ADMM for Efficient Deep Learning with Global Convergence

Junxiang Wang, Fuxun Yu, Xiang Chen and Liang Zhao
George Mason University
{jwang40,fyu2,xchen26,lzhao9}@gmu.edu

ABSTRACT

Alternating Direction Method of Multipliers (ADMM) has been used successfully in many conventional machine learning applications and is considered to be a useful alternative to Stochastic Gradient Descent (SGD) as a deep learning optimizer. However, as an emerging domain, several challenges remain, including 1) The lack of global convergence guarantees, 2) Slow convergence towards solutions, and 3) Cubic time complexity with regard to feature dimensions. In this paper, we propose a novel optimization framework for deep learning via ADMM (dlADMM) to address these challenges simultaneously. 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 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.

CCS CONCEPTS

• **Theory of computation → Nonconvex optimization**; • **Computing methodologies → Neural networks**.

KEYWORDS

Deep Learning, Global Convergence, Alternating Direction Method of Multipliers

ACM Reference Format:
Junxiang Wang, Fuxun Yu, Xiang Chen and Liang Zhao. 2019. ADMM for Efficient Deep Learning with Global Convergence. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330936

1 INTRODUCTION

Deep learning has been a hot topic in the machine learning community for the last decade. While conventional machine learning techniques have limited capacity to process natural data in their raw form, deep learning methods are composed of non-linear modules that can learn multiple levels of representation automatically [11]. Since deep learning methods are usually applied in large-scale datasets, such approaches require efficient optimizers to obtain a feasible solution within realistic time limits.

Stochastic Gradient Descent (SGD) and many of its variants are popular state-of-the-art methods for training deep learning models due to their efficiency. However, SGD suffers from many limitations that prevent its more widespread use: for example, the error signal diminishes as the gradient is backpropagated (i.e. the gradient vanishes); and SGD is sensitive to poor conditioning, which means a small input can change the gradient dramatically. Recently, the use of the Alternating Direction Method of Multipliers (ADMM) has been proposed as an alternative to SGD. 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 [3]. 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 [19].

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.** Despite the fact that 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 directly 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 [3]. It is often the case that ADMM becomes trapped in a modest solution and hence performs worse than SGD, as the experiment described later in this paper in Section 5 demonstrates. **3. Cubic time complexity with regard to 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 [19]. This computational bottleneck mainly originates from the matrix inversion required to update the weight parameters. Computing an inverse matrix needs further subiterations, and its time complexity is approximately $O(n^3)$, where n is a feature dimension [3].

In order to deal with these difficulties simultaneously, in this paper we propose a novel optimization framework for a deep learning Alternating Direction Method of Multipliers (dlADMM) algorithm. Specifically, our new dlADMM algorithm updates parameters first in a backward direction and then forwards. This update approach propagates parameter information across the whole network and accelerates the convergence process. It also avoids the operation of

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330936

Table 1: Important Notations and Descriptions

Notations	Descriptions
L	Number of layers.
W_l	The weight matrix for the l-th layer.
b_l	The intercept vector for the l-th layer.
z_l	The temporary variable of the linear mapping for the l-th layer.
$f_l(z_l)$	The nonlinear activation function for the l-th layer.
a_l	The output for the l-th layer.
x	The input matrix of the neural network.
y	The predefined label vector.
$R(z_L, y)$	The risk function for the l-th layer.
$\Omega_l(W_l)$	The regularization term for the l-th layer.
n_l	The number of neurons for the l-th layer.

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 functions (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 convergence proof are highly flexible for fully-connected deep neural networks, as well as being easily extendable to other popular network architectures such as Convolutional Neural Networks [10] and Recurrent Neural Networks [13]. Our contributions in this paper include:

- We present a novel and efficient dlADMM algorithm to handle the fully-connected deep neural network 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 reducing 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 critical 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 SGD and its variants.

The rest of this paper is organized as follows. In Section 2, we summarize recent research related to this topic. In Section 3, we present the new dlADMM algorithm, quadratic approximation, and the backtracking techniques utilized. In Section 4, we introduce the main convergence results for the dlADMM algorithm. The results of extensive experiments conducted to show the convergence, efficiency, and effectiveness of our proposed new dlADMM algorithm are presented in in Section 5, and Section 6 concludes this paper by summarizing the research.

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 SGD algorithm and its variants play a dominant role in the research conducted by deep learning optimization community. The famous back-propagation algorithm was firstly introduced by Rumelhart et al. to train the neural network effectively [17]. Since the superior performance exhibited by AlexNet [10] in 2012, deep learning has attracted a great deal of researchers' attention and many new optimizers based on SGD have been proposed to accelerate the convergence process, including the use of Polyak momentum [14], as well as research on the Nesterov momentum and initialization by Sutskever et al. [18]. Adam is the most popular method because it is computationally efficient and requires little tuning [9]. Other well-known methods that incorporate adaptive learning rates include AdaGrad [6], RMSProp [20], and AMSGrad [15]. Recently, the Alternating Direction Method of Multipliers (ADMM) has become popular with researchers due to its excellent scalability [19]. However, even though these optimizers perform well in real-world applications, their convergence mechanisms remain mysterious. This is because convergence assumptions are not applicable to deep learning problems, which often require non-differentiable activation functions such as the Rectifier linear unit (ReLU).

ADMM for nonconvex problems: The good performance achieved by ADMM over a range wide of convex problems has attracted the attention of many researchers, who have now begun to investigate the behavior of ADMM on nonconvex problems and made significant advances. For example, Wang et al. proposed an ADMM to solve multi-convex problems with a convergence guarantee [22], while Wang et al. presented convergence conditions for a coupled objective function that is nonconvex and nonsmooth [23]. Chen et al. discussed the use of ADMM to solve problems with quadratic coupling terms [4] and Wang et al. studied the convergence behavior of the ADMM for problems with nonlinear equality constraints [21]. Even though ADMM has been proposed to solve deep learning applications [7, 19], there remains a lack theoretical convergence analysis for the application of ADMM to such problems.

3 THE DLADMM ALGORITHM

We present our proposed dlADMM algorithm in this section. Section 3.1 formulates the deep neural network problem, Section 3.2 introduces how the dlADMM algorithm works, and the quadratic approximation and backtracking techniques used to solve the subproblems are presented in Section 3.3.

3.1 Problem Formulation

Table 1 lists the important notation utilized in this paper. Even though there are many variants of formulations for deep neural networks, a typical neural network is defined by multiple linear mappings and nonlinear activation functions. A linear mapping for the l-th layer is composed of a weight matrix $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$ and an intercept vector $b_l \in \mathbb{R}^{n_l}$, 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

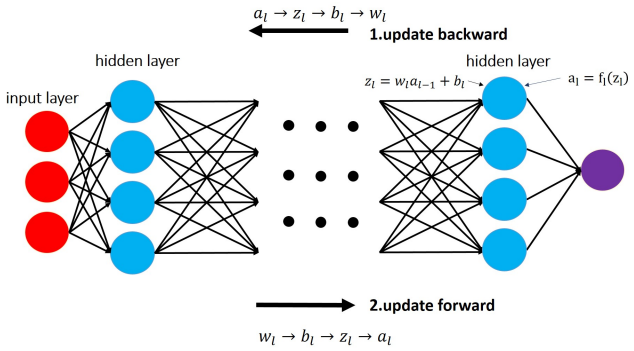


Figure 1: The dlADMM framework overview: update parameter backward and then forward.

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 $a_l = f_l(W_l a_{l-1} + b_l)$. Obviously, a_{l-1} is nested in $a_l = f_l(\bullet)$. By introducing an auxiliary variable z_l, the task of training a deep neural network problem is formulated mathematically as follows:

Problem 1.

$$\min_{W_l, b_l, z_l, a_l} R(z_L; y) + \sum_{l=1}^{L} \Omega_l(W_l)$$
$$s.t. z_l = W_l a_{l-1} + b_l (l = 1, \cdots, L),\ a_l = f_l(z_l)(l = 1, \cdots, L-1)$$

In Problem 1, $a_0 = x \in \mathbb{R}^{n_0}$ is the input of the deep neural network 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, and $\Omega_l(W_l)$ is a regularization term for the l-th layer, which is also convex, continuous, and proper. Rather than solving Problem 1 directly, we can relax Problem 1 by adding an ℓ_2 penalty to address Problem 2 as follows:

Problem 2.

$$\min_{W_l, b_l, z_l, a_l} F(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{z}, \boldsymbol{a}) = R(z_L; y) + \sum_{l=1}^{L} \Omega_l(W_l)$$
$$+ (\nu/2) \sum_{l=1}^{L-1} (\|z_l - W_l a_{l-1} - b_l\|_2^2 + \|a_l - f_l(z_l)\|_2^2)$$
$$s.t.\ z_L = W_L a_{L-1} + b_L$$

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

3.2 The dlADMM algorithm

We introduce the 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 [19]. 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 towards the previous layers to speed up convergence. To achieve this, we propose our novel dlADMM framework, as shown in Figure 1. Specifically, the dlADMM algorithm updates parameters in two steps. In the first, the 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 $a_l \rightarrow z_l \rightarrow b_l \rightarrow W_l$. In the second, the 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 $W_l \rightarrow b_l \rightarrow z_l \rightarrow a_l$. 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:

$$L_\rho(\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{a},u) = R(z_L; y) + \sum_{l=1}^{L} \Omega_l(W_l) + \phi(\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{a},u) \quad (1)$$

where $\phi(\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{a},u) = (\nu/2) \sum_{l=1}^{L-1} (\|z_l - W_l a_{l-1} - b_l\|_2^2 + \|a_l - f_l(z_l)\|_2^2) + u^T(z_L - W_L a_{L-1} - b_L) + (\rho/2)\|z_L - W_L a_{L-1} - b_L\|_2^2$, u is a dual variable and $\rho > 0$ is a hyperparameter of the dlADMM algorithm. We denote $\overline{W}_l^{k+1}$, $\overline{b}_l^{k+1}$, $\overline{z}_l^{k+1}$ and $\overline{a}_l^{k+1}$ as the backward update of the dlADMM for the l-th layer in the $(k+1)$-th iteration , while W_l^{k+1}, b_l^{k+1}, z_l^{k+1} and a_l^{k+1} are denoted as the forward update of the dlADMM for the l-th layer in the $(k+1)$-th iteration. Moreover, we denote $\overline{\mathbf{W}}_l^{k+1} = \{\{W_i^k\}_{i=1}^{l-1}, \{\overline{W}_i^{k+1}\}_{i=l}^{L}\}$, $\overline{\mathbf{b}}_l^{k+1} = \{\{b_i^k\}_{i=1}^{l-1}, \{\overline{b}_i^{k+1}\}_{i=l}^{L}\}$, $\overline{\mathbf{z}}_l^{k+1} = \{\{z_i^k\}_{i=1}^{l-1}, \{\overline{z}_i^{k+1}\}_{i=l}^{L}\}$, $\overline{\mathbf{a}}_l^{k+1} = \{\{a_i^k\}_{i=1}^{l-1}, \{\overline{a}_i^{k+1}\}_{i=l}^{L-1}\}$, $\mathbf{W}_l^{k+1} = \{\{W_i^{k+1}\}_{i=1}^{l}, \{\overline{W}_i^{k+1}\}_{i=l+1}^{L}\}$, $\mathbf{b}_l^{k+1} = \{\{b_i^{k+1}\}_{i=1}^{l}, \{\overline{b}_i^{k+1}\}_{i=l+1}^{L}\}$, $\mathbf{z}_l^{k+1} = \{\{z_i^{k+1}\}_{i=1}^{l}, \{\overline{z}_i^{k+1}\}_{i=l+1}^{L}\}$, $\mathbf{a}_l^{k+1} = \{\{a_i^{k+1}\}_{i=1}^{l}, \{\overline{a}_i^{k+1}\}_{i=l+1}^{L-1}\}$, $\overline{\mathbf{W}}^{k+1} = \{\overline{W}_i^{k+1}\}_{i=1}^{L}$, $\overline{\mathbf{b}}^{k+1} = \{\overline{b}_i^{k+1}\}_{i=1}^{L}$, $\overline{\mathbf{z}}^{k+1} = \{\overline{z}_i^{k+1}\}_{i=1}^{L}$, $\overline{\mathbf{a}}^{k+1} = \{\overline{a}_i^{k+1}\}_{i=1}^{L-1}$, $\mathbf{W}^{k+1} = \{W_i^{k+1}\}_{i=1}^{L}$, $\mathbf{b}^{k+1} = \{b_i^{k+1}\}_{i=1}^{L}$, $\mathbf{z}^{k+1} = \{z_i^{k+1}\}_{i=1}^{L}$, and $\mathbf{a}^{k+1} = \{a_i^{k+1}\}_{i=1}^{L-1}$. Then the dlADMM algorithm is shown in Algorithm 1. Specifically, Lines 5, 6, 10, 11, 14, 15, 17 and 18 solve eight subproblems, namely, $\overline{a}_l^{k+1}$, $\overline{z}_l^{k+1}$, $\overline{b}_l^{k+1}$, $\overline{W}_l^{k+1}$, W_l^{k+1}, b_l^{k+1}, z_l^{k+1} and a_l^{k+1}, respectively. Lines 21 and 22 update the residual r^{k+1} and the dual variable u^{k+1}, respectively.

3.3 The Quadratic Approximation and Backtracking

The eight 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 $\overline{a}_l^{k+1}$

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

$$\overline{a}_l^{k+1} \leftarrow \arg\min_{a_l} L_\rho(\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_{l+1}^{k+1}, \overline{\mathbf{z}}_{l+1}^{k+1}, \{a_i^k\}_{i=1}^{l-1}, a_l, \{\overline{a}_i^{k+1}\}_{i=l+1}^{L-1}, u^k)$$

The subproblem is transformed into the following form after it is replaced by Equation (1).

$$\overline{a}_l^{k+1} \leftarrow \arg\min_{a_l} \phi(\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_{l+1}^{k+1}, \overline{\mathbf{z}}_{l+1}^{k+1}, \{a_i^k\}_{i=1}^{l-1}, a_l, \{\overline{a}_i^{k+1}\}_{i=l+1}^{L-1}, u^k) \quad (2)$$

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 $\overline{W}_{l+1}^{k+1}$, which

Algorithm 1 the dlADMM Algorithm to Solve Problem 2

Require: y, $a_0 = x$, ρ, ν.
Ensure: $a_l (l = 1, \cdots, L-1)$, $W_l (l = 1, \cdots, L)$, $b_l (l = 1, \cdots, L)$, $z_l (l = 1, \cdots, L)$.
Initialize $k = 0$.
while $\mathbf{W}^{k+1}$, $\mathbf{b}^{k+1}$, $\mathbf{z}^{k+1}$, $\mathbf{a}^{k+1}$ not converged **do**
  for $l = L$ to 1 **do**
    if $l < L$ **then**
      Update $\overline{a}_l^{k+1}$ in Equation (3).
      Update $\overline{z}_l^{k+1}$ in Equation (4).
      Update $\overline{b}_l^{k+1}$ in Equation (6).
    else
      Update $\overline{z}_L^{k+1}$ in Equation (5).
      Update $\overline{b}_L^{k+1}$ in Equation (7).
    end if
    Update $\overline{W}_l^{k+1}$ in Equation (9).
  end for
  for $l = 1$ to L **do**
    Update W_l^{k+1} in Equation (11).
    if $l < L$ **then**
      Update b_l^{k+1} in Equation (12).
      Update z_l^{k+1} in Equation (14).
      Update a_l^{k+1} in Equation (16).
    else
      Update b_l^{k+1} in Equation (13).
      Update z_L^{k+1} in Equation (15).
      $r^{k+1} \leftarrow z_L^{k+1} - W_L^{k+1} a_{L-1}^{k+1} - b_L^{k+1}$.
      $u^{k+1} \leftarrow u^k + \rho r^{k+1}$.
    end if
  end for
  $k \leftarrow k + 1$.
end while
Output **W, b, z, a**.

involves subiterations and is computationally expensive [19]. In order to handle this challenge, we define $\overline{Q}_l(a_l; \overline{\tau}_l^{k+1})$ as a quadratic approximation of ϕ at a_l^k, which is mathematically reformulated as follows:

$$\begin{aligned}\overline{Q}_l(a_l; \overline{\tau}_l^{k+1}) = \phi(&\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_{l+1}^{k+1}, \overline{\mathbf{z}}_{l+1}^{k+1}, \overline{\mathbf{a}}_{l+1}^{k+1}, u^k) \\ &+ (\nabla_{a_l^k}\phi)^T(\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_{l+1}^{k+1}, \overline{\mathbf{z}}_{l+1}^{k+1}, \overline{\mathbf{a}}_{l+1}^{k+1}, u^k)(a_l - a_l^k) \\ &+ \|\overline{\tau}_l^{k+1} \circ (a_l - a_l^k)^{\circ 2}\|_1/2\end{aligned}$$

where $\overline{\tau}_l^{k+1} > 0$ is a parameter vector, $\circ$ denotes the Hadamard product (the elementwise product), and $a^{\circ b}$ denotes a to the Hadamard power of b and $\| \bullet \|_1$ is the ℓ_1 norm. $\nabla_{a_l^k}\phi$ is the gradient of $\overline{a}_l$ at a_l^k. Obviously, $\overline{Q}_l(a_l^k; \overline{\tau}_l^{k+1}) = \phi(\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_{l+1}^{k+1}, \overline{\mathbf{z}}_{l+1}^{k+1}, \overline{\mathbf{a}}_{l+1}^{k+1}, u^k)$. Rather than minimizing the original problem in Equation (2), we instead solve the following problem:

$$\overline{a}_l^{k+1} \leftarrow \arg\min_{a_l} \overline{Q}_l(a_l; \overline{\tau}_l^{k+1}) \tag{3}$$

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

$$\overline{a}_l^{k+1} \leftarrow a_l^k - \nabla_{a_l^k}\phi / \overline{\tau}_l^{k+1}$$

given a suitable $\overline{\tau}_l^{k+1}$. Now the main focus is how to choose $\overline{\tau}_l^{k+1}$. Algorithm 2 shows the backtracking algorithm utilized to find a suitable $\overline{\tau}_l^{k+1}$. Lines 2-5 implement a while loop until the condition $\phi(\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_{l+1}^{k+1}, \overline{\mathbf{z}}_{l+1}^{k+1}, \overline{\mathbf{a}}_l^{k+1}, u^k) \leq \overline{Q}_l(\overline{a}_l^{k+1}; \overline{\tau}_l^{k+1})$ is satisfied. As $\overline{\tau}_l^{k+1}$ becomes larger and larger, $\overline{a}_l^{k+1}$ 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 $\overline{a}_l^{k+1}$

Require: $\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_{l+1}^{k+1}, \overline{\mathbf{z}}_{l+1}^{k+1}, \overline{\mathbf{a}}_{l+1}^{k+1}, u^k, \rho$, some constant $\overline{\eta} > 1$.
Ensure: $\overline{\tau}_l^{k+1}, \overline{a}_l^{k+1}$.
1: Pick up $\overline{t}$ and $\overline{\beta} = a_l^k - \nabla_{a_l^k}\phi / \overline{t}$
2: **while** $\phi(\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_{l+1}^{k+1}, \overline{\mathbf{z}}_{l+1}^{k+1}, \{a_i^k\}_{i=1}^{l-1}, \overline{\beta}, \{\overline{a}_i^{k+1}\}_{i=l+1}^{L-1}, u^k) > \overline{Q}_l(\overline{\beta}; \overline{t})$ **do**
3: $\overline{t} \leftarrow \overline{t}\overline{\eta}$.
4: $\overline{\beta} \leftarrow a_l^k - \nabla_{a_l^k}\phi / \overline{t}$.
5: **end while**
6: Output $\overline{\tau}_l^{k+1} \leftarrow \overline{t}$.
7: Output $\overline{a}_l^{k+1} \leftarrow \overline{\beta}$.

2. Update $\overline{z}_l^{k+1}$

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

$$\overline{z}_l^{k+1} \leftarrow \arg\min_{z_l} L_\rho(\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_{l+1}^{k+1}, \{z_i^k\}_{i=1}^{l-1}, z_l, \{\overline{z}_i^{k+1}\}_{i=l+1}^{L}, \overline{\mathbf{a}}_l^{k+1}, u^k)$$

which is equivalent to the following forms: for $\overline{z}_l^{k+1} (l = 1, \cdots, L-1)$,

$$\overline{z}_l^{k+1} \leftarrow \arg\min_{z_l} \phi(\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_{l+1}^{k+1}, \{z_i^k\}_{i=1}^{l-1}, z_l, \{\overline{z}_i^{k+1}\}_{i=l+1}^{L}, \overline{\mathbf{a}}_l^{k+1}, u^k) \tag{4}$$

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

$$\overline{z}_L^{k+1} \leftarrow \arg\min_{z_L} \phi(\mathbf{W}^k, \mathbf{b}^k, \{z_i^k\}_{i=1}^{L-1}, z_L, \mathbf{a}^k, u^k) + R(z_L; y) \tag{5}$$

Equation (4) is highly nonconvex because the nonlinear activation function $f(z_l)$ is contained in $\phi(\bullet)$. For common activation functions such as the Rectified linear unit (ReLU) and leaky ReLU, Equation (4) has a closed-form solution; for other activation functions like sigmoid and hyperbolic tangent (tanh), a look-up table is recommended [19].

Equation (5) is a convex problem because $\phi(\bullet)$ and $R(\bullet)$ are convex with regard to z_L. Therefore, Equation (5) can be solved by Fast Iterative Soft-Thresholding Algorithm (FISTA) [1].

3. Update $\overline{b}_l^{k+1}$

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

$$\overline{b}_l^{k+1} \leftarrow \arg\min_{b_l} L_\rho(\overline{\mathbf{W}}_{l+1}^{k+1}, \{b_i^k\}_{i=1}^{l-1}, b_l, \{\overline{b}_i^{k+1}\}_{i=l+1}^{L}, \overline{\mathbf{z}}_l^{k+1}, \overline{\mathbf{a}}_l^{k+1}, u^k)$$

which is equivalent to the following form:

$$\overline{b}_l^{k+1} \leftarrow \arg\min_{b_l} \phi(\overline{\mathbf{W}}_{l+1}^{k+1}, \{b_i^k\}_{i=1}^{l-1}, b_l, \{\overline{b}_i^{k+1}\}_{i=l+1}^{L}, \overline{\mathbf{z}}_l^{k+1}, \overline{\mathbf{a}}_l^{k+1}, u^k)$$

Similarly to the update of $\overline{a}_l^{k+1}$, we define $\overline{U}_l(b_l; \overline{B})$ as a quadratic approximation of $\phi(\bullet)$ at b_l^k, which is formulated mathematically as follows [1]:

$$\begin{aligned}\overline{U}_l(b_l; \overline{B}) = \phi(&\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_{l+1}^{k+1}, \overline{\mathbf{z}}_l^{k+1}, \overline{\mathbf{a}}_l^{k+1}, u^k) \\ &+ (\nabla_{b_l^k}\phi)^T(\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_{l+1}^{k+1}, \overline{\mathbf{z}}_l^{k+1}, \overline{\mathbf{a}}_l^{k+1}, u^k)(b_l - b_l^k) \\ &+ (\overline{B}/2)\|b_l - b_l^k\|_2^2.\end{aligned}$$

where $\overline{B} > 0$ is a parameter. Here $\overline{B} \geq \nu$ for $l = 1, \cdots, L-1$ and $\overline{B} \geq \rho$ for $l = L$ are required for the convergence analysis [1].

Without loss of generality, we set $\overline{B} = \nu$, and solve the subsequent subproblem as follows:

$$\overline{b}_l^{k+1} \leftarrow \arg\min_{b_l} \overline{U}_l(b_l; \nu)(l = 1, \cdots, L-1) \tag{6}$$

$$\overline{b}_L^{k+1} \leftarrow \arg\min_{b_L} \overline{U}_L(b_L; \rho) \tag{7}$$

Equation (6) is a convex problem and has a closed-form solution as follows:

$$\overline{b}_l^{k+1} \leftarrow b_l^k - \nabla_{b_l^k}\phi/\nu.(l = 1, \cdots, L-1)$$

$$\overline{b}_L^{k+1} \leftarrow b_L^k - \nabla_{b_L^k}\phi/\rho.$$

4. Update $\overline{W}_l^{k+1}$

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

$$\overline{W}_l^{k+1} \leftarrow \text{argmin}_{W_l} L_\rho(\{W_i^k\}_{i=1}^{l-1}, W_l, \{\overline{W}_i^{k+1}\}_{i=l+1}^{L}, \overline{\mathbf{b}}_l^{k+1}, \overline{\mathbf{z}}_l^{k+1}, \overline{\mathbf{a}}_l^{k+1}, u^k)$$

which is equivalent to the following form:

$$\overline{W}_l^{k+1} \leftarrow \text{argmin}_{W_l} \phi(\{W_i^k\}_{i=1}^{l-1}, W_l, \{\overline{W}_i^{k+1}\}_{i=l+1}^{L}, \overline{\mathbf{b}}_l^{k+1}, \overline{\mathbf{z}}_l^{k+1}, \overline{\mathbf{a}}_l^{k+1}, u^k) + \Omega(W_l) \tag{8}$$

Due to the same challenge in updating $\overline{a}_l^{k+1}$, we define $\overline{P}_l(W_l; \overline{\theta}_l^{k+1})$ as a quadratic approximation of ϕ at W_l^k. The quadratic approximation is mathematically reformulated as follows [1]:

$$\begin{aligned}\overline{P}_l(W_l; \overline{\theta}_l^{k+1}) = {} & \phi(\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_l^{k+1}, \overline{\mathbf{z}}_l^{k+1}, \overline{\mathbf{a}}_l^{k+1}, u^k) \\ & + (\nabla_{W_l^k}\phi)^T(\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_l^{k+1}, \overline{\mathbf{z}}_l^{k+1}, \overline{\mathbf{a}}_l^{k+1}, u^k)(W_l - W_l^k) \\ & + \|\overline{\theta}_l^{k+1} \circ (W_l - W_l^k)^{\circ 2}\|_1/2\end{aligned}$$

where $\overline{\theta}_l^{k+1} > 0$ is a parameter vector, which is chosen by the Algorithm 3. Instead of minimizing the Equation (8), we minimize the following:

$$\overline{W}_l^{k+1} \leftarrow \arg\min_{W_l} \overline{P}_l(W_l; \overline{\theta}_l^{k+1}) + \Omega_l(W_l) \tag{9}$$

Equation (9) is convex and hence can be solved exactly. If Ω_l is either an ℓ_1 or an ℓ_2 regularization term, Equation (9) has a closed-form solution.

Algorithm 3 The Backtracking Algorithm to update $\overline{W}_l^{k+1}$

Require: $\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_l^{k+1}, \overline{\mathbf{z}}_l^{k+1}, \overline{\mathbf{a}}_l^{k+1}, u^k$, ρ, some constant $\overline{\gamma} > 1$.
Ensure: $\overline{\theta}_l^{k+1}, \overline{W}_l^{k+1}$.
1: Pick up $\overline{\alpha}$ and $\overline{\zeta} = W_l^k - \nabla_{W_l^k}\phi/\overline{\alpha}$.
2: **while** $\phi(\{W_i^k\}_{i=1}^{l-1}, \overline{\zeta}, \{\overline{W}_i^{k+1}\}_{i=l+1}^{L}, \overline{\mathbf{b}}_l^{k+1}, \overline{\mathbf{z}}_l^{k+1}, \overline{\mathbf{a}}_l^{k+1}, u^k) > \overline{P}_l(\overline{\zeta}; \overline{\alpha})$ **do**
3: $\overline{\alpha} \leftarrow \overline{\alpha}\,\overline{\gamma}$.
4: Solve $\overline{\zeta}$ by Equation (9).
5: **end while**
6: Output $\overline{\theta}_l^{k+1} \leftarrow \overline{\alpha}$.
7: Output $\overline{W}_l^{k+1} \leftarrow \overline{\zeta}$.

5. Update W_l^{k+1}

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

$$W_l^{k+1} \leftarrow \text{argmin}_{W_l} L_\rho(\{W_i^{k+1}\}_{i=1}^{l-1}, W_l, \{\overline{W}_i^{k+1}\}_{i=l+1}^{L}, \mathbf{b}_{l-1}^{k+1}, \mathbf{z}_{l-1}^{k+1}, \mathbf{a}_{l-1}^{k+1}, u^k)$$

which is equivalent to

$$W_l^{k+1} \leftarrow \text{argmin}_{W_l} \phi(\{W_i^{k+1}\}_{i=1}^{l-1}, W_l, \{\overline{W}_i^{k+1}\}_{i=l+1}^{L}, \mathbf{b}_{l-1}^{k+1}, \mathbf{z}_{l-1}^{k+1}, \mathbf{a}_{l-1}^{k+1}, u^k) + \Omega(W_l) \tag{10}$$

Similarly, we define $P_l(W_l; \theta_l^{k+1})$ as a quadratic approximation of ϕ at $\overline{W}_l^{k+1}$. The quadratic approximation is then mathematically reformulated as follows [1]:

$$\begin{aligned}P_l(W_l; \theta_l^{k+1}) = {} & \phi(\mathbf{W}_{l-1}^{k+1}, \mathbf{b}_{l-1}^{k+1}, \mathbf{z}_{l-1}^{k+1}, \mathbf{a}_{l-1}^{k+1}, u^k) \\ & + (\nabla_{\overline{W}_l^{k+1}}\phi)^T(\mathbf{W}_{l-1}^{k+1}, \mathbf{b}_{l-1}^{k+1}, \mathbf{z}_{l-1}^{k+1}, \mathbf{a}_{l-1}^{k+1}, u^k)(W_l - \overline{W}_l^{k+1}) \\ & + \|\theta_l^{k+1} \circ (W_l - \overline{W}_l^{k+1})^{\circ 2}\|_1/2\end{aligned}$$

where $\theta_l^{k+1} > 0$ is a parameter vector. Instead of minimizing the Equation (10), we minimize the following:

$$W_l^{k+1} \leftarrow \arg\min_{W_l} P_l(W_l; \theta_l^{k+1}) + \Omega_l(W_l) \tag{11}$$

The choice of θ_l^{k+1} is discussed in the supplementary materials[1].

6. Update b_l^{k+1}

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

$$b_l^{k+1} \leftarrow \arg\min_{b_l} L_\rho(\mathbf{W}_l^{k+1}, \{b_i^{k+1}\}_{i=1}^{l-1}, b_l, \{\overline{b}_i^{k+1}\}_{i=l+1}^{L}, \mathbf{z}_{l-1}^{k+1}, \mathbf{a}_{l-1}^{k+1}, u^k)$$

which is equivalent to the following formulation:

$$b_l^{k+1} \leftarrow \arg\min_{b_l} \phi(\mathbf{W}_l^{k+1}, \{b_i^{k+1}\}_{i=1}^{l-1}, b_l, \{\overline{b}_i^{k+1}\}_{i=l+1}^{L}, \mathbf{z}_{l-1}^{k+1}, \mathbf{a}_{l-1}^{k+1}, u^k)$$

$U_l(b_l; B)$ is defined as the quadratic approximation of ϕ at $\overline{b}_l^{k+1}$ as follows:

$$\begin{aligned}U_l(b_l; B) = {} & \phi(\mathbf{W}_l^{k+1}, \mathbf{b}_{l-1}^{k+1}, \mathbf{z}_{l-1}^{k+1}, \mathbf{a}_{l-1}^{k+1}, u^k) \\ & + \nabla_{\overline{b}_l^{k+1}}\phi^T(\mathbf{W}_l^{k+1}, \mathbf{b}_{l-1}^{k+1}, \mathbf{z}_{l-1}^{k+1}, \mathbf{a}_{l-1}^{k+1}, u^k)(b_l - \overline{b}_l^{k+1}) \\ & + (B/2)\|b_l - \overline{b}_l^{k+1}\|_2^2.\end{aligned}$$

where $B > 0$ is a parameter. We set $B = \nu$ for $l = 1, \cdots, L-1$ and $B = \rho$ for $l = L$, and solve the resulting subproblems as follows:

$$b_l^{k+1} \leftarrow \arg\min_{b_l} U_l(b_l; \nu)(l = 1, \cdots, L-1) \tag{12}$$

$$b_L^{k+1} \leftarrow \arg\min_{b_L} U_L(b_L; \rho) \tag{13}$$

. The solutions to Equations (12) and (13) are as follows:

$$b_l^{k+1} \leftarrow \overline{b}_l^{k+1} - \nabla_{\overline{b}_L^{k+1}}\phi/\nu(l = 1, \cdots, L-1)$$

$$b_L^{k+1} \leftarrow \overline{b}_L^{k+1} - \nabla_{\overline{b}_L^{k+1}}\phi/\rho$$

7. Update z_l^{k+1}

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

$$z_l^{k+1} \leftarrow \arg\min_{z_l} L_\rho(\mathbf{W}_l^{k+1}, \mathbf{b}_l^{k+1}, \{z_i^{k+1}\}_{i=1}^{l-1}, z_l, \{\overline{z}_i^{k+1}\}_{i=l+1}^{L}, \mathbf{a}_{l-1}^{k+1}, u^k)$$

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

$$z_l^{k+1} \leftarrow \arg\min_{z_l} \phi(\mathbf{W}_l^{k+1}, \mathbf{b}_l^{k+1}, \{z_i^{k+1}\}_{i=1}^{l-1}, z_l, \{\overline{z}_i^{k+1}\}_{i=l+1}^{L-1}, \mathbf{a}_{l-1}^{k+1}, u^k) \tag{14}$$

and for z_L:

$$\begin{aligned}z_L^{k+1} \leftarrow {} & \arg\min_{z_L} \phi(\mathbf{W}_L^{k+1}, \mathbf{b}_L^{k+1}, \{z_i^{k+1}\}_{i=1}^{L-1}, z_L, \mathbf{a}_{L-1}^{k}, u^k) \\ & + R(z_L; y)\end{aligned} \tag{15}$$

[1] The supplementary materials are available at http://mason.gmu.edu/~lzhao9/materials/papers/dlADMM_supp.pdf

Solving Equations (14) and (15) proceeds exactly the same as solving Equations (4) and (5), respectively.

8. Update a_l^{k+1}

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

$$a_l^{k+1} \leftarrow \arg\min_{a_l} L_\rho(\mathbf{W}_l^{k+1}, \mathbf{b}_l^{k+1}, \mathbf{z}_l^{k+1}, \{a_i^k\}_{i=1}^{l-1}, a_l, \{\overline{a}_i^{k+1}\}_{i=l+1}^{L-1}, u^k)$$

which is equivalent to the following form:

$$a_l^{k+1} \leftarrow \arg\min_{a_l} \phi(\mathbf{W}_l^{k+1}, \mathbf{b}_l^{k+1}, \mathbf{z}_l^{k+1}, \{a_i^k\}_{i=1}^{l-1}, a_l, \{\overline{a}_i^{k+1}\}_{i=l+1}^{L-1}, u^k)$$

$Q_l(a_l; \tau_l^{k+1})$ is defined as the quadratic approximation of ϕ at a_l^{k+1} as follows:

$$\begin{aligned} Q_l(a_l; \tau_l^{k+1}) = \phi(&\mathbf{W}_l^{k+1}, \mathbf{b}_l^{k+1}, \mathbf{z}_l^{k+1}, \mathbf{a}_{l-1}^{k+1}, u^k) \\ &+ (\nabla_{\overline{a}_l^{k+1}} \phi)^T (\mathbf{W}_l^{k+1}, \mathbf{b}_l^{k+1}, \mathbf{z}_l^{k+1}, \mathbf{a}_{l-1}^{k+1}, u^k)(a_l - \overline{a}_l^{k+1}) \\ &+ \|\tau_l^{k+1} \circ (a_l - \overline{a}_l^{k+1})^{\circ 2}\|_1/2 \end{aligned}$$

and we can solve the following problem instead:

$$a_l^{k+1} \leftarrow \arg\min_{a_l} Q_l(a_l; \tau_l^{k+1}) \tag{16}$$

where $\tau_l^{k+1} > 0$ is a parameter vector. The solution to Equation (16) can be obtained by

$$a_l^{k+1} \leftarrow \overline{a}_l^{k+1} - \nabla_{\overline{a}_l^{k+1}} \phi / \tau_l^{k+1}$$

To choice of an appropriate τ_l^{k+1} is shown in the supplementary materials[1].

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 dlADMM algorithms, Section 4.1 presents necessary assumptions to guarantee the global convgerence of dlADMM. In Section 4.2, we prove the global convergence of the dlADMM algorithm.

4.1 Assumptions

Assumption 1 (Closed-form Solution). *There exist activation functions $a_l = f_l(z_l)$ such that Equations (4) and (14) have closed form solutions $\overline{z}_l^{k+1} = \overline{h}(\overline{\mathbf{W}}_{l+1}^{k+1}, \overline{\mathbf{b}}_{l+1}^{k+1}, \overline{\mathbf{a}}_l^{k+1})$ and $z_l^{k+1} = h(\mathbf{W}_l^{k+1}, \mathbf{b}_l^{k+1}, \mathbf{a}_{l-1}^{k+1})$, 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 (14) has the following solution:

$$z_l^{k+1} = \begin{cases} \min(W_l^{k+1} a_{l-1}^{k+1} + b_l^{k+1}, 0) & z_l^{k+1} \le 0 \\ \max((W_l^{k+1} a_{l-1}^{k+1} + b_l^{k+1} + \overline{a}_l^{k+1})/2, 0) & z_l^{k+1} \ge 0 \end{cases}$$

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

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

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 the supplementary materials[1].

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

Property 1 concludes that all variables and the value of L_ρ have lower bounds. It is proven under Assumptions 1 and 2, and its proof can be found in the supplementary materials[1].

Property 2 (Sufficient Descent). *If $\rho > 2H$ so that $C_1 = \rho/2 - H/2 - H^2/\rho > 0$, then there exists $C_2 = \min(\nu/2, C_1, \{\overline{\theta}_l^{k+1}\}_{l=1}^L, \{\theta_l^{k+1}\}_{l=1}^L, \{\overline{\tau}_l^{k+1}\}_{l=1}^{L-1}, \{\tau_l^{k+1}\}_{l=1}^{L-1})$ such that*

$$\begin{aligned} &L_\rho(\boldsymbol{W}^k, \boldsymbol{a}^k, \boldsymbol{z}^k, \boldsymbol{a}^k, u^k) - L_\rho(\boldsymbol{W}^{k+1}, \boldsymbol{a}^{k+1}, \boldsymbol{z}^{k+1}, \boldsymbol{a}^{k+1}, u^{k+1}) \\ &\ge C_2\Big(\sum_{l=1}^L (\|\overline{W}_l^{k+1} - W_l^k\|_2^2 + \|W_l^{k+1} - \overline{W}_l^{k+1}\|_2^2 \\ &+ \|\overline{b}_l^{k+1} - b_l^k\|_2^2 + \|b_l^{k+1} - \overline{b}_l^{k+1}\|_2^2) \\ &+ \sum_{l=1}^{L-1} (\|\overline{a}_l^{k+1} - a_l^k\|_2^2 + \|a_l^{k+1} - \overline{a}_l^{k+1}\|_2^2) \\ &+ \|\overline{z}_L^{k+1} - z_L^k\|_2^2 + \|z_L^{k+1} - \overline{z}_L^{k+1}\|_2^2\Big) \end{aligned} \tag{17}$$

Property 2 depicts the monotonic decrease of the objective value during iterations. The proof of Property 2 is detailed in the supplementary materials[1].

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

$$\begin{aligned} \|g\| \le C(&\|\boldsymbol{W}^{k+1} - \overline{\boldsymbol{W}}^{k+1}\| + \|\boldsymbol{b}^{k+1} - \overline{\boldsymbol{b}}^{k+1}\| \\ &+ \|\boldsymbol{z}^{k+1} - \overline{\boldsymbol{z}}^{k+1}\| + \|\boldsymbol{a}^{k+1} - \overline{\boldsymbol{a}}^{k+1}\| + \|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|) \end{aligned} \tag{18}$$

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 the supplementary materials[1]. Now the global convergence of the dlADMM algorithm is presented. The following theorem states that Properties 1-3 are guaranteed.

Theorem 4.1. *For any $\rho > 2H$, if Assumptions 1 and 2 are satisfied, then Properties 1-3 hold.*

Proof. This theorem can be concluded by the proofs in the supplementary materials[1]. □

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

Theorem 4.2 (Global Convergence). *If $\rho > 2H$, then for the variables $(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{z}, \boldsymbol{a}, u)$ in Problem 2, starting from any $(\boldsymbol{W}^0, \boldsymbol{b}^0, \boldsymbol{z}^0, \boldsymbol{a}^0, u^0)$, it has at least a limit point $(\boldsymbol{W}^*, \boldsymbol{b}^*, \boldsymbol{z}^*, \boldsymbol{a}^*, u^*)$, and any limit point $(\boldsymbol{W}^*, \boldsymbol{b}^*, \boldsymbol{z}^*, \boldsymbol{a}^*, u^*)$ is a critical point of Problem 2. That is, $0 \in \partial L_\rho(\boldsymbol{W}^*, \boldsymbol{b}^*, \boldsymbol{z}^*, \boldsymbol{a}^*, u^*)$. Or equivalently,*

$$\begin{aligned} &z_L^* = W_L^* a_{L-1}^* + b_L^* \\ &0 \in \partial_{\boldsymbol{W}^*} L_\rho(\boldsymbol{W}^*, \boldsymbol{b}^*, \boldsymbol{z}^*, \boldsymbol{a}^*, u^*) \quad \nabla_{\boldsymbol{b}^*} L_\rho(\boldsymbol{W}^*, \boldsymbol{b}^*, \boldsymbol{z}^*, \boldsymbol{a}^*, u^*) = 0 \\ &0 \in \partial_{\boldsymbol{z}^*} L_\rho(\boldsymbol{W}^*, \boldsymbol{b}^*, \boldsymbol{z}^*, \boldsymbol{a}^*, u^*) \quad \nabla_{\boldsymbol{a}^*} L_\rho(\boldsymbol{W}^*, \boldsymbol{b}^*, \boldsymbol{z}^*, \boldsymbol{a}^*, u^*) = 0 \end{aligned}$$

Proof. Because $(\mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{a}^k, u^k)$ is bounded, there exists a subsequence $(\mathbf{W}^s, \mathbf{b}^s, \mathbf{z}^s, \mathbf{a}^s, \mathbf{u}^s)$ such that $(\mathbf{W}^s, \mathbf{b}^s, \mathbf{z}^s, \mathbf{a}^s, u^s) \to (\mathbf{W}^*, \mathbf{b}^*, \mathbf{z}^*, \mathbf{a}^*, u^*)$ where $(\mathbf{W}^*, \mathbf{b}^*, \mathbf{z}^*, \mathbf{a}^*, u^*)$ is a limit point. By Properties 1 and 2, $L_\rho(\mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{a}^k, u^k)$ is non-increasing and lower bounded and hence converges. By Property 2, we prove that $\|\overline{\mathbf{W}}^{k+1} - \mathbf{W}^k\| \to 0$, $\|\overline{\mathbf{b}}^{k+1} - \mathbf{b}^k\| \to 0$, $\|\overline{\mathbf{a}}^{k+1} - \mathbf{a}^k\| \to 0$, $\|\mathbf{W}^{k+1} - \overline{\mathbf{W}}^{k+1}\| \to 0$, $\|\mathbf{b}^{k+1} - \overline{\mathbf{b}}^{k+1}\| \to 0$, and $\|\mathbf{a}^{k+1} - \overline{\mathbf{a}}^{k+1}\| \to 0$, as $k \to \infty$. Therefore $\|\mathbf{W}^{k+1} - \mathbf{W}^k\| \to 0$, $\|\mathbf{b}^{k+1} - \mathbf{b}^k\| \to 0$, and $\|\mathbf{a}^{k+1} - \mathbf{a}^k\| \to 0$, as $k \to \infty$. Moreover, from Assumption 1, we know that $\overline{\mathbf{z}}^{k+1} \to \mathbf{z}^k$ and $\mathbf{z}^{k+1} \to \overline{\mathbf{z}}^{k+1}$ as $k \to \infty$. Therefore, $\mathbf{z}^{k+1} \to \mathbf{z}^k$. We infer there exists $g^k \in \partial L_\rho(\mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{a}^k, u^k)$ 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 (Defintion 8.3 in [16]), we have $0 \in \partial L_\rho(\mathbf{W}^*, \mathbf{b}^*, \mathbf{z}^*, \mathbf{a}^*, u^*)$. In other words, the limit point $(\mathbf{W}^*, \mathbf{b}^*, \mathbf{z}^*, \mathbf{a}^*, u^*)$ is a critical point of L_ρ defined in Equation (1). □

Theorem 4.2 shows that our dlADMM algorithm converges globally for sufficiently large ρ, which is consistent with previous literature [8, 23]. The next theorem shows that the dlADMM converges globally with a sublinear convergence rate $o(1/k)$.

Theorem 4.3 (Convergence Rate). *For a sequence* $(\boldsymbol{W}^k, \boldsymbol{b}^k, \boldsymbol{z}^k, \boldsymbol{a}^k, u^k)$, *define* $c_k = \min_{0 \le i \le k}(\sum_{l=1}^{L}(\|\overline{W}_l^{i+1} - W_l^i\|_2^2 + \|W_l^{i+1} - \overline{W}_l^{i+1}\|_2^2 + \|\overline{b}_l^{i+1} - b_l^i\|_2^2 + \|b_l^{i+1} - \overline{b}_l^{i+1}\|_2^2) + \sum_{l=1}^{L-1}(\|\overline{a}_l^{i+1} - a_l^i\|_2^2 + \|a_l^{i+1} - \overline{a}_l^{i+1}\|_2^2) + \|\overline{z}_L^{i+1} - z_L^i\|_2^2 + \|z_L^{i+1} - \overline{z}_L^{i+1}\|_2^2)$, *then the convergence rate of* c_k *is* $o(1/k)$.

Proof. The proof of this theorem is included in the supplementary materials[1]. □

5 EXPERIMENTS

In this section, we evaluate dlADMM algorithm using benchmark datasets. 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.

5.1 Experiment Setup

5.1.1 Dataset. In this experiment, two benchmark datasets were used for performance evaluation: MNIST [12] and Fashion MNIST [24]. The MNIST dataset has ten classes of handwritten-digit images, which was firstly introduced by Lecun et al. in 1998 [12]. It contains 55,000 training samples and 10,000 test samples with 784 features each, which is provided by the Keras library [5]. Unlike the MNIST dataset, the Fashion MNIST dataset has ten classes of assortment images on the website of Zalando, which is EuropeâĂŹs largest online fashion platform [24]. The Fashion-MNIST dataset consists of 60,000 training samples and 10,000 test samples with 784 features each.

5.1.2 Experiment Settings. We set up a network architecture which contained two hidden layers with 1, 000 hidden units each. The Rectified linear unit (ReLU) was used for the activation function for both network structures. The loss function was set as the deterministic cross-entropy loss. ν was set to 10^{-6}. ρ was initialized as 10^{-6} and was multiplied by 10 every 100 iterations. The number of iteration was set to 200. In the experiment, one iteration means one epoch.

5.1.3 Comparison Methods. Since this paper focuses on fully-connected deep neural networks, SGD and its variants and ADMM are state-of-the-art methods and hence were served as comparison methods. For SGD-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. Stochastic Gradient Descent (SGD) [2]. The SGD and its variants are the most popular deep learning optimizers, whose convergence has been studied extensively in the literature. The learning rate of SGD was set to 10^{-6} for both the MNIST and Fashion MNIST datasets.

2. Adaptive gradient algorithm (Adagrad) [6]. Adagrad is an improved version of SGD: 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.

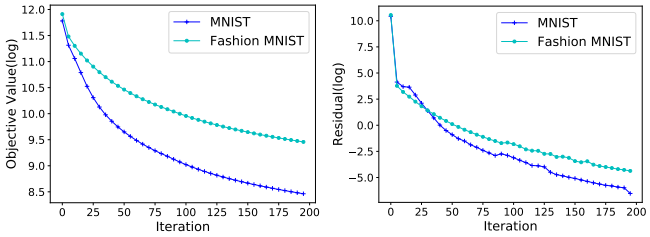
3. Adaptive learning rate method (Adadelta) [25]. As an improved version of the Adagrad, the Adadelta is proposed to overcome the sensitivity to hyperparameter selection. The learning rate of Adadelta was set to 0.1 for both the MNIST and Fashion MNIST datasets.

4. Adaptive momentum estimation (Adam) [9]. 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 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) [19]. 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 ρ of ADMM was set to 1 for both the MNIST and Fashion MNIST datasets.

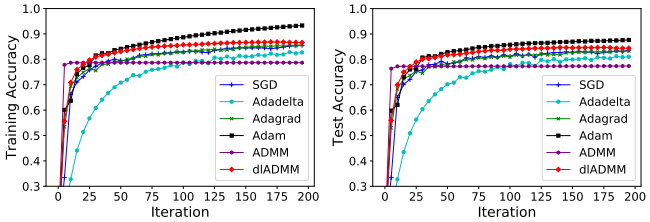
5.2 Experimental Results

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



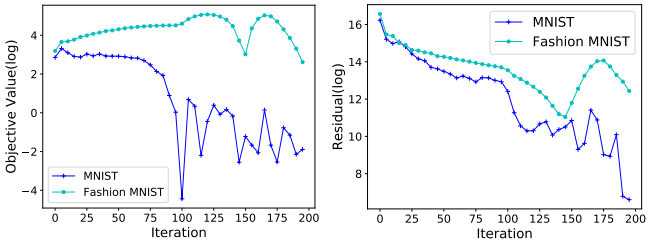
(a). Objective value (b). Residual

Figure 2: Convergence curves of dlADMM algorithm for MNIST and Fashion MNIST datasets when $\rho = 1$: dlADMM algorithm converged.



(a). Training Accuracy (b).Test Accuracy

Figure 5: Performance of all methods for the Fashion MNIST dataset: dlADMM algorithm outperformed most of the comparsion methods.

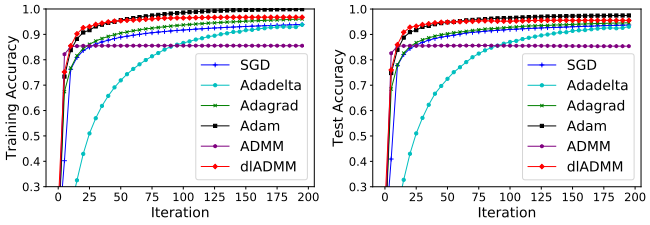


(a). Objective value (b). Residual

Figure 3: Divergence curves of the dlADMM algorithm for the MNIST and the Fashion MNIST datasets when $\rho = 10^{-6}$: dlADMM algorithm diverged.

5.2.1 Convergence. First, we show that our proposed dlADMM algorithm converges when ρ is sufficiently large and diverges when ρ is small for both the MNIST dataset and the Fashion MNIST dataset.

The convergence and divergence of dlADMM algorithm are shown in Figures 2 and 3 when $\rho = 1$ and $\rho = 10^{-6}$,respectively. In Figures 2(a) and 3(a), the X axis and Y axis denote the number of iterations and the logarithm of objective value, respectively. In Figures, 2(b) and 3(b), the X axis and Y axis denote the number of iterations and the logarithm of the residual, respectively. Figure 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 4.2. Moreover, Figure 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.



(a). Training accuracy (b). Test accuracy

Figure 4: Performance of all methods for the MNIST dataset: dlADMM algorithm outperformed most of the comparison methods.

5.2.2 Performance. Figure 4 and Figure 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. Specifically, the curves of our dlADMM algorithm soared to 0.8 at the early stage, and then raised steadily towards to 0.9 or more. The curves of the most SGD-related methods, SGD, 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.

MNIST dataset: From 200 to 1,000 neurons					
ρ \ neurons	200	400	600	800	1000
10^{-6}	1.9025	2.7750	3.6615	4.5709	5.7988
10^{-5}	2.8778	4.6197	6.3620	8.2563	10.0323
10^{-4}	2.2761	3.9745	5.8645	7.6656	9.9221
10^{-3}	2.4361	4.3284	6.5651	8.7357	11.3736
10^{-2}	2.7912	5.1383	7.8249	10.0300	13.4485
Fashion MNIST dataset: From 200 to 1,000 neurons					
ρ \ neurons	200	400	600	800	1000
10^{-6}	2.0069	2.8694	4.0506	5.1438	6.7406
10^{-5}	3.3445	5.4190	7.3785	9.0813	11.0531
10^{-4}	2.4974	4.3729	6.4257	8.3520	10.0728
10^{-3}	2.7108	4.7236	7.1507	9.4534	12.3326
10^{-2}	2.9577	5.4173	8.2518	10.0945	14.3465

Table 2: The relationship between running time per iteration (in second) and the number of neurons for each layer as well as value of ρ when the training size was fixed: generally, the running time increased as the number of neurons and the value of ρ became larger.

MNIST dataset: From 11,000 to 55,000 training samples					
ρ \ size	11,000	22,000	33,000	44,000	55,000
10^{-6}	1.0670	2.0682	3.3089	4.6546	5.7709
10^{-5}	2.3981	3.9086	6.2175	7.9188	10.2741
10^{-4}	2.1290	3.7891	5.6843	7.7625	9.8843
10^{-3}	2.1295	4.1939	6.5039	8.8835	11.3368
10^{-2}	2.5154	4.9638	7.6606	10.4580	13.4021
Fashion MNIST dataset: From 12,000 to 60,000 training samples					
ρ \ size	12,000	24,000	36,000	48,000	60,000
10^{-6}	1.2163	2.3376	3.7053	5.1491	6.7298
10^{-5}	2.5772	4.3417	6.6681	8.3763	11.0292
10^{-4}	2.3216	4.1163	6.2355	8.3819	10.7120
10^{-3}	2.3149	4.5250	6.9834	9.5853	12.3232
10^{-2}	2.7381	5.3373	8.1585	11.1992	14.2487

Table 3: The relationship between running time per iteration (in second) and the size of training samples as well as value of ρ when the number of neurons is fixed: generally, the running time increased as the training sample and the value of ρ became larger.

5.2.3 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 ρ, the size of 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. The number of neurons for each layer ranged from 200 to 1,000, with an increase of 200 each time. The value of ρ ranged from 10^{-6} to 10^{-2}, with multiplying by 10 each time. Generally, the running time increased as the number of neurons and the value of ρ became larger. However, there were a few exceptions: for example, when there were 200 neurons for the MNIST dataset, and ρ 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 ρ is shown in Table 3. The value of ρ ranged from 10^{-6} to 10^{-2}, with 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. Similiar to Table 3, the running time increased generally as the training sample and the value of ρ became larger and some exceptions exist.

6 CONCLUSION AND FUTURE WORK

Alternating Direction Method of Multipliers (ADMM) is a good alternative to Stochastic gradient descent (SGD) for deep learning problems. In this paper, we propose a novel deep learning Alternating Direction Method of Multipliers (dlADMM) to address some previously mentioned challenges. Firstly, the dlADMM updates parameters from backward to forward in order 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 dlADMM is guaranteed to converge to a critical solution under mild conditions. Experiments on benchmark datasets demonstrate that our proposed dlADMM algorithm outperformed most of the comparison methods.

In the future, we may extend our dlADMM from the fully-connected neural network to the famous Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN), because our convergence guarantee is also applied to them. We also consider other nonlinear activation functions such as sigmoid and hyperbolic tangent function (tanh).

ACKNOWLEDGEMENT

This work was supported by the National Science Foundation grant: #1755850.

REFERENCES

[1] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.
[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.
[3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.
[4] Caihua Chen, Min Li, Xin Liu, and Yinyu Ye. Extended admm and bcd for nonseparable convex minimization models with quadratic coupling terms: convergence analysis and insights. *Mathematical Programming*, pages 1–41, 2015.
[5] Francois Chollet. *Deep learning with python*. Manning Publications Co., 2017.
[6] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.
[7] Yuyang Gao, Liang Zhao, Lingfei Wu, Yanfang Ye, Hui Xiong, and Chaowei Yang. Incomplete label multi-task deep learning for spatio-temporal event subtype forecasting. 2019.
[8] Farkhondeh Kiaee, Christian Gagné, and Mahdieh Abbasi. Alternating direction method of multipliers for sparse convolutional neural networks. *arXiv preprint arXiv:1611.01590*, 2016.
[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.
[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
[13] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.
[14] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.
[15] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.
[16] R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.
[17] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.
[18] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.
[19] Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training neural networks without gradients: A scalable admm approach. In *International Conference on Machine Learning*, pages 2722–2731, 2016.
[20] T Tieleman and G Hinton. Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. Technical report, Technical Report. Available online: https://zh. coursera. org/learn/neuralnetworks/lecture/YQHki/rmsprop-divide-the-gradient-by-a-running-average-of-its-recent-magnitude (accessed on 21 April 2017).
[21] Junxiang Wang and Liang Zhao. Nonconvex generalizations of admm for nonlinear equality constrained problems. *arXiv preprint arXiv:1705.03412*, 2017.
[22] Junxiang Wang, Liang Zhao, and Lingfei Wu. Multi-convex inequality-constrained alternating direction method of multipliers. *arXiv preprint arXiv:1902.10882*, 2019.
[23] Yu Wang, Wotao Yin, and Jinshan Zeng. Global convergence of admm in nonconvex nonsmooth optimization. *Journal of Scientific Computing*, pages 1–35, 2015.
[24] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
[25] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.