} \circ (W_L^{k+1} - W_L^{k+1}) + \rho(W_L^{k+1} - W_L^{k+1})a_L^{k+1}(a_L^{k+1})^T - \rho(z_L^{k+1} - z_L^{k+1})(a_L^{k+1})^T$$

$$- (u^{k+1} - u^k)(a_L^{k+1})^T||$$

$$\leq || \theta_L^{k+1} \circ (W_L^{k+1} - W_L^{k+1})|| + \rho||W_L^{k+1} - W_L^{k+1}||a_L^{k+1}(a_L^{k+1})^T||$$

$$+ \rho||(z_L^{k+1} - z_L^{k+1})(a_L^{k+1})^T|| + ||(u^{k+1} - u^k)(a_L^{k+1})^T|| \quad \text{(Triangle Inequality)}$$

$$\leq || \theta_L^{k+1} \circ (W_L^{k+1} - W_L^{k+1})|| + \rho||W_L^{k+1} - W_L^{k+1}||a_L^{k+1}||a_L^{k+1}|| + \rho||z_L^{k+1} - z_L^{k+1}||a_L^{k+1}||$$

$$+ H||z_L^{k+1} - z_L^{k+1}||a_L^{k+1}||$$

(Cauchy-Schwarz inequality, Lemma 11, $R(z_L; y)$ is Lipschitz differentiable)

Because $a_L^{k+1}$ is bounded by Property 1 and Equation (A.11) holds, $||\partial_{W_L^{k+1}}L_\rho||$ can be upper bounded by a linear combination of $||W_L^{k+1} - W_L^{k+1}||$, $||z_L^{k+1} - z_L^{k+1}||$ and $||z_L^{k+1} - z_L^{k}||$. 
A.3. PROOF OF THEOREM 1

For $z_{l}^{k+1}(l < L)$,

$$\partial_{z_{l}^{k+1}}L_{\rho} = \partial_{z_{l}^{k+1}}\phi(W_{l}^{k+1}, z_{l}^{k+1}, a_{l-1}^{k+1}, u^{k+1})$$

$$= \nu(z_{l}^{k+1} - W_{l}^{k+1}a_{l-1}^{k+1}) + \nu\partial f_{l}(z_{l}^{k+1}) \circ (f(z_{l}^{k+1}) - a_{l}^{k+1})$$

$$= \partial_{z_{l}^{k+1}}\phi(W_{l}^{k+1}, z_{l}^{k+1}, a_{l-1}^{k+1}, u^{k})$$

$$= \partial_{z_{l}^{k+1}}\phi(W_{l}^{k+1}, z_{l}^{k+1}, a_{l-1}^{k+1}, u^{k}) - \partial_{z_{l}^{k+1}}\phi(W_{l}^{k+1}, z_{l}^{k+1}, a_{l-1}^{k+1}, u^{k})$$

$$+ \partial_{z_{l}^{k+1}}\phi(W_{l}^{k+1}, z_{l}^{k+1}, a_{l-1}^{k+1}, u^{k})$$

$$= \nu\partial f_{l}(z_{l}^{k+1}) \circ (a_{l}^{k+1} - a_{l}^{k}) + \partial_{z_{l}^{k+1}}\phi(W_{l}^{k+1}, z_{l}^{k+1}, a_{l-1}^{k+1}, u^{k})$$

because $z_{l}^{k+1}$ is bounded and $f_{l}(z_{l})$ is continuous and hence $f_{l}(z_{l}^{k+1})$ is bounded, and the optimality condition of Equation (2.10) yields

$$0 \in \partial_{z_{l}^{k+1}}\phi(W_{l}^{k+1}, z_{l}^{k+1}, a_{l-1}^{k+1}, u^{k})$$

Therefore, $\|\partial_{z_{l}^{k+1}}L_{\rho}\|$ is upper bounded by $\|a_{l}^{k+1} - a_{l}^{k}\|$.

For $z_{L}^{k+1}$,

$$\nabla_{z_{L}^{k+1}}L_{\rho} = \nabla_{z_{L}^{k+1}}R(z_{L}^{k+1}, y) + \nabla_{z_{L}^{k+1}}\phi(W_{L}^{k+1}, z^{k+1}, a_{L-1}^{k+1}, u^{k+1})$$

$$= \nabla_{z_{L}^{k+1}}R(z_{L}^{k+1}, y) + \nabla_{z_{L}^{k+1}}\phi(W_{L}^{k+1}, z^{k+1}, a_{L-1}^{k+1}, u^{k})$$

$$- \nabla_{z_{L}^{k+1}}\phi(W_{L}^{k+1}, z^{k+1}, a_{L-1}^{k+1}, u^{k}) + \nabla_{z_{L}^{k+1}}\phi(W_{L}^{k+1}, z^{k+1}, a_{L-1}^{k+1}, u^{k+1})$$

$$= \nabla_{z_{L}^{k+1}}R(z_{L}^{k+1}, y) + \nabla_{z_{L}^{k+1}}\phi(W_{L}^{k+1}, z^{k+1}, a_{L-1}^{k+1}, u^{k}) + u^{k+1} - u^{k}$$

The optimality condition of Equation (2.11) yields

$$0 \in \nabla_{z_{L}^{k+1}}R(z_{L}^{k+1}, y) + \nabla_{z_{L}^{k+1}}\phi(W_{L}^{k+1}, z^{k+1}, a_{L-1}^{k+1}, u^{k})$$
and

$$\|u^{k+1} - u^k\| = \|\nabla_{z_L}^k R(z_L^{k+1}; y) - \nabla_{z_L}^k R(z_L^k; y)\| \quad \text{(Lemma 11)} \leq H \|z_L^{k+1} - z_L^k\|$$

(Cauchy-Schwarz inequality, $R(z_L^k; y)$ is Lipschitz differentiable)

Therefore $\|\nabla_{z_L}^k \| \rho\|$ is upper bounded by $\|z_L^{k+1} - z_L^k\|$.

For $a_L^{k+1}$ ($l < L - 1$),

$$\nabla_{a_L^{k+1}} \phi(W^{k+1} z^{k+1} a^{k+1}, u^{k+1})$$

$$= \nu(W^{k+1})^T (W^{k+1} a_L^{k+1} - z_L^{k+1}) + (a_L^{k+1} - f_i(z_L^{k+1}))$$

$$= \nabla_{a_L^{k+1}} \phi(W^{k+1} z^{k+1} a^{k+1}, u^{k})$$

$$= \tau_{l+1}^k o (a_L^{k+1} - \blacksquare_k^{k+1}) + \nabla_{a_L^{k+1}} \phi(W^{k+1} z^{k+1} a^{k+1}, u^{k}) - \tau_{l+1}^k o (a_L^k - \blacksquare_k^{k+1})$$

$$- \nabla_{a_L^{k+1}} \phi(W^{k+1} z^{k+1} a^{k+1}, u^{k}) + \nabla_{a_L^{k+1}} \phi(W^{k+1} z^{k+1} a^{k+1}, u^{k})$$

$$= \tau_{l+1}^k o (a_L^{k+1} - \blacksquare_k^{k+1}) + \nabla_{a_L^{k+1}} \phi(W^{k+1} z^{k+1} a^{k+1}, u^{k}) - \tau_{l+1}^k o (a_L^k - \blacksquare_k^{k+1})$$

$$- \nu(W^{k+1} z^{k+1} a^{k+1}, u^{k}) + \nu(a_L^{k+1} - f_i(z_L^{k+1}))$$

$$+ \nu(W^{k+1} z^{k+1} a^{k+1}, u^{k})$$

Because

\[
\begin{align*}
\|\tau_{l+1}^k o (a_L^{k+1} - \blacksquare_k^{k+1}) - \nu(W^{k+1} z^{k+1} a^{k+1}, u^{k}) & - \nu(a_L^{k+1} - f_i(z_L^{k+1})) \\
+ \nu(W^{k+1} z^{k+1} a^{k+1}, u^{k}) - \nu(a_L^k - f_i(z_L^{k+1}))\| & \\
& \leq \|\tau_{l+1}^k o (a_L^{k+1} - \blacksquare_k^{k+1})\| + \|a_L^{k+1} - \blacksquare_k^{k+1}\| + \|W^{k+1} z^{k+1} a^{k+1} - \nu(W^{k+1} z^{k+1} a^{k+1}, u^{k})\| & \\
& + \|W^{k+1} z^{k+1} a^{k+1} - \nu(W^{k+1} z^{k+1} a^{k+1}, u^{k})\| & \\
\end{align*}
\]

(Triangle Inequality)

Then we need to show that $\nu(W^{k+1} z^{k+1} a^{k+1} - \nu(W^{k+1} z^{k+1} a^{k+1}, u^{k})$ and $\nu(W^{k+1} z^{k+1} a^{k+1} - \nu(W^{k+1} z^{k+1} a^{k+1}, u^{k})$ are upper bounded by $\|W^{k+1} - W^{k+1}\|$, $\|z^{k+1} - \bar{z}^{k+1}\|$, and
\[ \| \mathbf{a}^{k+1} - \mathbf{\bar{a}}^{k+1} \|. \]

\[ \nu \|(W_{l+1}^{k+1})^T z_{l+1}^{k+1} - (\overline{W}_{l+1}^{k+1})^T \overline{z}_{l+1}^{k+1}\| \]

\[ = \nu \|(W_{l+1}^{k+1})^T z_{l+1}^{k+1} - (W_{l+1}^{k+1})^T z_{l+1}^{k+1} + (W_{l+1}^{k+1})^T z_{l+1}^{k+1} - (\overline{W}_{l+1}^{k+1})^T \overline{z}_{l+1}^{k+1}\| \]

\[ \leq \nu \|z_{l+1}^{k+1}\| \|W_{l+1}^{k+1} - \overline{W}_{l+1}^{k+1}\| + \nu \|W_{l+1}^{k+1}\| \|z_{l+1}^{k+1} - \overline{z}_{l+1}^{k+1}\| \]

(Triangle Inequality, Cauchy-Schwarz inequality)

Because \(\|z_{l+1}^{k+1}\|\) and \(\|\overline{W}_{l+1}^{k+1}\|\) are upper bounded, \(\nu \|(W_{l+1}^{k+1})^T z_{l+1}^{k+1} - (\overline{W}_{l+1}^{k+1})^T \overline{z}_{l+1}^{k+1}\|\) is therefore upper bounded by a combination of \(\|W_{l+1}^{k+1} - \overline{W}_{l+1}^{k+1}\|\) and \(\|z_{l+1}^{k+1} - \overline{z}_{l+1}^{k+1}\|\).

\[ \nu \|(W_{l+1}^{k+1})^T W_{l+1}^{k+1} a_{l+1}^{k+1} - (\overline{W}_{l+1}^{k+1})^T \overline{W}_{l+1}^{k+1} \overline{a}_{l+1}^{k+1}\| \]

\[ = \nu \|(W_{l+1}^{k+1})^T W_{l+1}^{k+1} a_{l+1}^{k+1} - (W_{l+1}^{k+1})^T W_{l+1}^{k+1} \overline{a}_{l+1}^{k+1} + (W_{l+1}^{k+1})^T W_{l+1}^{k+1} \overline{a}_{l+1}^{k+1} - (\overline{W}_{l+1}^{k+1})^T \overline{W}_{l+1}^{k+1} \overline{a}_{l+1}^{k+1}\| \]

\[ \leq \nu \|(W_{l+1}^{k+1})^T W_{l+1}^{k+1} (a_{l+1}^{k+1} - \overline{a}_{l+1}^{k+1})\| + \nu \|(W_{l+1}^{k+1})^T (W_{l+1}^{k+1} - \overline{W}_{l+1}^{k+1}) \overline{a}_{l+1}^{k+1}\| \]

\[ + \nu \|(W_{l+1}^{k+1} - \overline{W}_{l+1}^{k+1})^T \overline{W}_{l+1}^{k+1} \overline{a}_{l+1}^{k+1}\| \text{(Triangle Inequality)} \]

\[ \leq \nu \|W_{l+1}^{k+1}\| \|W_{l+1}^{k+1}\| \|a_{l+1}^{k+1} - \overline{a}_{l+1}^{k+1}\| + \nu \|W_{l+1}^{k+1}\| \|W_{l+1}^{k+1} - \overline{W}_{l+1}^{k+1}\| \|\overline{a}_{l+1}^{k+1}\| \]

\[ + \nu \|W_{l+1}^{k+1} - \overline{W}_{l+1}^{k+1}\| \|W_{l+1}^{k+1}\| \|\overline{a}_{l+1}^{k+1}\| \text{(Cauchy-Schwarz Inequality)} \]

Because \(\|W_{l+1}^{k+1}\|\), \(\|\overline{W}_{l+1}^{k+1}\|\) and \(\|\overline{a}_{l+1}^{k+1}\|\) are upper bounded, \(\nu \|(W_{l+1}^{k+1})^T W_{l+1}^{k+1} a_{l+1}^{k+1} - (\overline{W}_{l+1}^{k+1})^T \overline{W}_{l+1}^{k+1} \overline{a}_{l+1}^{k+1}\|\) is therefore upper bounded by a combination of \(\|W_{l+1}^{k+1} - \overline{W}_{l+1}^{k+1}\|\) and \(\|a_{l+1}^{k+1} - \overline{a}_{l+1}^{k+1}\|\).
A.3. PROOF OF THEOREM 1

For $a^{k+1}_{L-1}$,

$$\nabla a_{L-1}^{k+1} \phi(W^{k+1}, z^{k+1}, a^{k+1}, u^{k+1})$$

$$= \tau_{L-1}^{k+1} \circ (a^{k+1}_{L-1} - \overline{a}_{L-1}^{k+1}) + \nabla a_{L-1}^{k+1} \phi(W_{L-1}^{k+1}, z_{L-1}^{k+1}, a_{L-2}^{k+1}, u^{k}) - \tau_{L-1}^{k+1} \circ (a^{k+1}_{L-1} - \overline{a}_{L-1}^{k+1})$$

$$- \nabla a_{L-1}^{k+1} \phi(W_{L-1}^{k+1}, z_{L-1}^{k+1}, a_{L-2}^{k+1}, u^{k}) + \nabla a_{L-1}^{k+1} \phi(W^{k+1}, z^{k+1}, a^{k+1}, u^{k+1})$$

$$= \tau_{L-1}^{k+1} \circ (a^{k+1}_{L-1} - \overline{a}_{L-1}^{k+1}) + \nabla a_{L-1}^{k+1} \phi(W_{L-1}^{k+1}, z_{L-1}^{k+1}, a_{L-2}^{k+1}, u^{k}) - \tau_{L-1}^{k+1} \circ (a^{k+1}_{L-1} - \overline{a}_{L-1}^{k+1})$$

$$- \rho(W_{L}^{k+1})^T(W_{L}^{k+1} a_{L-1}^{k+1} - z_{L}^{k+1} - u^{k}/\rho) - \nu(a_{L-1}^{k+1} - f_{L-1}(z_{L-1}^{k+1}))$$

$$+ \rho(W_{L}^{k+1})^T(W_{L}^{k+1} a_{L-1}^{k+1} - z_{L}^{k+1} - u^{k+1}/\rho) + \nu(a_{L-1}^{k+1} - f_{L-1}(z_{L-1}^{k+1}))$$

Because

$$\| \tau_{L-1}^{k+1} \circ (a^{k+1}_{L-1} - \overline{a}_{L-1}^{k+1}) - \rho(W_{L}^{k+1})^T(W_{L}^{k+1} a_{L-1}^{k+1} - z_{L}^{k+1} - u^{k}/\rho)$$

$$- \nu(a_{L-1}^{k+1} - f_{L-1}(z_{L-1}^{k+1})) + \rho(W_{L}^{k+1})^T(W_{L}^{k+1} a_{L-1}^{k+1} - z_{L}^{k+1} - u^{k+1}/\rho)$$

$$+ \nu(a_{L-1}^{k+1} - f_{L-1}(z_{L-1}^{k+1}))\| \leq \|\tau_{L-1}^{k+1} \circ (a^{k+1}_{L-1} - \overline{a}_{L-1}^{k+1})\| + \nu\|a_{L-1}^{k+1} - \overline{a}_{L-1}^{k+1}\|$$

Then we need to show that $\rho\|(W_{L}^{k+1})^T z_{L}^{k+1} - (W_{L}^{k+1})^T z_{L}^{k+1}\|$, $\|(W_{L}^{k+1})^T u^{k} - (W_{L}^{k+1})^T u^{k}\|$ and $\rho\|(W_{L}^{k+1})^T W_{L}^{k+1} a_{L-1}^{k+1} - (W_{L}^{k+1})^T W_{L}^{k+1} a_{L-1}^{k+1}\|$ are upper bounded by $\|W^{k+1} - \overline{W}^{k+1}\|$, $\|z^{k+1} - \overline{z}^{k+1}\|$, $\|a^{k+1} - \overline{a}^{k+1}\|$ and $\|z^{k+1} - z^{k}\|$. 

$$\rho\|(W_{L}^{k+1})^T z_{L}^{k+1} - (W_{L}^{k+1})^T z_{L}^{k+1}\|$$

$$= \rho\|(W_{L}^{k+1})^T z_{L}^{k+1} + (W_{L}^{k+1})^T z_{L}^{k+1} - (W_{L}^{k+1})^T z_{L}^{k+1}\|$$

$$\leq \rho\|z_{L}^{k+1}\|\|W_{L}^{k+1} - \overline{W}^{k+1}\| + \rho\|W_{L}^{k+1}\|\|z_{L}^{k+1} - \overline{z}^{k+1}\|$$

(Triangle Inequality, Cauchy-Schwarz inequality)
A.3. PROOF OF THEOREM 1

Because $\|z_L^{k+1}\|$ and $\|\overline{W}_L^{k+1}\|$ are upper bounded, $\rho \|W_L^{k+1}Tz^{k+1}_L - (\overline{W}_L^{k+1})Tz^{k+1}_L\|$ is therefore upper bounded by a combination of $\|W_L^{k+1} - \overline{W}_L^{k+1}\|$ and $\|z^{k+1}_L - \overline{z}^{k+1}_L\|$. 

$$
\|W_L^{k+1}T u^{k+1} - (\overline{W}_L^{k+1})^T u^k\|
= \|(W_L^{k+1})^T u^{k+1} - (W_L^{k+1})^T u^k + (W_L^{k+1})^T u^k - (\overline{W}_L^{k+1})^T u^k\|
\leq \|(W_L^{k+1})^T (u^{k+1} - u^k)\| + \|(W_L^{k+1})^T u^k\| \text{(Triangle Inequality)}
\leq \|W_L^{k+1}\| \|u^{k+1} - u^k\| + \|W_L^{k+1} - \overline{W}_L^{k+1}\| \|u^k\| \text{(Cauchy-Schwarz inequality)}
= \|W_L^{k+1}\| \|\nabla_{z_L} R(z^{k+1}_L; y) - \nabla_{z_L} R(z^{k}_L; y)\| + \|W_L^{k+1} - \overline{W}_L^{k+1}\| \|u^k\| \text{(Lemma 11)}
\leq H \|W_L^{k+1}\| \|z^{k+1}_L - z^k_L\| + \|W_L^{k+1} - \overline{W}_L^{k+1}\| \|u^k\|
\text{(}\nabla z_L; y\text{) is Lipschitz differentiable)}
$$

Because $\|W_L^{k+1}\|$ and $\|u^k\|$ are bounded, $\|W_L^{k+1})^T u^{k+1} - (\overline{W}_L^{k+1})^T u^k\|$ can be upper bounded by a combination of $\|z^{k+1}_L - z^k_L\|$ and $\|W_L^{k+1} - \overline{W}_L^{k+1}\|$. 

$$
\rho \|(W_L^{k+1})^T W_L^{k+1} a_L^{k+1} - (W_L^{k+1})^T \overline{W}_L^{k+1} \overline{a}_L^{k+1}\|
\leq \rho \|(W_L^{k+1})^T W_L^{k+1} a_L^{k+1} - (W_L^{k+1})^T \overline{W}_L^{k+1} \overline{a}_L^{k+1}\| + \rho \|(W_L^{k+1})^T (W_L^{k+1} - \overline{W}_L^{k+1}) \overline{a}_L^{k+1}\| \text{(Triangle Inequality)}
\leq \rho \|W_L^{k+1}\| \|W_L^{k+1}\| \|a_L^{k+1} - \overline{a}_L^{k+1}\| + \rho \|W_L^{k+1}\| \|W_L^{k+1} - \overline{W}_L^{k+1}\| \|\overline{a}_L^{k+1}\| \text{(Cauchy-Schwarz inequality)}
$$

Because $\|W_L^{k+1}\|$, $\|\overline{W}_L^{k+1}\|$ and $\|\overline{a}_L^{k+1}\|$ are upper bounded, $\rho \|(W_L^{k+1})^T W_L^{k+1} a_L^{k+1} - (W_L^{k+1})^T \overline{W}_L^{k+1} \overline{a}_L^{k+1}\|$ is therefore upper bounded by a combination of $\|W_L^{k+1} - \overline{W}_L^{k+1}\|$ and $\|a_L^{k+1} - \overline{a}_L^{k+1}\|$. 

A.4. PROOF OF THEOREM 3

For $u^{k+1}$,

$$\nabla u^{k+1}_L = \nabla u^{k+1}_L \phi(W^{k+1}, z^{k+1}, a^{k+1}, u^{k+1})$$

$$= z^{k+1}_L - W^{k+1} L \rho a^{k+1}_L$$

$$= (1/\rho)(u^{k+1} - u^k)$$

$$= (1/\rho)(\nabla z^k_L R(z^k_L; y) - \nabla z^{k+1}_L R(z^{k+1}_L; y))(\text{Lemma 11})$$

Because

$$\|(1/\rho)(\nabla z^k_L R(z^k_L; y) - \nabla z^{k+1}_L R(z^{k+1}_L; y))\|$$

$$\leq (H/\rho)\|z^{k+1}_L - z^k_L\|(R(z_L; y) \text{ is Lipschitz differentiable})$$

Therefore, $\|\nabla u^{k+1}_L \rho\|$ is upper bounded by $\|z^{k+1}_L - z^k_L\|$. \hfill \square

A.4 Proof of Theorem 3

Proof. To prove this theorem, we will first show that $c_k$ satisfies two conditions: (1). $c_k \geq c_{k+1}$. (2). $\sum_{k=0}^{\infty} c_k$ is bounded. We then conclude the convergence rate of $o(1/k)$ based on these two conditions. Specifically, first, we have

$$c_k = \min_{0 \leq i \leq k} \left( \sum_{l=1}^{L} (\| \bar{\theta}^{i+1}_l \circ (W^{i+1}_l - W^i_l)^{\circ 2}\|/2 + \| \bar{\theta}^{i+1}_l \circ (W^i_l - W^{i+1}_l)^{\circ 2}\|/2)$$

$$+ \| \bar{\tau}^{i+1}_l \circ (\bar{a}^{i+1}_l - a^i_l)^{\circ 2}\|/2 + \| \bar{\tau}^{i+1}_l \circ (a^{i+1}_l - \bar{a}^{i+1}_l)^{\circ 2}\|/2 + C_2 \| \bar{z}^{i+1}_L - z^i_L \|^2 \right)$$

$$+ \min_{0 \leq i \leq k+1} \left( \sum_{l=1}^{L} (\| \bar{\theta}^{i+1}_l \circ (W^{i+1}_l - W^i_l)^{\circ 2}\|/2 + \| \bar{\theta}^{i+1}_l \circ (W^i_l - W^{i+1}_l)^{\circ 2}\|/2)$$

$$+ \sum_{l=1}^{L-1} (\| \bar{\tau}^{i+1}_l \circ (\bar{a}^{i+1}_l - a^i_l)^{\circ 2}\|/2 + \| \bar{\tau}^{i+1}_l \circ (a^{i+1}_l - \bar{a}^{i+1}_l)^{\circ 2}\|/2 + C_2 \| \bar{z}^{i+1}_L - z^i_L \|^2 \right)$$

$$= c_{k+1}$$
Therefore $c_k$ satisfies the first condition. Second,

$$\sum_{k=0}^{\infty} c_k$$

$$= \sum_{k=0}^{\infty} \min_{0 \leq i \leq k} \left( \sum_{l=1}^{L} (\| \theta_l^{i+1} \circ (W_l^{i+1} - W_l^k) \|_2^2 / 2 + \| \theta_l^{i+1} \circ (W_l^{i+1} - W_l^k) \|_1^2 / 2) + \sum_{l=1}^{L-1} (\| \tau_l^{i+1} \circ (\overline{a}_l^{i+1} - a_l^i) \|_2^2 / 2 + \| \tau_l^{i+1} \circ (\overline{a}_l^{i+1} - a_l^i) \|_1^2 / 2) + C_2 \| \overline{z}_L^{i+1} - z_L^i \|_2^2 + C_1 \| \overline{z}_L^{i+1} - z_L^i \|_2^2 \right)$$

$$\leq \sum_{k=0}^{\infty} \left( \sum_{l=1}^{L} (\| \theta_l^{k+1} \circ (W_l^{k+1} - W_l^k) \|_2^2 / 2 + \| \theta_l^{k+1} \circ (W_l^{k+1} - W_l^k) \|_1^2 / 2) + \sum_{l=1}^{L-1} (\| \tau_l^{k+1} \circ (\overline{a}_l^{k+1} - a_l^k) \|_2^2 / 2 + \| \tau_l^{k+1} \circ (\overline{a}_l^{k+1} - a_l^k) \|_1^2 / 2) + C_2 \| \overline{z}_L^{k+1} - z_L^k \|_2^2 + C_1 \| \overline{z}_L^{k+1} - z_L^k \|_2^2 \right)$$

$$\leq L_\rho(W^0, z^0, a^0, u^0) - L_\rho(W^*, z^*, a^*, u^*) \quad \text{(Property 2)}$$

So $\sum_{k=0}^{\infty} c_k$ is bounded and $c_k$ satisfies the second condition. Finally, it has been proved that the sufficient conditions of convergence rate $o(1/k)$ are: (1) $c_k \geq c_{k+1}$, and (2) $\sum_{k=0}^{\infty} c_k$ is bounded, and (3) $c_k \geq 0$ (Lemma 1.2 in [26]). Since we have proved the first two conditions and the third one $c_k \geq 0$ is obvious, the convergence rate of $o(1/k)$ is proven. \qed
Appendix B

Appendix of the mDLAM Algorithm

B.1 Definition

Several definitions are shown here for the sake of convergence analysis.

Definition 7 (Coercivity). Any arbitrary function $G_2(x)$ is coercive over a nonempty set $\text{dom}(G_2)$ if as $\|x\| \to \infty$ and $x \in \text{dom}(G_2)$, we have $G_2(x) \to \infty$, where $\text{dom}(G_2)$ is a domain set of $G_2$.

Definition 8 (Multi-convexity). A function $f(x_1, x_2, \ldots, x_m)$ is a multi-convex function if $f$ is convex with regard to $x_i (i = 1, \ldots, m)$ while fixing other variables.

Definition 9 (Lipschitz Differentiability). A function $f(x)$ is Lipschitz differentiable with Lipschitz coefficient $L > 0$ if for any $x_1, x_2 \in \mathbb{R}$, the following inequality holds:

$$\|\nabla f(x_1) - \nabla f(x_2)\| \leq L\|x_1 - x_2\|.$$ 

For Lipschitz differentiability, we have the following lemma (Lemma 2.1 in [2]):
Lemma 15. If $f(x)$ is Lipschitz differentiable with $L > 0$, then for any $x_1, x_2 \in \mathbb{R}$

$$f(x_1) \leq f(x_2) + \nabla f^T (x_2)(x_1 - x_2) + \frac{L}{2} \| x_1 - x_2 \|^2.$$ 

**Definition 10** (Fréchet Subdifferential). For each $x_1 \in \text{dom}(u_1)$, the Fréchet subdifferential of $u_1$ at $x_1$, which is denoted as $\hat{\partial}u_1(x_1)$, is the set of vectors $v$, which satisfy

$$\lim \inf_{x_2 \neq x_1, x_2 \to x_1} (u_1(x_2) - u_1(x_1) - v^T(x_2 - x_1))/\|x_2 - x_1\| \geq 0.$$ 

The vector $v \in \hat{\partial}u_1(x_1)$ is a Fréchet subgradient.

Then the definition of the limiting subdifferential, which is based on Fréchet subdifferential, is given in the following [84]:

**Definition 11** (Limiting Subdifferential). For each $x \in \text{dom}(u_2)$, the limiting subdifferential (or subdifferential) of $u_2$ at $x$ is

$$\partial u_2(x) = \{ v_1 | \exists x_k \to x, \text{s.t. } u_2(x_k) \to u_2(x), v^k \in \hat{\partial}u_2(x_k), v^k \to v \}.$$ 

where $x_k$ is a sequence whose limit is $x$ and the limit of $u_2(x_k)$ is $u_2(x)$, $v^k$ is a sequence, which is a Fréchet subgradient of $u_2$ at $x_k$ and whose limit is $v$. The vector $v \in \partial u_2(x)$ is a limiting subgradient.

Specifically, when $u_2$ is convex, its limiting subdifferential is reduced to regular subdifferential [84], which is defined as follows:

**Definition 12** (Regular Subdifferential). For each $x_1 \in \text{dom}(f)$, the regular subdifferential of a convex function $f$ at $x_1$, which is denoted as $\partial f(x_1)$, is the set of vectors
The vector \( v \in \partial f(x_1) \) is a regular subgradient.

**Definition 13** (Quasilinearity). A function \( f(x) \) is quasiconvex if for any sublevel set \( S_\nu(f) = \{x|f(x) \leq \nu\} \) is a convex set. Likewise, A function \( f(x) \) is quasiconcave if for any superlevel set \( S_\nu(f) = \{x|f(x) \geq \nu\} \) is a convex set. A function \( f(x) \) is quasilinear if it is both quasiconvex and quasiconcave.

**Definition 14** (Locally Strong Convexity). A function \( f(x) \) is locally strongly convex within a bound set \( D \) with a constant \( \mu \) if

\[
f(y) \geq f(x) + g^T(y - x) + \frac{\mu}{2} \|x - y\|^2 \quad \forall \ g \in \partial f(x) \text{ and } x, y \in D.
\]

Simply speaking, a locally strongly convex function lies above a quadratic function within a bounded set.

**Definition 15** (Kurdyka-Lojasiewicz (KL) Property). A function \( f(x) \) has the KL Property at \( x \in dom \partial f = \{x \in \mathbb{R} : \partial f(x) \neq \emptyset\} \) if there exists \( \eta \in (0, +\infty) \), a neighborhood \( X \) of \( x \) and a function \( \psi \in \Psi_\eta \), such that for all

\[
x \in X \cap \{x \in \mathbb{R} : f(\overline{x}) < f(x) < f(\overline{x}) + \eta\},
\]

the following inequality holds

\[
\psi'(f(x) - f(\overline{x})) dist(0, \partial f(x)) \geq 1,
\]

where \( \Psi_\eta \) stands for a class of function \( \psi : [0, \eta] \to \mathbb{R}^+ \) satisfying: (1). \( \phi \) is concave and \( \psi'(x) \) continuous on \((0, \eta)\); (2). \( \psi \) is continuous at 0, \( \psi(0) = 0 \); and (3). \( \psi'(x) > 0 \) for all \( x \in (0, \eta) \).
0, ∀x ∈ (0, η).

The following lemma shows that a locally strongly convex function satisfies the KL Property:

**Lemma 16 ([115]).** A locally strongly convex function \( f(x) \) with a constant \( \mu \) satisfies the KL Property at any \( x \in \mathbb{D} \) with \( \psi(x) = \frac{2}{\mu} \sqrt{x} \) and \( X = \mathbb{D} \cap \{ y : f(y) \geq f(x) \} \).

### B.2 Preliminary Lemmas

In this section, we present preliminary lemmas of the proposed mDLAM algorithm. The limiting subdifferential is used to prove the convergence of the proposed mD-LAM algorithm in the following convergence analysis. Without loss of generality, \( \partial R \) is assumed to be nonempty, and the limiting subdifferential of \( F \) defined in Problem 3 is [115]:

\[
\partial F(W, z, a) = \partial_W F \times \partial_z F \times \partial_a F,
\]

where \( \times \) means the Cartesian product.

**Lemma 17.** If Equation (3.3) holds, then

\[
\nabla_{W_{i+1}} \phi + \theta^{k+1}_i (W_{i+1}^k - \overline{W}_{i+1}^k) = 0.
\]

Likewise, if Equation (3.4) holds, then there exists \( q \) such that

\[
\nabla_{z_{i+1}} \phi + \rho(z_{i+1}^k - \overline{z}_{i+1}^k) + q = 0,
\]

where \( q \) is a subgradient with regard to \( z_{i+1}^k \) to satisfy the constraint \( f_i(z_{i+1}^k) - \varepsilon \leq a_i^k \leq f_i(z_{i+1}^k) + \varepsilon \).
If Equation (3.5) holds, then there exists $u \in \partial R(z_{L}^{k+1}; y)$ such that

$$\nabla z_{L}^{k+1} \phi + \rho(z_{L}^{k+1} - z_{L}^{k+1}) + u = 0.$$ 

If Equation (3.6) holds, then there exists $v$ such that

$$\nabla a_{l}^{k+1} \phi + \tau_{l}^{k+1}(a_{l}^{k+1} - a_{l}^{k+1}) + v = 0,$$

where $v$ is a subgradient with regard to $a_{l}^{k+1}$ to satisfy the constraint $f_{l}(z_{l}^{k+1}) - \varepsilon \leq a_{l}^{k+1} \leq f_{l}(z_{l}^{k+1}) + \varepsilon$.

**Proof.** These can be obtained by directly applying the optimality conditions of Equation (3.3), Equation (3.4), Equation (3.5) and Equation (3.6), respectively.

**Lemma 18.** For Equation (3.4) and Equation (3.5), the following inequalities hold:

$$V_{l}^{k+1}(z_{l}^{k+1}) \geq \phi(a_{l-1}^{k+1}, W_{l}^{k+1}, z_{l}^{k+1}).$$  \hspace{1cm} (B.1)

**Proof.** Because $\phi(a_{l-1}, W_{l}, z_{l})$ is Lipschitz differentiable with respect to $z_{l}$ with Lipschitz coefficient $\rho$, we directly apply Lemma 15 to $\phi$ to obtain Equation (B.1).

**B.3 Main Proofs**

**Proof of Lemma 1**

**Proof.** In Algorithm 4, we only show Equation (3.7) because Equation (3.8) and Equation (3.9) follow the same routine of Equation (3.7).

In Line 7 of Algorithm 4, if $F(W_{\leq l}^{k+1}, z_{\leq l-1}^{k+1}, a_{\leq l-1}^{k+1}) < F(W_{\leq l-1}^{k+1}, z_{\leq l-1}^{k+1}, a_{\leq l-1}^{k+1})$, then obviously there exists $\alpha_{l}^{k+1} > 0$ such that Equation (3.7) holds. Otherwise, according
B.3. MAIN PROOFS

Let \( \phi(a_{l-1}, W_l, z_l) \) is convex with regard to \( W_l \), according to the definition of regular subgradient, we have

\[
\phi(a_{l-1}^{k+1}, W_l^k, z_l^k) \geq \phi(a_{l-1}^{k+1}, W_l^{k+1}, z_l^k) + \nabla_{W_l} \phi^T(W_l^k - W_l^{k+1}), \tag{B.2}
\]

Therefore, we have

\[
F(W_{\leq l-1}^{k+1}, z_{\leq l-1}^{k+1}, a_{\leq l-1}^{k+1}) - F(W_{\leq l-1}^{k+1}, z_{\leq l-1}^{k+1}, a_{\leq l-1}^{k+1}) \\
= \phi(a_{l-1}^{k+1}, W_l^k, z_l^k) - \phi(a_{l-1}^{k+1}, W_l^{k+1}, z_l^k) \quad \text{(Definition of } F \text{ in Problem 3)} \\
\geq -\left(\nabla_{W_l} \phi^T(W_l^{k+1} - W_l^k) - \frac{\theta_l^{k+1}}{2} \|W_l^{k+1} - W_l^k\|_2^2 \right) \\
- \phi(a_{l-1}^{k+1}, W_l^{k+1}, z_l^k) + \phi(a_{l-1}^{k+1}, W_l^k, z_l^k) \quad \text{(Equation (3.2))} \\
\geq -\left(\nabla_{W_l} \phi^T(W_l^{k+1} - W_l^k) - \frac{\theta_l^{k+1}}{2} \|W_l^{k+1} - W_l^k\|_2^2 \right) \quad \text{(Equation (B.2))} \\
= \theta_l^{k+1}(W_l^{k+1} - W_l^k)^T(W_l^{k+1} - W_l^k) - \frac{\theta_l^{k+1}}{2} \|W_l^{k+1} - W_l^k\|_2^2 \quad \text{(Lemma 17)} \\
= \frac{\theta_l^{k+1}}{2} \|W_l^{k+1} - W_l^k\|_2^2 \quad \text{($W_l^{k+1} = W_l^k$).}
\]

Let \( \alpha_{l-1}^{k+1} = \theta_l^{k+1} \), then Equation (3.7) still holds. \( \square \)

**Proof of Lemma 3**

**Proof.** In Algorithm 4:

(a). We sum Equation (3.7), Equation (3.8) and Equation (3.9) from \( l = 1 \) to \( L \) and from \( k = 0 \) to \( K \) to obtain

\[
F(W^0, z^0, a^0) - F(W^K, z^K, a^K) \\
\geq \sum_{k=0}^{K} \sum_{l=1}^{L} \left( \frac{\alpha_{l-1}^{k+1}}{2} \|W_l^{k+1} - W_l^k\|_2^2 + \frac{\gamma_l^{k+1}}{2} \|z_l^{k+1} - z_l^k\|_2^2 \right) \\
+ \sum_{l=1}^{L-1} \frac{\delta_l^{k+1}}{2} \|a_{l-1}^{k+1} - a_l^k\|_2^2. \tag{B.3}
\]

So \( F(W^K, z^K, a^K) \leq F(W^0, z^0, a^0) \). This proves the upper boundness of \( F \). Let
$K \to \infty$ in Equation (B.3), since $F > 0$ is lower bounded, we have

$$\sum_{k=0}^{K} \left( \sum_{l=1}^{L} \left( \frac{\delta^{k+1}}{2} \|W_{l}^{k+1} - W_{l}^{k}\|_{2}^{2} + \frac{\gamma^{k+1}}{2} \|z_{l}^{k+1} - z_{l}^{k}\|_{2}^{2} \right) + \sum_{l=1}^{L-1} \frac{\delta^{k+1}}{2} \|a_{l}^{k+1} - a_{l}^{k}\|_{2}^{2} \right) < \infty. \quad (B.4)$$

Since the sum of this infinite series is finite, every term converges to 0. This means that $\lim_{k \to \infty} W_{l}^{k+1} - W_{l}^{k} = 0$, $\lim_{k \to \infty} z_{l}^{k+1} - z_{l}^{k} = 0$ and $\lim_{k \to \infty} a_{l}^{k+1} - a_{l}^{k} = 0$. In other words, $\lim_{k \to \infty} W^{k+1} - W^{k} = 0$, $\lim_{k \to \infty} z^{k+1} - z^{k} = 0$, and $\lim_{k \to \infty} a^{k+1} - a^{k} = 0$.

(b). Because $F(W^{k}, z^{k}, a^{k})$ is bounded, by the definition of coercivity and Assumption 4, $(W^{k}, z^{k}, a^{k})$ is bounded.

$\square$

**Proof of Lemma 4**

*Proof.* As shown in Remark 2.2 in [115],

$$\partial_{W_{k+1}^{+}} F = \{\partial_{W_{1}^{k+1}} F\} \times \{\partial_{W_{2}^{k+1}} F\} \times \cdots \times \{\partial_{W_{L}^{k+1}} F\},$$

where $\times$ denotes Cartesian Product.

In Algorithm 4, for $W_{l}^{k+1}$, according to Line 6 of Algorithm 4, if

$$F(W_{\leq l}^{k+1}, z_{\leq l-1}^{k+1}, a_{\leq l-1}^{k+1}) < F(W_{\leq l-1}^{k+1}, z_{\leq l-1}^{k+1}, a_{\leq l-1}^{k+1}),$$

then

$$\partial_{W_{l}^{k+1}} F = \nabla_{W_{l}^{k+1}} F = \nabla_{W_{l}^{k+1}} \phi(a_{l-1}^{k+1}, W_{l}^{k+1}, z_{l}^{k+1}) \quad (\text{Definition of } F \text{ in Problem 3})$$

$$= \nabla_{W_{l}^{k+1}} \phi(a_{l-1}^{k+1}, W_{l}^{k+1}, z_{l}^{k+1}) - \nabla_{W_{l}^{k+1}} \phi(a_{l-1}^{k+1}, W_{l}^{k+1}, z_{l}^{k}) - \theta_{l}^{k+1}(W_{l}^{k+1} - W_{l}^{k})$$

$$+ \nabla_{W_{l}^{k+1}} \phi(a_{l-1}^{k+1}, W_{l}^{k+1}, z_{l}^{k}) + \theta_{l}^{k+1}(W_{l}^{k+1} - W_{l}^{k})$$

$$= \rho(W_{l}^{k+1} - W_{l}^{k})a_{l-1}^{k+1}(a_{l-1}^{k+1})^{T} - \rho(z_{l}^{k+1} - z_{l})^{T}(a_{l-1}^{k+1})^{T} - \theta_{l}^{k+1}(W_{l}^{k+1} - W_{l}^{k})$$

$$+ \nabla_{W_{l}^{k+1}} \phi(a_{l-1}^{k+1}, W_{l}^{k+1}, z_{l}^{k}) + \theta_{l}^{k+1}(W_{l}^{k+1} - W_{l}^{k}). \quad (B.5)$$
B.3. MAIN PROOFS

On one hand, we have

\[
\| \rho(W_t^{k+1} - \overline{W}_t^{k+1})a_{t-1}^{k+1}(a_{t-1}^{k+1})^T - \rho(z_t^{k+1} - z_t^k)(a_{t-1}^{k+1})^T - \theta_t^{k+1}(W_t^{k+1} - \overline{W}_t^{k+1}) \|
\]

\[
\leq \rho \| (W_t^{k+1} - \overline{W}_t^{k+1})a_{t-1}^{k+1}(a_{t-1}^{k+1})^T \| + \rho \| (z_t^{k+1} - z_t^k)(a_{t-1}^{k+1})^T \| + \theta_t^{k+1}\| W_t^{k+1} - \overline{W}_t^{k+1} \|
\]

(Triangle Inequality)

\[
\leq \rho \| W_t^{k+1} - \overline{W}_t^{k+1} \| \| a_{t-1}^{k+1} \| \| a_{t-1}^{k+1} \| + \rho \| z_t^{k+1} - z_t^k \| \| a_{t-1}^{k+1} \| + \theta_t^{k+1}\| W_t^{k+1} - \overline{W}_t^{k+1} \|
\]

(Cauchy-Schwarz Inequality)

\[
\leq \rho M_a \| z_t^{k+1} - z_t^k \| + (\rho M_a^2 + \theta_t^{k+1})\| W_t^{k+1} - \overline{W}_t^{k+1} \| \quad \text{(Lemma 3)} \quad (B.6)
\]

\[
\leq \rho M_a \| z_t^{k+1} - z_t^k \| + (\rho M_a^2 + \theta_t^{k+1})\| W_t^{k+1} - (W_t^k + \omega^k(W_t^k - W_t^{k-1})) \|
\]

(Nesterov Acceleration)

\[
\leq \rho M_a \| z_t^{k+1} - z_t^k \| + (\rho M_a^2 + \theta_t^{k+1})\| W_t^{k+1} - W_t^k \| + (\rho M_a^2 + \theta_t^{k+1})\| W_t^k - W_t^{k-1} \|
\]

(Triangle Inequality and \( \omega^k < 1 \)).

On the other hand, Lemma 17 holds, that is

\[
\nabla_{\overline{W}_t^{k+1}} \phi(a_{t-1}^{k+1}, \overline{W}_t^{k+1}, z_t^k) + \theta_t^{k+1}(W_t^{k+1} - \overline{W}_t^{k+1}) = 0.
\]

Therefore, there exists \( g_{t,1}^{k+1} \in \partial_{W_t^{k+1}}F \) such that

\[
\| g_{t,1}^{k+1} \| \leq \rho M_a \| z_t^{k+1} - z_t^k \| + (\rho M_a^2 + \theta_t^{k+1})\| W_t^{k+1} - W_t^k \| + (\rho M_a^2 + \theta_t^{k+1})\| W_t^k - W_t^{k-1} \|.
\]

This shows that there exists \( g_1^{k+1} = g_{1,1}^{k+1} \times g_{1,2}^{k+1} \times \cdots \times g_{1,L}^{k+1} \in \partial_{W_t^{k+1}}F \) and \( C_2 = \max(\rho M_a, \rho M_a^2 + \theta_1^{k+1}, \rho M_a^2 + \theta_2^{k+1}, \cdots, \rho M_a^2 + \theta_L^{k+1}) \) such that

\[
\| g_{t,1}^{k+1} \| \leq C_2(\| W_t^{k+1} - W_t^k \| + \| z_t^{k+1} - z_t^k \| + \| W_t^k - W_t^{k-1} \|). \quad (B.7)
\]
B.3. MAIN PROOFS

Otherwise, we have

\[
\left\| \rho(W_{t}^{k+1} - W_{t}^{k+1})a_{t-1}^{k+1}(a_{t-1}^{k+1})^T - \rho(z_{t}^{k+1} - z_{t}^{k})(a_{t-1}^{k+1})^T - \theta_{t}^{k+1}(W_{t}^{k+1} - W_{t}^{k+1}) \right\|
\]

\[
\leq \rho M_a \left\| z_{t}^{k+1} - z_{t}^{k} \right\| + (\rho M_a^2 + \theta_{t}^{k+1}) \left\| W_{t}^{k+1} - W_{t}^{k+1} \right\| (\text{Equation (B.6)})
\]

\[
= \rho M_a \left\| z_{t}^{k+1} - z_{t}^{k} \right\| + (\rho M_a^2 + \theta_{t}^{k+1}) \left\| W_{t}^{k+1} - W_{t}^{k} \right\| (W_{t}^{k+1} = W_{t}^{k}).
\]

Similarly, Lemma 17 holds, that is

\[
\nabla W_{t}^{k+1} \phi(a_{t-1}^{k+1}, W_{t}^{k+1}, z_{t}^{k}) + \theta_{t}^{k+1}(W_{t}^{k+1} - W_{t}^{k-1}) = 0.
\]

By Equation (B.5), we know that there exists \( g_{1,l}^{k+1} \in \partial W_{t}^{k+1}F \) such that

\[
\left\| g_{1,l}^{k+1} \right\| \leq \rho M_a \left\| z_{t}^{k+1} - z_{t}^{k} \right\| + (\rho M_a^2 + \theta_{t}^{k+1}) \left\| W_{t}^{k+1} - W_{t}^{k} \right\|. \tag{B.8}
\]

Combining Equation (B.7) with Equation (B.8), we show that there exists \( g_{1,l}^{k+1} = g_{1,1}^{k+1} \times g_{1,2}^{k+1} \times \cdots \times g_{1,L}^{k+1} \in \partial W_{t}^{k+1}F \) and \( C_2 = \max(\rho M_a, \rho M_a^2 + \theta_{t}^{k+1}, \rho M_a^2 + \theta_{t}^{k+1}, \cdots, \rho M_a^2 + \theta_{t}^{k+1}) \) such that

\[
\left\| g_{l}^{k+1} \right\| \leq C_2(\left\| W_{t}^{k+1} - W_{t}^{k} \right\| + \left\| z_{t}^{k+1} - z_{t}^{k} \right\| + \left\| W_{t}^{k} - W_{t}^{k-1} \right\|).
\]

\[\square\]

**Proof of Lemma 2**

**Proof.** We add Equation (3.7), Equation (3.8), and Equation (3.9) from \( l = 1 \) to \( L \) to obtain

\[
F(W_{t}^{k}, z_{t}^{k}, a_{t}^{k}) - F(W_{t}^{k+1}, z_{t}^{k+1}, a_{t}^{k+1})
\]

\[
\geq \sum_{l=1}^{L} \left( \frac{\alpha_{l}^{k+1}}{2} \left\| W_{l}^{k+1} - W_{l}^{k} \right\|_{2}^{2} + \frac{\beta_{l}^{k+1}}{2} \left\| z_{l}^{k+1} - z_{l}^{k} \right\|_{2}^{2} + \sum_{l=1}^{L-1} \frac{\delta_{l}^{k+1}}{2} \left\| a_{l}^{k+1} - a_{l}^{k} \right\|_{2}^{2}. \right)
\]
Let $C_5 = \min(\alpha k,\gamma k,\delta k) > 0$, we have

$$
F(W^k, z^k, a^k) - F(W^{k+1}, z^{k+1}, a^{k+1}) \\
\geq C_5(\sum_{l=1}^{L}(\|W^{k+1}_l - W^k_l\|_2^2 + \|z^{k+1}_l - z^k_l\|_2^2) + \sum_{l=1}^{L-1} \|a^{k+1}_l - a^k_l\|_2^2) \\
= C_5(\|W^{k+1} - W^k\|_2^2 + \|z^{k+1} - z^k\|_2^2 + \|a^{k+1} - a^k\|_2^2) \\
\geq 0. 
$$

(B.9)

By Lemma 3(b) and a monotone sequence is convergent if it is bounded, then $F(W^k, z^k, a^k)$ is convergent. \(\square\)

**Proof of Theorem 4**

*Proof.* By Lemma 3(a), $\lim_{k \to \infty} W^{k+1} - W^k = 0$. By Lemma 3(b), there exists a subsequence $W^s$ such that $W^s \to W^*$, where $W^*$ is a limit point. From Lemma 4, there exist $g^s_i \in \partial_{W^s} F$ such that $\|g^s_i\| \to 0$ as $s \to \infty$. According to the definition of limiting subgradient, we have $0 \in \partial_{W^*} F$. In other words, $W^*$ is a stationary point of $F$ in Problem 3. \(\square\)

**Proof of Theorem 5**

*Proof.* In Algorithm 4, we prove this by the KL Property. Firstly, we consider Equation (3.4) and Equation (3.6), by Lemma 3, $f_l(z^{k+1}_l - \nabla \phi_{z^{k+1}_l}/\rho) - a^k_l$ and $f_l(z^{k+1}_l - a^{k+1}_l + \nabla_{\tau_{l}^{k+1}} \phi/\tau_{l}^{k+1}$ are bounded, i.e. there exist constants $D_1$ and $D_2$ such that

$$
| f_l(z^{k+1}_l - \nabla \phi_{z^{k+1}_l}/\rho) - a^k_l | < D_1. \\
| f_l(z^{k+1}_l - a^{k+1}_l + \nabla_{\tau_{l}^{k+1}} \phi/\tau_{l}^{k+1} | | < D_2. 
$$

Let $\varepsilon = \max(D_1, D_2)$, then the solutions to Equation (3.4) and Equation (3.6) are
simplified as follows:

\[ z_{i}^{k+1} \leftarrow z_{i}^{k+1} - \nabla_{z_{i}^{k+1}} \phi / \rho. \]  
(B.10)

\[ a_{i}^{k+1} \leftarrow a_{i}^{k+1} - \nabla_{a_{i}^{k+1}} \phi / \tau_{i}^{k+1}. \]  
(B.11)

This is because \( f_{i}(z_{i}^{k+1}) - \varepsilon \leq a_{i}^{k} \leq f_{i}(z_{i}^{k+1}) + \varepsilon \) and \( f_{i}(z_{i}^{k+1}) - \varepsilon \leq a_{i}^{k+1} \leq f_{i}(z_{i}^{k+1}) + \varepsilon \) hold in Equation (3.4) and Equation (3.6), respectively.

Next, we prove that given \( \varepsilon = \max(D_{1}, D_{2}) \), there exists \( C_{3} = \max(\rho M_{w}^{2} + \tau_{1}^{k+1}, \rho M_{w}^{2} + \tau_{2}^{k+1}, \rho M_{w}^{2} + \tau_{3}^{k+1}, \cdots, \rho M_{w}^{2} + \tau_{L-1}^{k+1}, 2\rho M_{w}M_{a} + \rho M_{z}) \), some \( g_{3}^{k+1} \in \partial z_{k+1} F \) and \( g_{4}^{k+1} \in \partial a_{k+1} F \) such that

\[
\|g_{3}^{k+1}\| = 0,
\]

\[
\|g_{4}^{k+1}\| \leq C_{3}(\|a^{k+1} - a^{k}\| + \|a^{k} - a^{k-1}\| + \|W^{k+1} - W^{k}\| + \|z^{k+1} - z^{k}\|).
\]

As shown in [107, 115],

\[
\partial z_{k+1} F = \partial z_{1} F \times \partial z_{2} F \times \cdots \times \partial z_{L} F,
\]

\[
\nabla a_{k+1} F = \nabla a_{1} F \times \nabla a_{2} F \times \cdots \times \nabla a_{L-1} F,
\]

where \( \times \) denotes Cartesian Product.

For \( z_{i}^{k+1}(l < L) \), according to Line 18 of Algorithm 4, no matter
B.3. MAIN PROOFS

\[ F(W_{k+1}^l, z_{k+1}^l, a_{k+1}^{l-1}) \geq F(W_{k+1}^{l'}, z_{k+1}^{l'}, a_{k+1}^{l'-1}) \] or not, we have

\[
\partial z_{k+1} F = \nabla z_{k+1} \phi(a_{l-1}^{k+1}, W_{l}^{k+1}, z_{l+1}^{k+1}) \\
= \nabla z_{k+1} \phi(a_{l-1}^{k+1}, W_{l}^{k+1}, z_{l+1}^{k+1}) - \nabla z_{k+1} \phi(a_{l-1}^{k+1}, W_{l}^{k+1}, z_{l+1}^{k+1}) - \rho(z_{l+1}^{k+1} - z_{l}^{k+1}) \\
\text{(Equation (B.10))} \\
= 0.
\]

For \( z_{L+1}^{k+1} \), according to Line 12 of Algorithm 4, no matter

\[ F(W_{k+1}^{L}, z_{k+1}^{L}, a_{k+1}^{L-1}) \geq F(W_{k+1}^{L'}, z_{k+1}^{L'}, a_{k+1}^{L'-1}) \] or not, we have

\[
\partial z_{L+1} F = \nabla z_{L+1} \phi(a_{L-1}^{k+1}, W_{L}^{k+1}, z_{L+1}^{k+1}) + \partial R(z_{L}^{k+1}; y) \\
= \nabla z_{L+1} \phi(a_{L-1}^{k+1}, W_{L}^{k+1}, z_{L+1}^{k+1}) + \partial R(z_{L}^{k+1}; y) + \nabla z_{L+1} \phi(a_{L-1}^{k+1}, W_{L}^{k+1}, z_{L+1}^{k+1}) + \rho(z_{L}^{k+1} - z_{L}^{k+1}) \\
+ \rho(z_{L}^{k+1} - z_{L}^{k+1}) - \nabla z_{L+1} \phi(a_{L-1}^{k+1}, W_{L}^{k+1}, z_{L+1}^{k+1}) - \rho(z_{L}^{k+1} - z_{L}^{k+1}) \\
= \nabla z_{L+1} \phi(a_{L-1}^{k+1}, W_{L}^{k+1}, z_{L+1}^{k+1}) - \nabla z_{L+1} \phi(a_{L-1}^{k+1}, W_{L}^{k+1}, z_{L+1}^{k+1}) - \rho(z_{L}^{k+1} - z_{L}^{k+1}) \\
(0 \in \partial R(z_{L}^{k+1}; y) + \nabla z_{L+1} \phi(a_{L-1}^{k+1}, W_{L}^{k+1}, z_{L+1}^{k+1}) + \rho(z_{L}^{k+1} - z_{L}^{k+1}) \\
\text{by the optimality condition of Equation (3.5))} \\
= 0.
\]

Therefore, there exists \( g_{3,l}^{k+1} = \nabla_{z_l} F \) such that \( \| g_{3,l}^{k+1} \| = 0 \). This shows that there exists \( g_{3}^{k+1} = g_{3,1}^{k+1} \times g_{3,2}^{k+1} \times \cdots \times g_{3,L}^{k+1} = \nabla_{w} F \) such that

\[ \| g_{3}^{k+1} \| = 0. \] (B.12)
For $a_i^{k+1}$, we have

$$
\partial_{a_i^{k+1}} F = \nabla_{a_i^{k+1}} \phi(a_i^{k+1}, W_{l+1}^k, z_{l+1}^{k+1})
$$

$$
= \nabla_{a_i^{k+1}} \phi(a_i^{k+1}, W_{l+1}^k, z_{l+1}^{k+1}) - \nabla_{\bar{a}_i^{k+1}} \phi(\bar{a}_i^{k+1}, W_{l+1}^k, z_{l+1}^{k+1}) - \tau_l^{k+1} (a_i^{k+1} - \bar{a}_i^{k+1})
$$

(Equation (B.11))

$$
= \rho(W_{l+1}^{k+1})^T (W_{l+1}^{k+1} a_i^{k+1} - z_{l+1}^{k+1}) - \rho(W_{l+1}^{k+1})^T (W_{l+1}^{k+1} \bar{a}_i^{k+1} - z_{l+1})
$$

$$
- \tau_l^{k+1} (a_i^{k+1} - \bar{a}_i^{k+1})
$$

$$
= \rho(W_{l+1}^{k+1})^T (W_{l+1}^{k+1} a_i^{k+1} - \bar{a}_i^{k+1}) + \rho(W_{l+1}^{k+1})^T (W_{l+1}^{k+1} - W_{l+1}^k) \bar{a}_i^{k+1}
$$

$$
+ \rho(W_{l+1}^{k+1} - W_{l+1}^k)^T W_{l+1}^k \bar{a}_i^{k+1} - \rho(W_{l+1}^{k+1})^T (z_{l+1}^{k+1} - z_{l+1}^k) - \rho(W_{l+1}^{k+1} - W_{l+1}^k)^T z_{l+1}^k
$$

$$
- \tau_l^{k+1} (a_i^{k+1} - \bar{a}_i^{k+1}).
$$

Therefore

$$
\| \partial_{a_i^{k+1}} F \| \leq \rho \| W_{l+1}^{k+1} \| \| W_{l+1}^{k+1} \| \| a_i^{k+1} - \bar{a}_i^{k+1} \| + \rho \| W_{l+1}^{k+1} \| \| W_{l+1}^{k+1} - W_{l+1}^k \| \| \bar{a}_i^{k+1} \|
$$

$$
+ \rho \| W_{l+1}^{k+1} - W_{l+1}^k \| \| W_{l+1}^{k+1} \| \| \bar{a}_i^{k+1} \| + \rho \| W_{l+1}^{k+1} \| \| z_{l+1}^{k+1} - z_{l+1}^k \|
$$

$$
+ \rho \| W_{l+1}^{k+1} - W_{l+1}^k \| \| z_{l+1}^{k+1} \| + \tau_l^{k+1} \| a_i^{k+1} - \bar{a}_i^{k+1} \|
$$

(Triangle Inequality and Cauhtry-Schwarz Inequality)

$$
\leq \rho M_W \| a_i^{k+1} - \bar{a}_i^{k+1} \|
$$

$$
+ \rho M_W \| W_{l+1}^{k+1} - W_{l+1}^k \| M_a + \rho \| W_{l+1}^{k+1} - W_{l+1}^k \| M_W M_a + \rho M_W \| z_{l+1}^{k+1} - z_{l+1}^k \|
$$

$$
+ \rho \| W_{l+1}^{k+1} - W_{l+1}^k \| M_z + \tau_l^{k+1} \| a_i^{k+1} - \bar{a}_i^{k+1} \| \quad \text{(Lemma 3)}
$$

$$
= (\rho M_W^2 + \tau_l^{k+1}) \| a_i^{k+1} - \bar{a}_i^{k+1} \| + (2\rho M_W M_a + \rho M_z) \| W_{l+1}^{k+1} - W_{l+1}^k \|
$$

$$
+ \rho M_W \| z_{l+1}^{k+1} - z_{l+1}^k \|.
$$

According to Line 22 of Algorithm 4, if
Therefore, there exists $g_{4,i}^{k+1} \in \partial_{a_i}^{k+1} F$ such that

$$\|g_{4,i}^{k+1}\| \leq (\rho M_W^2 + \tau_i^{k+1})\|a_i^{k+1} - a_i^{k}\| + (\rho M_W^2 + \tau_i^{k+1})\|a_i^{k} - a_i^{k-1}\| + (2\rho M_W M_a + \rho M_z)\|w_{l+1}^{k+1} - w_{l+1}^k\| + \rho M_W\|z_{l+1}^{k+1} - z_{l+1}^k\|. \quad (B.13)$$

Otherwise,

$$\|\partial_{a_i}^{k+1} F\| \leq (\rho M_W^2 + \tau_i^{k+1})\|a_i^{k+1} - a_i^{k}\| + (2\rho M_W M_a + \rho M_z)\|w_{l+1}^{k+1} - w_{l+1}^k\| + \rho M_W\|z_{l+1}^{k+1} - z_{l+1}^k\| (\bar{a}_i^{k+1} = a_i^{k}).$$

Therefore, there exists $g_{4,i}^{k+1} \in \partial_{a_i}^{k+1} F$ such that

$$\|g_{4,i}^{k+1}\| \leq (\rho M_W^2 + \tau_i^{k+1})\|a_i^{k+1} - a_i^{k}\| + (2\rho M_W M_a + \rho M_z)\|w_{l+1}^{k+1} - w_{l+1}^k\| + \rho M_W\|z_{l+1}^{k+1} - z_{l+1}^k\|. \quad (B.14)$$

Combining Equation (B.13) and Equation (B.14), we show that there exists $g_i^{k+1} = g_{4,1}^{k+1} \times g_{4,2}^{k+1} \times \cdots \times g_{4,L}^{k+1} \in \partial_{a}^{k+1} F$ and $C_3 = \max(\rho M_W^2 + \tau_1^{k+1}, \rho M_W^2 + \tau_2^{k+1}, \rho M_W^2 + \tau_3^{k+1}, \cdots, \rho M_W^2 + \tau_L^{k+1})$. 

Therefore, there exists $g_i^{k+1} \in \partial_{a_i}^{k+1} F$ such that

$$\|g_i^{k+1}\| \leq (\rho M_W^2 + \tau_i^{k+1})\|a_i^{k+1} - a_i^{k}\| + (2\rho M_W M_a + \rho M_z)\|w_{l+1}^{k+1} - w_{l+1}^k\| + \rho M_W\|z_{l+1}^{k+1} - z_{l+1}^k\|.$$
B.3. MAIN PROOFS

\[
\tau_{3}^{k+1}, \cdots, \rho M_{W}^{2} + \tau_{L-1}^{k+1}, 2 \rho M_{W}M_{a} + \rho M_{z}) \text{ such that }
\]

\[
\|g_{k+1}^{4}\| \leq C_{3}(\|a_{k+1}^{k+1} - a_{k}^{k}\| + \|a_{k}^{k} - a_{k-1}^{k}\| + \|W_{k+1}^{k} - W_{k}^{k}\| + \|z_{k+1}^{k} - z_{k}^{k}\|). \tag{B.15}
\]

Combining Lemma 4, Equation (B.12) and Equation (B.15), we prove that there exists \(g_{k+1}^{k+1} \in \partial F(W_{k+1}^{k}, z_{k}^{k}, a_{k}^{k})) = \{\partial W_{k+1}^{k}F, \partial z_{k+1}^{k}F, \partial a_{k+1}^{k}F\} \) and \(C_{4} = \max(C_{2}, C_{3}, \rho)\) such that

\[
\|g_{k+1}^{k+1}\| \leq C_{4}(\|a_{k+1}^{k} - a_{k}^{k}\| + \|a_{k}^{k} - a_{k-1}^{k}\| + \|W_{k+1}^{k} - W_{k}^{k}\|
\]

\[
+ \|W_{k}^{k} - W_{k-1}^{k}\| + \|z_{k+1}^{k} - z_{k}^{k}\|). \tag{B.16}
\]

Finally, we prove the linear convergence rate by the KL Property given Equation (B.16) and Equation (B.9). Because \(F\) is locally strongly convex with a constant \(\mu\), \(F\) satisfies the KL Property by Lemma 16. Let \(F^{*} = F(W^{*}, z^{*}, a^{*})\) be the convergent value of \(F\), by Lemma 2, \(F(W^{k}, z^{k}, a^{k}) \to F^{*}\), then for any \(\eta_{1} > 0\) there exists \(k_{2} \in \mathbb{N}\) such that it holds for \(k > k_{2}\) that \(F^{*} < F(W^{k}, z^{k}, a^{k}) < F^{*} + \eta_{1}\). Also by Lemma 3(a) and Equation (B.16), \(g_{k+1}^{k+1} \to 0\) as \(k \to \infty\), then for any \(\eta_{2} > 0\) there exists \(k_{3} \in \mathbb{N}\), such that it holds for \(k > k_{3}\) that \(\|g_{k+1}^{k+1}\| < \eta_{2}\). Therefore, for any \(k > k_{1} = \max(k_{2}, k_{3})\), \((W^{k}, z^{k}, a^{k}) \in \{W, z, a) : |F^{*} < F(W, z, a) < F^{*} + \eta_{1} \cap \exists g \in
\]

\[
(B.16)
\]

\[
\|g_{k+1}^{k+1}\| \leq C_{4}(\|a_{k+1}^{k} - a_{k}^{k}\| + \|a_{k}^{k} - a_{k-1}^{k}\| + \|W_{k+1}^{k} - W_{k}^{k}\|
\]

\[
+ \|W_{k}^{k} - W_{k-1}^{k}\| + \|z_{k+1}^{k} - z_{k}^{k}\|). \tag{B.16}
\]

Finally, we prove the linear convergence rate by the KL Property given Equation (B.16) and Equation (B.9). Because \(F\) is locally strongly convex with a constant \(\mu\), \(F\) satisfies the KL Property by Lemma 16. Let \(F^{*} = F(W^{*}, z^{*}, a^{*})\) be the convergent value of \(F\), by Lemma 2, \(F(W^{k}, z^{k}, a^{k}) \to F^{*}\), then for any \(\eta_{1} > 0\) there exists \(k_{2} \in \mathbb{N}\) such that it holds for \(k > k_{2}\) that \(F^{*} < F(W^{k}, z^{k}, a^{k}) < F^{*} + \eta_{1}\). Also by Lemma 3(a) and Equation (B.16), \(g_{k+1}^{k+1} \to 0\) as \(k \to \infty\), then for any \(\eta_{2} > 0\) there exists \(k_{3} \in \mathbb{N}\), such that it holds for \(k > k_{3}\) that \(\|g_{k+1}^{k+1}\| < \eta_{2}\). Therefore, for any \(k > k_{1} = \max(k_{2}, k_{3})\), \((W^{k}, z^{k}, a^{k}) \in \{W, z, a) : |F^{*} < F(W, z, a) < F^{*} + \eta_{1} \cap \exists g \in
\]

\[
(B.16)
\]
B.3. MAIN PROOFS

157

\[ F(W, z, a) \text{ s.t. } \|g\| < \eta_2 \}. \] By the KL Property and Lemma 16, it holds that

\[
1 \leq \|g^{k+1}\|/(\mu \sqrt{F(W^{k+1}, z^{k+1}, a^{k+1}) - F^*})
\]
\[
\leq C_4(||a^{k+1} - a^k|| + ||a^k - a^{k-1}|| + \|W^{k+1} - W^k\| + \|W^k - W^{k-1}\| + \|z^{k+1} - z^k\|)
\]
\[ / (\mu \sqrt{F(W^{k+1}, z^{k+1}, a^{k+1}) - F^*}) \] (Equation (B.16))
\[
\leq C_4^2(||a^{k+1} - a^k|| + ||a^k - a^{k-1}|| + \|W^{k+1} - W^k\| + \|W^k - W^{k-1}\| + \|z^{k+1} - z^k\|)^2
\]
\[ / (\mu^2(F(W^{k+1}, z^{k+1}, a^{k+1}) - F^*)) \]
\[
\leq (5C_4^2(||a^{k+1} - a^k||^2 + ||a^k - a^{k-1}||^2 + \|W^{k+1} - W^k\|^2) + \|W^k - W^{k-1}\|^2)
\]
\[ + \|z^{k+1} - z^k\|^2) / (\mu^2(F(W^{k+1}, z^{k+1}, a^{k+1}) - F^*)) \] (Mean Inequality)
\[
\leq (5C_4^2(F(W^{k-1}, z^{k-1}, a^{k-1}) - F(W^{k+1}, z^{k+1}, a^{k+1})))
\]
\[ / (C_5 \mu^2(F(W^{k+1}, z^{k+1}, a^{k+1}) - F^*)) \] (Equation (B.9)).

This indicates that

\[
(C_5 \mu^2 + 5C_4^2)(F(W^{k+1}, z^{k+1}, a^{k+1}) - F^*) \leq 5C_4^2(F(W^{k-1}, z^{k-1}, a^{k-1}) - F^*).
\]

Let \(0 < C_1 = \frac{5C_4^2}{C_5 \mu^2 + 5C_4^2} < 1\), we have

\[
F(W^{k+1}, z^{k+1}, a^{k+1}) - F^* \leq C_1(F(W^{k-1}, z^{k-1}, a^{k-1}) - F^*).
\]

So in summary, for any \(\rho\), there exist \(\varepsilon = \max(D_1, D_2)\), \(k_1 = \max(k_2, k_3)\), and

\[
0 < C_1 = \frac{5C_4^2}{C_5 \mu^2 + 5C_4^2} < 1 \text{ such that}
\]
\[
F(W^{k+1}, z^{k+1}, a^{k+1}) - F^* \leq C_1(F(W^{k-1}, z^{k-1}, a^{k-1}) - F^*).
\]

for \(k > k_1\). In other words, the linear convergence rate is proven. \(\square\)
Appendix C

Appendix of the pdADMM Algorithm

C.1 Preliminary Results

Lemma 19. It holds for every $k \in \mathbb{N}$ and $l = 1, \cdots, L - 1$ that

$$u^k_l = \nu(q^k_l - f_l(z^k_l))$$

Proof. This follows directly from the optimality condition of $q^k_l$ and Equation (4.6).

Lemma 20. It holds for every $k \in \mathbb{N}$ and $l = 1, \cdots, L - 1$ that

$$\|u^{k+1}_l - u^k_l\| \leq \nu\|q^{k+1}_l - q^k_l\| + \nu S\|z^{k+1}_l - z^k_l\|$$
Proof.

\[ \|u_{t}^{k+1} - u_{t}^{k}\| \]
\[ = \|\nu(q_{t}^{k+1} - f_{t}(z_{t}^{k+1})) - \nu(q_{t}^{k} - f_{t}(z_{t}^{k}))\| \quad \text{(Lemma 19)} \]
\[ \leq \nu\|q_{t}^{k+1} - q_{t}^{k}\| + \nu\|f_{t}(z_{t}^{k+1}) - f_{t}(z_{t}^{k})\| \quad \text{(Triangle Inequality)} \]
\[ \leq \nu\|q_{t}^{k+1} - q_{t}^{k}\| + \nu S\|z_{t}^{k+1} - z_{t}^{k}\| \quad \text{(Assumption 5)} \]

\[ \square \]

**Lemma 21.** It holds for every \( k \in \mathbb{N} \) and \( l = 1, \cdots, L - 1 \) that

\[ \|u_{t}^{k+1} - u_{t}^{k}\|_{2}^{2} \leq 2\nu^{2}(\|q_{t}^{k+1} - q_{t}^{k}\|_{2}^{2} + S^{2}\|z_{t}^{k+1} - z_{t}^{k}\|_{2}^{2}) \]

Proof.

\[ \|u_{t}^{k+1} - u_{t}^{k}\|_{2}^{2} = \nu^{2}\|q_{t}^{k+1} - f_{t}(z_{t}^{k+1}) - q_{t}^{k} + f_{t}(z_{t}^{k})\|_{2}^{2} \quad \text{(Lemma 19)} \]
\[ \leq 2\nu^{2}(\|q_{t}^{k+1} - q_{t}^{k}\|_{2}^{2} + \|f_{t}(z_{t}^{k+1}) - f_{t}(z_{t}^{k})\|_{2}^{2}) \quad \text{(Mean Inequality)} \]
\[ \leq 2\nu^{2}(\|q_{t}^{k+1} - q_{t}^{k}\|_{2}^{2} + S^{2}\|z_{t}^{k+1} - z_{t}^{k}\|_{2}^{2}) \quad \text{(Assumption 5)} \]

\[ \square \]

**Lemma 22.** For every \( k \in \mathbb{N} \), it holds that

\[ L_{\rho}(p^{k}, W^{k}, z^{k}, q^{k}, u^{k}) - L_{\rho}(p^{k+1}, W^{k}, z^{k}, q^{k}, u^{k}) \geq \sum_{l=2}^{L}(\tau_{l}^{k+1}/2)\|p_{t}^{k+1} - p_{t}^{k}\|_{2}^{2} \quad \text{(C.1)} \]
\[ L_{\rho}(p^{k+1}, W^{k}, z^{k}, q^{k}, u^{k}) - L_{\rho}(p^{k+1}, W^{k+1}, z^{k}, q^{k}, u^{k}) \geq \sum_{l=1}^{L}(\theta_{l}^{k+1}/2)\|W_{l}^{k+1} - W_{l}^{k}\|_{2}^{2} \quad \text{(C.2)} \]
\[ L_{\rho}(p^{k+1}, W^{k+1}, z^{k}, q^{k}, u^{k}) - L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k}, u^{k}) \geq (\nu/2)\sum_{l=1}^{L}\|z_{l}^{k+1} - z_{l}^{k}\|_{2}^{2} \quad \text{(C.3)} \]
C.1. PRELIMINARY RESULTS

Proof. Generally, all inequalities can be obtained by applying optimality conditions of updating $p$, $W$, and $z$, respectively. We only prove Inequalities (C.1), and (C.3). This is because Inequality (C.2) follows the same routine of Inequality (C.1).

Firstly, we focus on Inequality (C.1). The choice of $\tau_{k+1}^{l_1}$ requires

$$\phi(p_{i_{l_1}}^{k+1}, W_{i_{l_1}}^{k}, z_{i_{l_1}}^k, q_{i_{l_1}-1}^k, u_{i_{l_1}-1}^k) \leq U_i(p_{i_{l_1}}^{k+1}, \tau_{i_{l_1}}^{k+1}) \quad (C.4)$$

Moreover, the optimality condition of Equation (4.1) leads to

$$\nabla_{p_i^k} \phi(p_{i_{l_1}}^k, W_{i_{l_1}}^k, z_{i_{l_1}}^k, q_{i_{l_1}-1}^k, u_{i_{l_1}-1}^k) + \tau_{i_{l_1}}^{k+1}(p_{i_{l_1}}^{k+1} - p_{i_{l_1}}^k) = 0 \quad (C.5)$$

Therefore

$$L_{\mu}(p^k, W^k, z^k, q^k, u^k) - L_{\mu}(p^{k+1}, W^k, z^k, q^k, u^k)$$

$$= \sum_{l_1=2}^L (\phi(p_{i_{l_1}}^k, W_{i_{l_1}}^k, z_{i_{l_1}}^k, q_{i_{l_1}-1}^k, u_{i_{l_1}-1}^k) - \phi(p_{i_{l_1}}^{k+1}, W_{i_{l_1}}^k, z_{i_{l_1}}^k, q_{i_{l_1}-1}^k, u_{i_{l_1}-1}^k))$$

$$\geq \sum_{l_1=2}^L (\phi(p_{i_{l_1}}^k, W_{i_{l_1}}^k, z_{i_{l_1}}^k, q_{i_{l_1}-1}^k, u_{i_{l_1}-1}^k) - U_i(p_{i_{l_1}}^{k+1}, \tau_{i_{l_1}}^{k+1}))(\text{Inequality (C.4)})$$

$$= \sum_{l_1=2}^L (-\nabla_{p_i^k} \phi^T(p_{i_{l_1}}^k, W_{i_{l_1}}^k, z_{i_{l_1}}^k, q_{i_{l_1}-1}^k, u_{i_{l_1}-1}^k)(p_{i_{l_1}}^{k+1} - p_i^k) - (\tau_{i_{l_1}}^{k+1}/2)||p_{i_{l_1}}^{k+1} - p_i^k||_2^2)$$

$$= \sum_{l_1=2}^L (\tau_{i_{l_1}}^{k+1}/2)||p_{i_{l_1}}^{k+1} - p_i^k||_2^2 \quad (\text{Equation (C.5)})$$

Next, we prove Inequality (C.3). Because $z_{i_{l_1}}^{k+1}$ minimizes Equation (4.3), we have

$$\leq \frac{\nu}{2}||z_{i_{l_1}}^{k+1} - W_{i_{l_1}}^{k+1}p_i^k||_2^2 + \frac{\nu}{2}||q_i^k - f_i(z_{i_{l_1}}^{k+1})||_2^2 + \frac{\nu}{2}||z_{i_{l_1}}^{k+1} - z_i^k||_2^2$$

$$\leq \frac{\nu}{2}||z_{i_{l_1}}^k - W_{i_{l_1}}^{k+1}p_i^k||_2^2 + \frac{\nu}{2}||q_i^k - f_i(z_i^k)||_2^2 \quad (C.6)$$
C.1. PRELIMINARY RESULTS

And

\[ R(z_L^k; y) + \left( \frac{\nu}{2} \right) \| z^k_L - W_L^{k+1} p_L^{k+1} \|^2_2 - R(z_L^{k+1}; y) \]
\[ = R(z_L^k; y) - R(z_L^{k+1}; y) + \left( \frac{\nu}{2} \right) \| z^k_L - z_L^{k+1} \|^2_2 + \nu (z_L^{k+1} - W_L^{k+1} p_L^{k+1})^T (z_L^k - z_L^{k+1}) \]
\[ (\| a - b \|^2_2 - \| a - c \|^2_2) = \| b - c \|^2_2 + 2(c - a)^T (b - c) \]

where

\[ a = W_L^{k+1} p_L^{k+1}, \quad b = z_L^k, \quad \text{and} \quad c = z_L^{k+1} \] (C.7)

\[ \geq s^T (z_L^k - z_L^{k+1}) + \left( \frac{\nu}{2} \right) \| z_L^k - z_L^{k+1} \|^2_2 + \nu (z_L^{k+1} - W_L^{k+1} p_L^{k+1})^T (z_L^k - z_L^{k+1}) \]

\[ (s \in \partial R(z_L^{k+1}; y) \text{ is a subgradient of } R(z_L^{k+1}; y)) \]
\[ = \left( \frac{\nu}{2} \right) \| z_L^{k+1} - z_L^k \|^2_2 \] (C.8)

\[ (0 \in s + \nu (z_L^{k+1} - W_L^{k+1} p_L^{k+1}) \text{ by the optimality condition of Equation (4.4))} \]

Therefore

\[ L_\rho(p^{k+1}, W^{k+1}, z^k, q^k, u^k) - L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^k, u^k) \]
\[ = \sum_{i=1}^{L-1} \left( \left( \frac{\nu}{2} \right) \| z_i^k - W_i^{k+1} p_i^{k+1} \|^2_2 \right) \]
\[ \left( \frac{\nu}{2} \right) \| z_i^{k+1} - W_i^{k+1} p_i^{k+1} \|^2_2 - \left( \frac{\nu}{2} \right) \| q_i^k - f_i(z_i^k) \|^2_2 \]
\[ + R(z_L^k; y) + \left( \frac{\nu}{2} \right) \| z_L^k - W_L^{k+1} p_L^{k+1} \|^2_2 - R(z_L^{k+1}; y) \]
\[ \geq \left( \frac{\nu}{2} \right) \sum_{i=1}^{L} \| z_i^{k+1} - z_i^k \|^2_2 \quad \text{(Inequalities (C.6) and (C.8))} \]

Lemma 23. For every \( k \in N \), it holds that

\[ L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^k, u^k) - L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) \]
\[ \geq \sum_{i=1}^{L-1} \left( \left( \frac{\rho}{2} - 2\nu^2 / \rho - \nu / 2 \right) \| q_i^{k+1} - q_i^k \|^2_2 - (2\nu^2 S^2 / \rho) \| z_i^{k+1} - z_i^k \|^2_2 \right) \] (C.9)
Proof.

\[ L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^k, u^k) - L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) \]
\[ = \sum_{l=1}^{L-1} \left( (\nu/2)\|f_l(z^{k+1}) - q_l^k\|_2^2 - (\nu/2)\|f_l(z^{k+1}) - q_l^{k+1}\|_2^2 - (u_l^{k+1})^T(q_l^k - q_l^{k+1}) \right) \]
\[ + (\rho/2)\|q_l^{k+1} - q_l^k\|_2^2 - (1/\rho)\|u_l^{k+1} - u_l^k\|_2^2 \]
\[ = \sum_{l=1}^{L-1} \left( (\nu/2)\|f_l(z^{k+1}) - q_l^k\|_2^2 - (\nu/2)\|f_l(z^{k+1}) - q_l^{k+1}\|_2^2 \right. \]
\[ - \nu(q_l^{k+1} - f_l(z_l^{k+1}))^T(q_l^k - q_l^{k+1}) + (\rho/2)\|q_l^{k+1} - q_l^k\|_2^2 - (1/\rho)\|u_l^{k+1} - u_l^k\|_2^2 \]

(Lemma 19)
\[ \geq \sum_{l=1}^{L-1} \left( -\nu(\nu/2)\|q_l^{k+1} - q_l^k\|_2^2 - (\rho/2)\|q_l^{k+1} - q_l^k\|_2^2 - (1/\rho)\|u_l^{k+1} - u_l^k\|_2^2 \right) \]
\[ - \nu(\nu/2)\nabla_q l \|q_l - f_l(z_l^{k+1})\|_2^2 \text{ is lipschitz continuous} \]

with regard to \( q_l \) and Lemma 2.1 in [2])
\[ \geq \sum_{l=1}^{L-1} \left( -\nu(\nu/2)\|q_l^{k+1} - q_l^k\|_2^2 + (\rho/2)\|q_l^{k+1} - q_l^k\|_2^2 - (2\nu^2/\rho)\|q_l^{k+1} - q_l^k\|_2^2 \right. \]
\[ - (2\nu^2 S^2/\rho)\|z_l^{k+1} - z_l^{k}\|_2^2 \text{ (Lemma 21)} \]
\[ = \sum_{l=1}^{L-1} \left( (\rho/2 - 2\nu^2/\rho - \nu/2)\|q_l^{k+1} - q_l^k\|_2^2 - (2\nu^2 S^2/\rho)\|z_l^{k+1} - z_l^{k}\|_2^2 \right) \]

\[ \square \]

C.2 Main Proofs

Proof of Lemma 5

Proof. We sum up Inequalities (C.1), (C.2), (C.3), and (C.9) to obtain Inequality (4.7).

Proof of Lemma 6
Proof. There exists \( \mathbf{q}' \) such that \( p_{t+1}^k = q_t' \) and

\[
F(p^k, W^k, z^k, q') \geq \min_{p, W, z, q} \{ F(p, W, z, q) | p_{t+1} = q_t \} > -\infty
\]

Therefore, we have

\[
L_\rho(p^k, W^k, z^k, q^k, u^k) = F(p^k, W^k, z^k, q^k) + \sum_{l=1}^L (u_l^k)^T (p_{l+1}^k - q_{l+1}^k) + (\rho/2)\|p_{l+1}^k - q_{l+1}^k\|^2
\]

\[
= R(z_L^k; y) + (\nu/2)(\sum_{l=1}^L \|z_l^k - W_l^k p_l^k\|^2 + \sum_{l=1}^{L-1} \|q_{l+1}^k - f_l(z_l^k)\|^2) + (\rho/2)\|p_{l+1}^k - q_{l+1}^k\|^2
\]

\[
\geq R(z_L^k; y) + (\nu/2)(\sum_{l=1}^L \|z_l^k - W_l^k p_l^k\|^2 + \sum_{l=1}^{L-1} \|q_{l+1}^k - f_l(z_l^k)\|^2) - \sum_{l=1}^{L-1} (\nu/2)\|q_{l+1}^k - q_l^k\|^2 + \sum_{l=1}^{L-1} (\rho/2)\|p_{l+1}^k - q_{l+1}^k\|^2
\]

\[
(\nu(q_t - f_l(z_{l+1}^k))) = (\nu/2)\|q_t - f_l(z_{l+1}^k)\|^2
\]

is lipschitz continuous with regard to \( q_t \) and Lemma 2.1 in [2]

\[
= F(p^k, W^k, z^k, q') + (\rho - \nu)/2\|p_{l+1}^k - q_{l+1}^k\|^2 > -\infty
\]

Therefore, \( F(p^k, W^k, z^k, q') \) and \( (\rho - \nu)/2\|p_{l+1}^k - q_{l+1}^k\|^2 \) are upper bounded by \( L_\rho(p^k, W^k, z^k, q^k, u^k) \) and hence \( L_\rho(p^0, W^0, z^0, q^0, u^0) \) (Lemma 5). From Assumption 5, \((p^k, W^k, z^k)\) is bounded. \( q^k \) is also bounded because \( (\rho - \nu)/2\|p_{l+1}^k - q_{l+1}^k\|^2 \) is upper bounded. \( u^k \) is bounded because of Lemma 19. \( \square \)

Proof of Lemma 7

Proof. We know that \( \partial L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) \)

\[
= \{ \nabla_{p^{k+1}} L_\rho, \nabla_{W^{k+1}} L_\rho, \partial_{z^{k+1}} L_\rho, \nabla_{q^{k+1}} L_\rho, \nabla_{u^{k+1}} L_\rho \} [101]. \text{ Specifically, we prove that}
\]
\[ \|g\| \text{ is upper bounded by the linear combination of } \|p^{k+1} - p^k\|, \|W^{k+1} - W^k\|, \|z^{k+1} - z^k\|, \|q^{k+1} - q^k\|, \text{ and } \|u^{k+1} - u^k\|. \]

For \( p_{l_i}^{k+1} \),

\[
\nabla_{p_{l_i}^{k+1}} L_{p_l}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})
= \nabla_{p_{l_i}^{k+1}} \phi(p_{l_i}^{k+1}, W_{l_i}^{k+1}, z_{l_i}^{k+1}, q_{l_i-1}^{k+1}, u_{l_i-1}^{k+1}) \\
= \nabla_{p_{l_i}^{k}} \phi(p_{l_i}^{k}, W_{l_i}^{k}, z_{l_i}^{k}, q_{l_i-1}^{k}, u_{l_i-1}^{k}) \quad \text{ with } \tau_{l_i}^{k+1}(p_{l_i}^{k+1} - p_{l_i}^{k}) - \tau_{l_i}^{k}(p_{l_i}^{k+1} - p_{l_i}^{k}) \\
+ \nu(W_{l_i}^{k+1})^T W_{l_i}^{k+1} p_{l_i}^{k+1} - \nu(W_{l_i}^{k})^T W_{l_i}^{k} p_{l_i}^{k} - \nu(W_{l_i}^{k+1})^T z_{l_i}^{k+1} + \nu(W_{l_i}^{k})^T z_{l_i}^{k} \\
+ (u_{l_i-1}^{k+1} - u_{l_i-1}^{k}) + \rho(p_{l_i}^{k+1} - p_{l_i}^{k}) - \rho(q_{l_i-1}^{k+1} - q_{l_i-1}^{k}) \\
= -\tau_{l_i}^{k+1}(p_{l_i}^{k+1} - p_{l_i}^{k}) + \nu(W_{l_i}^{k+1})^T W_{l_i}^{k+1} p_{l_i}^{k+1} - \nu(W_{l_i}^{k})^T W_{l_i}^{k} p_{l_i}^{k} \\
- \nu(W_{l_i}^{k+1})^T z_{l_i}^{k+1} + \nu(W_{l_i}^{k})^T z_{l_i}^{k} + (u_{l_i-1}^{k+1} - u_{l_i-1}^{k}) + \rho(p_{l_i}^{k+1} - p_{l_i}^{k}) - \rho(q_{l_i-1}^{k+1} - q_{l_i-1}^{k})
\]

(The optimality condition of Equation (4.1))
\[
\| \nabla_p^{k+1} L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) \| \\
= \| \tau_i^{k+1}(p_i^{k+1} - p_i^k) + \nu(W_i^{k+1})^T W_i^{k+1} p_i^{k+1} - \nu(W_i^k)^T W_i^k p_i^k \\
- \nu(W_i^{k+1})^T z_i^{k+1} + \nu(W_i^k)^T z_i^k + (u_{i-1}^k - u_i^k) + \rho(p_i^{k+1} - p_i^k) - \rho(q_i^{k+1} - q_i^k) \| \\
\leq \tau_i^{k+1} \| p_i^{k+1} - p_i^k \| + \nu \| (W_i^{k+1})^T W_i^{k+1} p_i^{k+1} - (W_i^k)^T W_i^k p_i^k \| \\
+ \nu \| (W_i^{k+1})^T z_i^{k+1} - (W_i^k)^T z_i^k \| + \| u_{i-1}^k - u_i^k \| + \rho \| p_i^{k+1} - p_i^k \| + \rho \| q_i^{k+1} - q_i^k \| \\
\text{(Triangle Inequality)} \\
\leq \tau_i^{k+1} \| p_i^{k+1} - p_i^k \| + \nu \| W_i^{k+1} \|^2 \| p_i^{k+1} - p_i^k \| + \nu \| W_i^{k+1} \| \| W_i^{k+1} - W_i^k \| \| p_i^k \| \\
+ \nu \| W_i^{k+1} - W_i^k \| \| W_i^k \| \| p_i^k \| + \nu \| W_i^{k+1} \| \| z_i^{k+1} - z_i^k \| + \nu \| W_i^{k+1} - W_i^k \| \| z_i^k \| \\
+ \nu \| q_i^{k+1} - q_i^k \| + S \| z_i^{k+1} - z_i^k \| + \rho \| p_i^{k+1} - p_i^k \| + \rho \| q_i^{k+1} - q_i^k \| \\
\text{(Triangle Inequality, Cauhthy-Schwartz Inequality and Lemma 20)} \\
\leq \tau_i^{k+1} \| p_i^{k+1} - p_i^k \| + \nu N_{W}^2 \| p_i^{k+1} - p_i^k \| + 2\nu N_{W} N_{p} \| W_i^{k+1} - W_i^k \| \\
+ \nu N_{W} \| z_i^{k+1} - z_i^k \| + \nu N_{z} \| W_i^{k+1} - W_i^k \| \\
+ 2\nu^2 (\| q_i^{k+1} - q_i^k \|^2 + S^2 \| z_i^{k+1} - z_i^k \|^2) + \rho \| p_i^{k+1} - p_i^k \| + \rho \| q_i^{k+1} - q_i^k \| \\
\text{(Lemma 6)}
For $W_1^{k+1}$,

$$\nabla_{W_1^{k+1}} L_\rho(p_1^{k+1}, W_1^{k+1}, z_1^{k+1}, q_1^{k+1}, u_1^{k+1})$$

$$= \nabla_{W_1^{k+1}} \phi(p_1^{k+1}, W_1^{k+1}, z_1^{k+1})$$

$$= \nabla_{W_1^{k}} \phi(p_1^{k+1}, W_1^{k}, z_1^{k}) + \theta_1^{k+1}(W_1^{k+1} - W_1^{k}) + \nu(W_1^{k+1} - W_1^{k}) p_1^{k+1} (p_1^{k+1})^T$$

$$- \nu(z_1^{k+1} - z_1^{k})(p_1^{k+1})^T - \theta_1^{k+1}(W_1^{k+1} - W_1^{k})$$

$$= \nu(W_1^{k+1} - W_1^{k}) p_1^{k+1} (p_1^{k+1})^T - \nu(z_1^{k+1} - z_1^{k})(p_1^{k+1})^T - \theta_1^{k+1}(W_1^{k+1} - W_1^{k})$$

(The optimality condition of Equation (4.2))

So

$$\|\nabla_{W_1^{k+1}} L_\rho(p_1^{k+1}, W_1^{k+1}, z_1^{k+1}, q_1^{k+1}, u_1^{k+1})\|$$

$$= \|\nu(W_1^{k+1} - W_1^{k}) p_1^{k+1} (p_1^{k+1})^T - \nu(z_1^{k+1} - z_1^{k})(p_1^{k+1})^T - \theta_1^{k+1}(W_1^{k+1} - W_1^{k})\|$$

$$\leq \nu\|W_1^{k+1} - W_1^{k}\|\|p_1^{k+1}\|^2 + \nu\|z_1^{k+1} - z_1^{k}\|\|p_1^{k+1}\| + \theta_1^{k+1}\|W_1^{k+1} - W_1^{k}\|$$

(Triangle Inequality and Cauthy-Schwartz Inequality)

$$\leq \nu\|W_1^{k+1} - W_1^{k}\|N_p^2 + \nu\|z_1^{k+1} - z_1^{k}\|N_p + \theta_1^{k+1}\|W_1^{k+1} - W_1^{k}\| \quad \text{(Theorem 6)}$$

For $W_l^{k+1} (1 < l \leq L)$,

$$\nabla_{W_l^{k+1}} L_\rho(p_l^{k+1}, W_l^{k+1}, z_l^{k+1}, q_l^{k+1}, u_l^{k+1})$$

$$= \nabla_{W_l^{k+1}} \phi(p_l^{k+1}, W_l^{k+1}, z_l^{k+1}, p_l^{k-1}, u_l^{k+1})$$

$$= \nabla_{W_l^{k}} \phi(p_l^{k+1}, W_l^{k}, z_l^{k}, p_l^{k-1}, u_l^{k+1}) + \theta_l^{k+1}(W_l^{k+1} - W_l^{k}) + \nu(W_l^{k+1} - W_l^{k}) p_l^{k+1} (p_l^{k+1})^T$$

$$- \nu(z_l^{k+1} - z_l^{k})(p_l^{k+1})^T - \theta_l^{k+1}(W_l^{k+1} - W_l^{k})$$

$$= \nu(W_l^{k+1} - W_l^{k}) p_l^{k+1} (p_l^{k+1})^T - \nu(z_l^{k+1} - z_l^{k})(p_l^{k+1})^T - \theta_l^{k+1}(W_l^{k+1} - W_l^{k})$$

(The optimality condition of Equation (4.2))
So

\[\|\nabla_{\hat{W}^{k+1}_{l}}L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})\|\]

\[= \|\nu(W_{l}^{k+1} - W_{l}^{k})p_{l}^{k+1}(p_{l}^{k+1})^{T} - \nu(z_{l}^{k+1} - z_{l}^{k})(p_{l}^{k+1})^{T} - \theta_{l}^{k+1}(W_{l}^{k+1} - W_{l}^{k})\|\]

\[\leq \nu\|W_{l}^{k+1} - W_{l}^{k}\|\|p_{l}^{k+1}\|^{2} + \nu\|z_{l}^{k+1} - z_{l}^{k}\||p_{l}^{k+1}\| + \theta_{l}^{k+1}\|W_{l}^{k+1} - W_{l}^{k}\|\]

(Triangle Inequality and Cauchy-Schwartz Inequality)

\[\leq \nu\|W_{l}^{k+1} - W_{l}^{k}\|N_{p}^{2} + \nu\|z_{l}^{k+1} - z_{l}^{k}\|N_{p} + \theta_{l}^{k+1}\|W_{l}^{k+1} - W_{l}^{k}\| \quad \text{(Theorem 6)}\]

For \(z_{l}^{k+1}(l < L)\),

\[\partial_{z_{l}^{k+1}}L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})\]

\[= \partial_{z_{l}^{k+1}}L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k}, u^{k}) + \nu(z_{l}^{k+1} - z_{l}^{k}) - \nu(z_{l}^{k} - z_{l}^{k})\]

\[- \nu\partial f_{i}(z_{l}^{k+1}) \circ (q_{l}^{k+1} - q_{l}^{k}) \quad \circ \text{is Hadamard product}\]

\[= -\nu(z_{l}^{k+1} - z_{l}^{k}) - \nu\partial f_{i}(z_{l}^{k+1}) \circ (q_{l}^{k+1} - q_{l}^{k})\]

\((0 \in \partial_{z_{l}^{k+1}}L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k}, u^{k}) + \nu(z_{l}^{k+1} - z_{l}^{k}))\)

So

\[\|\partial_{z_{l}^{k+1}}L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})\|\]

\[= \| - \nu(z_{l}^{k+1} - z_{l}^{k}) - \nu\partial f_{i}(z_{l}^{k+1}) \circ (q_{l}^{k+1} - q_{l}^{k})\|\]

\[\leq \nu\|z_{l}^{k+1} - z_{l}^{k}\| + \nu\|\partial f_{i}(z_{l}^{k+1})\|\|q_{l}^{k+1} - q_{l}^{k}\|\]

(Cauchy-Schwartz Inequality and Triangle Inequality)

\[\leq \nu\|z_{l}^{k+1} - z_{l}^{k}\| + \nu M\|q_{l}^{k+1} - q_{l}^{k}\|(\|\partial f_{i}(z_{l}^{k+1})\| \leq M)\]

For \(z_{L}^{k+1}, \partial_{z_{L}^{k+1}}L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) = 0\) by the optimality condition of Equation (4.4).
C.2. MAIN PROOFS

For $q_k^{k+1}$,

$$
\nabla_{q_i^{k+1}} L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})
$$

$$
= \nabla_{q_i^{k+1}} L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^k) + u_i^{k+1} - u_i^k
$$

$$
= u_i^{k+1} - u_i^k (\nabla_{q_i^{k+1}} L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^k) = 0
$$

by the optimality condition of Equation (4.5))

So $\|\nabla_{q_i^{k+1}} L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})\| = \|u_i^{k+1} - u_i^k\|.$

For $u_i^{k+1}$,

$$
\nabla_{u_i^{k+1}} L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) = (p_i^{k+1} - q_i^{k+1}) = (u_i^{k+1} - u_i^k)/\rho
$$

So $\|\nabla_{u_i^{k+1}} L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})\| = \|u_i^{k+1} - u_i^k\|/\rho.$

In summary, we prove that $\nabla_{p^{k+1}} L_\rho, \nabla_{W^{k+1}} L_\rho, \nabla_{q^{k+1}} L_\rho, \nabla_{u^{k+1}} L_\rho$ are upper bounded by the linear combination of $\|p^{k+1} - p^k\|, \|W^{k+1} - W^k\|, \|z^{k+1} - z^k\|, \|q^{k+1} - q^k\|$, and $\|u^{k+1} - u^k\|$. □

Proof of Theorem 8

Proof. To prove this theorem, we will first show that $c_k$ satisfies two conditions: (1). $c_k \geq c_{k+1}$. (2). $\sum_{k=0}^{\infty} c_k$ is bounded. Specifically, first, we have

$$
c_k = \min_{0 \leq i \leq k} \left(\sum_{t=2}^{L} (\tau_{t-1}^{i+1}/2) ||p_{t-1}^{i+1} - p_t^{i+1}||_2^2 + \sum_{t=1}^{L} (\theta_{t-1}^{i+1}/2) ||W_{t}^{i+1} - W_{t}^{i+1}||_2^2
+ \sum_{t=1}^{L-1} C_1 ||z_{t}^{i+1} - z_{t}^{i}||_2^2 + (\nu/2) ||z_{L}^{i+1} - z_{L}^{i}||_2^2 + \sum_{t=1}^{L-1} C_2 ||q_{t}^{i+1} - q_{t}^{i}||_2^2\right)
\geq \min_{0 \leq i \leq k+1} \left(\sum_{t=2}^{L} (\tau_{t-1}^{i+1}/2) ||p_{t-1}^{i+1} - p_t^{i+1}||_2^2 + \sum_{t=1}^{L} (\theta_{t-1}^{i+1}/2) ||W_{t}^{i+1} - W_{t}^{i+1}||_2^2
+ \sum_{t=1}^{L-1} C_1 ||z_{t}^{i+1} - z_{t}^{i}||_2^2 + (\nu/2) ||z_{L}^{i+1} - z_{L}^{i}||_2^2 + \sum_{t=1}^{L-1} C_2 ||q_{t}^{i+1} - q_{t}^{i}||_2^2\right)
= c_{k+1}
$$
Therefore $c_k$ satisfies the first condition. Second,

$$
\sum_{k=0}^{\infty} c_k = \sum_{k=0}^{\infty} \min_{0 \leq i \leq k} (\sum_{l=2}^{L} (\tau_l^{i+1}/2)\|p_l^{i+1} - p_l^i\|_2^2 + \sum_{l=1}^{L} (\theta_l^{i+1}/2)\|W_l^{i+1} - W_l^i\|_2^2 \\
+ \sum_{l=1}^{L-1} C_1\|z_l^{i+1} - z_l^i\|_2^2 + (\nu/2)\|z_L^{i+1} - z_L^i\|_2^2 + \sum_{l=1}^{L-1} C_2\|q_l^{i+1} - q_l^i\|_2^2)
\leq \sum_{k=0}^{\infty} (\sum_{l=2}^{L} (\tau_l^{k+1}/2)\|p_l^{k+1} - p_l^k\|_2^2 + \sum_{l=1}^{L} (\theta_l^{k+1}/2)\|W_l^{k+1} - W_l^k\|_2^2 \\
+ \sum_{l=1}^{L-1} C_1\|z_l^{k+1} - z_l^k\|_2^2 + (\nu/2)\|z_L^{k+1} - z_L^k\|_2^2 + \sum_{l=1}^{L-1} C_2\|q_l^{k+1} - q_l^k\|_2^2)
\leq L_\rho(p^0, W^0, z^0, q^0, u^0) - L_\rho(p^*, W^*, z^*, q^*, u^*)
$$

(Lemma 5)

So $c_k$ satisfies the second condition. Finally, since we have proved the first two conditions and the third one $c_k \geq 0$ is obvious, the convergence rate of $o(1/k)$ is proven (Lemma 1.2 in [26]). \qed
Appendix D

Appendix of the pdADMM-G Algorithm

D.1 Convergence Proofs

D.1.1 Preliminary Results

Lemma 24. It holds for every $k \in \mathbb{N}$ and $l = 1, \cdots, L - 1$ that

$$u^k_l = \nu(q^k_l - f_i(z^k_l)).$$

Proof. This follows directly from the optimality condition of $q^k_l$ and Equation (4.6).

Lemma 25. It holds for every $k \in \mathbb{N}$ and $l = 1, \cdots, L - 1$ that

$$\|u^{k+1}_l - u^k_l\| \leq \nu\|q^{k+1}_l - q^k_l\| + \nu S\|z^{k+1}_l - z^k_l\|.$$
D.1. CONVERGENCE PROOFS

**Proof.**

\[
\|u_{k+1}^l - u_k^l\| = \|\nu(q_{k+1}^l - f_l(z_{k+1}^l)) - \nu(q_k^l - f_l(z_k^l))\| \quad \text{(Lemma 24)}
\]

\[
\leq \nu\|q_{k+1}^l - q_k^l\| + \nu\|f_l(z_{k+1}^l) - f_l(z_k^l)\| \quad \text{(Triangle Inequality)}
\]

\[
\leq \nu\|q_{k+1}^l - q_k^l\| + \nu S\|z_{k+1}^l - z_k^l\| \quad \text{(Assumption 6)}.
\]

**Lemma 26.** It holds for every \(k \in \mathbb{N}\) and \(l = 1, \cdots, L - 1\) that

\[
\|u_{k+1}^l - u_k^l\| \leq 2\nu^2(\|q_{k+1}^l - q_k^l\|^2 + S^2\|z_{k+1}^l - z_k^l\|^2).
\]

**Proof.**

\[
\|u_{k+1}^l - u_k^l\|^2 = \nu^2\|q_{k+1}^l - f_l(z_{k+1}^l) - q_k^l + f_l(z_k^l)\|^2 \quad \text{(Lemma 24)}
\]

\[
\leq 2\nu^2(\|q_{k+1}^l - q_k^l\|^2 + \|f_l(z_{k+1}^l) - f_l(z_k^l)\|^2) \quad \text{(Mean Inequality)}
\]

\[
\leq 2\nu^2(\|q_{k+1}^l - q_k^l\|^2 + S^2\|z_{k+1}^l - z_k^l\|^2) \quad \text{(Assumption 6)}.
\]
Lemma 27. For every $k \in \mathbb{N}$, it holds that

\[
L_\rho(p^k, W^k, z^k, q^k, u^k) - L_\rho(p^{k+1}, W^k, z^k, q^k, u^k) \geq \sum_{l=2}^{L} (\tau_{l}^{k+1}/2) \|p_{l}^{k+1} - p_{l}^{k}\|^2,
\]

(D.1)

\[
L_\rho(p^{k+1}, W^k, z^k, q^k, u^k) - L_\rho(p^{k+1}, W^{k+1}, z^k, q^k, u^k) \geq \sum_{l=1}^{L} (\theta_{l}^{k+1}/2) \|W_{l}^{k+1} - W_{l}^{k}\|^2,
\]

(D.2)

\[
L_\rho(p^{k+1}, W^{k+1}, z^k, q^k, u^k) - L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^k, u^k) \geq (\nu/2) \sum_{l=1}^{L} \|z_{l}^{k+1} - z_{l}^{k}\|^2,
\]

(D.3)

\[
\beta_\rho(p^{k}, W^k, z^k, q^k, u^k) \geq \beta_\rho(p^{k+1}, W^k, z^k, q^k, u^k),
\]

(D.4)

\[
\beta_\rho(p^{k+1}, W^k, z^k, q^k, u^k) - \beta_\rho(p^{k+1}, W^{k+1}, z^k, q^k, u^k) \geq \sum_{l=1}^{L} (\theta_{l}^{k+1}/2) \|W_{l}^{k+1} - W_{l}^{k}\|^2,
\]

(D.5)

\[
\beta_\rho(p^{k+1}, W^{k+1}, z^k, q^k, u^k) - \beta_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^k, u^k) \geq (\nu/2) \sum_{l=1}^{L} \|z_{l}^{k+1} - z_{l}^{k}\|^2.
\]

(D.6)

Proof. Generally, all inequalities can be obtained by applying optimality conditions of updating $p$, $W$, and $z$, respectively. We only prove Inequalities (D.1), (D.3) and (D.4). This is because Inequalities (D.2) and (D.5) follow the same routine of Inequality (D.1), and Inequality (D.6) follows the same routine of Inequality (D.3).

Firstly, we focus on Inequality (D.1). The choice of $\tau_{l}^{k+1}$ requires

\[
\phi(p_{l}^{k+1}, W_{l}^{k}, z_{l}^{k}, q_{l-1}^{k}, u_{l-1}^{k}) \leq U_l(p_{l}^{k+1}; \tau_{l}^{k+1}).
\]

(D.7)

Moreover, the optimality condition of Equation (4.1) leads to

\[
\nabla_{p_{l}^{k}} \phi(p_{l}^{k}, W_{l}^{k}, z_{l}^{k}, q_{l-1}^{k}, u_{l-1}^{k}) + \tau_{l}^{k+1}(p_{l}^{k+1} - p_{l}^{k}) = 0.
\]

(D.8)
Therefore

\[ L_\rho(p^k, W^k, z^k, q^k, u^k) - L_\rho(p^{k+1}, W^k, z^k, q^k, u^k) \]
\[ = \sum_{l=2}^{L} (\phi(p_l^k, W_l^k, z_l^k, q_l^k, u_{l-1}^k) - \phi(p_l^{k+1}, W_l^k, z_l^k, q_l^k, u_{l-1}^k)) \]
\[ \geq \sum_{l=2}^{L} (\phi(p_l^k, W_l^k, z_l^k, q_l^k, u_{l-1}^k) - U_l(p_l^{k+1}, \tau_l^{k+1})) \text{(Inequality (D.7))} \]
\[ = \sum_{l=2}^{L} (-\nabla_{p_l^k}\phi^T(p_l^k, W_l^k, z_l^k, q_l^k, u_{l-1}^k)(p_l^{k+1} - p_l^k) - (\tau_l^{k+1}/2)\|p_l^{k+1} - p_l^k\|_2^2) \]
\[ = \sum_{l=2}^{L} (\tau_l^{k+1}/2)\|p_l^{k+1} - p_l^k\|_2^2 \text{(Equation (D.8)).} \]

Next we prove Inequality (D.3). Because \( z_l^{k+1} \) minimizes Equation (4.3) and Equation (4.4), we have

\[ (\nu/2)\|z_l^{k+1} - W_l^{k+1} p_l^{k+1}\|_2^2 + (\nu/2)\|q_l^k - f_i(z_l^{k+1})\|_2^2 + (\nu/2)\|z_l^k - z_l^k\|_2^2 \]
\[ \leq (\nu/2)\|z_l^k - W_l^{k+1} p_l^{k+1}\|_2^2 + (\nu/2)\|q_l^k - f_i(z_l^k)\|_2^2, \quad \text{(D.9)} \]

and

\[ R(z_l^k; y) + (\nu/2)\|z_l^k - W_l^{k+1} p_l^{k+1}\|_2^2 - R(z_l^{k+1}; y) - (\nu/2)\|z_l^{k+1} - W_l^{k+1} p_l^{k+1}\|_2^2 \]
\[ = R(z_l^k; y) - R(z_l^{k+1}; y) + (\nu/2)\|z_l^k - z_l^{k+1}\|_2^2 + \nu(z_l^{k+1} - W_l^{k+1} p_l^{k+1})^T(z_l^k - z_l^{k+1}) \]
\[ \|a - b\|_2^2 - \|a - c\|_2^2 = \|b - c\|_2^2 + 2(c - a)^T(b - c) \text{ where } a = W_l^{k+1} p_l^{k+1}, b = z_l^k, \]
\[ \text{and } c = z_l^{k+1} \]
\[ \geq s^T(z_l^k - z_l^{k+1}) + (\nu/2)\|z_l^k - z_l^{k+1}\|_2^2 + \nu(z_l^{k+1} - W_l^{k+1} p_l^{k+1})^T(z_l^k - z_l^{k+1}) \]
\[ (s \in \partial R(z_l^{k+1}; y) \text{ is a subgradient of } R(z_l^{k+1}; y)) \]
\[ = (\nu/2)\|z_l^{k+1} - z_l^k\|_2^2 \quad \text{(D.10)} \]
\[ (0 \in s + \nu(z_l^{k+1} - W_l^{k+1} p_l^{k+1}) \text{ by the optimality condition of Equation (4.4))}. \]
Therefore

\[
L_\rho(p^{k+1}, W^{k+1}, z^k, q^k, u^k) - L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^k, u^k)
\]

\[
= \sum_{i=1}^{L-1} \left( (\nu/2)\|z_i^k - W_i^{k+1}p_i^{k+1}\|_2^2 + (\nu/2)\|q_i^k - f_i(z_i^k)\|_2^2 - (\nu/2)\|z_i^{k+1} - W_i^{k+1}p_i^{k+1}\|_2^2 - (\nu/2)\|q_i^{k+1} - f_i(z_i^{k+1})\|_2^2 \right)
\]

\[
+ R(z_L^k; y) + (\nu/2)\|z_L^k - W_L^{k+1}p_L^{k+1}\|_2^2 - R(z_L^{k+1}; y) - (\nu/2)\|z_L^{k+1} - W_L^{k+1}p_L^{k+1}\|_2^2
\]

\[
\geq (\nu/2) \sum_{l=1}^{L} \|z_l^{k+1} - z_l^k\|_2^2 (\text{Inequalities (D.9) and (D.10)}).
\]

Finally Inequality (D.4) follows directly the optimality condition of \(p^{k+1}\). \(\Box\)

**Lemma 28.** For every \(k \in N\), it holds that

\[
L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^k, u^k) - L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})
\]

\[
\geq \sum_{l=1}^{L-1} \left( (\rho/2 - 2\nu^2/\rho - \nu/2)\|q_l^{k+1} - q_l^k\|_2^2 - (2\nu^2S^2/\rho)\|z_l^{k+1} - z_l^k\|_2^2 \right) \quad (D.11)
\]

\[
\beta_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^k, u^k) - \beta_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})
\]

\[
\geq \sum_{l=1}^{L-1} \left( (\rho/2 - 2\nu^2/\rho - \nu/2)\|q_l^{k+1} - q_l^k\|_2^2 - (2\nu^2S^2/\rho)\|z_l^{k+1} - z_l^k\|_2^2 \right). \quad (D.12)
\]

**Proof.** We only prove Inequality (D.11) because Inequality (D.12) follows the same
D.1. CONVERGENCE PROOFS

Routine of Inequality (D.11).

\[ L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^k, u^k) - L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) \]
\[ = \sum_{t=1}^{L-1} ((\nu/2)\|f_t(z_t^{k+1}) - q_t^k\|^2 - (\nu/2)\|f_t(z_t^{k+1}) - q_t^{k+1}\|^2 - (u_t^{k+1})^T(q_t^k - q_t^{k+1}) + (\rho/2)\|q_t^{k+1} - q_t^k\|^2 - (1/\rho)\|u_t^{k+1} - u_t^k\|^2) \]
\[ = \sum_{t=1}^{L-1} ((\nu/2)\|f_t(z_t^{k+1}) - q_t^k\|^2 - (\nu/2)\|f_t(z_t^{k+1}) - q_t^{k+1}\|^2 - (u_t^{k+1})^T(q_t^k - q_t^{k+1}) + (\rho/2)\|q_t^{k+1} - q_t^k\|^2 - (1/\rho)\|u_t^{k+1} - u_t^k\|^2) \]
\[ = \sum_{t=1}^{L-1} ((\nu/2)\|f_t(z_t^{k+1}) - q_t^k\|^2 - (\nu/2)\|f_t(z_t^{k+1}) - q_t^{k+1}\|^2 - (u_t^{k+1})^T(q_t^k - q_t^{k+1}) + (\rho/2)\|q_t^{k+1} - q_t^k\|^2 - (1/\rho)\|u_t^{k+1} - u_t^k\|^2) \]
\[ \geq \sum_{t=1}^{L-1} ((\nu/2)\|q_t^{k+1} - q_t^k\|^2 + (\rho/2)\|q_t^{k+1} - q_t^k\|^2 - (1/\rho)\|u_t^{k+1} - u_t^k\|^2) \]
\[ = \sum_{t=1}^{L-1} ((\nu/2)\|q_t^{k+1} - q_t^k\|^2 + (\rho/2)\|q_t^{k+1} - q_t^k\|^2 - (1/\rho)\|u_t^{k+1} - u_t^k\|^2) \]
\[ \geq \sum_{t=1}^{L-1} ((\nu/2)\|q_t^{k+1} - q_t^k\|^2 + (\rho/2)\|q_t^{k+1} - q_t^k\|^2 - (2\nu^2/\rho)\|q_t^{k+1} - q_t^k\|^2) \]
\[ - (2\nu^2S^2/\rho)\|z_t^{k+1} - z_t^k\|^2 \]

\[ \geq \sum_{t=1}^{L-1} ((\rho/2 - 2\nu^2/\rho - \nu/2)\|q_t^{k+1} - q_t^k\|^2 - (2\nu^2S^2/\rho)\|z_t^{k+1} - z_t^k\|^2) \]

\[ \geq \sum_{t=1}^{L-1} ((\rho/2 - 2\nu^2/\rho - \nu/2)\|q_t^{k+1} - q_t^k\|^2 - (2\nu^2S^2/\rho)\|z_t^{k+1} - z_t^k\|^2) \]

D.1.2 Proof of Lemma 8

Proof. We sum up Inequalities (D.1), (D.2), (D.3), and (D.11) to obtain Inequality (5.2), and we sum up Inequalities (D.4), (D.5), (D.6), and (D.12) to obtain Inequality (5.3).

D.1.3 Proof of Lemma 9

Proof. (1) There exists \( q' \) such that \( p_{i+1}^k = q'_i \) and

\[ F(p^k, W^k, z^k, q') \geq \min_{p, W, z, q}{F(p, W, z, q)|p_{i+1} = q_i} > -\infty. \]
Therefore, we have

\[ L_\rho(p^k, W^k, z^k, q^k, u^k) \]

\[ = F(p^k, W^k, z^k, q^k) + \sum_{t=1}^{L} (u_t^k)^T (p_{t+1}^k - q_t^k) + (\rho/2)\|p_{t+1}^k - q_t^k\|_2^2 \]

\[ = R(z_L^k; y) + (\nu/2)(\sum_{t=1}^{L} \|z_t^k - W_t^k p_t^k\|_2^2 + \sum_{t=1}^{L-1} \|q_t^k - f_t(z_t^k)\|_2^2) \]

\[ \geq R(z_L^k; y) + (\nu/2)(\sum_{t=1}^{L} \|z_t^k - W_t^k p_t^k\|_2^2 + \sum_{t=1}^{L-1} \|q_t^k - f_t(z_t^k)\|_2^2) \]

\[ - \sum_{t=1}^{L-1} (\nu/2)\|q_t^k - q_t^k\|_2^2 + \sum_{t=1}^{L-1} (\rho/2)\|p_{t+1}^k - q_t^k\|_2^2 \]

(\nu(q_t - f_t(z_t^{k+1}))) = (\nu/2)\nabla q_t\|q_t - f_t(z_t^{k+1})\|_2^2 \text{ is lipschitz continuous concerning } q_t \]

and Lemma 21 in [2])

\[ = F(p^k, W^k, z^k, q^') + ((\rho - \nu)/2)p_{t+1}^k - q_t^k\|_2^2 > -\infty. \]

Therefore, \( F(p^k, W^k, z^k, q^') \) and \(((\rho - \nu)/2)p_{t+1}^k - q_t^k\|_2^2 \) are upper bounded by \( L_\rho(p^k, W^k, z^k, q^k, u^k) \) and hence \( L_\rho(p^0, W^0, z^0, q^0, u^0) \) (Lemma 8). From Assumption 6, \( (p^k, W^k, z^k) \) is bounded. \( q^k \) is also bounded because \((\rho - \nu)/2\|p_{t+1}^k - q_t^k\|_2^2 \) is upper bounded. \( u^k \) is bounded because of Lemma 24.

(2). It follows the same routine as (1). \( \square \)

### D.1.4 Proof of Theorem 9

**Proof.** (1). From Lemmas 8 and 9, we know that \( L_\rho(p^k, W^k, z^k, q^k, u^k) \) is convergent because a monotone bounded sequence converges. Moreover, we take the limit on
both sides of Inequality (5.2) to obtain

\[
0 = \lim_{k \to \infty} L_{\rho}(p^{k}, W^{k}, z^{k}, q^{k}, u^{k}) - \lim_{k \to \infty} L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) \\
\geq \lim_{k \to \infty} \left( \sum_{l=2}^{L} \left( \frac{\tau^{k+1,l}}{2} \right) \|p_{l}^{k+1} - p_{l}^{k}\|_{2}^{2} + \sum_{l=1}^{L} \left( \frac{\theta^{k+1,l}}{2} \right) \|W^{k+1}_{l} - W^{k}_{l}\|_{2}^{2} \right) \\
+ \sum_{l=1}^{L-1} C_{1} \|z^{k+1}_{l} - z^{k}_{l}\|_{2}^{2} + (\nu/2) \|z^{k+1}_{l} - z^{k}_{l}\|_{2}^{2} + \sum_{l=1}^{L-1} C_{2} \|q^{k+1}_{l} - q^{k}_{l}\|_{2}^{2} \geq 0. 
\]

Because \(L_{\rho}(p^{k}, W^{k}, z^{k}, q^{k}, u^{k})\) is convergent, then \(\lim_{k \to \infty} \|p^{k+1} - p^{k}\|_{2}^{2} = 0, \lim_{k \to \infty} \|W^{k+1} - W^{k}\|_{2}^{2} = 0, \lim_{k \to \infty} \|z^{k+1} - z^{k}\|_{2}^{2} = 0,\) and \(\lim_{k \to \infty} \|q^{k+1} - q^{k}\|_{2}^{2} = 0.\)

\(\lim_{k \to \infty} \|u^{k+1} - u^{k}\|_{2}^{2} = 0\) is derived from Lemma 26 in Section D.1 in the Appendix. (2). The proof follows the same procedure as (1).

\[\square\]

**Proof of Lemma 10**

**Proof.** (1). We know that \(\partial L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})\) is upper bounded by the linear combination of \(\|p^{k+1} - p^{k}\|, \|W^{k+1} - W^{k}\|, \|z^{k+1} - z^{k}\|, \|q^{k+1} - q^{k}\|,\) and \(\|u^{k+1} - u^{k}\|\).

For \(p^{k+1}_{l}\),

\[
\nabla_{p^{k+1}_{l}} L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) \\
= \nabla_{p^{k+1}_{l}} \phi(p^{k+1}_{l}, W^{k+1}_{l}, z^{k}_{l}, q^{k+1}_{l-1}, u^{k+1}_{l-1}) \\
= \nabla_{p^{k+1}_{l}} \phi(p^{k}_{l}, W^{k}_{l}, z^{k}_{l}, q^{k}_{l-1}, u^{k}_{l-1}) + \tau^{k+1}_{l}(p^{k+1}_{l} - p^{k}_{l}) - \tau^{k+1}_{l}(p^{k+1}_{l} - p^{k}_{l}) \\
+ \nu(W^{k+1}_{l} + W^{k}_{l}) - \nu(W^{k}_{l}) + \nu(W^{k+1}_{l}) z^{k+1}_{l} + \nu(W^{k}_{l}) z^{k}_{l} \\
+ (u^{k+1}_{l-1} - u^{k}_{l-1}) + \rho(p^{k+1}_{l} - p^{k}_{l}) - \rho(q^{k+1}_{l} - q^{k}_{l}) \\
= -\tau^{k+1}_{l}(p^{k+1}_{l} - p^{k}_{l}) + \nu(W^{k+1}_{l} + W^{k}_{l}) - \nu(W^{k}_{l}) W^{k}_{l} p^{k}_{l} \\
- \nu(W^{k+1}_{l} + W^{k}_{l}) z^{k+1}_{l} + \nu(W^{k}_{l}) z^{k}_{l} + (u^{k+1}_{l-1} - u^{k}_{l-1}) + \rho(p^{k+1}_{l} - p^{k}_{l}) - \rho(q^{k+1}_{l} - q^{k}_{l}) \\
\text{(The optimality condition of Equation (4.1))}
\]
\[ \| \nabla_{\eta_{k+1}} L_{\rho}(P^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) \| \]
\[ = \| - \tau^{k+1}_{l} (p^{k+1}_{l} - p^{k}_{l}) + \nu(W^{k+1}_{l})^{T} W^{k+1}_{l} p^{k+1}_{l} - \nu(W^{k}_{l})^{T} W^{k}_{l} p^{k}_{l} \]
\[ - \nu(W^{k+1}_{l})^{T} z^{k+1}_{l} + \nu(W^{k}_{l})^{T} z^{k}_{l} + (u^{k+1}_{l-1} - u^{k}_{l-1}) + \rho(p^{k+1}_{l} - p^{k}_{l}) - \rho(q^{k+1}_{l-1} - q^{k}_{l-1}) \| \]
\[ \leq \tau^{k+1}_{l} \| p^{k+1}_{l} - p^{k}_{l} \| + \nu\| (W^{k+1}_{l})^{T} W^{k}_{l} p^{k+1}_{l} - (W^{k}_{l})^{T} W^{k}_{l} p^{k}_{l} \| \]
\[ + \nu\| (W^{k+1}_{l})^{T} z^{k+1}_{l} - (W^{k}_{l})^{T} z^{k}_{l} \| + \| u^{k+1}_{l-1} - u^{k}_{l-1} \| + \rho\| p^{k+1}_{l} - p^{k}_{l} \| + \rho\| q^{k+1}_{l-1} - q^{k}_{l-1} \| \]

(Triangle Inequality)
\[ \leq \tau^{k+1}_{l} \| p^{k+1}_{l} - p^{k}_{l} \| + \nu\| W^{k+1}_{l} \|^2 \| p^{k+1}_{l} - p^{k}_{l} \| + \nu\| W^{k+1}_{l} \| \| W^{k}_{l} - W^{k+1}_{l} \| \| p^{k}_{l} \| \]
\[ + \nu\| W^{k}_{l} - W^{k+1}_{l} \| \| W^{k}_{l} \| \| p^{k}_{l} \| + \nu\| W^{k+1}_{l} \| \| z^{k+1}_{l} - z^{k}_{l} \| + \nu\| W^{k+1}_{l} - W^{k}_{l} \| \| z^{k}_{l} \| \]
\[ + \nu((q^{k+1}_{l-1} - q^{k}_{l-1}) + S\| z^{k+1}_{l-1} - z^{k}_{l-1} \|) + \rho\| p^{k+1}_{l} - p^{k}_{l} \| + \rho\| q^{k+1}_{l-1} - q^{k}_{l-1} \| \]

(Triangle Inequality, Cauhthy-Schwartz Inequality and Lemma 20)
\[ \leq \tau^{k+1}_{l} \| p^{k+1}_{l} - p^{k}_{l} \| + \nu N_{W}^2 \| p^{k+1}_{l} - p^{k}_{l} \| + 2\nu N_{W} N_{p} \| W^{k+1}_{l} - W^{k}_{l} \| \]
\[ + \nu N_{W} \| z^{k+1}_{l} - z^{k}_{l} \| + \nu N_{z} \| W^{k+1}_{l} - W^{k}_{l} \| + 2\nu^2 \| \| q^{k+1}_{l-1} - q^{k}_{l-1} \| + S^2 \| z^{k+1}_{l-1} - z^{k}_{l-1} \|^2 \]
\[ + \rho\| p^{k+1}_{l} - p^{k}_{l} \| + \rho\| q^{k+1}_{l-1} - q^{k}_{l-1} \| \] (Lemma 6).
D.1. CONVERGENCE PROOFS

For $W_1^{k+1}$,

\[
\nabla_{W_1^{k+1}} L_\rho(P^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) \\
= \nabla_{W_1^{k+1}} \phi(p_1^{k+1}, W_1^{k+1}, z_1^{k+1}) \\
= \nabla_{W_1^{k+1}} \phi(p_1^{k+1}, W_1^{k}, z_1^{k}) + \theta_1^{k+1}(W_1^{k+1} - W_1^{k}) + \nu(W_1^{k+1} - W_1^{k})p_1^{k+1}(p_1^{k+1})^T \\
- \nu(z_1^{k+1} - z_1^{k})(p_1^{k+1})^T - \theta_1^{k+1}(W_1^{k+1} - W_1^{k}) \\
= \nu(W_1^{k+1} - W_1^{k})p_1^{k+1}(p_1^{k+1})^T - \nu(z_1^{k+1} - z_1^{k})(p_1^{k+1})^T - \theta_1^{k+1}(W_1^{k+1} - W_1^{k}) \\
\]

(The optimality condition of Equation (4.2)).

So

\[
\|\nabla_{W_1^{k+1}} L_\rho(P^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})\| \\
= \|\nu(W_1^{k+1} - W_1^{k})p_1^{k+1}(p_1^{k+1})^T - \nu(z_1^{k+1} - z_1^{k})(p_1^{k+1})^T - \theta_1^{k+1}(W_1^{k+1} - W_1^{k})\| \\
\leq \nu\|W_1^{k+1} - W_1^{k}\|^2 + \nu\|z_1^{k+1} - z_1^{k}\|\|p_1^{k+1}\| + \theta_1^{k+1}\|W_1^{k+1} - W_1^{k}\| \\
\]

(Triangle Inequality and Cauchy-Schwartz Inequality)

\[
\leq \nu\|W_1^{k+1} - W_1^{k}\|^2 + \nu\|z_1^{k+1} - z_1^{k}\|\|N_p\| + \theta_1^{k+1}\|W_1^{k+1} - W_1^{k}\| \text{ (Theorem 6).} 
\]

For $W_l^{k+1}(1 < l \leq L)$,

\[
\nabla_{W_l^{k+1}} L_\rho(P^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) \\
= \nabla_{W_l^{k+1}} \phi(p_l^{k+1}, W_l^{k+1}, z_l^{k+1}, p_{l-1}^{k+1}, u_{l-1}^{k+1}) \\
= \nabla_{W_l^{k+1}} \phi(p_l^{k+1}, W_l^{k}, z_l^{k}, p_{l-1}^{k}, u_{l-1}^{k}) + \theta_l^{k+1}(W_l^{k+1} - W_l^{k}) + \nu(W_l^{k+1} - W_l^{k})p_l^{k+1}(p_l^{k+1})^T \\
- \nu(z_l^{k+1} - z_l^{k})(p_l^{k+1})^T - \theta_l^{k+1}(W_l^{k+1} - W_l^{k}) \\
= \nu(W_l^{k+1} - W_l^{k})p_l^{k+1}(p_l^{k+1})^T - \nu(z_l^{k+1} - z_l^{k})(p_l^{k+1})^T - \theta_l^{k+1}(W_l^{k+1} - W_l^{k}) \\
\]

(The optimality condition of Equation (4.2)).
D.1. CONVERGENCE PROOFS

So

\[ \| \nabla W_{l+1}L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) \| 
= \| \nu(W_{l+1}^{k+1} - W_{l}^{k})p_{l+1}^{k+1}(q_{l+1}^{k+1})^T - \nu(z_{l+1}^{k+1} - z_{l}^{k})(p_{l}^{k+1})^T - \theta_{l+1}^{k+1}(W_{l+1}^{k+1} - W_{l}^{k}) \| 
\leq \nu\| W_{l+1}^{k+1} - W_{l}^{k} \| \| p_{l+1}^{k+1} \| + \nu\| z_{l+1}^{k+1} - z_{l}^{k} \| \| p_{l}^{k+1} \| + \theta_{l+1}^{k+1}\| W_{l+1}^{k+1} - W_{l}^{k} \| 
\]

(Triangle Inequality and Cauhy-Schwartz Inequality)

\[ \leq \nu\| W_{l+1}^{k+1} - W_{l}^{k} \| \| q_{l+1}^{k+1} \| + \nu\| z_{l+1}^{k+1} - z_{l}^{k} \| \| q_{l}^{k+1} \| + \theta_{l+1}^{k+1}\| W_{l+1}^{k+1} - W_{l}^{k} \| \quad (\text{Theorem 6}). \]

For \( z_{l}^{k+1}(l < L) \),

\[
\partial z_{l+1}L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) \\
= \partial z_{l+1}L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k}, u^{k}) + \nu(z_{l+1}^{k+1} - z_{l}^{k}) - \nu(z_{l+1}^{k+1} - z_{l}^{k}) \\
- \nu\partial f_{1}(z_{l+1}^{k+1}) \circ (q_{l+1}^{k+1} - q_{l}^{k}) \quad (\text{o is Hadamard product}) \\
= -\nu(z_{l+1}^{k+1} - z_{l}^{k}) - \nu\partial f_{1}(z_{l+1}^{k+1}) \circ (q_{l+1}^{k+1} - q_{l}^{k}) \\
(0 \in \partial z_{l+1}L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k}, u^{k}) + \nu(z_{l+1}^{k+1} - z_{l}^{k})).
\]

So

\[
\| \partial z_{l+1}L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) \| \\
= \| - \nu(z_{l+1}^{k+1} - z_{l}^{k}) - \nu\partial f_{1}(z_{l+1}^{k+1}) \circ (q_{l+1}^{k+1} - q_{l}^{k}) \| \\
\leq \nu\| z_{l+1}^{k+1} - z_{l}^{k} \| + \nu\| \partial f_{1}(z_{l+1}^{k+1}) \| \| q_{l+1}^{k+1} - q_{l}^{k} \| 
\]

(Cauchy-Schwartz Inequality and Triangle Inequality)

\[ \leq \nu\| z_{l}^{k+1} - z_{l}^{k} \| + \nu M\| q_{l+1}^{k+1} - q_{l}^{k} \| (\| \partial f_{1}(z_{l+1}^{k+1}) \| \leq M). \]

For \( z_{L}^{k+1}, \partial z_{L+1}L_{\rho}(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) = 0 \) by the optimality condition of Equation (4.4).
For $q_{i+1}^k$,

$$\nabla q_{i+1}^k L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})$$

$$= \nabla q_{i+1}^k L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^k) + u^{k+1} - u^k$$

(by the optimality condition of Equation (4.5)).

So $\|\nabla q_{i+1}^k L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})\| = \|u^{k+1} - u^k\|$.

For $u_{i+1}^k$,

$$\nabla u_{i+1}^k L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1}) = (p^{k+1}_i - q^{k+1}_i) = (u^{k+1}_i - u^k_i)/\rho.$$

So $\|\nabla u_{i+1}^k L_\rho(p^{k+1}, W^{k+1}, z^{k+1}, q^{k+1}, u^{k+1})\| = \|u^{k+1}_i - u^k_i\|/\rho$. 

In summary, we prove that $\nabla p^{k+1}_i L_\rho, \nabla W^{k+1}_i L_\rho, \partial q^{k+1}_i L_\rho, \nabla u^{k+1}_i L_\rho$ are upper bounded by the linear combination of $\|p^{k+1} - p^k\|, \|W^{k+1} - W^k\|, \|z^{k+1} - z^k\|, \|q^{k+1} - q^k\|$, and $\|u^{k+1} - u^k\|$.

(2). It follows exactly the proof of (1) except for $p_{i+1}^k$. □

D.1.5 The proof of Theorem 10

Proof. From Lemma 9(1), $(p^k, W^k, z^k, q^k, u^k)$ has at least a limit point $(p^*, W^*, z^*, q^*, u^*)$ because a bounded sequence has at least a limit point. From Lemma 10 and Theorem 9, $\|g^{k+1}\| \to 0$ as $k \to \infty$. According to the definition of general subgradient (Definition 8.3 in [84]), we have $0 \in \partial L_\rho(p^*, W^*, z^*, q^*, u^*)$. In other words, every limit point $(p^*, W^*, z^*, q^*, u^*)$ is a stationary point. □

D.1.6 The proof of Theorem 11

Proof. From Lemma 9(2), $(W^k, z^k, q^k, u^k)$ has at least a limit point $(W^*, z^*, q^*, u^*)$ because a bounded sequence has at least a limit point. $p^k$ has at least a limit point
Proof. (1). To prove this, we will first show that $p^k \in \Delta$ and $\Delta$ is finite. From Lemma 10(2) and Theorem 9, $\|g_{W}^{k+1}\| \to 0$, $\|g_{z}^{k+1}\| \to 0$, $\|g_{q}^{k+1}\| \to 0$, $\|g_{u}^{k+1}\| \to 0$ as $k \to \infty$. According to the definition of general subgradient (Definition 8.3 in [84]), we have $\nabla W_{\beta}(p^*, W^*, z^*, q^*, u^*) = 0$, $0 \in \partial z^* \beta(p^*, W^*, z^*, q^*, u^*) = 0$, $0 \in \partial z^* \beta(p^*, W^*, z^*, q^*, u^*) = 0$ (i.e. $p^*_{l+1} = q^*_{l+1}$). In other words, every limit point $(W^*, z^*, u^*)$ is a stationary point of Problem 7. Moreover, $\tau^k_l$ has a limit point $\tau^*_l$ because it is bounded. Let $\tau^k = \{\tau^k_l\}_{l=2}^L$.

Consider a subsequence $(p^s, W^s, z^s, q^s, u^s, \tau^{s+1}) \to (p^*, W^*, z^*, q^*, u^*, \tau^*)$. Because $u^s_{l+1} = u^s_l + \rho(p^s_{l+1} - q^s_l)$ and $u^s_{l+1} \to u^s_l$, thus $p^s_{l+1} \to q^s_l$, and $p^s_{l+1} \to q^s_{l+1}$. Because $q^s_{l+1} \to q^s_l$, then $p^s_{l+1} \to p^s_{l+1}$ for any $l$. In other words, $p^s_{l+1} \to p^s_*$. Because $p^s \to p^*$, then $p^s_{l+1} \to p^*$. The optimality condition of $p^s_{l+1}$ (i.e. Equation (5.1)) leads to

$$p^s_{l+1} \leftarrow \arg \min_{\delta \in \Delta} \|\delta - p^s_l - \nabla_{p_l^s} \phi(p^s_l, W^s_l, z^s_l, q^s_{l-1}, u^s_{l-1})/\tau^s_{l+1})\|.$$ 

Taking $s \to \infty$ on both sides, we have

$$p^s_l \leftarrow \arg \min_{\delta \in \Delta} \|\delta - (p^s_l - \nabla_{p_l^s} \phi(p^s_l, W^s_l, z^s_l, q^s_{l-1}, u^s_{l-1})/\tau^s_l)\|.$$ 

Because $\nabla_{p_l} F(p^*, W^*, z^*, q^*) = \nu W^T_l (z^*_l - W^*_l p^*_l) = \nabla_{p_l^s} \phi(p^s_l, W^s_l, z^s_l, q^s_{l-1}, u^s_{l-1})$. Then

$$p^s_l \leftarrow \arg \min_{\delta \in \Delta} \|\delta - (p^s_l - \nabla_{p_l^s} F(p^*, W^*, z^*, q^*)/\tau^s_l)\|.$$ 

Namely, $p^*$ is a quantized stationary point of Problem 7. 

D.1.7 The proof of Theorem 12

Proof. (1). To prove this, we will first show that $c_k$ satisfies two conditions: (1). $c_k \geq c_{k+1}$. (2). $\sum_{k=0}^{\infty} c_k$ is bounded. We then conclude the convergence rate of $o(1/k)$
D.1. CONVERGENCE PROOFS

Based on these two conditions. Specifically, first, we have

\[
c_k = \min_{0 \leq i \leq k} \left( \sum_{t=2}^{L} \frac{(\tau^{i+1}_t)}{2} \|p^{i+1}_t - p^i_t\|_2^2 + \sum_{t=1}^{L} \frac{(\theta^{i+1}_t)}{2} \|W^{i+1}_t - W^i_t\|_2^2 \
+ \sum_{t=1}^{L-1} C_1 \|z^{i+1}_t - z^i_t\|_2^2 + \frac{(\nu/2)}{2} \|z^{i+1}_0 - z^i_0\|_2^2 + \sum_{t=1}^{L-1} C_2 \|q^{i+1}_t - q^i_t\|_2^2 \right)
\]

\[
\geq \min_{0 \leq i \leq k+1} \left( \sum_{t=2}^{L} \frac{(\tau^{i+1}_t)}{2} \|p^{i+1}_t - p^i_t\|_2^2 + \sum_{t=1}^{L} \frac{(\theta^{i+1}_t)}{2} \|W^{i+1}_t - W^i_t\|_2^2 \
+ \sum_{t=1}^{L-1} C_1 \|z^{i+1}_t - z^i_t\|_2^2 + \frac{(\nu/2)}{2} \|z^{i+1}_0 - z^i_0\|_2^2 + \sum_{t=1}^{L-1} C_2 \|q^{i+1}_t - q^i_t\|_2^2 \right)
\]

\[
= c_{k+1}.
\]

Therefore \( c_k \) satisfies the first condition. Second,

\[
\sum_{t=0}^{\infty} c_k = \sum_{k=0}^{\infty} \min_{0 \leq i \leq k} \left( \sum_{t=2}^{L} \frac{(\tau^{i+1}_t)}{2} \|p^{i+1}_t - p^i_t\|_2^2 + \sum_{t=1}^{L} \frac{(\theta^{i+1}_t)}{2} \|W^{i+1}_t - W^i_t\|_2^2 \
+ \sum_{t=1}^{L-1} C_1 \|z^{i+1}_t - z^i_t\|_2^2 + \frac{(\nu/2)}{2} \|z^{i+1}_0 - z^i_0\|_2^2 + \sum_{t=1}^{L-1} C_2 \|q^{i+1}_t - q^i_t\|_2^2 \right)
\]

\[
\leq \sum_{k=0}^{\infty} \left( \sum_{t=2}^{L} \frac{(\tau^{k+1}_t)}{2} \|p^{k+1}_t - p^k_t\|_2^2 + \sum_{t=1}^{L} \frac{(\theta^{k+1}_t)}{2} \|W^{k+1}_t - W^k_t\|_2^2 \
+ \sum_{t=1}^{L-1} C_1 \|z^{k+1}_t - z^k_t\|_2^2 + \frac{(\nu/2)}{2} \|z^{k+1}_0 - z^k_0\|_2^2 + \sum_{t=1}^{L-1} C_2 \|q^{k+1}_t - q^k_t\|_2^2 \right)
\]

\[
\leq L_p(p^0, W^0, z^0, q^0, u^0) - L_p(p^*, W^*, z^*, q^*, u^*)(\text{Lemma 8}).
\]

So \( \sum_{k=0}^{\infty} c_k \) is bounded and \( c_k \) satisfies the second condition. Finally, it has been proved that the sufficient conditions of convergence rate \( o(1/k) \) are: (1) \( c_k \geq c_{k+1} \), and (2) \( \sum_{k=0}^{\infty} c_k \) is bounded, and (3) \( c_k \geq 0 \) (Lemma 1.2 in [26]). Since we have proved the first two conditions and the third one \( c_k \geq 0 \) is obvious, the convergence rate of \( o(1/k) \) is proven.

(2). It follows the same procedure as (1).
D.2 More Experimental Results

D.2.1 The Settings of All Hyperparameters

This section provides more details on the hyperparameter settings of all datasets, lr denotes learning rate, which are shown in the following tables.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cora</th>
<th>PubMed</th>
<th>Citeseer</th>
<th>Dataset</th>
<th>Cora</th>
<th>PubMed</th>
<th>Citeseer</th>
</tr>
</thead>
<tbody>
<tr>
<td>lr(GD)</td>
<td>10^{-4}</td>
<td>5 \times 10^{-2}</td>
<td>10^{-4}</td>
<td>lr(GD)</td>
<td>10^{-4}</td>
<td>5 \times 10^{-3}</td>
<td>10^{-4}</td>
</tr>
<tr>
<td>lr(Adadelta)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>lr(Adadelta)</td>
<td>10^{-3}</td>
<td>10^{-4}</td>
<td>10^{-3}</td>
</tr>
<tr>
<td>lr(Adagrad)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>lr(Adagrad)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
</tr>
<tr>
<td>lr(Adam)</td>
<td>10^{-4}</td>
<td>10^{-4}</td>
<td>10^{-3}</td>
<td>lr(Adam)</td>
<td>10^{-4}</td>
<td>10^{-4}</td>
<td>10^{-4}</td>
</tr>
<tr>
<td>\rho, \nu(pdADMM-G)</td>
<td>10^{-4}</td>
<td>10^{-4}</td>
<td>10^{-4}</td>
<td>\rho, \nu(pdADMM-G)</td>
<td>10^{-4}</td>
<td>10^{-4}</td>
<td>10^{-3}</td>
</tr>
<tr>
<td>\rho, \nu(pdADMM-G-Q)</td>
<td>10^{-4}</td>
<td>10^{-4}</td>
<td>10^{-3}</td>
<td>\rho, \nu(pdADMM-G-Q)</td>
<td>10^{-4}</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Amazon Computers</th>
<th>Amazon Photo</th>
<th>Coauthor CS</th>
<th>Dataset</th>
<th>Amazon Computers</th>
<th>Amazon Photo</th>
<th>Coauthor CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>lr(GD)</td>
<td>10^{-2}</td>
<td>10^{-2}</td>
<td>10^{-1}</td>
<td>lr(GD)</td>
<td>10^{-2}</td>
<td>10^{-2}</td>
<td>10^{-1}</td>
</tr>
<tr>
<td>lr(Adadelta)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>lr(Adadelta)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
</tr>
<tr>
<td>lr(Adagrad)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>lr(Adagrad)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
</tr>
<tr>
<td>lr(Adam)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>lr(Adam)</td>
<td>10^{-4}</td>
<td>10^{-4}</td>
<td>10^{-4}</td>
</tr>
<tr>
<td>\rho, \nu(pdADMM-G)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-2}</td>
<td>\rho, \nu(pdADMM-G)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
</tr>
<tr>
<td>\rho, \nu(pdADMM-G-Q)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-2}</td>
<td>\rho, \nu(pdADMM-G-Q)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Coauthor CS</th>
<th>Coauthor Physics</th>
<th>Flickr</th>
<th>Ogbn-ArxivDataset</th>
<th>Coauthor CS</th>
<th>Coauthor Physics</th>
<th>Flickr</th>
<th>Ogbn-Arxiv</th>
</tr>
</thead>
<tbody>
<tr>
<td>lr(GD)</td>
<td>10^{-1}</td>
<td>10^{-3}</td>
<td>10^{-2}</td>
<td>lr(GD)</td>
<td>10^{-2}</td>
<td>10^{-2}</td>
<td>10^{-2}</td>
<td></td>
</tr>
<tr>
<td>lr(Adadelta)</td>
<td>10^{-3}</td>
<td>10^{-2}</td>
<td>10^{-1}</td>
<td>lr(Adadelta)</td>
<td>10^{-3}</td>
<td>10^{-2}</td>
<td>10^{-1}</td>
<td></td>
</tr>
<tr>
<td>lr(Adagrad)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>lr(Adagrad)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td></td>
</tr>
<tr>
<td>lr(Adam)</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td>10^{-2}</td>
<td>lr(Adam)</td>
<td>10^{-4}</td>
<td>10^{-3}</td>
<td>10^{-3}</td>
<td></td>
</tr>
<tr>
<td>\rho, \nu(pdADMM-G)</td>
<td>10^{-3}</td>
<td>10^{-4}</td>
<td>10^{-3}</td>
<td>\rho, \nu(pdADMM-G)</td>
<td>10^{-2}</td>
<td>10^{-4}</td>
<td>10^{-4}</td>
<td></td>
</tr>
<tr>
<td>\rho, \nu(pdADMM-G-Q)</td>
<td>10^{-2}</td>
<td>10^{-4}</td>
<td>10^{-4}</td>
<td>\rho, \nu(pdADMM-G-Q)</td>
<td>10^{-4}</td>
<td>10^{-4}</td>
<td>10^{-4}</td>
<td></td>
</tr>
</tbody>
</table>

Table D.1: Hyperparameter settings of all methods on nine benchmark datasets when the number of neurons is 100.

Table D.2: Hyperparameter settings of all methods on nine benchmark datasets when the number of neurons is 500.

D.2.2 The Performance of Validation Sets

This section provides more experimental results on the validation sets of all datasets, which are shown in the following tables.
D.2. MORE EXPERIMENTAL RESULTS

#### Table D.4: The validation performance of all methods when the number of neurons is 500.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cora</th>
<th>PubMed</th>
<th>Citeseer</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>0.704 ± 0.037</td>
<td>0.626 ± 0.072</td>
<td>0.619 ± 0.045</td>
</tr>
<tr>
<td>Adadelta</td>
<td>0.652 ± 0.034</td>
<td>0.720 ± 0.035</td>
<td>0.620 ± 0.022</td>
</tr>
<tr>
<td>Adagrad</td>
<td>0.720 ± 0.022</td>
<td>0.762 ± 0.012</td>
<td>0.604 ± 0.027</td>
</tr>
<tr>
<td>Adam</td>
<td>0.720 ± 0.034</td>
<td>0.745 ± 0.014</td>
<td>0.624 ± 0.014</td>
</tr>
<tr>
<td>pdADMM-G</td>
<td>0.750 ± 0.005</td>
<td>0.788 ± 0.004</td>
<td>0.724 ± 0.005</td>
</tr>
<tr>
<td>pdADMM-G-Q</td>
<td><strong>0.754 ± 0.002</strong></td>
<td><strong>0.793 ± 0.002</strong></td>
<td><strong>0.722 ± 0.002</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Amazon Computers</th>
<th>Amazon Photo</th>
<th>Coauthor CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>0.654 ± 0.033</td>
<td>0.730 ± 0.165</td>
<td>0.875 ± 0.007</td>
</tr>
<tr>
<td>Adadelta</td>
<td>0.136 ± 0.062</td>
<td>0.343 ± 0.046</td>
<td>0.781 ± 0.084</td>
</tr>
<tr>
<td>Adagrad</td>
<td>0.750 ± 0.095</td>
<td>0.808 ± 0.018</td>
<td>0.899 ± 0.006</td>
</tr>
<tr>
<td>Adam</td>
<td>0.740 ± 0.010</td>
<td>0.850 ± 0.006</td>
<td>0.887 ± 0.009</td>
</tr>
<tr>
<td>pdADMM-G</td>
<td><strong>0.753 ± 0.005</strong></td>
<td>0.846 ± 0.014</td>
<td><strong>0.913 ± 0.003</strong></td>
</tr>
<tr>
<td>pdADMM-G-Q</td>
<td><strong>0.688 ± 0.063</strong></td>
<td>0.822 ± 0.013</td>
<td><strong>0.916 ± 0.003</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Coauthor Physics</th>
<th>Flickr</th>
<th>Ogbn-Arxiv</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>0.921 ± 0.009</td>
<td>0.464 ± 0.008</td>
<td>0.378 ± 0.004</td>
</tr>
<tr>
<td>Adadelta</td>
<td>0.918 ± 0.014</td>
<td>0.461 ± 0.006</td>
<td>0.514 ± 0.014</td>
</tr>
<tr>
<td>Adagrad</td>
<td>0.928 ± 0.005</td>
<td>0.480 ± 0.003</td>
<td>0.574 ± 0.008</td>
</tr>
<tr>
<td>Adam</td>
<td>0.919 ± 0.010</td>
<td>0.512 ± 0.004</td>
<td><strong>0.681 ± 0.003</strong></td>
</tr>
<tr>
<td>pdADMM-G</td>
<td><strong>0.933 ± 0.001</strong></td>
<td><strong>0.514 ± 0.001</strong></td>
<td><strong>0.649 ± 0.012</strong></td>
</tr>
<tr>
<td>pdADMM-G-Q</td>
<td><strong>0.935 ± 0.002</strong></td>
<td>0.506 ± 0.004</td>
<td><strong>0.661 ± 0.004</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Coauthor Physics</th>
<th>Flickr</th>
<th>Ogbn-Arxiv</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>0.731 ± 0.018</td>
<td>0.651 ± 0.034</td>
<td>0.679 ± 0.008</td>
</tr>
<tr>
<td>Adadelta</td>
<td>0.716 ± 0.061</td>
<td>0.688 ± 0.024</td>
<td>0.597 ± 0.025</td>
</tr>
<tr>
<td>Adagrad</td>
<td>0.765 ± 0.014</td>
<td>0.776 ± 0.006</td>
<td>0.668 ± 0.028</td>
</tr>
<tr>
<td>Adam</td>
<td><strong>0.758 ± 0.013</strong></td>
<td>0.778 ± 0.008</td>
<td>0.668 ± 0.020</td>
</tr>
<tr>
<td>pdADMM-G</td>
<td>0.753 ± 0.004</td>
<td><strong>0.792 ± 0.004</strong></td>
<td><strong>0.729 ± 0.003</strong></td>
</tr>
<tr>
<td>pdADMM-G-Q</td>
<td><strong>0.757 ± 0.005</strong></td>
<td><strong>0.792 ± 0.003</strong></td>
<td><strong>0.730 ± 0.004</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Coauthor Physics</th>
<th>Flickr</th>
<th>Ogbn-Arxiv</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>0.727 ± 0.012</td>
<td>0.809 ± 0.012</td>
<td>0.897 ± 0.003</td>
</tr>
<tr>
<td>Adadelta</td>
<td>0.246 ± 0.073</td>
<td>0.371 ± 0.075</td>
<td>0.884 ± 0.003</td>
</tr>
<tr>
<td>Adagrad</td>
<td>0.766 ± 0.011</td>
<td>0.860 ± 0.003</td>
<td><strong>0.912 ± 0.004</strong></td>
</tr>
<tr>
<td>Adam</td>
<td>0.750 ± 0.017</td>
<td><strong>0.872 ± 0.020</strong></td>
<td>0.893 ± 0.013</td>
</tr>
<tr>
<td>pdADMM-G</td>
<td><strong>0.778 ± 0.007</strong></td>
<td>0.861 ± 0.005</td>
<td><strong>0.912 ± 0.003</strong></td>
</tr>
<tr>
<td>pdADMM-G-Q</td>
<td><strong>0.764 ± 0.008</strong></td>
<td>0.850 ± 0.009</td>
<td>0.910 ± 0.003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Coauthor Physics</th>
<th>Flickr</th>
<th>Ogbn-Arxiv</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>0.928 ± 0.001</td>
<td>0.466 ± 0.001</td>
<td>0.451 ± 0.003</td>
</tr>
<tr>
<td>Adadelta</td>
<td>0.932 ± 0.006</td>
<td>0.462 ± 0.004</td>
<td>0.591 ± 0.017</td>
</tr>
<tr>
<td>Adagrad</td>
<td><strong>0.935 ± 0.005</strong></td>
<td>0.488 ± 0.007</td>
<td><strong>0.646 ± 0.010</strong></td>
</tr>
<tr>
<td>Adam</td>
<td>0.933 ± 0.007</td>
<td><strong>0.516 ± 0.002</strong></td>
<td><strong>0.692 ± 0.008</strong></td>
</tr>
<tr>
<td>pdADMM-G</td>
<td><strong>0.932 ± 0.001</strong></td>
<td><strong>0.514 ± 0.003</strong></td>
<td><strong>0.661 ± 0.005</strong></td>
</tr>
<tr>
<td>pdADMM-G-Q</td>
<td><strong>0.933 ± 0.002</strong></td>
<td><strong>0.514 ± 0.001</strong></td>
<td><strong>0.667 ± 0.005</strong></td>
</tr>
</tbody>
</table>
Bibliography


[17] François Chollet et al. Keras. https://keras.io, 2015.


[23] Hanjun Dai, Zornitsa Kozareva, Bo Dai, Alex Smola, and Le Song. Learning


[69] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev


