Journal of Machine Learning Research 22 (2021) 1-67 Submitted 9/20; Revised 8/21; Published 9/21

On ADMM in Deep Learning: Convergence and Saturation-Avoidance

Jinshan Zeng JINSHANZENG@JXNU.EDU.CN
School of Computer and Information Engineering, Jiangxi Normal University, Nanchang, China
Liu Bie Ju Centre for Mathematical Sciences, City University of Hong Kong, Hong Kong
Department of Mathematics, Hong Kong University of Science and Technology, Hong Kong

Shao-Bo Lin[*] SBLIN1983@GMAIL.COM
Center of Intelligent Decision-Making and Machine Learning, School of Management, Xi'an Jiaotong University, Xi'an, China

Yuan Yao[*] YUANY@UST.HK
Department of Mathematics, Hong Kong University of Science and Technology, Hong Kong

Ding-Xuan Zhou MAZHOU@CITYU.EDU.HK
School of Data Science and Department of Mathematics, City University of Hong Kong, Hong Kong

Editor: Marc Schoenauer

Abstract

In this paper, we develop an alternating direction method of multipliers (ADMM) for deep neural networks training with sigmoid-type activation functions (called *sigmoid-ADMM pair*), mainly motivated by the gradient-free nature of ADMM in avoiding the saturation of sigmoid-type activations and the advantages of deep neural networks with sigmoid-type activations (called deep sigmoid nets) over their rectified linear unit (ReLU) counterparts (called deep ReLU nets) in terms of approximation. In particular, we prove that the approximation capability of deep sigmoid nets is not worse than that of deep ReLU nets by showing that ReLU activation function can be well approximated by deep sigmoid nets with two hidden layers and finitely many free parameters but not vice-verse. We also establish the global convergence of the proposed ADMM for the nonlinearly constrained formulation of the deep sigmoid nets training from arbitrary initial points to a Karush-Kuhn-Tucker (KKT) point at a rate of order $\mathcal{O}(1/k)$. Besides sigmoid activation, such a convergence theorem holds for a general class of smooth activations. Compared with the widely used stochastic gradient descent (SGD) algorithm for the deep ReLU nets training (called ReLU-SGD pair), the proposed sigmoid-ADMM pair is practically stable with respect to the algorithmic hyperparameters including the learning rate, initial schemes and the pro-processing of the input data. Moreover, we find that to approximate and learn simple but important functions the proposed sigmoid-ADMM pair numerically outperforms the ReLU-SGD pair.

Keywords: Deep learning, ADMM, sigmoid, global convergence, saturation avoidance

1. Introduction

In the era of big data, data of massive size are collected in a wide range of applications including image processing, recommender systems, search engineering, social activity mining

∗. Corresponding author.

and natural language processing (Zhou et al., 2014). These massive data provide a springboard to design machine learning systems matching or outperforming human capability but pose several challenges on how to develop learning systems to sufficiently exploit the data. As shown in Figure 1, the traditional approach comes down to a three-step learning process. It at first adopts delicate data transformations to yield a tractable representation of the original massive data; then develops some interpretable and computable optimization models based on the transformed data to embody the utility of data; finally designs efficient algorithms to solve the proposed optimization problems. These three steps are called feature extraction, model selection and algorithm designation respectively. Since feature extraction usually involves human ingenuity and prior knowledge, it is labor intensive, especially when the data size is huge. Therefore, it is highly desired to reduce the human factors in the learning process.

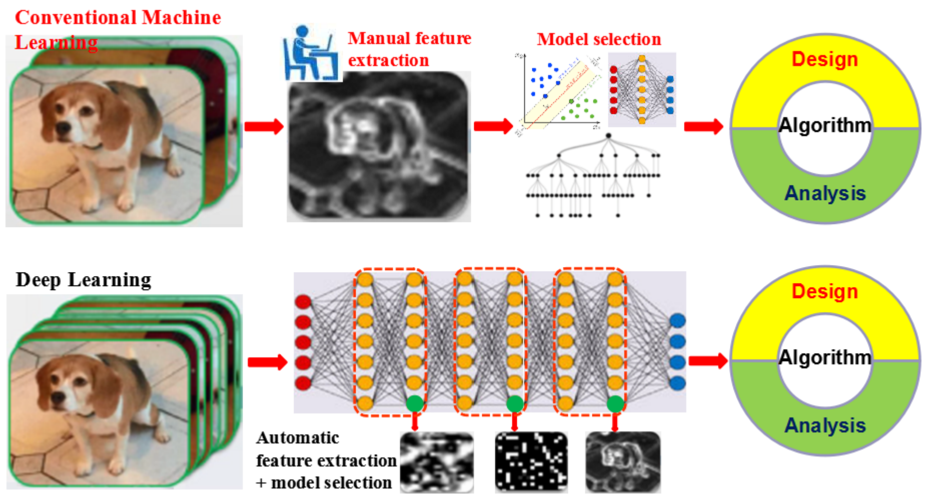

Conventional Machine Learning
Manual feature extraction
Model selection
Design
Algorithm
Analysis
Deep Learning
Automatic feature extraction + model selection
Design
Algorithm
Analysis

Figure 1: Philosophy behind deep learning

Deep learning (Hinton and Salakhutdinov, 2006; LeCun et al., 2015), which utilizes deep neural networks (deep nets for short) for feature extraction and model selection simultaneously, provides a promising way to reduce human factors in machine learning. Just as Figure 1 purports to show, deep learning transforms the classical three-step strategy into a two-step approach: neural networks selection and algorithm designation. It is thus important to pursue why such a transformation is feasible and efficient. In particular, we are interested in making clear of when deep nets are better than classical methods such as shallow neural networks (shallow nets) and kernel methods, and which optimization algorithm is good enough to realize the benefits brought from deep nets.

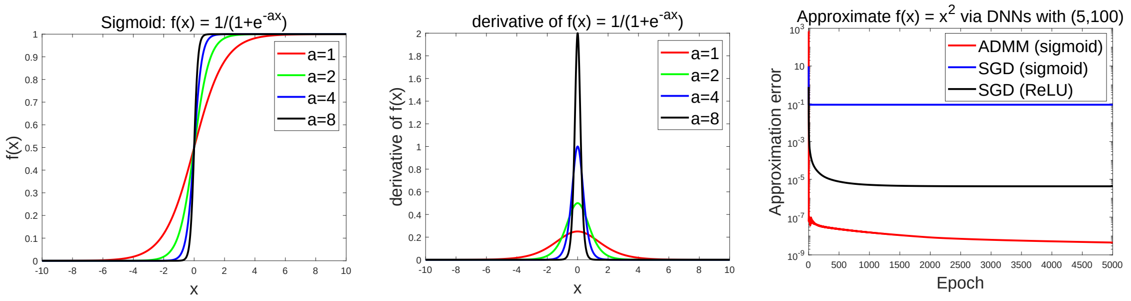
In the past decade, deep nets with ReLU activations (deep ReLU nets) equipped with the well known stochastic gradient descent (SGD) algorithm have been successfully used in image classification (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012; Sainath et al., 2013), natural language processing (Devlin et al., 2014), demonstrating the power of ReLU-SGD pair in deep learning. The problem is, however, that there is a crucial inconsistency between approximation and optimization for the ReLU-SGD pair. To be detailed, from the approximation theory viewpoint, it is necessary to deepen the network to approximate smooth function (Yarotsky, 2017), extract manifold structures (Shaham et al., 2018), realize rotation-invariance features (Han et al., 2020) and provide localized approximation (Safran and Shamir, 2017). However, from the optimization viewpoint, it is difficult to solve optimization problems associated with too deep networks with theoretical guarantees (Goodfellow et al., 2016). Besides the lack of convergence (to a global minima) guarantees, deep ReLU nets equipped with SGD may suffer from the issue of gradient explosion/vanishing (Goodfellow et al., 2016) and is usually sensitive to its algorithmic hyper-parameters such as the initialization (Glorot and Bengio, 2010; Sutskever et al., 2013; Hanin and Rolnick, 2018) and learning rate (Senior et al., 2013; Daniel et al., 2016; Ruder, 2016) in the sense that these parameters have dramatic impacts on the performance of SGD and thus should be carefully tuned in practice. In a nutshell, deep ReLU nets should be deep enough to exhibit excellent approximation capability while too deep networks frequently impose additional difficulty in optimization.

There are numerous remedies to tackle the aforementioned inconsistency for the ReLU-SGD pair with intuition that SGD as well as its variants is capable of efficiently solving the optimization problem associated with deep ReLU nets. In particular, some tricks on either the network architectures such as ResNets (He et al., 2016) or the training procedure such as the batch normalization (Ioffe and Szegedy, 2015) and weight normalization (Salimans and Kingma, 2016) have been developed to address the issue of gradient vanishing/explosion; several efficient initialization schemes including the *MSRA* initialization (He et al., 2015) have been proposed for deep ReLU nets; some guarantees have been established (Allen-Zhu et al., 2019; Du et al., 2019; Zou and Gu, 2019) in the over-parametrized setting to verify the convergence of SGD; and numerous strategies of learning rates (Chollet et al., 2015; Gotmare et al., 2019; Smith and Topin, 2017) have been provided to enhance the feasibility of SGD.

Different from the aforementioned approach focusing on modifying SGD for deep ReLU nets, we pursue an alternative direction to ease the training via reducing the depth. Our studies stem from an interesting observation in neural networks approximation. As far as the approximation capability is concerned, deep nets with sigmoid-type activation functions (deep sigmoid nets) theoretically perform better than deep ReLU nets for some function classes in the sense that to attain the same approximation accuracy, the depth and number of parameters of the former is much smaller than those of the latter. This phenomenon was observed in approximating smooth functions (Mhaskar, 1996; Yarotsky, 2017), reflecting the rotation invariance feature (Chui et al., 2019; Han et al., 2020) and capturing sparse signals (Lin et al., 2017; Schwab and Zech, 2019).

In spite of their advantages in approximation, deep sigmoid nets have not been widely used in the deep learning community. The major reason is due to the saturation problem of the sigmoid function [1] (Goodfellow et al., 2016, Section 6.3), which is easy to cause gradient vanishing for gradient-descent based algorithms in the deep sigmoid nets training (Bengio et al., 1994; LeCun et al., 1998). Specifically, as shown in Figure 2 (b), derivatives of sigmoid functions vanish numerically in a large range. In this paper, we aim at developing a gradient-free algorithm for the deep sigmoid nets training to avoid saturation of deep sigmoid nets and sufficiently embody their theoretical advantages. As a typical gradient-

1. A function $f : \mathbb{R} \to \mathbb{R}$ is said to be *saturating* if it is differentiable and its derivative $f'(x)$ satisfies $\lim_{|x|\to+\infty} f'(x) = 0$.



(a) Sigmoid functions (b) Saturation of sigmoid (c) SGD vs. ADMM

Figure 2: The cons of SGD and pros of ADMM in solving deep sigmoid nets. The setting of numerical simulation in (c) can be found in the Table 2 below.

free optimization algorithm, alternating direction method of multipliers (ADMM) can be regarded as a primal-dual method based on an augmented Lagrangian by introducing nonlinear constraints and enables a convergent sequence satisfying the nonlinear constraints. Therefore, ADMM attracted rising attention in deep learning with various implementations (Carreira-Perpinan and Wang, 2014; Taylor et al., 2016; Kiaee et al., 2016; Yang et al., 2016; Gotmare et al., 2018; Murdock et al., 2018). Under this circumstance, we propose an efficient ADMM algorithm based on a novel update order and an efficient sub-problem solver. Surprisingly, as shown in Figure 2 (c), the proposed sigmoid-ADMM pair performs better than ReLU-SGD pair in approximating the simple but extremely important square function (Yarotsky, 2017; Petersen and Voigtlaender, 2018; Han et al., 2020). This implies that ADMM may be an efficient algorithm to sufficiently realize theoretical advantages of deep sigmoid nets. Our contributions of this paper can be summarized as the following three folds.

- **Methodology Novelty:** We develop a novel sigmoid-ADMM pair for deep learning. Compared with the widely used ReLU-SGD pair, the proposed sigmoid-ADMM pair is stable with respect to algorithmic hyperparameters including learning rates, initial schemes and the pro-processing of input data. Furthermore, we find that to approximate and learn simple but important functions including the square function, radial functions and product gate, deep sigmoid nets theoretically beat deep ReLU nets and the proposed sigmoid-ADMM pair outperforms the ReLU-SGD pair. In terms of algorithm designs, different from existing ADMM methods in deep learning, our proposed ADMM adopts a backward-forward update order that is similar as BackProp (Rumelhart et al., 1986) and a local linear approximation for sub-problems, and more importantly keeps all the nonlinear constraints such that the solution found by the proposed algorithm can converge to a solution satisfying these nonlinear constraints.

- **Theoretical Novelty:** To demonstrate the theoretical advantages of deep sigmoid nets, we rigorously prove that the approximation capability of deep sigmoid nets is not worse than deep ReLU nets by showing that ReLU can be well approximated by deep sigmoid nets with two hidden layers and finitely many free parameters but not vice-verse. We also establish the global convergence of the proposed ADMM for the nonlinearly constrained formulation of the deep sigmoid nets training from arbitrary initial points to a Karush-Kuhn-Tucker (KKT) point at a rate of order $\mathcal{O}(1/k)$. Different from the existing literature on convergence of nonconvex ADMM (Hong et al., 2016; Wang et al., 2019; Gao et al., 2020) for linear or multiaffine constrained optimization problems, our analysis provides a new methodology to deal with the nonlinear constraints in deep learning. In a word, our approach actually leads to a general convergence framework for ADMM with "smooth" enough activations.

• **Numerical Novelty:** In terms of numerical performance, the effectiveness (particularly the stability to initial schemes and the easy-to-tune property of algorithmic parameters) of the proposed ADMM has been demonstrated by numerous experiments including a series of toy simulations and three real-data experiments. Numerical results illustrate the outperformance of the sigmoid-ADMM pair over the ReLU-SGD pair in approximating the extremely important square function, product gate, piecewise L_1 radial and smooth L_2 radial functions with stable algorithmic hyperparameters. Together with some other important functions such as the localized approximation (Chui et al., 1994), these natural functions realized in this paper can represent some important data features such as piecewise smoothness in image processing (Krizhevsky et al., 2012), sparseness in computer vision (LeCun et al., 2015), and rotation-invariance in earthquake prediction (Vikraman, 2016). The effectiveness of the proposed ADMM is further demonstrated by real-data experiments, i.e., earthquake intensity, extended Yale B databases and PTB Diagnostic ECG databases, which reflect the partially radial and low-dimensional manifold features in some extent.

The rest of this paper is organized as follows. In the next section, we demonstrate the advantage of deep sigmoid nets in approximation (see Theorem 3). Section 3 describes the proposed ADMM method for the considered DNN training model followed by the main convergence theorem (see Theorem 4). Section 4 provides some discussions on related work and key ideas of our proofs. Section 5 provides some toy simulations to show the effectiveness of the proposed ADMM method in realizing some important natural functions. Section 6 provides two real data experiments to further demonstrate the effectiveness of the proposed method. All proofs are presented in Appendix.

Notations: For any matrix $A \in \mathbb{R}^{m\times n}$, $[A]_{ij}$ denotes its (i,j)-th entry. Given a matrix A, $\|A\|_F$, $\|A\|_2$ and $\|A\|_{\max}$ denote the Frobenius norm, operator norm, and max-norm of A, respectively, where $\|A\|_{\max} = \max_{i,j} |[A]_{ij}|$. Then obviously, $\|A\|_{\max} \le \|A\|_2 \le \|A\|_F$. We let $W_{<i} := [W_1, W_2, \dots, W_{i-1}]$, $W_{>i} := [W_{i+1}, \dots, W_N]$ for $i = 1, \dots, N$, $W_{<1} = \emptyset$ and $W_{>N} = \emptyset$. $\mathbf{I}$ denotes the identity matrix whose size can be determined according to the text. Denote by $\mathbb{R}$ and $\mathbb{N}$ the real and natural number sets, respectively.

2. Deep Sigmoid Nets in Approximation

For the depth $N \in \mathbb{N}$ of a neural network, let $d_i \in \mathbb{N}$ be the number of hidden neurons at the i-th hidden layer for $i = 1, \dots, N-1$. Denote an affine mapping $\mathcal{J}_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ by $\mathcal{J}_i(x) := W_i x + b_i$ for $d_i \times d_{i-1}$ weight matrix and thresholds $b_i \in \mathbb{R}^{d_i}$. For a univariate activation function $\sigma_i, i = 1 \dots, N$, denote further $\sigma_i(x)$ when σ_i is applied component-wise to the vector x. Define an N-layer feedforward neural network by

$$\mathcal{N}_{N,d_1,\dots,d_N,\sigma} = a \cdot \sigma_N \circ \mathcal{J}_N \circ \sigma_{N-1} \circ \mathcal{J}_{N-1} \circ \cdots \circ \sigma_1 \circ \mathcal{J}_1(x), \tag{1}$$

where $a \in \mathbb{R}^{d_N}$. The deep net defined in (1) is called the deep ReLU net and deep sigmoid net, provided $\sigma_i(t) \equiv \sigma_{relu}(t) = \max\{t, 0\}$ and $\sigma_i(t) \equiv \sigma(t) = \frac{1}{1+e^{-t}}$ respectively.

Since the (sub-)gradient computation of ReLU is very simple, deep ReLU nets have attracted enormous research activities in the deep learning community (Nair and Hinton, 2010). The power of deep ReLU nets, compared with shallow nets with ReLU (shallow ReLU nets) has been sufficiently explored in the literature (Yarotsky, 2017; Petersen and Voigtlaender, 2018; Shaham et al., 2018; Schwab and Zech, 2019; Chui et al., 2020; Han et al., 2020). In particular, it was proved in (Yarotsky, 2017, Proposition 2) that the following "square-gate" property holds for deep ReLU nets, which is beyond the capability of shallow ReLU nets due to the non-smoothness of ReLU.

Lemma 1 *The function $f(t) = t^2$ on the segment $[-M, M]$ for $M > 0$ can be approximated within any accuracy $\varepsilon > 0$ by a deep ReLU net with the depth and free parameters of order $\mathcal{O}(\log(1/\varepsilon))$.*

The above lemma exhibits the necessity of the depth for deep ReLU nets to act as a "square-gate". Since the depth depends on the accuracy, it requires many hidden layers for deep ReLU nets for such an easy task and too many hidden layers enhance the difficulty for analyzing SGD (Goodfellow et al., 2016, Sec.8.2). This presents the reason why the numerical accuracy of deep ReLU nets in approximating t^2 is not so good, just as Figure 2 exhibits. Differently, due to the infinitely differentiable property of the sigmoid function, it is easy for shallow sigmoid nets (Chui et al., 2019, Proposition 1) to play as a "square-gate", as shown in the following lemma.

Lemma 2 *Let $M > 0$. For $\varepsilon > 0$, there is a shallow sigmoid net $\mathcal{N}_3$ with 3 free parameters bounded by $\mathcal{O}(\varepsilon^{-6})$ such that*

$$|t^2 - \mathcal{N}_3(t)| \leq \varepsilon, \qquad t \in [-M, M].$$

Besides the "square-gate", deep sigmoid nets are capable of acting as a "product-gate" (Chui et al., 2019), providing localized approximation (Chui et al., 1994), extracting the rotation-invariance property (Chui et al., 2019) and reflecting the sparseness in spatial domain (Lin, 2019) and frequency domain (Lin et al., 2017) with much fewer hidden layers than deep ReLU nets. The following Table 1 presents a comparison between deep sigmoid nets and deep ReLU nets in feature selection and approximation.

Table 1 presents theoretical advantages of deep sigmoid nets over deep ReLU nets. In fact, as the following theorem shows, it is easy to construct a sigmoid net with two hidden layers and small number of free parameters to approximate ReLU, implying that the approximation capability of deep sigmoid nets is at least not worse than that of deep ReLU nets with comparable hidden layers and free parameters.

Theorem 3 *Let $1 \leq p < \infty$ and $M \geq 1$. Then for any $\varepsilon \in (0, 1/2)$ and $M > 0$, there is a sigmoid net h^* with 2 hidden layers and at most 27 free parameters bounded by $\mathcal{O}(\varepsilon^{-7})$ such that*

$$\|h^* - \sigma_{relu}\|_{L^p([-M,M])} \leq \varepsilon, \tag{2}$$

where $L^p([-M, M])$ denotes the L^p space of functions defined on $[-M, M]$.

Table 1: Depth required for deep nets in feature extraction and approximation within accuracy ε

Features	sigmoid	ReLU
Square-gate	1 (Chui et al., 2019)	$\log(\varepsilon^{-1})$ (Yarotsky, 2017)
Product-gate	1 (Chui et al., 2019)	$\log(\varepsilon^{-1})$ (Yarotsky, 2017)
Localized approximation	2 (Chui et al., 1994)	2(Chui et al., 2020)
k-spatially sparse+smooth	2 (Lin, 2019)	> 8 (Chui et al., 2020)
Smooth+Manifold	3 (Chui et al., 2018)	4 (Shaham et al., 2018)
Smooth	1 (Mhaskar, 1996)	$\log(\varepsilon^{-1})$ (Yarotsky, 2017)
k-sparse (frequency)	k (Lin et al., 2017)	$k\log(\varepsilon^{-1})$ (Schwab and Zech, 2019)
Radial+smooth	4 (Chui et al., 2019)	> 8 (Han et al., 2020)

The proof of Theorem 3 is postponed in Appendix A. For an arbitrary deep ReLU net

$$\mathcal{N}^{relu}_{L,d_1,\dots,d_L} = a \cdot \sigma_{relu} \circ \mathcal{J}_L \circ \sigma_{relu} \circ \mathcal{J}_{L-1} \circ \cdots \circ \sigma_{relu} \circ \mathcal{J}_1(x)$$

with bounded free parameters, Theorem 3 shows that we can construct a deep sigmoid net

$$\mathcal{N}^{sigmoid}_{L,d_1,\dots,d_L} = a \cdot h^* \circ \mathcal{J}_L \circ h^* \circ \mathcal{J}_{L-1} \circ \cdots \circ h^* \circ \mathcal{J}_1(x)$$

that possesses at least similar approximation capability. However, due to the infinitely differentiable property of the sigmoid function, it is difficult to construct a deep ReLU net with accuracy-independent depth and width to approximate it. Indeed, it can be found in (Petersen and Voigtlaender, 2018, Theorem 4.5) that for any open interval Ω and deep ReLU net $\mathcal{N}^{relu}_{L,n}$ with L hidden layers and n free parameters, there holds

$$\|\sigma - \mathcal{N}^{relu}_{L,n}\|_{L^p(\Omega)} \geq C'' n^{-2L}, \tag{3}$$

where σ is the sigmoid activation and C'' is a constant independent of n or L. Comparing (2) with (3), we find that any functions being well approximated by deep ReLU nets can also be well approximated by deep sigmoid nets, but not vice-verse.

It should be mentioned that though the depth and width in Theorem 3 are independent of ε and relatively small, the magnitude of free parameters depends heavily on the accuracy and may be large. Since such large free parameters are difficult to realize for an optimization algorithm, a preferable way to shrink them is to deepen the network further. In particular, it can be found in (Chui et al., 2019) that there is a shallow sigmoid net $\mathcal{N}^{sigmoid}_2$ with 2 free parameters of order $\mathcal{O}(1/\varepsilon)$ such that

$$|\sigma(\sqrt{w}\mathcal{N}^{sigmoid}_2(\sqrt{w}t)) - \sigma(wt)| \leq \varepsilon,$$

where $t, w \in \mathbb{R}$. Because we only focus on the power of deep sigmoid nets in approximation, we do not shrink free parameters in Theorem 3.

3. ADMM for Deep Sigmoid Nets

Let $\mathcal{Z} := \{(x_j, y_j)\}_{j=1}^n \subset \mathbb{R}^{d_0} \times \mathbb{R}^{d_N}$ be n samples. Denote $X := (x_1, x_2, \ldots, x_n) \in \mathbb{R}^{d_0 \times n}$ and $Y := (y_1, y_2, \ldots, y_n) \in \mathbb{R}^{d_N \times n}$. It is natural to consider the following regularized DNN training problem

$$\min_{\mathcal{W}} \left\{ \frac{1}{n} \sum_{i=1}^n \|\Phi(x_i, \mathcal{W}) - y_i\|_2^2 + \lambda' \|W_i\|_F^2 \right\}, \tag{4}$$

where $\Phi(x_i, \mathcal{W})$ denotes a deep sigmoid net with N layers, $\mathcal{W} = \{W_i\}_{i=1}^N$ and $\lambda' > 0$ is the regularization parameter. Here, we consider the square loss as analyzed in the literature (Allen-Zhu et al., 2019; Du et al., 2019; Zou and Gu, 2019). We also absorb thresholds into the weight matrices for the sake of simplicity. Based on the advantage of deep sigmoid nets in approximation, (Chui et al., 2019; Lin, 2019) proved that the model defined by (4) with $N = 2$ are optimal in embodying data features such as the spatial sparseness, smoothness and rotation-invariance in the sense that it can achieve almost optimal generalization error bounds in the framework of learning theory. The aim of this section is to introduce an efficient algorithm to solve the optimization problem (4).

Due to the saturation problem of the sigmoid function (see Figure 2 (b)), the issue of gradient vanishing or explosion frequently happens for running SGD on deep sigmoid nets (see Figure 4 (a) for example), implying that the classical SGD is not a good candidate to solve (4). We then turn to designing a gradient-free optimization algorithm, like ADMM, to efficiently solve (4). For DNN training, there are generally two important ingredients in designing ADMM: update order and solution to each sub-problem. The novelty of our proposed algorithm is the use of backward-forward update order similar to BackProp in (Rumelhart et al., 1986) and local linear approximation to sub-problems.

3.1 Update order in ADMM for deep learning training

The optimization problem (4) can be equivalently reformulated as the following constrained optimization problem

$$\begin{aligned} &\underset{\mathcal{W},\mathcal{V}}{\text{minimize}} \ \frac{1}{2}\|V_N - Y\|_F^2 + \frac{\lambda}{2}\sum_{i=1}^N \|W_i\|_F^2 \\ &\text{subject to} \quad V_i = \sigma(W_i V_{i-1}),\ i = 1, \ldots, N-1, \quad V_N = W_N V_{N-1}, \end{aligned} \tag{5}$$

where $\mathcal{V} := \{V_i\}_{i=1}^N$ represents the set of responses of all layers and $\lambda = \frac{\lambda' n}{2}$. We define the augmented Lagrangian of (5) as follows:

$$\begin{aligned} \mathcal{L}(\mathcal{W}, \mathcal{V}, \{\Lambda_i\}_{i=1}^N) := {} & \frac{1}{2}\|V_N - Y\|_F^2 + \frac{\lambda}{2}\sum_{i=1}^N \|W_i\|_F^2 \\ & + \sum_{i=1}^{N-1} \left(\frac{\beta_i}{2} \|\sigma(W_i V_{i-1}) - V_i\|_F^2 + \langle \Lambda_i, \sigma(W_i V_{i-1}) - V_i \rangle \right) \\ & + \frac{\beta_N}{2}\|W_N V_{N-1} - V_N\|_F^2 + \langle \Lambda_N, W_N V_{N-1} - V_N \rangle, \end{aligned} \tag{6}$$

where $\Lambda_i \in \mathbb{R}^{d_i \times n}$ is the multiplier matrix associated with the i-th constraint, and β_i is the associated penalty parameter for $i = 1, \ldots, N$.

ADMM is an augmented-Lagrangian based primal-dual method, which updates the primal variables ($\{W_i\}_{i=1}^N$ and $\{V_i\}_{i=1}^N$ in (6)) via a Gauss-Seidel scheme and then multipliers ($\{\Lambda_i\}_{i=1}^N$ in (6)) via a gradient ascent scheme in a parallel way (Boyd et al., 2011). As suggested in (Wang et al., 2019), the update order of the primal variables is tricky for ADMM in terms of the convergence analysis in the nonconvex setting. In light of (Wang et al., 2019), the key idea to yield a *desired update order* with convergence guarantee is to arrange the updates of some *special primal variables* followed by the updates of multipliers such that the updates of multipliers can be *explicitly expressed* by the updates of these special primal variables, and thus the dual ascent quantities arisen by the updates of multipliers shall be controlled by the descent quantities brought by the updates of these special primal variables. Hence, the arrangement of these special primal variables is crucial.

It can be noted that there are $2N$ blocks of primal variables, i.e., $\{W_i\}_{i=1}^N$ and $\{V_i\}_{i=1}^N$ and N blocks of multipliers $\{\Lambda_i\}_{i=1}^N$ involved in (6). For better elaboration of our idea, we take $N = 3$ for an example. Notice that the multipliers $\{\Lambda_i\}_{i=1}^3$ are only involved in these inner product terms $\langle \Lambda_1, \sigma(W_1X) - V_1 \rangle$, $\langle \Lambda_2, \sigma(W_2V_1) - V_2 \rangle$ and $\langle \Lambda_3, W_3V_2 - V_3 \rangle$. By these terms, the gradient of the i-th inner product with respect to V_i is $-\Lambda_i$, while the associated gradient with respect to W_i is a more complex term (namely, $(\Lambda_1 \odot \sigma'(W_1X))X^T$ for W_1, $(\Lambda_2 \odot \sigma'(W_2V_1))V_1^T$ for W_2, and $\Lambda_3 V_2^T$ for W_3, where $\odot$ represents Hadamard product). If the update of W_i is used to express Λ_i, then according to the W_i subproblem, an inverse operation of a nonlinear or linear mapping is required, while such an inverse does not necessarily exist. Specifically, following the analysis of Lemma 8 shown later and taking the expression of W_3 for example, the term $\Lambda_3 V_2^T$ will be involved in the expression of W_3. In this case, if we wish to express Λ_3 by W_3, then the inverse of V_2 is generally required, while it does not necessarily exist. Due to this, it should be more convenient to express Λ_i ($i = 1, 2, 3$) via exploiting the V_i subproblem instead of the W_i subproblem. Therefore, we suggest firstly update the blocks of W_i's and then V_i's such that Λ_i's can be explicitly expressed via the latest updates of V_i's. To be detailed, for each loop, we update $\{W_i\}_{i=1}^N$ in the backward order, i.e., $W_N \to W_{N-1} \to \cdots \to W_1$, then update $\{V_j\}_{j=1}^N$ in the forward order, i.e., $V_1 \to V_2 \to \cdots \to V_N$, motivated by BackProp in (Rumelhart et al., 1986), and finally update the multipliers $\{\Lambda_i\}_{i=1}^N$ in a parallel way, as shown by the following Figure 3.

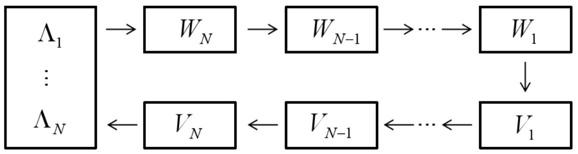


Figure 3: Update order of ADMM

Specifically, given an initialization $\{W_i^0\}_{i=1}^N$, we set

$$V_j^0 = \sigma(W_j^0 V_{j-1}^0), \ j = 1, \ldots, N-1, \quad V_N^0 = W_N^0 V_{N-1}^0, \text{ and } \Lambda_i^0 = 0, \ i = 1, \ldots, N, \tag{7}$$

where $V_0^0 = X$. Given the (k-1)-th iterate $\left(\{W_i^{k-1}\}_{i=1}^N, \{V_i^{k-1}\}_{i=1}^N, \{\Lambda_i^{k-1}\}_{i=1}^N\right)$, we define the W_i- and V_i-subproblems at the k-th iteration via minimizing the augmented Lagrangian

(6) with respect to only one block but fixing the other blocks at the latest updates, according to the update order specified in Figure 3, shown as follows:

$$W_N^k = \arg\min_{W_N} \left\{ \frac{\lambda}{2}\|W_N\|_F^2 + \frac{\beta_N}{2}\|W_N V_{N-1}^{k-1} - V_N^{k-1}\|_F^2 + \langle \Lambda_N^{k-1}, W_N V_{N-1}^{k-1} - V_N^{k-1}\rangle \right\}, \quad (8)$$

and for $i = N-1, \ldots, 1$,

$$W_i^k = \arg\min_{W_i} \left\{ \frac{\lambda}{2}\|W_i\|_F^2 + \frac{\beta_i}{2}\|\sigma(W_i V_{i-1}^{k-1}) - V_i^{k-1}\|_F^2 + \langle \Lambda_i^{k-1}, \sigma(W_i V_{i-1}^{k-1}) - V_i^{k-1}\rangle \right\}, \quad (9)$$

and for $j = 1, \ldots, N-2$,

$$\begin{aligned} V_j^k = \arg\min_{V_j} \Big\{ & \frac{\beta_j}{2}\|\sigma(W_j^k V_{j-1}^k) - V_j\|_F^2 + \langle \Lambda_j^{k-1}, \sigma(W_j^k V_{j-1}^k) - V_j\rangle \quad (10) \\ & + \frac{\beta_{j+1}}{2}\|\sigma(W_{j+1}^k V_j) - V_{j+1}^{k-1}\|_F^2 + \langle \Lambda_{j+1}^{k-1}, \sigma(W_{j+1}^k V_j) - V_{j+1}^{k-1}\rangle \Big\}, \end{aligned}$$

$$\begin{aligned} V_{N-1}^k = \arg\min_{V_{N-1}} \Big\{ & \frac{\beta_{N-1}}{2}\|\sigma(W_{N-1}^k V_{N-2}^k) - V_{N-1}\|_F^2 + \langle \Lambda_{N-1}^{k-1}, \sigma(W_{N-1}^k V_{N-2}^k) - V_{N-1}\rangle \\ & + \frac{\beta_N}{2}\|W_N^k V_{N-1} - V_N^{k-1}\|_F^2 + \langle \Lambda_N^{k-1}, W_N^k V_{N-1} - V_N^{k-1}\rangle \Big\}, \quad (11) \end{aligned}$$

$$V_N^k = \arg\min_{V_N} \left\{ \frac{1}{2}\|V_N - Y\|_F^2 + \frac{\beta_N}{2}\|W_N^k V_{N-1}^k - V_N\|_F^2 + \langle \Lambda_N^{k-1}, W_N^k V_{N-1}^k - V_N\rangle \right\}. \quad (12)$$

Once $\left(\{W_i^k\}_{i=1}^N, \{V_i^k\}_{i=1}^N\right)$ have been updated, we then update the multipliers $\{\Lambda_i^k\}_{i=1}^N$ parallelly according to the following: for $i = 1, \ldots, N-1$,

$$\Lambda_i^k = \Lambda_i^{k-1} + \beta_i(\sigma(W_i^k V_{i-1}^k) - V_i^k), \quad \Lambda_N^k = \Lambda_N^{k-1} + \beta_N(W_N^k V_{N-1}^k - V_N^k). \quad (13)$$

Based on these, each iterate of ADMM only involves several relatively simpler sub-problems.

It should be mentioned that the suggested update order is actually a technical requirement in the convergence proof (see, Lemma 8 below), which also appears in the previous work (Wang et al., 2019). Moreover, the following local linear approximation is also required to establish Lemma 8.

3.2 Local linear approximation for sub-problems

Note that W_i-subproblems ($i = 1, \ldots, N-1$) involve functions of the following form

$$H_\sigma(W; A, B) = \frac{1}{2}\|\sigma(WA) - B\|_F^2, \quad (14)$$

while V_j-subproblems ($j = 1, \ldots, N-2$) involve functions of the following form

$$M_\sigma(V; \tilde{A}, \tilde{B}) = \frac{1}{2}\|\sigma(\tilde{A}V) - \tilde{B}\|_F^2, \quad (15)$$

where $A, B, \tilde{A}, \tilde{B}$ are four given matrices related to the previous updates. Due to the nonlinearity of the sigmoid activation function, the subproblems are generally difficult to be solved, or at least some additional numerical solvers are required to solve these subproblems. To break such computational hurdle, we adopt the first-order approximations of the original functions presented in (14) and (15) at the latest updates, instead of themselves, to update the variables, that is,

$$\begin{aligned} H_\sigma^k(W;A,B) &:= H_\sigma(W^{k-1};A,B) + \langle(\sigma(W^{k-1}A)-B)\odot\sigma'(W^{k-1}A),(W-W^{k-1})A\rangle \\ &\quad + \frac{h^k}{4}\|(W-W^{k-1})A\|_F^2, \qquad (16)\\ M_\sigma^k(V;\tilde{A},\tilde{B}) &:= M_\sigma(V^{k-1};\tilde{A},\tilde{B}) + \langle(\sigma(\tilde{A}V^{k-1})-\tilde{B})\odot\sigma'(\tilde{A}V^{k-1}),\tilde{A}(V-V^{k-1})\rangle \\ &\quad + \frac{\mu^k}{4}\|\tilde{A}(V-V^{k-1})\|_F^2, \qquad (17)\end{aligned}$$

where W^{k-1} and V^{k-1} are the (k-1)-th iterate, and $\sigma'(W^{k-1}A)$ and $\sigma'(\tilde{A}V^{k-1})$ represent the componentwise derivatives, h^k and μ^k can be specified as the upper bounds of twice of the locally Lipschitz constants of functions H_σ and M_σ, respectively, shown as

$$h^k = \mathbb{L}(\|B\|_{\max}), \quad \mu^k = \mathbb{L}(\|\tilde{B}\|_{\max}).$$

Here, for any given $c \in \mathbb{R}$,

$$\mathbb{L}(|c|) := 2L_2(L_0+|c|) + 2L_1^2 \qquad (18)$$

is an upper bound of the Lipschitz constant of the gradient of function $(\sigma(u)-c)^2$ with constants $L_0 = 1, L_1 = \frac{1}{4}$ and $L_2 = \frac{1}{4}$ related to the sigmoid activation σ.

Henceforth, we call this treatment as the **local linear approximation (LLA)**, which can be viewed as adopting certain *prox-linear scheme* (Xu and Yin, 2013) to update the subproblems of ADMM. Based on (16) and (17), the original updates (9) of $\{W_i^k\}_{i=1}^{N-1}$ are replaced by

$$W_i^k = \arg\min_{W_i}\left\{\frac{\lambda}{2}\|W_i\|_F^2 + \beta_i H_\sigma^k(W_i; V_{i-1}^{k-1}, V_i^{k-1} - \beta_i^{-1}\Lambda_i^{k-1})\right\}, \qquad (19)$$

and by completing perfect squares and some simplifications, the original updates (10) of $\{V_j^k\}_{j=1}^{N-2}$ are replaced by

$$V_j^k = \arg\min_{V_j}\left\{\frac{\beta_j}{2}\|\sigma(W_j^kV_{j-1}^k) + \beta_j^{-1}\Lambda_j^{k-1} - V_j\|_F^2 + \beta_{j+1}M_\sigma^k(V_j; W_{j+1}^k, V_{j+1}^{k-1} - \beta_{j+1}^{-1}\Lambda_{j+1}^{k-1})\right\}, \qquad (20)$$

with h_i^k and μ_j^k being specified as follows

$$h_i^k = \mathbb{L}(\|V_i^{k-1} - \beta_i^{-1}\Lambda_i^{k-1}\|_{\max}),\ i = 1,\ldots,N-1, \qquad (21)$$

$$\mu_j^k = \mathbb{L}(\|V_{j+1}^{k-1} - \beta_{j+1}^{-1}\Lambda_{j+1}^{k-1}\|_{\max}),\ j = 1,\ldots,N-2, \qquad (22)$$

where $\mathbb{L}(\cdot)$ is defined in (18). Note that with these alternatives, all the subproblems can be solved with analytic expressions (see, Lemma 8 in Appendix C.1).

3.3 ADMM for deep sigmoid nets

The ADMM algorithm for DNN training problem (5) is summarized in Algorithm 1. As shown in Figure 4 (b) and Figure 2 (c), ADMM does not suffer from either the issue of gradient explosion or the issue of gradient vanishing caused by the saturation of sigmoid activation and thus can approximate the square function within high precision. The intuition behind ADMM to avoid the issues of gradient explosion and vanishing is that the suggested ADMM does not exactly follow the chain rule as exploited in BackProp and SGD, but introduces the multipliers as certain compensation to eventually fit the chain rule at the stationary point. From Algorithm 1, besides the regularization parameter λ related to the DNN training model, only the penalty parameters β_i's should be tuned. In the algorithmic perspective, penalty parameters can be regarded as the dual step sizes for the updates of multipliers, which play similar roles as learning rates in SGD. As shown by our experiment results below, the performance of ADMM is not sensitive to penalty parameters, making the parameters be easy-to-tune. Moreover, by exploiting the LLA, the updates for all variables can be very cheap with analytic expressions (see Lemma 8 in Appendix C.1).

As compared to the existing ADMM methods for deep learning (Carreira-Perpinan and Wang, 2014; Taylor et al., 2016; Kiaee et al., 2016; Murdock et al., 2018), there are two major differences shown as follows. The first one is that the existing ADMM type methods in deep learning only keep partial nonlinear constraints for the sake of reducing the difficulty of optimization, while the ADMM method suggested in this paper keeps all the nonlinear constraints, and thus our proposed ADMM can come back to the original DNN training model in the sense that its convergent limit fits all the nonlinear constraints as shown in Theorem 4 below. To overcome the difficulty from optimization, we introduce an elegant update order and the LLA technique for subproblems. The second one is that most of existing ADMM methods focus on deep ReLU nets, while our proposed ADMM is designed for deep sigmoid nets.

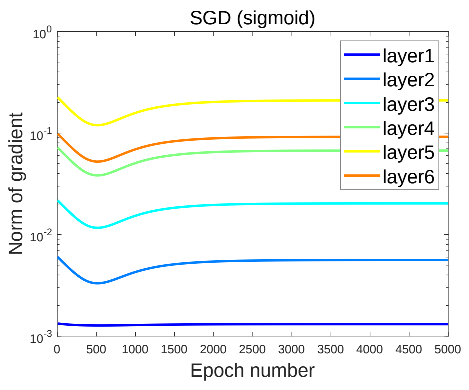
It should be pointed out that the subproblems of the proposed algorithm require inverting matrices at each iteration, which could be expensive. Although there are some practical tricks like warm-start and solving inexactly via doing gradient descent by a fixed number of times to improve the computational efficiency of the proposed ADMM (e.g., in (Liu et al., 2021)), the major focus of this paper is mainly on the development of an effective ADMM method with theoretical guarantees for the training of deep sigmoid nets, and we will consider its practical acceleration in the future.

3.4 Convergence of ADMM for deep sigmoid nets

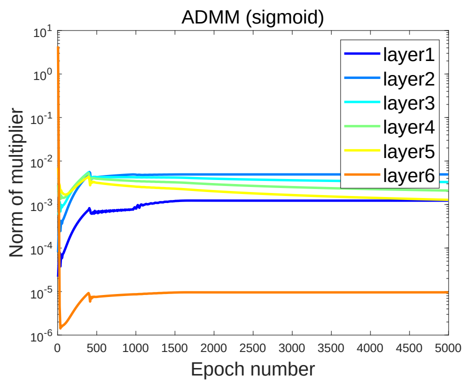
Without loss of generality, we assume that X, Y and $\{W_i^0\}_{i=1}^N$ are normalized with $\|X\|_F = 1$, $\|Y\|_F = 1$ and $\|W_i^0\|_F = 1$, $i = 1, \ldots, N$, and all numbers of hidden layers are the same, i.e., $d_i = d$, $\forall i = 1, \ldots, N-1$. Under these settings, we present the main convergence theorem of ADMM in the following, while that of ADMM under more general settings is presented in Theorem 7 in Appendix B.

Algorithm 1 ADMM for Deep Sigmoid Nets Training

```
Samples: $X := [x_1, \dots, x_n] \in \mathbb{R}^{d_0 \times n}$, $Y := [y_1, \dots, y_n] \in \mathbb{R}^{d_N \times n}$.
Initialization: $(\{W_i^0\}_{i=1}^N, \{V_i^0\}_{i=1}^N, \{\Lambda_i^0\}_{i=1}^N)$ is set according to (7). $V_0^k \equiv X, \forall k \in \mathbb{N}$.
Parameters: $\lambda > 0$, $\beta_i > 0, i = 1, \dots, N$.
for $k = 1, \dots$ do
    ▶ (Backward Estimation)
    for $i = N : -1 : 1$ do
        Update $W_N^k$ via (8) and the other $W_i^k$ via (19).
    end for
    ▶(Forward Prediction)
    for $j = 1 : N$ do
        Update $V_j^k (j = 1, \dots, N-2)$ via (20), $V_{N-1}^k$ via (11), and $V_N^k$ via (12).
    end for
    ▶(Updating Multipliers)
    $\Lambda_i^k = \Lambda_i^{k-1} + \beta_i(\sigma(W_i^k V_{i-1}^k) - V_i^k),\ i = 1, \dots, N-1,$
    $\Lambda_N^k = \Lambda_N^{k-1} + \beta_N(W_N^k V_{N-1}^k - V_N^k).$
    $k \leftarrow k + 1$
end for
```



(a) Gradient vanishing of SGD



(b) Saturation-avoidance of ADMM

Figure 4: Gradient vanishing of SGD and saturation-avoidance of ADMM in the training of deep sigmoid nets. The numerical setting is the same as that of Figure 2.

Theorem 4 *Let $\{\mathcal{Q}^k := (\{W_i^k\}_{i=1}^N, \{V_i^k\}_{i=1}^N, \{\Lambda_i^k\}_{i=1}^N)\}$ be a sequence generated by Algorithm 1. If $2 \le N \le \sqrt{n}$, $\lambda \ge \tilde{c} N^{\frac{N-3}{2}} (nd)^{\frac{N}{2} - \frac{1}{4}}$ and $\{\beta_i\}_{i=1}^N$ satisfy*

$$\beta_N \ge 3.5,\ \beta_{N-1} \ge 16\beta_N,\ \beta_i \ge \tilde{c}_1 \beta_{N-1} (Nnd)^{\frac{N-1-i}{2}},\ i = 1, \dots, N-2 \tag{23}$$

for some constants $\tilde{c}, \tilde{c}_1 > 0$ independent of n, N, then we have:

(a) the augmented Lagrangian sequence $\{\mathcal{L}(\mathcal{Q}^k)\}$ is convergent.

(b) $\{\mathcal{Q}^k\}$ converges to a stationary point $\mathcal{Q}^* := (\{W_i^*\}_{i=1}^N, \{V_i^*\}_{i=1}^N, \{\Lambda_i^*\}_{i=1}^N)$ of the augmented Lagrangian $\mathcal{L}$, which is also a KKT point (defined in(24) below) of problem (5), implying that $\{W_i^*\}_{i=1}^N$ is a stationary point of problem (4) with $\lambda' = 2\lambda/n$.

(c) $\frac{1}{K}\sum_{k=1}^{K}\|\nabla\mathcal{L}(\mathcal{Q}^k)\|_F^2 \to 0$ *at a rate of order* $\mathcal{O}(\frac{1}{K})$.

Theorem 4 establishes the global convergence of ADMM to a KKT point at a rate of $\mathcal{O}(1/K)$. By (23), the parameters $\{\beta_i\}_{i=1}^N$ increase exponentially fast from the output layer to the input layer. Moreover, by Theorem 4, the regularization parameter λ is also required to grow exponentially fast as the depth increases. Back to the original DNN training model (4), the requirement on the regularization parameter λ' is $\lambda' \geq \tilde{c}N^{\frac{N-3}{2}}d^{\frac{N}{2}-\frac{1}{4}}n^{\frac{N}{2}-\frac{5}{4}}$. Particularly, when $N=2$, namely, the neural networks with single hidden layer, then $\lambda' = \tilde{c}\sqrt[4]{d^3/n}$ is a good choice, which implies that the regularization parameter can be small when the sample size n is sufficiently large. Despite these convergence conditions seem a little stringent, by the existing literature (Chui et al., 2018, 2019), the depth of deep sigmoid nets is usually small, say, 2 or 3 for realizing some important data features in deep learning. Moreover, as shown in the numerical results to be presented in Sections 5 and 6, a moderately large augmented Lagrangian parameter (say, each $\beta_i = 1$) and a small regularization parameter (say, $\lambda = 10^{-6}$) are empirically enough for the proposed ADMM. In this case, the KKT point found by ADMM should be close to the optimal solutions to the empirical risk minimization of DNN training.

Remark 1: KKT conditions. Based on (6), the Karush-Kuhn-Tucker (KKT) conditions of the problem (5) can be derived as follows. Specifically, let $\{W_i, V_i\}_{i=1}^N$ be an optimal solution of problem (5), then there exit multipliers $\{\Lambda_i\}_{i=1}^N$ such that the following hold:

$$\begin{aligned}
&0 = \lambda W_1 + (\Lambda_1 \odot \sigma'(W_1V_0))V_0^T,\\
&0 = \lambda W_i + (\Lambda_i \odot \sigma'(W_iV_{i-1}))V_{i-1}^T,\ i = 2,\ldots,N-1,\\
&0 = \lambda W_N + \Lambda_N V_{N-1}^T,\\
&0 = -\Lambda_i + W_{i+1}^T(\Lambda_{i+1} \odot \sigma'(W_{i+1}V_i)),\ i = 1,\ldots,N-2,\\
&0 = -\Lambda_{N-1} + W_N^T\Lambda_N,\\
&0 = -\Lambda_N + (V_N - Y),\\
&0 = \sigma(W_iV_{i-1}) - V_i,\ i = 1,\ldots,N-1,\\
&0 = W_NV_{N-1} - V_N
\end{aligned} \tag{24}$$

where $V_0 = X$. From (24), the KKT point of problem (5) exactly fits these nonlinear constraints. Moreover, given a KKT point $(\{W_i^*\}_{i=1}^N, \{V_i^*\}_{i=1}^N, \{\Lambda_i^*\}_{i=1}^N)$ of (5), substituting the last five equations into the first three equations of (24) shows that $\{W_i^*\}_{i=1}^N$ is also a stationary point of the original DNN training model (4).

Remark 2: More general activations: As presented in Theorem 7 in Appendix B, the convergence results in Theorem 4 still hold for a general class of smooth activations such as the hyperbolic tangent activation as studied in (Lin et al., 2019). Actually, the approximation result yielded in Theorem 3 can be also easily extended to a class of twice differentiable sigmoid-type activations.

4. Related Work and Discussions

In this section, we present some related works and show the novelty of our studies.

4.1 Deep sigmoid nets versus deep ReLU nets in approximation

Deep ReLU nets are the most popular neural networks in deep learning. Compared with deep sigmoid nets, there are commonly three advantages of deep ReLU nets (Nair and Hinton, 2010). At first, the piecewise linear property makes it easy to compute the derivative to ease the training via gradient-type algorithms. Then, the derivative of ReLU is either 1 or 0, which in a large extent alleviates the saturation phenomenon for deep sigmoid nets and particularly the gradient vanishing/explotion issue of the gradient-descent based algorithms for the training of deep neural networks. Finally, $\sigma_{relu}(t) = 0$ for $t < 0$ enables the sparseness of the neural networks, which coincides with the biological mechanism for neural systems.

Theoretical verification for the power of depth in deep ReLU nets is a hot topic in deep learning theory. It stems from the study in (Eldan and Shamir, 2016), where some functions were constructed to be well approximated by deep ReLU nets but cannot be expressed by shallow ReLU nets with similar number of parameters. Then, numerous interesting results on the expressivity and generalization of deep ReLU nets have been provided in (Yarotsky, 2017; Safran and Shamir, 2017; Shaham et al., 2018; Petersen and Voigtlaender, 2018; Schwab and Zech, 2019; Guo et al.; Zhou, 2018, 2020; Chui et al., 2020; Han et al., 2020). Typically, it was proved in (Yarotsky, 2017) that deep ReLU nets perform at least not worse than the classical linear approaches in approximating smooth functions, and are beyond the capability of shallow ReLU nets. Furthermore, it was also exhibited in (Shaham et al., 2018) that deep ReLU nets can extract the manifold structure of the input space and the smoothness of the target functions simultaneously.

The problem is, however, that there are frequently too many hidden layers for deep ReLU nets to extract data features. Even for approximating the extremely simple square function, Lemma 1 requires $\log(\varepsilon^{-1})$ depth, which is totally different from deep sigmoid nets. Due to its infinitely differentiable property, sigmoid function is the most popular activation for shallow nets (Pinkus, 1999). The universal approximation property of shallow sigmoid nets has been verified in (Cybenko, 1989) for thirty years. Furthermore, (Mhaskar, 1993, 1996) showed that the approximation capability of shallow sigmoid nets is at least not worse than that of polynomials. However, there are also several bottlenecks for shallow sigmoid nets in embodying the locality (Chui et al., 1994), extracting the rotation-invariance (Chui et al., 2019) and producing sparse estimators (Lin et al., 2017), which show the necessity to deepen the neural networks. Different from deep ReLU nets, adding only a few hidden layers can significantly improve the approximation capability of shallow sigmoid nets. In particular, deep sigmoid nets with two hidden layers are capable of providing localized approximation (Chui et al., 1994), reflecting the spatially sparseness (Lin, 2019) and embodying the rotation-invariance (Chui et al., 2019).

In a nutshell, as shown in Table 1, it was proved in the existing literature that any function expressible for deep ReLU nets can also be well approximated by deep sigmoid nets with fewer hidden layers and free parameters. Our Theorem 3 partly reveals the reason for such a phenomenon in the sense that ReLU can be well approximated by sigmoid nets but not vice-verse.

4.2 Algorithms for DNN training

In order to address the choice of learning rate in SGD, there are many variants of SGD incorporated with adaptive learning rates called *adaptive gradient* methods. Some important adaptive gradient methods are Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), RMSprop (Tieleman and Hinton, 2012), Adam (Kingma and Ba, 2015), and AMSGrad (Reddi et al., 2018). Although these adaptive gradient methods have been widely used in deep learning, there are few theoretical guarantees when applied to the deep neural network training, a highly nonconvex and possibly nonsmooth optimization problem (Wu et al., 2019). Regardless the lack of theoretical guarantees of the existing variants of SGD, another major flaw is that they may suffer from the issue of gradient explosion/vanishing (Goodfellow et al., 2016), essentially due to the use of BackProp (Rumelhart et al., 1986) for updating the gradient during the iteration procedure.

To address the issue of gradient vanishing, there are some tricks that focus on either the design of the network architectures such as ResNets (He et al., 2016) or the training procedure such as the batch normalization (Ioffe and Szegedy, 2015) and weight normalization (Salimans and Kingma, 2016). Besides these tricks, there are many works in the perspective of algorithm design, aiming to propose some alternatives of SGD to alleviate the issue of gradient vanishing. Among these alternatives, the so called *gradient-free* type methods have recently attracted rising attention in deep learning since they may in principle alleviate this issue by their gradient-free natures, where the alternating direction method of multipliers (ADMM) and block coordinate descent (BCD) methods are two most popular ones (see, (Carreira-Perpinan and Wang, 2014; Taylor et al., 2016; Kiaee et al., 2016; Yang et al., 2016; Murdock et al., 2018; Gotmare et al., 2018; Zhang and Brand, 2017; Gu et al., 2018; Lau et al., 2018; Zeng et al., 2019)). Besides the gradient-free nature, another advantage of both ADMM and BCD is that they can be easily implemented in a distributed and parallel manner, and thus are capable of solving distributed/decentralized large-scale problems (Boyd et al., 2011).

In the perspective of constrained optimization, all the BackProp (BP), BCD and ADMM can be regarded as certain Lagrangian methods or variants for the nonlinearly constrained formulation of DNN training problem. In (LeCun, 1988), BP was firstly reformulated as a Lagrangian multiplier method. The fitting of nonlinear equations motivated by the forward pass of the neural networks plays a central role in the development of BP. Following the Lagrangian framework, the BCD methods for DNN training proposed by (Zhang and Brand, 2017; Lau et al., 2018; Zeng et al., 2019; Gu et al., 2018) can be regarded as certain Lagrangian relaxation methods without requiring the exact fitting of nonlinear constraints. Unlike in BP, such nonlinear constraints are directly lifted as quadratic penalties to the objective function in BCD, rather than involving these nonlinear constraints with Lagrangian multipliers. However, such a lifted treatment of nonlinear constraints in BCD as penalties suffers from an inconsistent issue in the sense that the solution found by BCD cannot converge to a solution satisfying these nonlinear constraints. To tackle this issue, ADMM, a primal-dual method based on the augmented Lagrangian by introducing the nonlinear constraints via Lagrangian multipliers, enables a convergent sequence satisfying the nonlinear constraints. Therefore, ADMM attracted rising attention in deep learning with various implementations (Carreira-Perpinan and Wang, 2014; Taylor et al., 2016; Kiaee et al., 2016;

Yang et al., 2016; Gotmare et al., 2018; Murdock et al., 2018). However, most of the existing ADMM type methods in deep learning only keep partial nonlinear constraints for the sake of reducing the difficulty of optimization, and there are few convergence guarantees (Gao et al., 2020).

4.3 Convergence of ADMM and challenges

Most results in the literature on the convergence of nonconvex ADMM focused on linear constrained optimization problems (e.g. (Hong et al., 2016; Wang et al., 2019)). Following the similar analysis of (Wang et al., 2019), (Gao et al., 2020) extended the convergence results of ADMM to multiaffine constrained optimization problems. In the analysis of (Hong et al., 2016; Wang et al., 2019; Gao et al., 2020), the separation of a special block of variables is crucial for the convergence of ADMM in both linear and multiaffine scenarios. Following the notations of (Wang et al., 2019), the linear constraint considered in (Wang et al., 2019) is of the form

$$\mathbf{A}\mathbf{x} + By = 0 \tag{25}$$

where $\mathbf{x} := (x_0, \ldots, x_p)$ includes $p+1$ blocks of variables, y is a special block of variables, $\mathbf{A} := [A_0, \ldots, A_p]$ and B are two matrices satisfying $\mathrm{Im}(\mathbf{A}) \subseteq \mathrm{Im}(B)$, where $\mathrm{Im}(\cdot)$ returns the image of a matrix. Similarly, (Gao et al., 2020) extended (25) to multiaffine constraint of the form, $\mathcal{A}(x_1, x_2) + \mathcal{B}(y) = 0$, where $\mathcal{A}$ and $\mathcal{B}$ are respectively some multiaffine and linear maps satisfying $\mathrm{Im}(\mathcal{A}) \subseteq \mathrm{Im}(\mathcal{B})$. Leveraging this special block variable y, the dual variables (namely, multipliers) is expressed solely by y (Wang et al., 2019, Lemma 3), and the amount of dual ascent part is controlled by the amount of descent part brought by the primal y-block update (Wang et al., 2019, Lemma 5). Together with the descent quantity arisen by the $\mathbf{x}$-block update, the total progress of one step ADMM update is descent along the augmented Lagrangian. Such a technique is in the core of analysis in (Wang et al., 2019) and (Gao et al., 2020) to deal with some multiaffine constraints in deep learning.

However, in a general formulation of ADMM for DNN training (e.g. (5)), it is impossible to separate such a special variable block y satisfying these requirements. Let us take a three-layer neural network for example. Let $\mathcal{W} := \{W_i\}_{i=1}^3$ be the weight matrices of the neural network, and $\mathcal{V} := \{V_i\}_{i=1}^3$ be the response matrices of the neural network and X be the input matrix, then the nonlinear constraints are of the following form,

$$\sigma(W_1 X) - V_1 = 0, \tag{26a}$$

$$\sigma(W_2 V_1) - V_2 = 0, \tag{26b}$$

$$W_3 V_2 - V_3 = 0, \tag{26c}$$

where σ is the sigmoid activation. Note that in (26b) and (26c), W_2 is coupled with V_1 and W_3 is coupled with V_2, respectively, so none of these four variable blocks can be separated from the others. Although W_1 in (26a) and V_3 in (26c) can be separated, the image inclusion constraint above is not satisfied. Therefore, one cannot exploit the structure in (Wang et al., 2019; Gao et al., 2020) to study such constraints in deep learning.

4.4 Key stones to the challenges and main idea of proof

In order to address the challenge of such nonlinear constraints $\sigma(W_iV_{i-1}) - V_i = 0$, we introduce a *local linear approximation* (LLA) technique. Let us take (26) for example to illustrate this idea. The most difficult block of variable is V_1 which involves two constraints, namely, a linear constraint in (26a), and a nonlinear constraint in (26b). Now we fix W_1, W_2 and V_2 as the previous updates, say W_1^0, W_2^0 and V_2^0, respectively. For the update of V_1-block, we keep the linear constraint, but relax the nonlinear constraint with its linear approximation at the previous update V_1^0,

$$\sigma(W_2^0V_1^0) - V_2^0 + \sigma'(W_2^0V_1^0) \odot \left[W_2^0(V_1 - V_1^0)\right] \approx 0, \tag{27}$$

assuming the differentiability of activation function σ. The other blocks can be handled in a similar way. Taking W_1 block for example, we relax the related nonlinear constraint via its linear approximation at the previous update W_1^0, namely, $\sigma(W_1^0X) - V_1^0 + \sigma'(W_1^0X) \odot ((W_1 - W_1^0)X) \approx 0$. The operations of LLA on the nonlinear constraints can be regarded as applying certain *prox-linear updates* (Xu and Yin, 2013) to replace the subproblems of ADMM involving nonlinear constraints as shown in Section 3.2.

To make such a local linear approximation valid, intuitively one needs: (a) the activation function σ is smooth enough; and (b) the linear approximation occurs in a small enough neighbourhood around the previous updates. Condition (a) is mild and naturally satisfied by the sigmoid type activations. But condition (b) requires us to introduce a new Lyapunov function defined in (29) by adding to the original augmented Lagrangian a proximal control between V_i and its previous updates. Equipped with such a Lyapunov function, we are able to show that an auxiliary sequence converges to a stationary point of the new Lyapunov function (see Theorem 5 below), which leads to the convergence of the original sequence generated by ADMM (see Theorem 4 in Section 3.4). Specifically, denote $\{\hat{\mathcal{Q}}^k\}$ as

$$\hat{\mathcal{Q}}^k := (\mathcal{Q}^k, \{\hat{V}_i^k\}_{i=1}^N), \tag{28}$$

with $\hat{V}_i^k := V_i^{k-1}$ for $i = 1, \ldots, N$ and $k \geq 1$, and $\hat{\mathcal{L}}(\hat{\mathcal{Q}}^k)$ as

$$\hat{\mathcal{L}}(\hat{\mathcal{Q}}^k) := \mathcal{L}(\mathcal{Q}^k) + \sum_{i=1}^N \xi_i \|V_i^k - \hat{V}_i^k\|_F^2 \tag{29}$$

for some positive constant $\xi_i > 0$ ($i = 1, \ldots, N$) specified later in Appendix D.1.4. Then we state the convergence of $\{\hat{\mathcal{Q}}^k\}$ as follows.

Theorem 5 (Convergence of $\{\hat{\mathcal{Q}}^k\}$) *Under conditions of Theorem 4, we have:*

(a) $\hat{\mathcal{L}}(\hat{\mathcal{Q}}^k)$ *is convergent.*

(b) $\hat{\mathcal{Q}}^k$ *converges to some stationary point* $\hat{\mathcal{Q}}^*$ *of* $\hat{\mathcal{L}}$.

(c) $\frac{1}{K}\sum_{k=1}^K \|\nabla\hat{\mathcal{L}}(\hat{\mathcal{Q}}^k)\|_F^2 \to 0$ *at a* $\mathcal{O}(\frac{1}{K})$ *rate.*

Theorem 5 presents the function value convergence and sequence convergence to a stationary point at a $\mathcal{O}(1/K)$ rate of the auxiliary sequence $\{\hat{\mathcal{Q}}^k\}$. By the definitions (28) and (29) of $\hat{\mathcal{Q}}^k$ and $\hat{\mathcal{L}}$, Theorem 5 directly implies Theorem 4. As shown by the proofs in Appendix D, the claims in Theorem 5 also hold under the more general assumptions for Theorem 7 in Appendix B. In Theorem 5, we only give the convergence guarantee for the proposed ADMM. It would be interesting to derive the convergence rate to highlight the role of algorithmic parameters. We will keep in study and report the result in future work.

Our main idea of proof for Theorem 5 can be summarized as follows: we firstly establish a *sufficient descent* lemma along the new Lyapunov function (see Lemma 14 in Appendix D.1), then show a *relative error* lemma (see Lemma 21 in Appendix D.1.5), and later verify the *Kurdyka-Łojasiewicz property* (Łojasiewicz, 1993; Kurdyka, 1998) (see Lemma 13 in Appendix C.2) and the *limiting continuity* property of the new Lyapunov function by Assumption 1, and finally establish Theorem 5 via following the analysis framework formulated in (Attouch et al., 2013, Theorem 2.9). In order to prove Lemma 14, we prove the following three lemmas, namely, a one-step progress lemma (see Lemma 15 in Appendix D.1.1), a dual-bounded-by-primal lemma (see Lemma 18 in Appendix D.1.2), and a boundedness lemma (see Lemma 19 in Appendix D.1.3), while to prove Lemma 21, besides Lemmas 18 and 19, we also use the Lipschitz continuity of the activation and its derivative by Assumption 1 in Appendix B. The proof sketch can be illustrated by Figure 5.

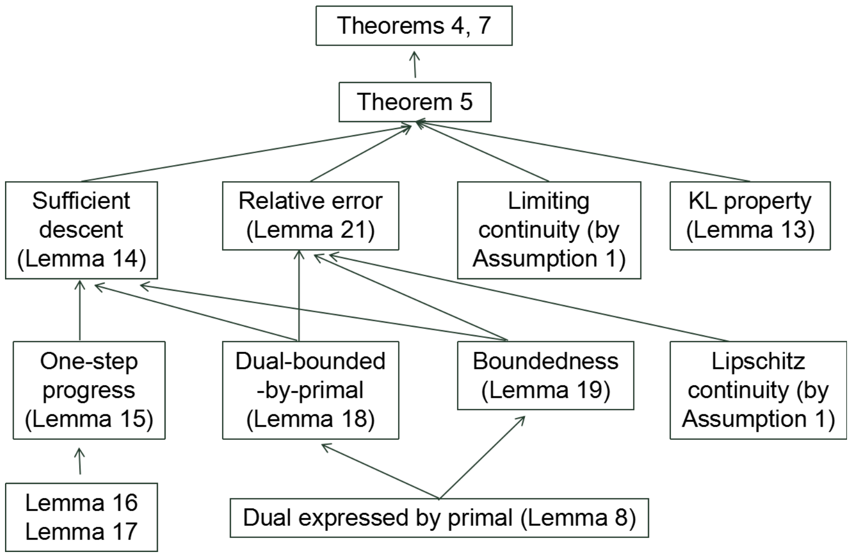

Theorems 4, 7
Theorem 5
Sufficient descent (Lemma 14)
Relative error (Lemma 21)
Limiting continuity (by Assumption 1)
KL property (Lemma 13)
One-step progress (Lemma 15)
Dual-bounded -by-primal (Lemma 18)
Boundedness (Lemma 19)
Lipschitz continuity (by Assumption 1)
Lemma 16 Lemma 17
Dual expressed by primal (Lemma 8)

Figure 5: Proof sketch of convergence of ADMM.

According to Figure 5, we show the boundedness of the sequence before the establishment of the sufficient descent lemma (i.e., Lemma 14). Such a proof procedure is different from the existing ones in the literature (say, (Wang et al., 2019)), where a sequence boundedness is usually implied by firstly showing the (sufficient) descent lemma (Wang et al., 2019, Lemma 6).

5. Toy Simulations

In this section, we conduct a series of simulations to show the effectiveness of the proposed ADMM in approximating and learning some natural functions including the square function, product gate, a piecewise L_1 radial function, and a smooth L_2 radial function, which play important roles in reflecting some commonly used data features (Safran and Shamir, 2017; Shaham et al., 2018; Chui et al., 2019; Guo et al.). In particular, we provide empirical studies to show that these important natural functions can be numerically well approximated or learned by the proposed ADMM-sigmoid pair. Furthermore we also show that the proposed ADMM-sigmoid pair is stable to its algorithmic hyper-parameters, via comparing to the popular deep learning optimizers including the vanilla SGD, SGD with momentum (called SGDM for short henceforth) and Adam (Kingma and Ba, 2015). There are four experiments concerning approximation and learning tasks: **(a)** approximation of square function, **(b)** approximation of product gate, **(c)** learning of a piecewise L_1 radial function, and **(d)** learning of a smooth L_2 radial function. All numerical experiments were carried out in Matlab R2015b environment running Windows 10, Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.2GHz 3.2GH. The codes are available at `https://github.com/JinshanZeng/ADMM-DeepLearning`.

5.1 Experimental settings

In all our experiments, we use deep fully connected neural networks with different depths and widths. Throughout the paper, the depth and width of deep neural networks are respectively the number of hidden layers and number of neurons in each hidden layer. For simplicity, we only consider deep neural networks with the same width for all the hidden layers. We consider both deep sigmoid nets and deep ReLU nets in the simulation. Motivated by the existing literature (Guo et al.), the depth required for deep ReLU nets is in general much more than that for deep sigmoid nets in the aforementioned approximation or learning tasks. For the fairness of comparison, we consider deep ReLU nets with more hidden layers, i.e., the maximal depth of deep ReLU nets is 20 while that of deep sigmoid nets is only 5 or 6, as presented in Table 2. Besides the vanilla SGD-ReLU pair (*SGD (ReLU)* for short), we also consider SGD-sigmoid pair (*SGD (sigmoid)* for short), SGDM-ReLU pair (*SGDM (ReLU)* for short), and Adam-ReLU pair (*Adam (ReLU)* for short) as the competitors. Similarly, we denote by *ADMM (sigmoid)* the proposed ADMM-sigmoid pair.

For each experiment, our purposes are mainly two folds: excellent approximation or learning performance, and stability with respect to initialization schemes and penalty parameters with appropriate neural network structures for the proposed ADMM-sigmoid pair. For the first purpose, we consider deep neural networks with different depths and widths as presented in Table 2. Moreover, for ADMM, we empirically set the regularization parameter $\lambda = 10^{-6}$ and the augmented Lagrangian parameters β_i's as the same 1, while for SGD methods, we empirically utilize the step exponential decay (or, called *geometric decay*) learning rate schedule with the decay exponent 0.95 for every 10 epochs. For SGDM and Adam, we use the default settings as presented in Table 2. The number of epochs in all experiments is empirically set as 2000. The specific settings of these experiments are presented in Table 2.

Table 2: Experimental settings for toy simulations. Sample sizes for approximation tasks are set as 1000, and training and test samples sizes for learning tasks are set as 1000 respectively. The notation $[a : b]$ is denoted by the set $\{a, a+1, \ldots, b\}$ for two integers a, b. † In the case of learning the L_2 radial function, ranges of depth of deep sigmoid and ReLU nets are $[2, 6]$ and $[2, 20]$, respectively.

task	functions	deep fully-connected NNs		SGDs (sigmoid/ReLU), SGDM		SGDM	Adam	ADMM
		width	depth	learning rate (lr)	batch size	(momentum)		(λ, β)
Approx.	x^2	$20 \times [1:5]$	$[1,5]$ (sigmoid), $[1,20]$ (ReLU)	0.1×0.95^k per 10 epochs,	50	0.5	lr: 0.001 β_1: 0.9 β_2: 0.999 ϵ: 1e-8	$(10^{-6}, 1)$
	uv	$60 \times [1:5]$						
Learn.	$(\\|x\\|_1 - 1)_+$	$10 \times [1:6]$						
	$g(\|x\|^2)$	$100 \times [1:5]$	$[2,6]$, $[2,20]$ †					

For the second purpose, we consider different regularization and penalty parameters ($\lambda \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}, \beta \in \{0.01, 0.1, 0.5, 1, 5, 10, 100\}$), as well as several existing initialization schemes for ADMM under the optimal neural network structure determined by the first part. Particularly, we consider the following four types of schemes yielding six typical initializations:

(1) **LeCun random initialization** (LeCun et al., 1998): the components of the weight matrix W_l at the l-th layer are generated i.i.d. according to some random distribution with zero mean and variance $\frac{1}{d_{l-1}}$, $l = 1, \ldots, N$. Particularly, we consider two special *LeCun random* initialization schemes generated respectively according to the uniform and Gaussian distributions, i.e., $W_l \sim U([-\sqrt{\frac{3}{d_{l-1}}}, \sqrt{\frac{3}{d_{l-1}}}])$ (*LeCun-Unif* for short) and $W_l \sim \mathcal{N}(0, \frac{1}{d_{l-1}})$ (*LeCun-Gauss* for short).

(2) **Random orthogonal initialization** (Saxe et al., 2014): the weight matrix W is set as some random orthogonal matrix such that $W^T W = \mathbf{I}$ or $WW^T = \mathbf{I}$. We call it *Orth-Unif* (or *Orth-Gauss*) for short if the random matrix is generated i.i.d. according to the uniform random (or, Gaussian) distribution.

(3) **Xavier initialization** (Glorot and Bengio, 2010): each $W_l \sim U([-\sqrt{\frac{6}{d_{l-1}+d_l}}, \sqrt{\frac{6}{d_{l-1}+d_l}}])$, $l = 1, \ldots, N$.

(4) **MSRA initialization** (He et al., 2015): $W_l \sim \mathcal{N}(0, \frac{2}{d_l})$, $l = 1, \ldots, N-1$, and $W_N \sim \mathcal{N}(0, \frac{1}{d_N})$ since there is no ReLU activation for the last layer.

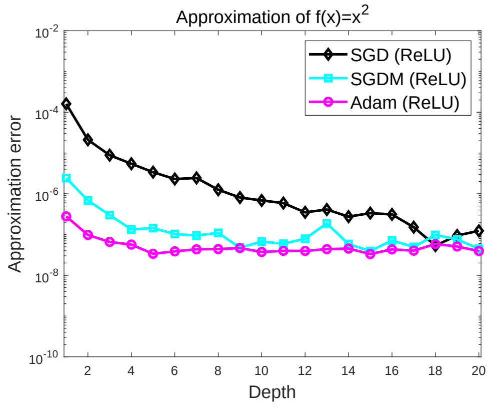
The default settings for initial threshold vectors in the above initialization schemes are set to be 0. For each group of parameters, we run 20 trails for average. Specifically, for the approximation tasks, the performance of an algorithm is measured by the *approximation error*, defined as the average of these 20 trails's mean square errors, while for the learning tasks, the performance of an algorithm is measured by the *test error*, defined as the average of the mean square errors of these trails over the test data.

5.2 Approximation of square function

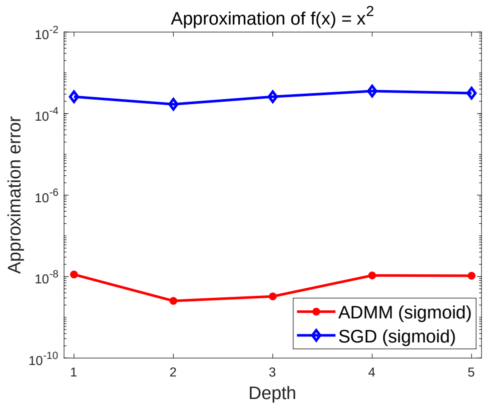
In these experiments, we consider the performance of the ADMM-sigmoid pair in approximating the univariate square function, that is, $f(x) = x^2$ on $[-1, 1]$. The specific experimental settings can be found in Table 2.

Table 3: Experimental results of different algorithms in approximating $f(x) = x^2$. The standard derivation of the approximation error is presented in the parentheses. The running time is recorded in seconds. The depth and width of the optimal network structure in terms of approximation error is presented in the last row.

Algorithm	SGD (ReLU)	SGDM (ReLU)	Adam (ReLU)	SGD (sigmoid)	ADMM (sigmoid)
Approx. Error	5.34e-8(2.34e-8)	3.95e-8(1.25e-8)	3.33e-8(1.46e-8)	2.46e-4(1.69e-4)	**2.53e-9(1.18e-9)**
Run Time (s)	26.99	41.35	38.26	3.45	9.47
(depth, width)	(18,100)	(15,100)	(15,80)	(2,60)	(2,100)



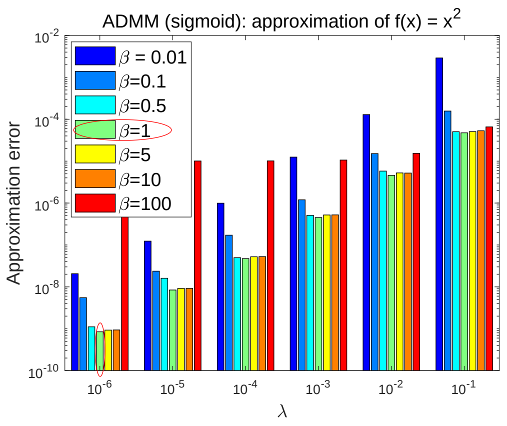
(a) Deep ReLU nets



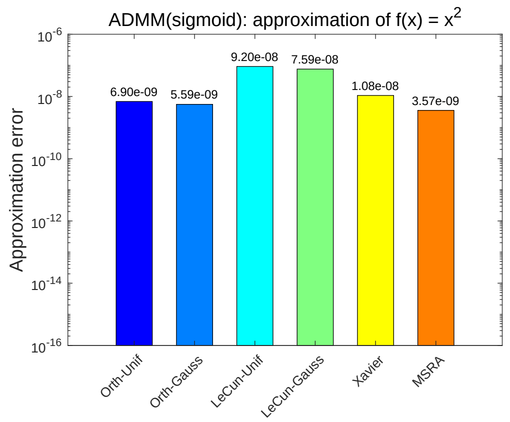
(b) Deep sigmoid nets

Figure 6: Effect of the depth of neural networks in approximating the square function.

A. Approximation performance of ADMM. Experiment results over the best neural network structures are presented in Table 3, and trends of approximation errors with respect to the depth are shown in Figure 6. From Table 3, the ADMM-SGD pair can approximate the square function within very high precision, i.e., in the order of 10^{-9}, which is slightly better than that of competitors for deeper ReLU nets, and is much better than the SGD-sigmoid pair with the same depth. Specifically, optimal depths for SGD (ReLU), SGDM (ReLU) and Adam (ReLU) are 18, 15, 15, respectively, while the optimal depth for ADMM (sigmoid) is only 2, which matches the theoretical results in approximation theory, as shown in (Chui et al., 2019, Proposition 2). In terms of running time, ADMM (sigmoid) with optimal network structures is generally faster than the SGDM (ReLU) and Adam (ReLU) with optimal network structures as presented in the third row of Table 3, mainly due to the depth required for ADMM (sigmoid) is much less than those for deep ReLU nets SGD (ReLU), SGDM (ReLU) and Adam (ReLU). Moreover, according to Figure 6, ADMM (sigmoid) can yield high approximation precision with less layers than the competitors. These experimental results demonstrate that the proposed ADMM can embody the advantage of deep sigmoid nets on approximating the square function, as pointed out in the existing theoretical literature (Chui et al., 2019).



(a) Effect of parameters of ADMM



(b) Stability of initialization

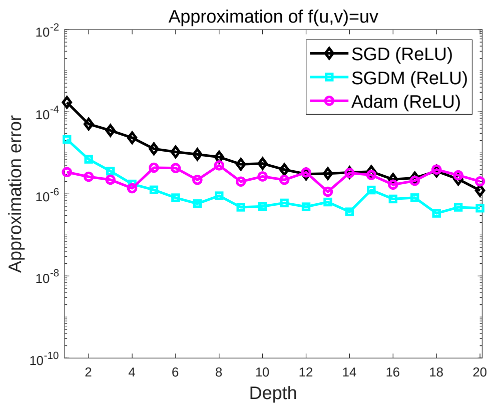
Figure 7: Effect of parameters for ADMM in approximating the univariate square function.

B. Effect of parameters for ADMM. There are mainly two parameters for the proposed ADMM, i.e., the model parameter λ (also called as the regularization parameter) and the algorithm parameter β involved in the augmented Lagrangian (also called as the penalty parameter). In this experiment, we consider the performance of ADMM in approximating the univariate square function with different model and algorithmic parameters, under the optimal neural networks, i.e., deep fully-connected neural networks with depth 2 and width 100. Specifically, the regularization and penalty parameters vary from $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and $\{0.01, 0.1, 0.5, 1, 5, 10, 100\}$, respectively. The approximation errors of ADMM with these parameters are shown in Figure 7(a). From Figure 7(a), considering the penalty parameter β, ADMM with $\beta = 1$ achieves the best performance in most cases, as also observed in the experiments later. Thus, in practice, we can empirically set the penalty parameter β as 1. Since λ is a model parameter, it usually has significant effect on the performance of the proposed ADMM. From Figure 7(a), a small regularization parameter (say, $\lambda = 10^{-6}$) is sufficient to yield an ADMM solver with high approximation precision.

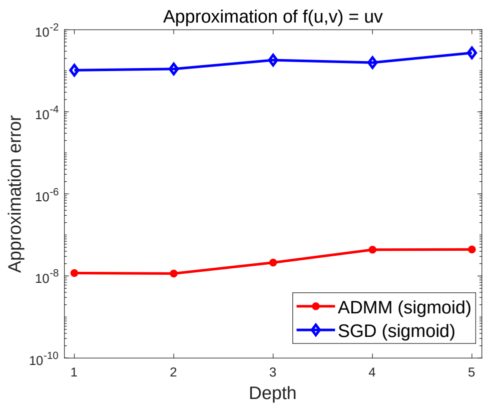
C. Effect of initial schemes. Besides the MSRA initialization (He et al., 2015), there are some other commonly used initial schemes such as the random orthogonal initializations (Saxe et al., 2014), LeCun random initializations (LeCun et al., 1998), and Xavier initialization (Glorot and Bengio, 2010). Under the optimal parameter settings presented in Table 3, the performance of the ADMM-sigmoid pair with different initialization schemes is presented in Figure 7(b). From Figure 7(b), the proposed ADMM performs well for all the initialization schemes. This demonstrates that the proposed ADMM is stable to the initial scheme.

Table 4: Experimental results of different algorithms in approximating $f(u, v) = uv$.

Algorithm	SGD (ReLU)	SGDM (ReLU)	Adam (ReLU)	SGD (sigmoid)	ADMM (sigmoid)
Approx. Error	1.22e-6(3.68e-7)	3.37e-7(1.29e-7)	1.13e-6(4.37e-7)	1.13e-3(2.33e-4)	**2.62e-9(1.05e-9)**
Run Time (s)	66.37	54.17	46.58	9.72	17.29
(depth, width)	(20,300)	(18,180)	(13,120)	(2,240)	(2,300)



(a) Deep ReLU nets



(b) Deep sigmoid nets

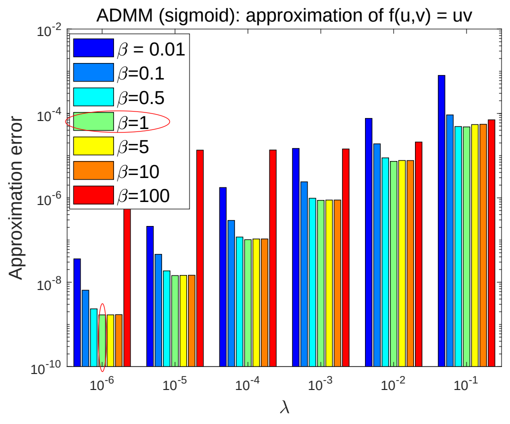
Figure 8: Effect of the depth of neural networks in approximating the product-gate function.

5.3 Approximation of product gate

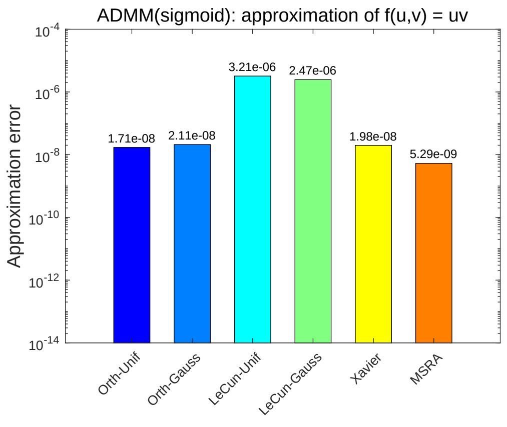
In this subsection, we present experimental results in approximating the product gate function, i.e., $f(u, v) = uv$ for $u, v \in [-1, 1]$. The specific experimental settings in approximating the product gate function can be found in Table 2.

A. Approximation performance of ADMM. The performance of ADMM and competitors is presented in Table 4, and their performance with respect to the depth is depicted in Figure 8. From Table 4, the product gate function can be well approximated by the ADMM-sigmoid pair with precision in the order of 10^{-9}, which is better than those of competitors including SGD (ReLU), SGDM (ReLU) and Adam (ReLU), even when more hidden layers are involved in the training. It follows from Figure 8(b) and Table 4 that the optimal depth for ADMM in approximating the product gate function is 2, which matches the theoretical depth for the approximation of product gate as shown in (Chui et al., 2019, Proposition 3). Similar to the case of approximating square function, the running time of the proposed ADMM-sigmoid pair is less than the SGD type competitors for deep ReLU nets with more hidden layers.

B. Effect of parameters of ADMM. Similar to Section 5.2 B, we also consider the effect of parameters λ and β for ADMM under the optimal network structures, which is presented in Figure 9(a). From Figure 9(a), the effect of the concerned parameters on the performance of ADMM in approximating the product gate function is very similar to that in the approximation of univariate square function. It can be observed that the



(a) Effect of parameters of ADMM



(b) Stability of initialization

Figure 9: Effect of parameters for ADMM in approximating the product-gate function.

settings of parameters with ($\lambda = 10^{-6}, \beta = 1$) are empirically good for ADMM in these two approximation tasks.

C. Effect of initial schemes. Moreover, in this experiment, we consider the performance of ADMM (sigmoid) for the aforementioned six different initialization schemes. The experimental results are shown in Figure 9(b). It can be observed in Figure 9(b) that all the initial schemes are generally effective in yielding an ADMM solver with high precision. Among these effective initialization schemes, the LeCun type of initializations perform slightly worse than the others. This, in some extent, also implies that the proposed ADMM is usually stable to initial schemes.

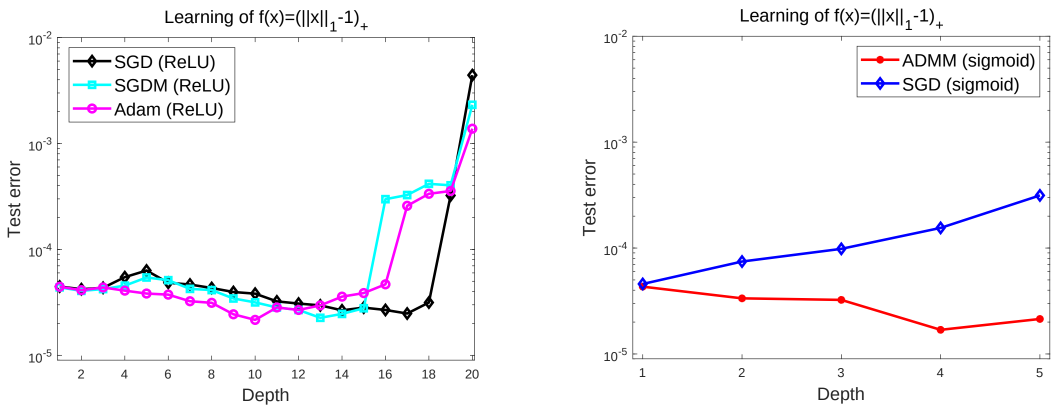
5.4 Learning L_1 radial function

In this subsection, we consider the performance of the ADMM-sigmoid pair for learning a two-dimensional L_1 radial function, i.e., $f(x) = (\|x\|_1 - 1)_+ := \max\{0, \|x\|_1 - 1\}$ for $x \in [r,(1+\epsilon)r] \times [r,(1+\epsilon)r]$ for some $r > 0$ and $\epsilon > 0$. Such an L_1 radial function was particularly considered in (Safran and Shamir, 2017). In our experiments, we let $\epsilon = 1/2$ and $r = 1 - \frac{\epsilon}{2}$ in light of the theoretical studies in (Safran and Shamir, 2017). Different from the approximation tasks in Sections 5.2 and 5.3, samples generated for the learning task include both training and test samples, where training samples are commonly generated with certain noise and the test samples are clean data. In these experiments, we consider Gaussian noises with different variances.

A. Learning performance of ADMM. Optimal test errors of different algorithms for learning the L_1 radial function are presented in Table 5, where the variance of Gaussian noise added into the training samples is 0.1. The associated test errors of these algorithms with respect to the depth of neural networks are presented in Figure 10. From Table 5, the considered L_1 radial function can be well learned by both ADMM and SGD type methods. Specifically, the performance of the proposed ADMM is slightly better than SGD type methods. In particular, the optimal depth of deep sigmoid nets trained by ADMM is only 4, which is much less than those of deep ReLU nets trained by SGD type methods. Under optimal network structures, the running time of the suggested ADMM-pair is slightly

Table 5: Performance of different algorithms for learning L_1 radial function with 0.1 Gaussian noise.

Algorithm	SGD (ReLU)	SGDM (ReLU)	Adam (ReLU)	SGD (sigmoid)	ADMM (sigmoid)
Test Error	2.48e-5(9.74e-6)	2.26e-5(7.88e-6)	2.16e-5(8.53e-6)	4.58e-5(1.59e-5)	**1.69e-5(4.34e-6)**
Run Time (s)	23.24	18.23	14.66	1.68	10.36
(depth, width)	(17,50)	(13,50)	(10,50)	(1,20)	(4,10)



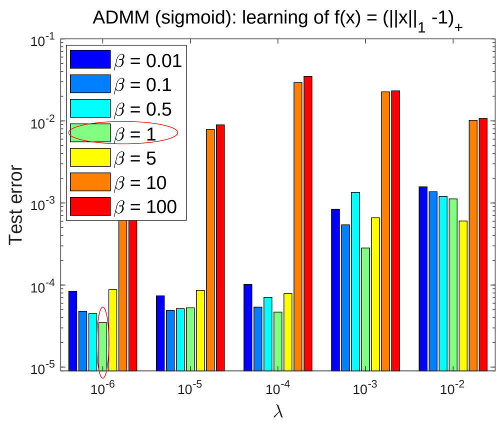
(a) Deep ReLU nets (b) Deep sigmoid nets

Figure 10: Effect of the depth of neural networks in learning the L_1 radial function.

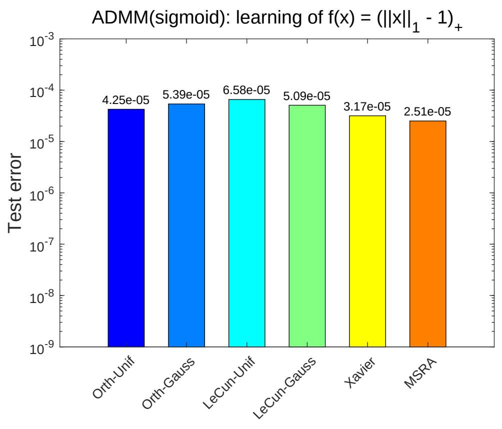
less than that of SGD type methods for deep ReLU nets, due to the less depth of deep sigmoid nets. According to Figure 10(b), the proposed ADMM performs better than SGD for training deep sigmoid nets, and as the depth increasing, the performance of SGD gets worse possibly due to the vanishing gradient issue, while our suggested ADMM can alleviate the issue of vanishing gradient and thus achieve better and better performance in general as the depth increases in our considered range of depth, i.e., $\{1, 2, 3, 4, 5\}$.

B. Effect of parameters and initialization. Under the optimal neural network structures specified in Table 5, we consider the effect of parameters, i.e., (λ, β) for ADMM, as well as the effect of the initialization schemes for both ADMM and SGD type methods. The numerical results are shown in Figure 11. From Figure 11(a), we can observe that the specific parametric setting, i.e., $\lambda = 10^{-6}$ and $\beta = 1$, is also empirically effective in learning L_1 radial function. By Figure 11(b), the proposed ADMM performs well for all the concerned random initialization schemes.

C. Robustness to the noise. Moreover, we consider the performance of the proposed ADMM for training data with different levels of noise. Specifically, under the optimal parameters specified in Table 5, we consider several levels of noise, where the variance of Gaussian noise varies from $\{0.1, 0.3, \ldots, 1.5\}$. Trends of training and test errors are presented in Figure 12(a) and (b) respectively. From Figure 12(a), the proposed ADMM is generally trained well in the sense that the training error almost fits the true noise level. In this case, we can observe from Figure 12(b) that the proposed ADMM is robustness to

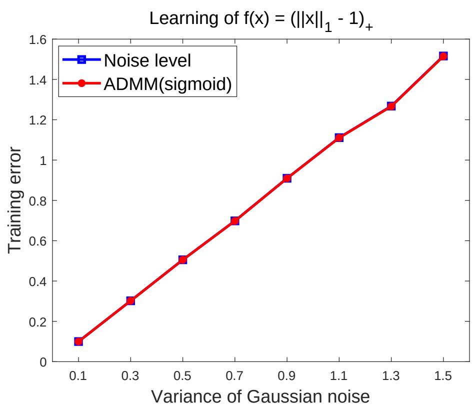


(a) Effect of parameters of ADMM

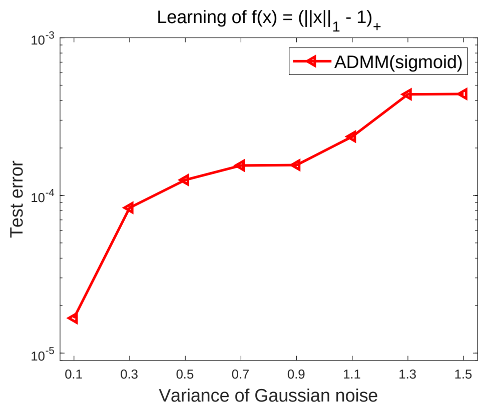


(b) Stability of initialization

Figure 11: Effect of parameters and initial schemes for ADMM in learning the L_1 radial function.



(a) Training error



(b) Test error

Figure 12: Robustness of the proposed ADMM to the noise in learning L_1 radial function.

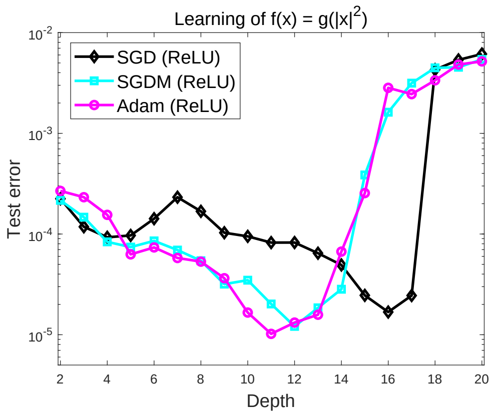
the noise in the sense that the test error increases much slower than the increasing of the variance of Gaussian noise.

5.5 Learning L_2 radial function

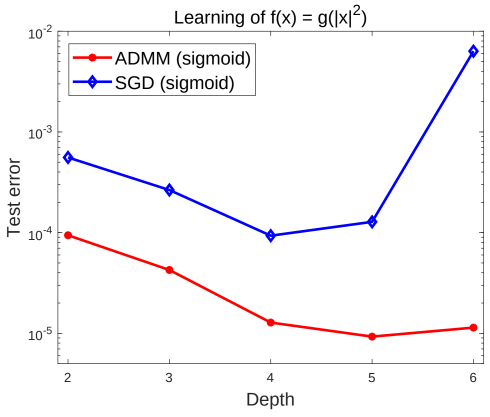
In this subsection, we consider to learn certain smooth L_2 radial function that frequently reflects the rotation-invariance feature in deep learning (Chui et al., 2019). Specifically, we adopt a two-dimensional smooth L_2 radial function, i.e., $f(x) = g(|x|^2)$, where $x \in [-1,1] \times [-1,1]$, $|x|^2 := \sum_{i=1}^2 x_i^2$, and $g(t) = (1-t)_+^5(8t^2+5t+1)$ on $\mathbb{R}$ is some Wendland function (Lin et al., 2019). Except the target function f, the experimental settings in these experiments are similar to those in Section 5.4.

Table 6: Test errors of different algorithms for learning L_2 radial function with 0.1 Gaussian noise.

Algorithm	SGD (ReLU)	SGDM (ReLU)	Adam (ReLU)	SGD (sigmoid)	ADMM (sigmoid)
Test Error	1.68e-5(6.43e-6)	1.21e-5(5.25e-6)	1.02e-5(4.88e-6)	9.33e-5(1.42e-5)	**9.28e-6(1.01e-6)**
Run Time (s)	104.52	116.44	108.49	18.13	47.36
(depth, width)	(16,300)	(12,400)	(11,400)	(4,200)	(5,300)



(a) Deep ReLU nets

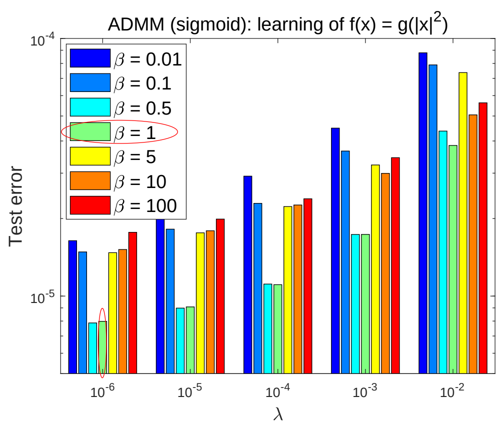


(b) Deep sigmoid nets

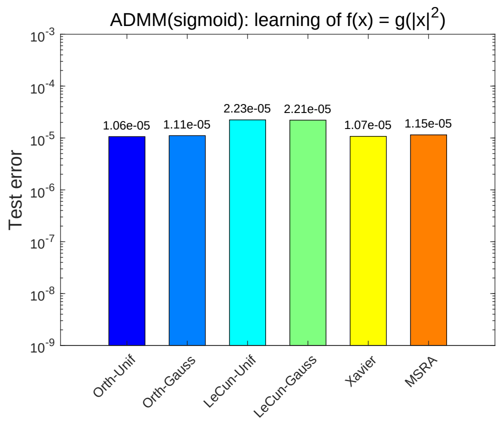
Figure 13: Effect of the depth of neural networks in learning the L_2 radial function.

A. Learning performance of ADMM. The test error of the considered algorithms in learning such a smooth L_2 radial function is presented in Table 6, while trends of test errors with respect to the depth are shown in Figure 13. By Table 6, the considered smooth L_2 radial function can be learned by the proposed ADMM well with a small test error. Specifically, in terms of test error, the performance of the ADMM-sigmoid pair is slightly better than that of SGD type methods for deep ReLU nets, and the optimal depth of deep sigmoid nets required by ADMM is much smaller than those of deep ReLU nets required by the concerned SGD type methods. Due to less depth, the running time of ADMM is less than that of the concerned SGD type methods for deep ReLU nets under the optimal settings of neural networks. Moreover, from Figure 13(a), a deeper ReLU network with about 10 layers is generally required to learn the L_2 radial function with a good test error, while from Figure 13(b), the depth of deep sigmoid nets trained by ADMM can be much smaller (i.e., about 5) to yield a good test error.

B. Effect of parameters and initialization. In this part, we consider the effect of parameters (i.e., λ and β) of ADMM as well as the effect of initialization for learning the L_2 radial function in the optimal settings specified in Table 6. The numerical results are presented in Figure 14. From Figure 14(a), the effect of parameters are similar to previous three simulations and it can be observed that the specific settings, i.e., $\lambda = 10^{-6}$ and $\beta = 1$,

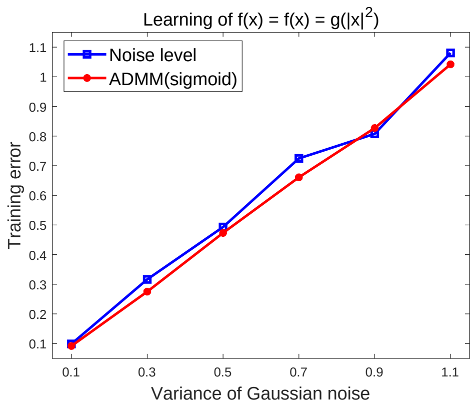


(a) Effect of parameters of ADMM

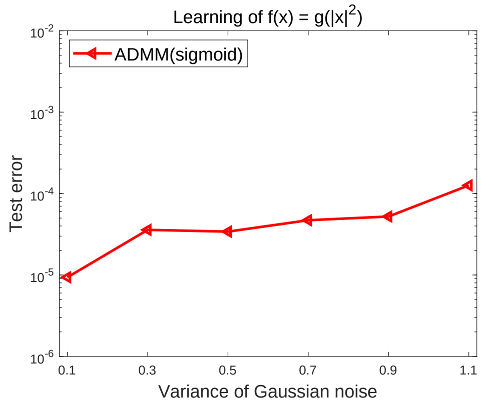


(b) Stability of initialization

Figure 14: Effect of parameters and initial schemes for ADMM in learning smooth L_2 radial function.



(a) Training error



(b) Test error

Figure 15: Robustness of the proposed ADMM to the noise in learning L_2 radial function.

are empirically effective. From Figure 14, we also observe that ADMM is effective to all the random initialization schemes.

C. Robustness to noise. Similar to the learning of L_1 radial function, we consider the performance of the proposed ADMM for noisy training data with different levels of noise. Specifically, the variance of the Gaussian noise added into the training samples varies from $\{0.1, 0.3, 0.5, 0.7, 0.9, 1.1\}$. Curves of training error and test error are shown respectively in Figure 15(a) and (b). From Figure 15, the behavior in learning L_2 radial function is similar to that in learning L_1 radial function as shown in Figure 12. This demonstrates that the proposed ADMM is also robust to noise in learning such a smooth L_2 radial function.

Table 7: Experimental settings for real-data experiments. The number of epochs for each case is set empirically to be 200.

dataset	(training size, test size)	Network structure		SGDs (sigmoid/ReLU),SGDM		SGDM	Adam	ADMM
		width	depth	batch size	learning rate	(momentum)	lr:0.001	(λ, β)
Earthquake	(4173,4000)	$20 \times [1:10]$	[1:6]	100	0.1×0.95^k, per 10 epochs	0.5	β_1: 0.9 β_2: 0.999 ϵ: 1e-8	$\lambda \in 10^{[-6:-2]}$ $\beta = 1$
EYB	(2432,2432)	$20 \times [1:10]$	1	50				
PTB	(7000,7552)	$64 \times [1:4]$	[1:10]	100				

6. Real Data Experiments

In this section, we provide three real-data experiments over the earthquake intensity database, the extended Yale B (EYB) face recognition database and the PTB Diagnostic ECG database, to demonstrate the effectiveness of the proposed ADMM. We choose these three datasets since they can in some sense reflect certain features that can be well approximated by deep sigmoid nets, and thus, the benefits of the proposed ADMM can be embodied over these datasets. Specific experimental settings are presented in Table 7, where the penalty parameter β is empirically set as 1 and the regularization parameter λ is chosen via cross validation from the set $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ according to the previous studies of toy simulations.

6.1 Earthquake intensity dataset

Earthquake Intensity Database is from: *https://www.ngdc.noaa.gov/hazard/intintro.shtml.* This database contains more than 157,000 reports on over 20,000 earthquakes that affected the United States from the year 1638 to 1985. For each record, the features include the geographic latitudes and longitudes of the epicentre and "reporting city" (or, locality) where the Modified Mercalli Intensity (MMI) was observed, magnitudes (as a measure of seismic energy), and the hypocentral depth (positive downward) in kilometers from the surface. The output label of each record is measured by MMI, varying from 1 to 12 in integer. An illustration of the generation procedure of each earthquake record is shown in Figure 16(a). In this paper, we transfer such multi-classification task into the binary classification since this database is very unbalanced (say, there is only one sample for the class with MMI being 1). Specifically, we set the labels lying in 1 to 4 as the positive class, while the other labels lying in 5 to 12 as the negative class, mainly according to the damage extent of the earthquake suggested by the referred website. Moreover, we removed those incomplete records with missing labels. After such preprocessing, there are total 8173 effective records, where the numbers of samples in positive and negative classes are respectively 5011 and 3162. We divide the total data set into the training and test sets randomly, where the training and test sample sizes are 4173 and 4000, respectively. Before training, we use the *z-scoring normalization* for each feature, that is, $\frac{x_i - \mu}{\sigma}$ with μ and σ being respectively the mean and standard deviation of the ith feature $\{x_i\}$. The classification accuracies of all algorithms are shown in Table 8. The effect of the depth of neural network, algorithmic parameters, and random initial schemes are shown in Figure 16 (b)-(d) respectively.

According to Table 8, the performance of the proposed ADMM is comparable to the state-of-the-art methods in terms of test accuracy. Specifically, the proposed ADMM is

Table 8: Test accuracies (%) of different algorithms for earthquake intensity database. The baseline of the test accuracy is 80.48% (Zeng et al., 2020).

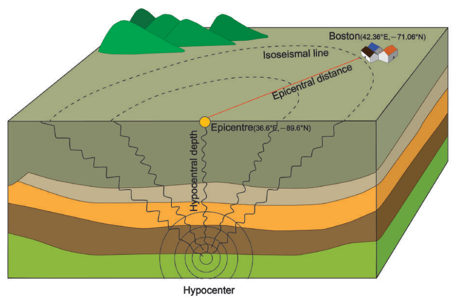
Algorithm	SGD (ReLU)	SGDM (ReLU)	Adam (ReLU)	SGD (sigmoid)	ADMM (sigmoid)
Test Acc(%)	81.24(0.45)	81.16(0.32)	**81.31(0.36)**	79.94(0.23)	81.26(0.31)
Run Time (s)	4.74	14.20	13.24	2.60	12.64
(depth, width)	(2,120)	(5,140)	(4,80)	(1,100)	(3,80)

slightly worse than Adam, and outperforms the other competitors in terms of test accuracy, while in terms of running time, the proposed ADMM is slightly faster than Adam and SGDM under the associated optimal network settings, mainly because the optimal depth of the deep sigmoid nets trained by ADMM is less than those of deep ReLU nets trained by Adam and SGDM. Compared to the SGD counterpart for deep sigmoid nets, the performance of the proposed ADMM is much better in terms of test accuracy. It can be observed from Figure 16(b) that the vanilla SGD may suffer from the gradient vanishing/explosion issue when training a slightly deeper sigmoid nets (say, the depth is larger than 5) due to the saturation of the sigmoid activation, while the proposed ADMM can avoid such saturation and thus alleviate the gradient vanishing/explosion issue. From Figure 16(c), the proposed ADMM with the default settings, i.e., λ = 1e-6 and $\beta = 1$ in general yields the best performance. Moreover, it can be observed from Figure 16(d) that the proposed ADMM is stable to the commonly used initialization schemes under the optimal neural network structure specified in Table 8.

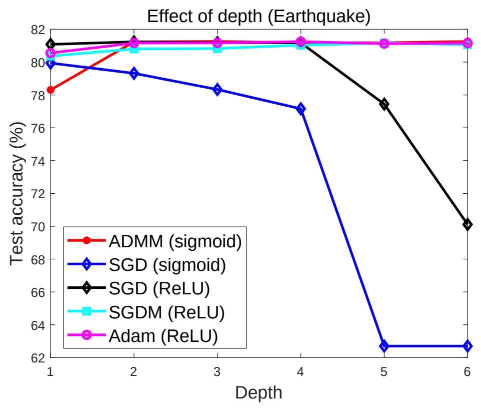
6.2 Extended Yale B face recognition database

In the extended Yale B (EYB) database, well-known face recognition database (Lee et al., 2005), there are in total 2432 images for 38 objects under 9 poses and 64 illumination conditions, where for each objective, there are 64 images. The pixel size of each image is 32×32. In our experiments, we randomly divide these 64 images for each objective into two equal parts, that is, one half of images are used for training while the rest half of images are used for testing. For each image, we normalize it via the z-scoring normalization. The specific experimental settings for this database can be found in Table 7. Particularly, we empirically use a shallow neural network with depth one and various of widths, since such shallow neural network is good enough to extract the low-dimensional manifold feature of this face recognition data, as shown in Table 9. The effect of network structures and stability of the proposed ADMM to initialization schemes are shown in Figure 17(a) and (b) respectively.

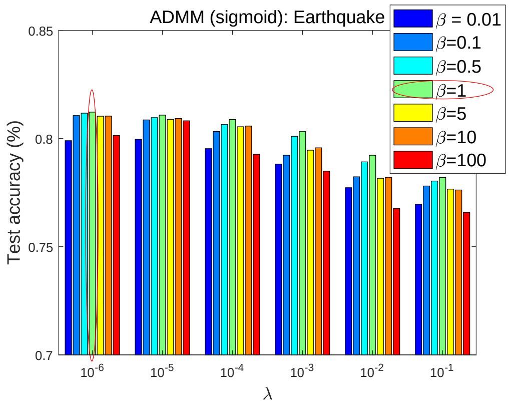
According to Table 9, the proposed ADMM achieves the state-of-the-art test accuracy (see, Lu et al. (2020)) with a smaller width of the sigmoid nets when compared to the concerned competitors. From Figure 17, the proposed ADMM can achieve a very high test accuracy for most of the concerned widths of the networks and is stable to the commonly used random initial schemes.



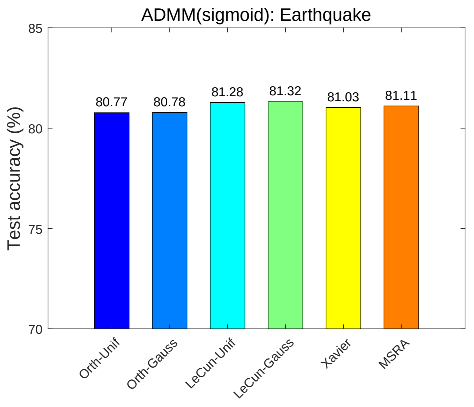
(a) An illustration of earthquake data



(b) Effect of depth of NNs



(c) Effect of parameters



(d) Stability to initial schemes

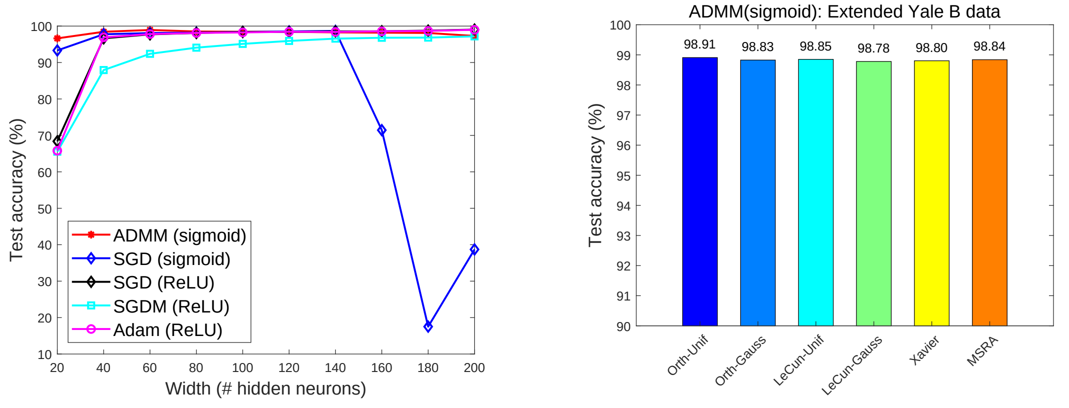
Figure 16: Performance of ADMM in earthquake intensity data: (a) an illustration of the earthquake intensity data (Zeng et al., 2020); (b) the effect of depth of the neural network for different algorithms; (c) the effect of algorithmic parameters for the proposed ADMM; (d) the stability of the proposed ADMM to different initial schemes.

6.3 PTB Diagnostic ECG database

An ECG is a 1D signal which is the result of recording the electrical activity of the heart using an electrode. It is one of popular tools that cardiologists use to diagnose heart anomalies and diseases. The PTB diagnostic ECG database is available at `https://github.com/CVxTz/ECG_Heartbeat_Classification` and was preprocessed by (Kachuee et al., 2018). There are 14,552 samples in total with 2 categories. The specific experimental settings for this database can be found in Table 7. The experiment results of the proposed ADMM and concerned competitors are presented in Table 10. The effect of network structures and stability of the proposed ADMM to initialization schemes are shown in Figure 18(a) and (b) respectively.

Table 9: Performance of different algorithms for extended Yale B database. The baseline of the test accuracy is about 96% in (Lu et al., 2020).

Algorithm	SGD (ReLU)	SGDM (ReLU)	Adam (ReLU)	SGD (sigmoid)	ADMM (sigmoid)
Test Acc(%)	98.84(0.28)	97.18(0.28)	98.91(0.34)	98.67(0.41)	**98.93(0.43)**
Run Time (s)	19.78	23.99	48.92	16.95	21.36
(depth, width)	(1,200)	(1,200)	(1,200)	(1,140)	(1,60)



(a) Effect of network structure. (b) Stability to initial schemes of ADMM

Figure 17: Performance of ADMM in extended Yale B database: (a) the effect of width for different algorithms; (b) the stability of the proposed ADMM to different random initial schemes.

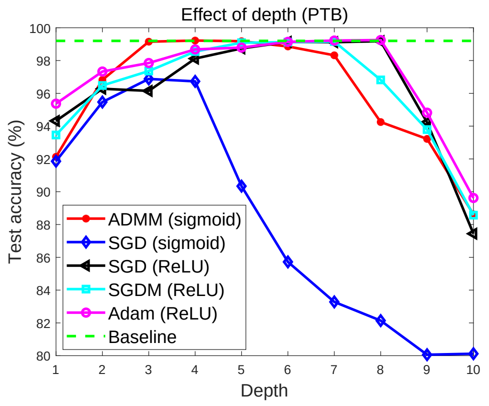
According to Table 10, the proposed ADMM achieves the state-of-the-art test accuracy (see, Kachuee et al. (2018)) with a less width of sigmoid nets when compared to the concerned competitors. Specifically, the optimal depth of deep sigmoid nets trained by ADMM is 4, while those of deep ReLU nets trained respectively by SGD, SGDM and Adam are 8, 7, 7. This also verifies our previous claim on the advantage of deep sigmoid nets in feature representation. Due to less hidden layers, the proposed ADMM is slightly faster than the SGD competitors for deep ReLU nets. From Figure 18(a), when the depth of deep sigmoid nets is larger than 8, the performance of all considered algorithms degrades much possibly due to the overfitting. From Figure 18(b), the proposed ADMM is stable to the commonly used random initial schemes under the optimal neural network setting as presented in Table 10.

Acknowledgments

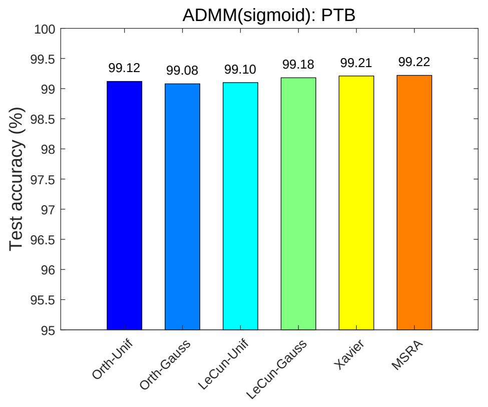
We would like to thank Prof. Wotao Yin at UCLA and Dr. Yugen Yi at Jiangxi Normal University for their helpful discussions on this work. The work is supported by National Key

Table 10: Performance of different algorithms for PTB diagnostic ECG database. The baseline of the test accuracy is 99.20% in (Kachuee et al., 2018).

Algorithm	SGD (ReLU)	SGDM (ReLU)	Adam (ReLU)	SGD (sigmoid)	ADMM (sigmoid)
Test Acc(%)	99.18(0.32)	99.16(0.28)	**99.25(0.25)**	96.88(0.46)	99.22(0.11)
Run Time (s)	29.82	40.77	30.83	12.28	29.17
(depth, width)	(8,192)	(7,192)	(7,256)	(3,256)	(4,128)



(a) Effect of network structure.



(b) Stability to initial schemes of ADMM

Figure 18: Performance of ADMM in PTB diagnostic ECG database: (a) the effect of depth for different algorithms; (b) the stability of the proposed ADMM to different initial schemes.

R&D Program of China (No.2020YFA0713900). The work of Jinshan Zeng is supported in part by the National Natural Science Foundation of China [Project No. 61977038] and by the Thousand Talents Plan of Jiangxi Province [Project NO. jxsq2019201124]. The work of Shao-Bo Lin is supported in part by the National Natural Science Foundation of China [Project No. 61876133]. This work of Yuan Yao is supported in part by Hong Kong Research Grant Council Project NO. RGC16308321 and 16303817, NSFC/RGC Joint Research Scheme N_HKUST635/20, and ITF UIM/390. The work of Ding-Xuan Zhou is supported partially by the Research Grants Council of Hong Kong [Project No. CityU 11307319], Laboratory for AI-Powered Financial Technologies and by the Hong Kong Institute for Data Science. This research made use of the computing resources of the X-GPU cluster supported by the Hong Kong Research Grant Council Collaborative Research Fund: C6021-19EF.

Appendix A. Proof of Theorem 3

To prove Theorem 3, we need the following "product-gate" for shallow sigmoid nets, which can be found in (Chui et al., 2019, Proposition 1).

Lemma 6 *Let $M > 0$. For any $\nu \in (0,1)$ there exists a shallow sigmoid net $h_{9,\nu}^{prod} : \mathbb{R}^2 \to \mathbb{R}$ with 9 free parameters bounded by $\mathcal{O}(\nu^{-6})$ such that for any $t, t' \in [-M, M]$,*

$$|tt' - h_{9,\nu}^{prod}(t,t')| \le \nu.$$

Then, we can give the proof of Theorem 3 as follows.

Proof [Proof of Theorem 3] Let $\sigma_0(t)$ be the heaviside function, i.e., $\sigma_0(t) = \begin{cases} 1, & t \ge 0 \\ 0, & t < 0. \end{cases}$ Then, $\sigma_{relu}(t) = t\sigma_0(t)$. A direct computation yields $\sigma(0) = 1/2$ and $\sigma'(0) = 1/4$. For $0 < \mu < 1/2$, according to the Taylor formula

$$\sigma(\mu t) = \frac{1}{2} + \frac{\mu t}{4} + \int_0^{\mu t} (\sigma'(u) - \sigma'(0)) du,$$

we have

$$t = \frac{4}{\mu}\sigma(\mu t) - \frac{2}{\mu} - \frac{4}{\mu}\int_0^{\mu t} (\sigma'(u) - \sigma'(0)) du.$$

Therefore,

$$\left| t - \frac{4}{\mu}\sigma(\mu t) - \frac{4}{\mu}\sigma(0 \cdot t) \right| \le \frac{4}{\mu}\int_0^{\mu t} |\sigma'(u) - \sigma'(0)| du \le \frac{4}{\mu}\max_{v \ge 0} |\sigma''(v)| \int_0^{\mu t} u du \le 2\mu t^2.$$

Denote

$$h_{2,\mu}^{linear} = \frac{4}{\mu}\sigma(\mu t) - \frac{4}{\mu}\sigma(0 \cdot t).$$

Then for $|t| \le M$, there holds

$$|t - h_{2,\mu}^{linear}(t)| \le 2M_0^2 \mu, \tag{30}$$

where $M_0 > 0$ satisfying $M_0 + M_0^2 = M$. This shows that $h_{2,\mu}^{linear}$ is a good approximation of t. On the other hand, for $\epsilon, \tau > 0$ and $A = \frac{1}{\tau}\log\frac{1}{\epsilon}$, we have

$$\sigma(At) = \frac{1}{1 + e^{-At}} \le \frac{1}{1 + e^{A\tau}} \le \epsilon, \qquad t \le -\tau$$

and

$$|\sigma(At) - 1| \le \frac{e^{-A\tau}}{1 + e^{-A\tau}} \le e^{-A\tau} \le \epsilon, \qquad t \ge \tau,$$

showing

$$|\sigma(At) - \sigma_0(t)| \le \epsilon, \qquad t \in [-M_0, -\tau] \cup [\tau, M_0]. \tag{31}$$

Since $|\sigma(At)| \le 1$ and $|h_{2,\mu}^{linear}(t)| \le |t| + 2M_0^2\mu \le M_0 + M_0^2$ for $|t| \le M_0$, we then utilize the "product-gate" exhibited in Lemma 6 with $M = M_0 + M_0^2$ to construct a deep sigmoid net with two hidden layers and at most 27 free parameters to approximate $\sigma_{relu}(t)$. Define

$$h_{9,2,\mu,\nu,A}^{relu}(t) = h_{9,\nu}^{prod}\left(\sigma(At), h_{2,\mu}^{linear}(t)\right)$$

for $t \in [-M_0, M_0]$. We then have from Lemma 6, (30) and (31) that for any $t \in [-M_0, -\tau] \cup [\tau, M_0]$

$$\begin{aligned} & |t\sigma_0(t) - h^{relu}_{9,2,\mu,\nu,A}(t)| \\ \leq\ & |t\sigma_0(t) - t\sigma(At)| + |t\sigma(At) - h^{linear}_{2,\mu}(t)\sigma(At)| + |h^{linear}_{2,\mu}(t)\sigma(At) - h^{relu}_{9,2,\mu,\nu,A}(t)| \\ \leq\ & M_0\epsilon + 2M_0^2\mu + \nu \end{aligned}$$

and

$$|h^{relu}_{9,2,\mu,\nu,A}(t)| \leq C\nu^{-6}, \qquad \forall t \in [-M, M].$$

Let $\epsilon = \mu = \nu = \varepsilon$. We have for any $0 < \varepsilon < 1/2$,

$$|\sigma_{relu}(t) - h^{relu}_{9,2,\mu,\nu,A}(t)| \leq (M_0 + 1 + 2M_0^2)\varepsilon, \qquad t \in [-M_0, -\tau] \cup [\tau, M_0] \tag{32}$$

and the free parameters of $h^{relu}_{9,2,\mu,\nu,A}(t)$ are bounded by $\max\{\mathcal{O}(\frac{1}{\varepsilon^6}), \frac{1}{\tau}\log\frac{1}{\varepsilon}\}$. Then, setting $\tau = \varepsilon^7$, we have

$$\begin{aligned} & \int_{-M}^{M} |\sigma_{relu}(t) - h^{relu}_{9,2,\mu,\nu,A}(t)|^p dt = \left(\int_{M}^{-\tau} + \int_{-\tau}^{\tau} + \int_{\tau}^{M}\right) |\sigma_{relu}(t) - h^{relu}_{9,2,\mu,\nu,A}(t)|^p \\ \leq\ & 2M\varepsilon + 2C\tau\varepsilon^{-6} \leq 2(M+C)\varepsilon. \end{aligned}$$

This completes the proof of Theorem 3 by a simple scaling. ■

Appendix B. Generic convergence of ADMM without normalization

In this appendix, we consider more general settings than that in Section 3.4, where X and Y are not necessarily normalized with unit norms, and the numbers of neurons of hidden layers can be different, and the activation function σ can be any twice differentiable activation satisfying the following assumptions.

Assumption 1 *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a twice-differentiable bounded function with bounded first- and second-order derivatives, namely, there exist positive constants $L_0 (\geq \frac{1}{8}), L_1, L_2$ such that: $|\sigma(u)| \leq L_0$, $|\sigma'(u)| \leq L_1$ and $|\sigma''(u)| \leq L_2$ for any $u \in \mathbb{R}$. Moreover, σ is either a real analytic (Krantz and Parks, 2002, Definition 1.1.5) or semialgebraic function (Bochnak et al., 1998).*

Besides the sigmoid activation, some typical activations satisfying Assumption 1 include the sigmoid-type activations (Lin et al., 2019) such as the hyperbolic tangent activation. For the abuse use of notation, in this appendix, we still use σ as any activation satisfying Assumption 1. Before presenting our main theorem under these generic settings, we define the following constants:

$$L_3 := 2(L_1^2 + L_2 L_0 + L_2), \tag{33}$$

$$\gamma := \max_{1\leq i\leq N} \|W_i^0\|_F, \tag{34}$$

$$d_{\min} := \min_{1\leq i\leq N-1} d_i, \tag{35}$$

$$f_{\min} := \sqrt{6}\left(\sqrt{3L_1} + 2(L_0L_3)^{1/2}(nd_{\min})^{1/4}\right), \tag{36}$$

$$\alpha_3 := \left(\frac{f_{\min}}{L_1}\right)^2, \tag{37}$$

$$C_3 := \max\left\{\max_{0\leq j\leq N-2} \frac{2L_0\sqrt{nd_{j+1}}}{\gamma^j}, \frac{\|Y\|_F}{(\beta_N-3)\gamma^{N-1}}\right\}, \tag{38}$$

$$\tilde{\lambda}_i := 3L_1C_3\beta_i\gamma^{i-3}(4C_3\gamma^{i-1} + L_0\sqrt{nd_i})\left(1 + \sqrt{\frac{6L_3C_3^2\gamma^{2i-2}}{L_1(4C_3\gamma^{i-1}+L_0\sqrt{nd_i})}}\right),\ 2\leq i\leq N-1,$$

$$\bar{\lambda} := \max_{2\leq i\leq N-1}\left\{\tilde{\lambda}_i, \frac{1}{6}(1+3L_1^{-1}L_2L_3\gamma^{i-1})^2C_3^2\gamma^{2(i-2)}\beta_i\right\},$$

$$\hat{\lambda} := L_1\beta_1\|X\|_F(4C_3 + L_0\sqrt{nd_1})\gamma^{-1}\left(1 + \sqrt{\frac{2L_3C_3\|X\|_F\gamma}{L_1(4C_3+L_0\sqrt{nd_1})}}\right).$$

With these defined constants, we impose some conditions on the the penalty parameters $\{\beta_i\}_{i=1}^N$ in the augmented Lagrangian, the regularization parameter λ, the minimal number of hidden neurons $d_{\min}$, and the initializations of $\{V_i^0\}_{i=1}^N$ and $\{\Lambda_i^0\}_{i=1}^N$ as follows

$$\beta_N \geq 3.5, \tag{39}$$

$$\frac{\beta_{N-1}}{\beta_N} \geq 16\gamma^2, \tag{40}$$

$$\frac{\beta_i}{\beta_{i+1}} \geq \max\left\{6\sqrt{N}(2L_1^2 + (4L_3+L_2)C_3\gamma^i)\gamma^2, 6(\sqrt{3L_1}+\sqrt{2L_3C_3\gamma^i})^2\gamma^2\right\},\ i=1,\dots,N-2, \tag{41}$$

$$\lambda \geq \max\left\{12\beta_NC_3^2\gamma^{2N-4}, \bar{\lambda}, \hat{\lambda}\right\}, \tag{42}$$

$$d_{\min} \geq \frac{\left(\max\left\{\sqrt{24N+1}L_1 - \sqrt{18L_1}, 0\right\}\right)^4}{n(24L_0L_3)^2}, \tag{43}$$

$$\|V_i^0\|_F \leq 3C_3\gamma^{i-1}, \quad \|\Lambda_i^0\|_F \leq C_3\beta_i\gamma^{i-1}, \quad i=1,\dots,N. \tag{44}$$

Under these assumptions, we state the main convergence theorem of ADMM as follows.

Theorem 7 *Let Assumption 1 hold. Let $\{\mathcal{Q}^k := (\{W_i^k\}_{i=1}^N, \{V_i^k\}_{i=1}^N, \{\Lambda_i^k\}_{i=1}^N)\}$ be a sequence generated by Algorithm 1 with $h_i^k = \mathbb{L}(\|V_i^{k-1} - \beta_i^{-1}\Lambda_i^{k-1}\|_{\max})$ for $i=1,\dots,N-1$. and $\mu_j^k = \mathbb{L}(\|V_{j+1}^{k-1} - \beta_{j+1}^{-1}\Lambda_{j+1}^{k-1}\|_{\max})$ for $j=1,\dots,N-2$, where $\mathbb{L}(\cdot)$ is defined in* (18). *Assume that* (39)-(44) *hold, then the following hold:*

(a) $\{\mathcal{L}(\mathcal{Q}^k)\}$ *is convergent.*

(b) $\{\mathcal{Q}^k\}$ *converges to a stationary point* $\mathcal{Q}^* := (\{W_i^*\}_{i=1}^N, \{V_i^*\}_{i=1}^N, \{\Lambda_i^*\}_{i=1}^N)$ *of* $\mathcal{L}$*, which is also a KKT point* (24) *of problem* (5)*, implying* $\{W_i^*\}_{i=1}^N$ *is a stationary point of problem* (4) *with* $\lambda' = 2\lambda/n$*.*

(c) $\frac{1}{K}\sum_{k=1}^K \|\nabla\mathcal{L}(\mathcal{Q}^k)\|_F^2 \to 0$ *at a* $\mathcal{O}(\frac{1}{K})$ *rate.*

Theorem 4 presented in the context is a special case of Theorem 7 with $\gamma = 1$, $\|X\|_F = \|Y\|_F = 1$, $\|W_i^0\|_F = 1$, $i = 1, \ldots, N$, and the initialization strategy (7). Actually, the initialization strategy (7) satisfies (44) shown as follows:

$$\|V_j^0\|_F \leq L_0\sqrt{nd_j} \leq \frac{1}{2}C_3\gamma^{j-1}, \ j = 1, \ldots, N-1, \tag{45}$$

$$\|V_N^0\|_F \leq \gamma \cdot \frac{1}{2}C_3\gamma^{N-2} = \frac{1}{2}C_3\gamma^{N-1}, \tag{46}$$

$$\|\Lambda_i^0\|_F = 0, \ i = 1, \ldots, N,$$

where the first inequality in (45) holds by the boundedness of activation, and the second inequality in (45) holds by the definition (38) of C_3, and the inequality in (46) holds for $\|W_N^0\|_F \leq \gamma$ and (45) with $j = N-1$. By the definitions (28) and (29) of $\hat{\mathcal{Q}}^k$ and $\hat{\mathcal{L}}$, if we can show that Theorem 5 holds under the assumptions of Theorem 7, then we directly yield Theorem 7. Thus, we only need to prove Theorem 5 under the assumptions of Theorem 7.

Appendix C. Preliminaries

Before presenting the proof of Theorem 5 under the assumptions of Theorem 7, we provide some preliminary definitions and lemmas which serve as the basis of our proof.

C.1 Dual expressed by primal

According to the specific updates of Algorithm 1, we show that the updates of dual variables $\{\Lambda_i^k\}_{i=1}^N$ can be expressed explicitly by the updates of primal variables $\{W_i^k\}_{i=1}^N$ and $\{V_i^k\}_{i=1}^N$ as in the following lemma.

Lemma 8 (Dual expressed by primal) *Suppose that Assumption 1 holds. Let* $\{\mathcal{Q}^k := (\{W_i^k\}_{i=1}^N, \{V_i^k\}_{i=1}^N, \{\Lambda_i^k\}_{i=1}^N)\}$ *be a sequence generated by Algorithm 1. Then we have*

$$\Lambda_N^k = V_N^k - Y, \quad \forall k \in \mathbb{N}, \tag{47}$$

$$\Lambda_{N-1}^k = (W_N^k)^T\Lambda_N^k + \beta_N(W_N^k)^T(V_N^k - V_N^{k-1}), \tag{48}$$

$$\begin{aligned}\Lambda_j^k = {} & (W_{j+1}^k)^T\left(\Lambda_{j+1}^k \odot \sigma'(W_{j+1}^k V_j^{k-1})\right) + \beta_{j+1}(W_{j+1}^k)^T\Big[\Big((\sigma(W_{j+1}^k V_j^{k-1}) - \sigma(W_{j+1}^k V_j^k)) \\ & + (V_{j+1}^k - V_{j+1}^{k-1})\Big) \odot \sigma'(W_{j+1}^k V_j^{k-1}) + \mu_j^k W_{j+1}^k(V_j^k - V_j^{k-1})/2\Big], \ j = N-2, \ldots, 1.\end{aligned} \tag{49}$$

Proof We firstly derive the explicit updates of $\{W_i^k\}_{i=1}^N$ and $\{V_j^k\}_{j=1}^N$, then based on these updates, we prove Lemma 8.

1) W_i-subproblems: According to the update (8), W_N^k is updated via

$$W_N^k = (\beta_N V_N^{k-1} - \Lambda_N^{k-1})(V_{N-1}^{k-1})^T \left(\lambda \mathbf{I} + \beta_N V_{N-1}^{k-1} V_{N-1}^{k-1\,T}\right)^{-1}. \tag{50}$$

By (19), for $i = 1, \ldots, N-1$, we get

$$\begin{aligned} W_i^k &= W_i^{k-1} \frac{\beta_i h_i^k}{2} V_{i-1}^{k-1} (V_{i-1}^{k-1})^T \left(\lambda \mathbf{I} + \frac{\beta_i h_i^k}{2} V_{i-1}^{k-1} (V_{i-1}^{k-1})^T\right)^{-1} \\ &- \left[\left(\Lambda_i^{k-1} + \beta_i(\sigma(W_i^{k-1} V_{i-1}^{k-1}) - V_i^{k-1})\right) \odot \sigma'(W_i^{k-1} V_{i-1}^{k-1})\right] (V_{i-1}^{k-1})^T \left(\lambda \mathbf{I} + \frac{\beta_i h_i^k}{2} V_{i-1}^{k-1} (V_{i-1}^{k-1})^T\right)^{-1} \\ &= W_i^{k-1} - W_i^{k-1} \left(\mathbf{I} + \frac{\beta_i h_i^k}{2\lambda} V_{i-1}^{k-1} (V_{i-1}^{k-1})^T\right)^{-1} \\ &- \left[\left(\Lambda_i^{k-1} + \beta_i(\sigma(W_i^{k-1} V_{i-1}^{k-1}) - V_i^{k-1})\right) \odot \sigma'(W_i^{k-1} V_{i-1}^{k-1})\right] (V_{i-1}^{k-1})^T \left(\lambda \mathbf{I} + \frac{\beta_i h_i^k}{2} V_{i-1}^{k-1} V_{i-1}^{k-1\,T}\right)^{-1}. \end{aligned} \tag{51}$$

Particularly, when $i = 1$, W_1^k is updated by

$$\begin{aligned} W_1^k &= W_1^{k-1} \frac{\beta_1 h_1^k}{2} V_0 V_0^T \left(\lambda \mathbf{I} + \frac{\beta_1 h_1^k}{2} V_0 V_0^T\right)^{-1} \\ &- \left[\left(\Lambda_1^{k-1} + \beta_1(\sigma(W_1^{k-1} V_0) - V_1^{k-1})\right) \odot \sigma'(W_1^{k-1} V_0)\right] V_0^T \left(\lambda \mathbf{I} + \frac{\beta_1 h_1^k}{2} V_0 V_0^T\right)^{-1} \\ &= W_1^{k-1} - W_1^{k-1} \left(\mathbf{I} + \frac{\beta_1 h_1^k}{2\lambda} V_0 V_0^T\right)^{-1} \\ &- \left[\left(\Lambda_1^{k-1} + \beta_1(\sigma(W_1^{k-1} V_0) - V_1^{k-1})\right) \odot \sigma'(W_1^{k-1} V_0)\right] V_0^T \left(\lambda \mathbf{I} + \frac{\beta_1 h_1^k}{2} V_0 V_0^T\right)^{-1}. \end{aligned} \tag{52}$$

2) V_j-subproblems: According to (12), it holds

$$V_N^k - Y - \left[\Lambda_N^{k-1} + \beta_N \left(W_N^k V_{N-1}^k - V_N^k\right)\right] = 0. \tag{53}$$

By the relation $\Lambda_N^k = \Lambda_N^{k-1} + \beta_N \left(W_N^k V_{N-1}^k - V_N^k\right)$, (53) implies

$$\Lambda_N^k = V_N^k - Y, \quad \forall k \in \mathbb{N}, \tag{54}$$

which shows (47) in Lemma 8. Substituting the equality (54) with the index value $k-1$ into (53) yields

$$V_N^k = \frac{1}{1+\beta_N} V_N^{k-1} + \frac{\beta_N}{1+\beta_N} W_N^k V_{N-1}^k. \tag{55}$$

According to (11), it holds

$$-\left[\Lambda_{N-1}^{k-1} + \beta_{N-1}(\sigma(W_{N-1}^k V_{N-2}^k) - V_{N-1}^k)\right] + (W_N^k)^T \left[\Lambda_N^{k-1} + \beta_N \left(W_N^k V_{N-1}^k - V_N^{k-1}\right)\right] = 0,$$

which implies

$$V_{N-1}^k = \tag{56}$$
$$(\beta_{N-1}\mathbf{I} + \beta_N (W_N^k)^T W_N^k)^{-1} \left[\Lambda_{N-1}^{k-1} + \beta_{N-1}\sigma(W_{N-1}^k V_{N-2}^k) - (W_N^k)^T \left(\Lambda_N^{k-1} - \beta_N V_N^{k-1}\right)\right],$$

and together with the updates of Λ_{N-1}^k and Λ_N^k in Algorithm 1 yields

$$\Lambda_{N-1}^k = (W_N^k)^T \left[\Lambda_N^{k-1} + \beta_N \left(W_N^k V_{N-1}^k - V_N^{k-1}\right)\right] = (W_N^k)^T \left(\Lambda_N^k + \beta_N (V_N^k - V_N^{k-1})\right).$$

This implies (48) in Lemma 8.

By (20), for $j = 1, \ldots, N-2$, V_j^k satisfies the following optimality condition

$$\begin{aligned} &-\left[\Lambda_j^{k-1} + \beta_j(\sigma(W_j^k V_{j-1}^k) - V_j^k)\right] + \frac{\beta_{j+1}\mu_j^k}{2} {W_{j+1}^k}^T W_{j+1}^k (V_j^k - V_j^{k-1}) \\ &+ {W_{j+1}^k}^T \left[\left(\Lambda_{j+1}^{k-1} + \beta_{j+1}(\sigma(W_{j+1}^k V_j^{k-1}) - V_{j+1}^{k-1})\right) \odot \sigma'(W_{j+1}^k V_j^{k-1})\right] = 0, \end{aligned}$$

which implies

$$\begin{aligned} V_j^k &= \left(\beta_j \mathbf{I} + \frac{\beta_{j+1}\mu_j^k}{2} {W_{j+1}^k}^T W_{j+1}^k\right)^{-1} \left[\frac{1}{2}\beta_{j+1}\mu_j^k {W_{j+1}^k}^T W_{j+1}^k V_j^{k-1} + \left(\Lambda_j^{k-1} + \beta_j \sigma(W_j^k V_{j-1}^k)\right)\right. \\ &\left. + {W_{j+1}^k}^T \left([\Lambda_{j+1}^{k-1} + \beta_{j+1}(\sigma(W_{j+1}^k V_j^{k-1}) - V_{j+1}^{k-1})] \odot \sigma'(W_{j+1}^k V_j^{k-1})\right)\right] \\ &= V_j^{k-1} - \left(\mathbf{I} + \frac{\beta_{j+1}\mu_j^k}{2\beta_j} {W_{j+1}^k}^T W_{j+1}^k\right)^{-1} V_j^{k-1} \\ &+ \left(\beta_j \mathbf{I} + \frac{\beta_{j+1}\mu_j^k}{2} {W_{j+1}^k}^T W_{j+1}^k\right)^{-1} \left[\left(\Lambda_j^{k-1} + \beta_j \sigma(W_j^k V_{j-1}^k)\right)\right. \\ &\left. + {W_{j+1}^k}^T \left(\left[\Lambda_{j+1}^{k-1} + \beta_{j+1}(\sigma(W_{j+1}^k V_j^{k-1}) - V_{j+1}^{k-1})\right] \odot \sigma'(W_{j+1}^k V_j^{k-1})\right)\right]. \end{aligned} \tag{57}$$

and together with the updates of Λ_j^k and Λ_{j+1}^k in Algorithm 1 yields

$$\begin{aligned} \Lambda_j^k &= (W_{j+1}^k)^T \left[\left(\Lambda_{j+1}^{k-1} + \beta_{j+1}\left(\sigma(W_{j+1}^k V_j^{k-1}) - V_{j+1}^{k-1}\right)\right) \odot \sigma'(W_{j+1}^k V_j^{k-1})\right] \\ &+ \frac{\beta_{j+1}\mu_j^k}{2} (W_{j+1}^k)^T W_{j+1}^k (V_j^k - V_j^{k-1}), \\ &= (W_{j+1}^k)^T \left(\Lambda_{j+1}^k \odot \sigma'(W_{j+1}^k V_j^{k-1})\right) + \beta_{j+1}(W_{j+1}^k)^T \left[\left((\sigma(W_{j+1}^k V_j^{k-1}) - \sigma(W_{j+1}^k V_j^k))\right.\right. \\ &\left.\left. + (V_{j+1}^k - V_{j+1}^{k-1})\right) \odot \sigma'(W_{j+1}^k V_j^{k-1}) + \mu_j^k W_{j+1}^k (V_j^k - V_j^{k-1})/2\right]. \end{aligned}$$

The final equality implies (49) in Lemma 8. This completes the proof of this lemma. ■

C.2 Kurdyka-Łojasiewicz property

The Kurdyka-Łojasiewicz (KL) property (Łojasiewicz, 1993; Kurdyka, 1998) plays a crucial role in the convergence analysis of nonconvex algorithm (see, Attouch et al. (2013)). The following definition is adopted from (Bolte et al., 2007).

Definition 9 (KL property) *An extended real valued function $h : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ is said to have the Kurdyka-Łojasiewicz property at $x^* \in \mathrm{dom}(\partial h)$ if there exist a neighborhood U of x^*, a constant $\eta > 0$, and a continuous concave function $\phi(s) = cs^{1-\theta}$ for some $c > 0$ and $\theta \in [0, 1)$ such that the Kurdyka-Lojasiewicz inequality holds*

$$\phi'(h(x) - h(x^*))\mathrm{dist}(0, \partial h(x)) \geq 1, \ \forall x \in U \cap \mathrm{dom}(\partial h) \ \textit{and} \ h(x^*) < h(x) < h(x^*) + \eta, \quad (58)$$

where $\partial h(x)$ denotes the limiting-subdifferential of h at $x \in \mathrm{dom}(h)$ (introduced in Mordukhovich (2006)), $\mathrm{dom}(h) := \{x \in \mathcal{X} : h(x) < +\infty\}$, $\mathrm{dom}(\partial h) := \{x \in \mathcal{X} : \partial h(x) \neq \emptyset\}$, and $\mathrm{dist}(0, \partial h(x)) := \min\{\|z\| : z \in \partial h(x)\}$, where $\|\cdot\|$ represents the Euclidean norm.

Proper lower semi-continuous functions which satisfy the Kurdyka-Lojasiewicz inequality at each point of $\mathrm{dom}(\partial h)$ are called KL functions.

Note that we have adopted in the definition of KL inequality (58) the following notational conventions: $0^0 = 1, \infty/\infty = 0/0 = 0$. This property was firstly introduced by (Łojasiewicz, 1993) on real analytic functions (Krantz and Parks, 2002) for $\theta \in [\frac{1}{2}, 1)$, then was extended to functions defined on the *o*-minimal structure in (Kurdyka, 1998), and later was extended to nonsmooth subanalytic functions in (Bolte et al., 2007). In the following, we give the definitions of real-analytic and semialgebraic functions.

Definition 10 (Real analytic, Definition 1.1.5 in (Krantz and Parks, 2002)) *A function h with domain being an open set $U \subset \mathbb{R}$ and range either the real or the complex numbers, is said to be* ***real analytic*** *at u if the function f may be represented by a convergent power series on some interval of positive radius centered at u:*

$$h(x) = \sum_{j=0}^{\infty} \alpha_j (x - u)^j,$$

for some $\{\alpha_j\} \subset \mathbb{R}$. The function is said to be ***real analytic*** *on $V \subset U$ if it is real analytic at each $u \in V$. The real analytic function f over $\mathbb{R}^p$ for some positive integer $p > 1$ can be defined similarly.*

According to (Krantz and Parks, 2002), some typical real analytic functions include polynomials, exponential functions, and the logarithm, trigonometric and power functions on any open set of their domains. One can verify whether a multivariable real function $h(\mathbf{x})$ on $\mathbb{R}^p$ is analytic by checking the analyticity of $g(t) := h(\mathbf{x} + t\mathbf{y})$ for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$. The following lemma shows some important properties of real analytic functions.

Lemma 11 (Krantz and Parks, 2002) *The sums, products, and compositions of real analytic functions are real analytic functions.*

Let $h : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ be an extended-real-valued function (respectively, $h : \mathbb{R}^p \rightrightarrows \mathbb{R}^q$ be a point-to-set mapping), its *graph* is defined by

$$\begin{aligned} \mathrm{Graph}(h) &:= \{(x, y) \in \mathbb{R}^p \times \mathbb{R} : y = h(x)\}, \\ (\text{resp. } \mathrm{Graph}(h) &:= \{(x, y) \in \mathbb{R}^p \times \mathbb{R}^q : y \in h(x)\}), \end{aligned}$$

and its domain by $\mathrm{dom}(h) := \{x \in \mathbb{R}^p : h(x) < +\infty\}$ (resp. $\mathrm{dom}(h) := \{x \in \mathbb{R}^p : h(x) \neq \emptyset\}$).

Definition 12 (Semialgebraic)

(a) A set $\mathcal{D} \subset \mathbb{R}^p$ is called semialgebraic (Bochnak et al., 1998) if it can be represented as

$$\mathcal{D} = \bigcup_{i=1}^{s} \bigcap_{j=1}^{t} \{x \in \mathbb{R}^p : P_{ij}(x) = 0, Q_{ij}(x) > 0\},$$

where P_{ij}, Q_{ij} are real polynomial functions for $1 \le i \le s, 1 \le j \le t$.

(b) A function $h : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ (resp. a point-to-set mapping $h : \mathbb{R}^p \rightrightarrows \mathbb{R}^q$) is called semialgebraic if its graph Graph(h) *is semialgebraic.*

According to (Łojasiewicz, 1965; Bochnak et al., 1998) and (Shiota, 1997, I.2.9, p.52), the class of semialgebraic sets is stable under the operation of finite union, finite intersection, Cartesian product or complementation. Some typical examples include polynomial functions, the indicator function of a semialgebraic set, and the Euclidean norm (Bochnak et al., 1998, p.26).

Lemma 13 (KL properties of $\mathcal{L}$ and $\hat{\mathcal{L}}$) *Suppose that Assumption 1 holds, then both $\mathcal{L}$ and $\hat{\mathcal{L}}$ are KL functions.*

Proof Let $\mathcal{Q} := \left(\{W_i\}_{i=1}^N, \{V_i\}_{i=1}^N, \{\Lambda_i\}_{i=1}^N\right)$, $\hat{\mathcal{Q}} := \left(\mathcal{Q}, \{\hat{V}_i\}_{i=1}^N\right)$ and

$$\begin{aligned} \mathcal{L}_1(\mathcal{Q}) &:= \frac{1}{2}\|V_N - Y\|_F^2 + \frac{\lambda}{2}\sum_{i=1}^N \|W_i\|_F^2 + \sum_{i=1}^N \frac{\beta_i}{2}\|\sigma_i(W_i V_{i-1}) - V_i\|_F^2, \\ \mathcal{L}_2(\mathcal{Q}) &:= \sum_{i=1}^N \langle \Lambda_i, \sigma_i(W_i V_{i-1}) - V_i \rangle. \end{aligned}$$

Then $\mathcal{L}(\mathcal{Q}) = \mathcal{L}_1(\mathcal{Q}) + \mathcal{L}_2(\mathcal{Q}), \hat{\mathcal{L}}(\hat{\mathcal{Q}}) = \mathcal{L}(\mathcal{Q}) + \sum_{i=1}^N \xi_i \|V_i - \hat{V}_i\|_F^2$, where $\xi_i > 0$, $i = 1, \ldots, N$. According to the same arguments as in the proof of (Zeng et al., 2019, Proposition 2), $\mathcal{L}_1$ is real analytic (resp. semialgebraic) if σ_i is real analytic (resp. semialgebraic). By the closedness of real analytic (resp. semialgebraic) functions under the sum, product and composition (see, Krantz and Parks (2002); Bochnak et al. (1998)), we can show that $\mathcal{L}_2$ is also real analytic (resp. semialgebraic) if σ_i is real analytic (resp. semialgebraic). Thus, $\mathcal{L}$ is a finite sum of real analytic or semialgebraic functions. According to Shiota (1997), $\mathcal{L}$ is a subanalytic function. By Assumption 1, $\mathcal{L}$ is continuous. Thus, $\mathcal{L}$ is a KL function by (Bolte et al., 2007, Theorem 3.1). Since $\sum_{i=1}^N \xi_i \|V_i - \hat{V}_i\|_F^2$ is polynomial, $\hat{\mathcal{L}}$ is also a KL function by a similar argument. This completes the proof. ■

Appendix D. Proofs for Theorem 5

As stated in Section 4.4, the main idea of proof of Theorem 5 is shown as follows: we firstly establish the desired sufficient descent lemma (see, Lemma 14) via estimating the progress made by one step update, and bounding dual by primal as well as showing the boundedness of the sequence, then develop the desired relative error lemma (see, Lemma 21) via the optimality conditions of all subproblems, the Lipschitz continuity of the activation as well as the boundedness of the sequence, and finally prove this theorem via (Attouch et al., 2013, Theorem 2.9), together with Lemma 13 and the continuous assumption of the activation. In the following, we establish these lemmas followed by the detailed proof of Theorem 5.

D.1 Proof for Lemma 14: Sufficient descent lemma

In order to prove Theorem 5, the following *sufficient descent* lemma plays a key role.

Lemma 14 (Sufficient descent) *Under assumptions of Theorem 7, for $k \geq 2$, there holds*

$$\hat{\mathcal{L}}(\hat{\mathcal{Q}}^k) \leq \hat{\mathcal{L}}(\hat{\mathcal{Q}}^{k-1}) - a\left(\sum_{i=1}^N \|W_i^k - W_i^{k-1}\|_F^2 + \sum_{i=1}^N \|V_i^k - V_i^{k-1}\|_F^2\right), \tag{59}$$

where a is some positive constant specified later in (98) *in Appendix D.1.4.*

From Lemma 14, we establish the sufficient descent property of an auxiliary sequence $\{\hat{\mathcal{Q}}^k\}$ instead of the sequence $\{\mathcal{Q}^k\}$ itself, along a new Laypunov function $\hat{\mathcal{L}}$ but not the original augmented Lagrangian $\mathcal{L}$. This is different from the convergence analysis of ADMM in (Wang et al., 2019) for linear constrained optimization problems, where the sufficient descent lemma is shown for the original sequence along the augmented Lagrangian (see, (Wang et al., 2019, Lemma 5)). In order to establish Lemma 14, the following three lemmas are required, where the first lemma shows the progress made by one step update (called, *one-step progress lemma*), the second lemma bounds the discrepancies of two successive dual updates via those of the primal updates (called, *dual-bounded-by-primal lemma*), and the third lemma shows the boundedness of the sequence (called, *boundedness lemma*).

D.1.1 Lemma 15: One-step progress lemma

We present the first lemma that estimates the progress made by a single update of ADMM.

Lemma 15 (One-step progress) *Let Assumption 1 hold. Let $\big\{\mathcal{Q}^k := \big(\{W_i^k\}_{i=1}^N, \{V_i^k\}_{i=1}^N, \{\Lambda_i^k\}_{i=1}^N\big)\big\}$ be a sequence generated by Algorithm 1 with $\{h_i^k\}_{i=1}^{N-1}$ and $\{\mu_j^k\}_{j=1}^{N-2}$ specified in* (21) *and* (22)*, respectively. Then for any integer $k \geq 1$, the following holds*

$$\begin{aligned}\mathcal{L}(\mathcal{Q}^k) \leq \mathcal{L}(\mathcal{Q}^{k-1}) &- \sum_{i=1}^N \left(\frac{\lambda}{2}\|W_i^k - W_i^{k-1}\|_F^2 + \frac{\beta_i h_i^k}{4}\|(W_i^k - W_i^{k-1})V_{i-1}^{k-1}\|_F^2\right) \\ &- \sum_{j=1}^{N-1}\left(\frac{\beta_j}{2}\|V_j^k - V_j^{k-1}\|_F^2 + \frac{\beta_{j+1}\mu_j^k}{4}\|W_{j+1}^k(V_j^k - V_j^{k-1})\|_F^2\right) - \frac{1+\beta_N}{2}\|V_N^k - V_N^{k-1}\|_F^2 \\ &+ \sum_{i=1}^N \beta_i^{-1}\|\Lambda_i^k - \Lambda_i^{k-1}\|_F^2,\end{aligned} \tag{60}$$

where $V_0^k \equiv X$, $h_N^k = 1$ *and* $\mu_{N-1}^k = 1$.

From Lemma 15, there are two key parts that contribute to the progress along the augmented Lagrangian sequence, namely, the descent part arisen by the primal updates and the ascent part brought by the dual updates. Due to the existence of the dual ascent part, the convergence of nonconvex ADMM is usually very challengeable. By (60), in order to further estimate the progress in terms of the primal updates, we shall bound these dual ascent parts via the primal updates as shown in Lemma 18 below.

To prove Lemma 15, we firstly establish two preliminary lemmas.

Lemma 16 *Given a constant* $c \in \mathbb{R}$, *let* f_c *be the function on* $\mathbb{R}$ *given by* $f_c(u) = (\sigma(u)-c)^2$. *Then the following holds*

$$f_c(v) \le f_c(u) + f_c'(u)(v-u) + \frac{\mathbb{L}(|c|)}{2}(v-u)^2, \forall u, v \in \mathbb{R}$$

where $\mathbb{L}(|c|)$ *is defined in* (18).

Proof According to Assumption 1, by some simple derivations, we can show $|f_c''(u)| \le \mathbb{L}(|c|), \forall u \in \mathbb{R}$. This yields the inequality $f_c(v) \le f_c(u) + f_c'(u)(v-u) + \frac{\mathbb{L}(|c|)}{2}(v-u)^2, \forall u, v \in \mathbb{R}$. ■

Note that the W_i^k $(i = 1, \ldots, N-1)$ and V_j^k $(j = 1, \ldots, N-2)$ updates involve the following update schemes, i.e.,

$$W^k = \arg\min_W \left\{ \frac{\lambda}{2}\|W\|_F^2 + \beta H_\sigma^k(W; A, B) \right\}, \tag{61}$$

$$V^k = \arg\min_V \left\{ \frac{\lambda}{2}\|V - C\|_F^2 + \beta M_\sigma^k(V; A, B) \right\} \tag{62}$$

for some matrices A, B and C, positive constants λ and β. Based on Lemma 16, we provide a lemma to estimate the descent quantities of the above two updates.

Lemma 17 *Suppose that Assumption 1 holds. Let* W^k *and* V^k *be updated according to* (61) *and* (62), *respectively, then*

$$\begin{aligned} \frac{\lambda}{2}\|W^k\|_F^2 + \beta H_\sigma(W^k; A, B) \le{}& \frac{\lambda}{2}\|W^{k-1}\|_F^2 + \beta H_\sigma(W^{k-1}; A, B) \\ &- \frac{\lambda}{2}\|W^k - W^{k-1}\|_F^2 - \frac{\beta h^k}{4}\|(W^k - W^{k-1})A\|_F^2, \end{aligned} \tag{63}$$

$$\begin{aligned} \frac{\lambda}{2}\|V^k - C\|_F^2 + \beta M_\sigma(V^k; A, B) \le{}& \frac{\lambda}{2}\|V^{k-1} - C\|_F^2 + \beta M_\sigma(V^{k-1}; A, B) \\ &- \frac{\lambda}{2}\|V^k - V^{k-1}\|_F^2 - \frac{\beta \mu^k}{4}\|A(V^k - V^{k-1})\|_F^2, \end{aligned} \tag{64}$$

where $h^k := \mathbb{L}(\|B\|_{\max})$ *and* $\mu^k := \mathbb{L}(\|B\|_{\max})$.

Proof We first establish the descent inequality (63) then similarly show (64).

Let $h(W) := \frac{\lambda}{2}\|W\|_F^2 + \beta H_\sigma^k(W; A, B)$. By Taylor's formula, the optimality of W^k, and noting that $h(W)$ is a quadratic function, there holds

$$h(W^{k-1}) = h(W^k) + \frac{\lambda}{2}\|W^k - W^{k-1}\|_F^2 + \frac{\beta h^k}{4}\|(W^k - W^{k-1})A\|_F^2,$$

which implies

$$\begin{aligned}
&\frac{\lambda}{2}\|W^{k-1}\|_F^2 + \beta H_\sigma(W^{k-1}; A, B) \\
&= \frac{\lambda}{2}\|W^k\|_F^2 + \beta\left(H_\sigma(W^{k-1}; A, B) + \langle \nabla H_\sigma(W^{k-1}; A, B), W^k - W^{k-1}\rangle + \frac{h^k}{4}\|(W^k - W^{k-1})A\|_F^2\right) \\
&+ \frac{\lambda}{2}\|W^k - W^{k-1}\|_F^2 + \frac{\beta h^k}{4}\|(W^k - W^{k-1})A\|_F^2 \\
&\geq \frac{\lambda}{2}\|W^k\|_F^2 + \beta H_\sigma(W^k; A, B) + \frac{\lambda}{2}\|W^k - W^{k-1}\|_F^2 + \frac{\beta h^k}{4}\|(W^k - W^{k-1})A\|_F^2,
\end{aligned}$$

where the final inequality holds by the definition (14) of $H_\sigma(W; A, B) = \frac{1}{2}\|\sigma(WA) - B\|_F^2$ and by specializing Lemma 16 with $v = [W^k A]_{ij}$, $u = [W^{k-1}A]_{ij}$ and $c = B_{ij}$ for any i, j, where $[W^k A]_{ij}$ and $[W^{k-1}A]_{ij}$ are the (i, j)-th entries of $W^k A$ and $W^{k-1}A$, respectively. This yields (63).

Similarly, we can establish the inequality (64). This completes the proof. ∎

Based on Lemma 17, we prove Lemma 15 as follows.

Proof [Proof of Lemma 15] We establish (60) via estimating the progress for each block update. At first, we consider the W_N^k update. By (8), it is easy to show

$$\begin{aligned}
&\mathcal{L}(W_{<N}^{k-1}, W_N^k, \{V_j^{k-1}\}_{j=1}^N, \{\Lambda_j^{k-1}\}_{j=1}^N) \leq \mathcal{L}(W_{<N}^{k-1}, W_N^{k-1}, \{V_j^{k-1}\}_{j=1}^N, \{\Lambda_j^{k-1}\}_{j=1}^N) \\
&- \frac{\lambda}{2}\|W_N^k - W_N^{k-1}\|_F^2 - \frac{\beta_N}{2}\|(W_N^k - W_N^{k-1})V_{N-1}^{k-1}\|_F^2. \qquad (65)
\end{aligned}$$

By (19), W_i^k $(i = 1, \ldots, N-1)$ is updated according to (61) with $\lambda = \lambda$, $\beta = \beta_i$, $A = V_{i-1}^{k-1}$ and $B = V_i^{k-1} - \beta_i^{-1}\Lambda_i^{k-1}$. Then by Lemma 17, it holds

$$\begin{aligned}
&\mathcal{L}(W_{<i}^{k-1}, W_i^k, W_{>i}^k, \{V_j^{k-1}\}_{j=1}^N, \{\Lambda_j^{k-1}\}_{j=1}^N) \leq \mathcal{L}(W_{<i}^{k-1}, W_i^{k-1}, W_{>i}^k, \{V_j^{k-1}\}_{j=1}^N, \{\Lambda_j^{k-1}\}_{j=1}^N) \\
&- \frac{\lambda}{2}\|W_i^k - W_i^{k-1}\|_F^2 - \frac{\beta_i h_i^k}{4}\|(W_i^k - W_i^{k-1})V_{i-1}^{k-1}\|_F^2. \qquad (66)
\end{aligned}$$

Similarly, for the V_j^k-update $(j = 1, \ldots, N-2)$, by (20) and Lemma 17, the following holds

$$\begin{aligned}
&\mathcal{L}(\{W_i^k\}_{i=1}^N, V_{<j}^k, V_j^k, V_{>j}^{k-1}, \{\Lambda_i^{k-1}\}_{i=1}^N) \leq \mathcal{L}(\{W_i^k\}_{i=1}^N, V_{<j}^k, V_j^{k-1}, V_{>j}^{k-1}, \{\Lambda_i^{k-1}\}_{i=1}^N) \\
&- \frac{\beta_j}{2}\|V_j^k - V_j^{k-1}\|_F^2 - \frac{\beta_{j+1}\mu_j^k}{4}\|W_{j+1}^k(V_j^k - V_j^{k-1})\|_F^2. \qquad (67)
\end{aligned}$$

For the V_{N-1}^k and V_N^k updates, by (11) and (12), we can easily obtain the following

$$\begin{aligned}&\mathcal{L}(\{W_i^k\}_{i=1}^N, V_{<N-1}^k, V_{N-1}^k, V_N^{k-1}, \{\Lambda_i^{k-1}\}_{i=1}^N) \leq \mathcal{L}(\{W_i^k\}_{i=1}^N, V_{<N-1}^k, V_{N-1}^{k-1}, V_N^{k-1}, \{\Lambda_i^{k-1}\}_{i=1}^N)\\ &- \frac{\beta_{N-1}}{2}\|V_{N-1}^k - V_{N-1}^{k-1}\|_F^2 - \frac{\beta_N}{2}\|W_N^k(V_{N-1}^k - V_{N-1}^{k-1})\|_F^2 \qquad (68)\end{aligned}$$

and

$$\begin{aligned}&\mathcal{L}(\{W_i^k\}_{i=1}^N, V_{<N}^k, V_N^k, \{\Lambda_i^{k-1}\}_{i=1}^N) \leq \mathcal{L}(\{W_i^k\}_{i=1}^N, V_{<N}^k, V_N^{k-1}, \{\Lambda_i^{k-1}\}_{i=1}^N)\\ &- \frac{1+\beta_N}{2}\|V_N^k - V_N^{k-1}\|_F^2. \qquad (69)\end{aligned}$$

Particularly, by the updates of Λ_j^k $(j = 1, \ldots, N)$, we have

$$\begin{aligned}&\mathcal{L}(\{W_i^k\}_{i=1}^N, \{V_j^k\}_{j=1}^N, \{\Lambda_i^k\}_{i=1}^N)\\ &= \mathcal{L}(\{W_i^k\}_{i=1}^N, \{V_j^k\}_{j=1}^N, \{\Lambda_i^{k-1}\}_{i=1}^N) + \sum_{i=1}^N \langle \Lambda_i^k - \Lambda_i^{k-1}, \sigma_i(W_i^k V_{i-1}^k) - V_i^k\rangle\\ &= \mathcal{L}(\{W_i^k\}_{i=1}^N, \{V_j^k\}_{j=1}^N, \{\Lambda_i^{k-1}\}_{i=1}^N) + \sum_{i=1}^N \beta_i^{-1}\|\Lambda_i^k - \Lambda_i^{k-1}\|_F^2. \qquad (70)\end{aligned}$$

Summing up (65)-(70) yields (60). ■

D.1.2 Lemma 18: Dual-bounded-by-primal lemma

By Lemma 15, how to control the amount of ascent part brought by the dual updates via the amount of descent part characterized by the primal updates is very important. The following lemma shows that the dual ascent quantity $\{\|\Lambda_j^k - \Lambda_j^{k-1}\|_F^2\}_{j=1}^N$ can be bounded by the discrepancies between two successive primal updates $\{\|W_i^k - W_i^{k-1}\|_F^2\}_{i=1}^N$, $\{\|V_i^k - V_i^{k-1}\|_F^2\}_{i=1}^N$, and $\{\|V_i^{k-1} - V_i^{k-2}\|_F^2\}_{i=1}^N$ via a recursive way.

Lemma 18 (Dual-bounded-by-primal) *Let Assumption 1 hold. For any positive integer $k \geq 2$, the following hold*

$$\|\Lambda_N^k - \Lambda_N^{k-1}\|_F = \|V_N^k - V_N^{k-1}\|_F, \qquad (71)$$

$$\begin{aligned}\|\Lambda_{N-1}^k - \Lambda_{N-1}^{k-1}\|_F &\leq \|W_N^k\|_F \cdot \|\Lambda_N^k - \Lambda_N^{k-1}\|_F + \|\Lambda_N^{k-1}\|_F \cdot \|W_N^k - W_N^{k-1}\|_F\\ &+ \beta_N\|W_N^k\|_F \cdot \|V_N^k - V_N^{k-1}\|_F + \beta_N\|W_N^{k-1}\|_F \cdot \|V_N^{k-1} - V_N^{k-2}\|_F, \qquad (72)\end{aligned}$$

and for $j = 1, \ldots, N-2$,

$$\begin{aligned}\|\Lambda_j^k - \Lambda_j^{k-1}\|_F \leq\; & L_1 \|W_{j+1}^k\|_F \cdot \|\Lambda_{j+1}^k - \Lambda_{j+1}^{k-1}\|_F \\ & + \left(L_1 \|\Lambda_{j+1}^{k-1}\|_F + L_2 \|W_{j+1}^{k-1}\|_F \cdot \|\Lambda_{j+1}^{k-1}\|_F \cdot \|V_j^{k-1}\|_F\right) \cdot \|W_{j+1}^k - W_{j+1}^{k-1}\|_F \\ & + L_1 \beta_{j+1} \left(\|W_{j+1}^k\|_F \cdot \|V_{j+1}^k - V_{j+1}^{k-1}\|_F + \|W_{j+1}^{k-1}\|_F \cdot \|V_{j+1}^{k-1} - V_{j+1}^{k-2}\|_F\right) \\ & + \left(L_1^2 + \frac{\mu_j^k}{2}\right) \beta_{j+1} \|W_{j+1}^k\|_F^2 \cdot \|V_j^k - V_j^{k-1}\|_F \\ & + \left((L_1^2 + \mu_j^{k-1}/2) \cdot \beta_{j+1} + L_2 \|\Lambda_{j+1}^{k-1}\|_F\right) \|W_{j+1}^{k-1}\|_F^2 \cdot \|V_j^{k-1} - V_j^{k-2}\|_F,\end{aligned} \tag{73}$$

where L_1 and L_2 are two constants specified in Assumption 1.

From Lemma 18, the amount of the dual ascent part at j-th layer is related to all the later layers (i.e., $i = j+1, \ldots, N$) via a recursive way. Besides these terms $\{\|W_i^k - W_i^{k-1}\|_F^2\}_{i=1}^N$ and $\{\|V_i^k - V_i^{k-1}\|_F^2\}_{i=1}^N$ exist in the upper bounds, the discrepancies between the previous two updates $\{\|V_i^{k-1} - V_i^{k-2}\|_F^2\}_{i=1}^N$ are also involved in the upper bounds. This may bring some challenge to construct the Lyapunov function such that the sequence or its variant is a descent sequence, because in this case, the augmented Lagrangian shall not be an appropriate Lyapunov function by Lemma 15, where the amount of descent part is only characterized by $\{\|W_i^k - W_i^{k-1}\|_F^2\}_{i=1}^N$ and $\{\|V_i^k - V_i^{k-1}\|_F^2\}_{i=1}^N$ without $\{\|V_i^{k-1} - V_i^{k-2}\|_F^2\}_{i=1}^N$.

Proof The equality (71) follows directly from (47). By the update (48) of Λ_{N-1}^k, the following holds

$$\begin{aligned}& \Lambda_{N-1}^k - \Lambda_{N-1}^{k-1} \\ & = (W_N^k)^T \Lambda_N^k - (W_N^{k-1})^T \Lambda_N^{k-1} + \beta_N (W_N^k)^T (V_N^k - V_N^{k-1}) - \beta_N (W_N^{k-1})^T (V_N^{k-1} - V_N^{k-2}) \\ & = (W_N^k)^T (\Lambda_N^k - \Lambda_N^{k-1}) + (W_N^k - W_N^{k-1})^T \Lambda_N^{k-1} + \beta_N (W_N^k)^T (V_N^k - V_N^{k-1}) \\ & - \beta_N (W_N^{k-1})^T (V_N^{k-1} - V_N^{k-2}),\end{aligned}$$

which implies (72) directly by the triangle inequality. For $j = 1, \ldots, N-2$, by the update of (49),

$$\begin{aligned}& \Lambda_j^k - \Lambda_j^{k-1} = (W_{j+1}^k)^T \left(\Lambda_{j+1}^k \odot \sigma'(W_{j+1}^k V_j^{k-1})\right) - (W_{j+1}^{k-1})^T \left(\Lambda_{j+1}^{k-1} \odot \sigma'(W_{j+1}^{k-1} V_j^{k-2})\right) \\ & + \beta_{j+1} (W_{j+1}^k)^T \Bigg[\left((\sigma(W_{j+1}^k V_j^{k-1}) - \sigma(W_{j+1}^k V_j^k)) + (V_{j+1}^k - V_{j+1}^{k-1})\right) \odot \sigma'(W_{j+1}^k V_j^{k-1}) \\ & + \frac{\mu_j^k}{2} W_{j+1}^k (V_j^k - V_j^{k-1}) \Bigg] \\ & - \beta_{j+1} (W_{j+1}^{k-1})^T \Bigg[\left((\sigma(W_{j+1}^{k-1} V_j^{k-2}) - \sigma(W_{j+1}^{k-1} V_j^{k-1})) + (V_{j+1}^{k-1} - V_{j+1}^{k-2})\right) \odot \sigma'(W_{j+1}^{k-1} V_j^{k-2}) \\ & + \frac{\mu_j^{k-1}}{2} W_{j+1}^{k-1} (V_j^{k-1} - V_j^{k-2}) \Bigg].\end{aligned}$$

By Assumption 1 and the triangle inequality, the above equality implies that

$$\begin{aligned}
&\|\Lambda_j^k - \Lambda_j^{k-1}\|_F \le \|(W_{j+1}^k)^T \left(\Lambda_{j+1}^k \odot \sigma'(W_{j+1}^k V_j^{k-1})\right) - (W_{j+1}^{k-1})^T \left(\Lambda_{j+1}^{k-1} \odot \sigma'(W_{j+1}^{k-1} V_j^{k-2})\right) \|_F \\
&+ \beta_{j+1}\|W_{j+1}^k\|_F \left((L_1^2 + \mu_j^k/2)\|W_{j+1}^k\|_F\|V_j^k - V_j^{k-1}\|_F + L_1\|V_{j+1}^k - V_{j+1}^{k-1}\|_F\right) \qquad (74)\\
&+ \beta_{j+1}\|W_{j+1}^{k-1}\|_F \left((L_1^2 + \mu_j^{k-1}/2)\|W_{j+1}^{k-1}\|_F\|V_j^{k-1} - V_j^{k-2}\|_F + L_1\|V_{j+1}^{k-1} - V_{j+1}^{k-2}\|_F\right).
\end{aligned}$$

Note that

$$\begin{aligned}
&\|(W_{j+1}^k)^T \left(\Lambda_{j+1}^k \odot \sigma'(W_{j+1}^k V_j^{k-1})\right) - (W_{j+1}^{k-1})^T \left(\Lambda_{j+1}^{k-1} \odot \sigma'(W_{j+1}^{k-1} V_j^{k-2})\right) \|_F \\
&\le \|W_{j+1}^k - W_{j+1}^{k-1}\|_F\|\Lambda_{j+1}^k \odot \sigma'(W_{j+1}^k V_j^{k-1})\|_F \\
&+ \|W_{j+1}^{k-1}\|_F\|\Lambda_{j+1}^k \odot \sigma'(W_{j+1}^k V_j^{k-1}) - \Lambda_{j+1}^{k-1} \odot \sigma'(W_{j+1}^{k-1} V_j^{k-2})\|_F \\
&\le L_1\|\Lambda_{j+1}^k\|_F\|W_{j+1}^k - W_{j+1}^{k-1}\|_F + L_1\|W_{j+1}^{k-1}\|_F\|\Lambda_{j+1}^k - \Lambda_{j+1}^{k-1}\|_F \\
&+ L_2\|W_{j+1}^{k-1}\|_F\|\Lambda_{j+1}^{k-1}\|_F\|V_j^{k-1}\|_F\|W_{j+1}^k - W_{j+1}^{k-1}\|_F \\
&+ L_2\|W_{j+1}^{k-1}\|_F^2\|\Lambda_{j+1}^{k-1}\|_F\|V_j^{k-1} - V_j^{k-2}\|_F, \qquad (75)
\end{aligned}$$

where the final inequality holds for

$$\begin{aligned}
&\|\Lambda_{j+1}^k \odot \sigma'(W_{j+1}^k V_j^{k-1}) - \Lambda_{j+1}^{k-1} \odot \sigma'(W_{j+1}^{k-1} V_j^{k-2})\|_F \\
&\le \|(\Lambda_{j+1}^k - \Lambda_{j+1}^{k-1}) \odot \sigma'(W_{j+1}^k V_j^{k-1})\|_F + \|\Lambda_{j+1}^{k-1} \odot (\sigma'(W_{j+1}^k V_j^{k-1}) - \sigma'(W_{j+1}^{k-1} V_j^{k-2}))\|_F \\
&\le L_1\|\Lambda_{j+1}^k - \Lambda_{j+1}^{k-1}\|_F + L_2\|\Lambda_{j+1}^{k-1}\|_F\|W_{j+1}^k V_j^{k-1} - W_{j+1}^{k-1} V_j^{k-2}\|_F \\
&\le L_1\|\Lambda_{j+1}^k - \Lambda_{j+1}^{k-1}\|_F + L_2\|\Lambda_{j+1}^{k-1}\|_F \left(\|W_{j+1}^k - W_{j+1}^{k-1}\|_F\|V_j^{k-1}\|_F + \|W_{j+1}^{k-1}\|_F\|V_j^{k-1} - V_j^{k-2}\|_F\right)
\end{aligned}$$

by Assumption 1 and the triangle inequality. Substituting (75) into (74) yields (73). This completes the proof of this lemma. ■

D.1.3 Lemma 19: Boundedness lemma

Note that in the upper bounds of Lemma 18, the terms $\{\|W_i^k - W_i^{k-1}\|_F\}_{i=1}^N$, $\{\|V_i^k - V_i^{k-1}\|_F\}_{i=1}^N$ and $\{\|V_i^{k-1} - V_i^{k-2}\|_F^2\}_{i=1}^N$ are multiplied by many other terms including $\{\|W_i^k\|_F\}_{i=1}^N$, $\{\|V_i^k\|_F\}_{i=1}^N$, $\{\|\Lambda_i^k\|_F\}_{i=1}^N$, and the locally Lipschitz constants $\{h_i^k := \mathbb{L}(\|V_i^{k-1} - \beta_i^{-1}\Lambda_i^{k-1}\|_{\max})\}_{i=1}^{N-1}$ and $\{\mu_j^k := \mathbb{L}(\|V_{j+1}^{k-1} - \beta_{j+1}^{-1}\Lambda_{j+1}^{k-1}\|_{\max})\}_{j=1}^{N-2}$, highly depending on the current or previous updates. In order to make these bounds in Lemma 18 only depend on those desired terms, the following boundedness property of the sequence is required.

Instead of the conditions of Theorem 7, we impose the following weaker conditions:

$$\beta_N \geq 3.5, \tag{76}$$
$$\frac{\beta_{N-1}}{\beta_N} \geq 7\gamma^2, \tag{77}$$
$$\frac{\beta_i}{\beta_{i+1}} \geq 6\left(\sqrt{3L_1}+\sqrt{2L_3C_3\gamma^i}\right)^2\gamma^2, \quad i=1,\dots,N-2, \tag{78}$$
$$\lambda \geq \max\left\{\hat{\lambda}, 12\beta_N C_3^2\gamma^{2N-4}, \max_{2\leq j\leq N-1}\tilde{\lambda}_j\right\}, \tag{79}$$
$$\|W_i^0\|_F \leq \gamma, \quad \|V_i^0\|_F \leq 3C_3\gamma^{i-1}, \quad \|\Lambda_i^0\|_F \leq C_3\beta_i\gamma^{i-1}, \quad i=1,\dots,N. \tag{80}$$

It can be seen that the conditions (76)-(78) on β_i's are slightly weaker than the conditions (39)-(41).

Lemma 19 (Boundedness) *Under Assumption 1 and the above conditions* (76)-(80)*, for any $k\in\mathbb{N}$, there hold*

$$\|W_i^k\|_F \leq \gamma, \quad \|V_i^k\|_F \leq 3C_3\gamma^{i-1}, \quad \|\Lambda_i^k\|_F \leq C_3\beta_i\gamma^{i-1}, \quad i=1,\dots,N, \tag{81}$$
$$h_i^k \leq 4L_3C_3\gamma^{i-1}, \quad i=1,\dots,N-1, \tag{82}$$
$$\mu_i^k \leq 4L_3C_3\gamma^{i}, \quad i=1,\dots,N-2, \tag{83}$$

where $\gamma := \max_{1\leq i\leq N}\|W_i^0\|_F$ (particularly, $\gamma=1$ in the normalized case), C_3 and L_3 are specified later in (38) *and* (33)*, respectively.*

The boundedness of the sequence is mainly derived by the specific updates of the algorithm and the introduced ℓ_2 regularization.

Proof [Proof of Lemma 19] We first show that the boundedness condition holds for $k=1$. By the definitions of (18) and (33), it holds

$$\mathbb{L}(|c|) \leq L_3|c|, \quad \forall\ |c|\geq 1.$$

By the settings (21), (22) of h_i^k and μ_i^k, and (80),

$$h_i^1 \leq \mathbb{L}(\|V_i^0\|_F+\beta_i^{-1}\|\Lambda_i^0\|_F) \leq \mathbb{L}(4C_3\gamma^{i-1}) \leq 4L_3C_3\gamma^{i-1}, \quad i=1,\dots,N-1, \tag{84}$$
$$\mu_i^1 \leq \mathbb{L}(\|V_{i+1}^0\|_F+\beta_{i+1}^{-1}\|\Lambda_{i+1}^0\|_F) \leq \mathbb{L}(4C_3\gamma^{i}) \leq 4L_3C_3\gamma^{i}, \quad i=1,\dots,N-2, \tag{85}$$

where the final inequalities in both (84) and (85) hold for $4C_3\gamma^{i-1}\geq 1$ by the definition (38) and $L_0\geq \frac{1}{8}$ in Assumption 1. In the following, we show that (80) holds.

(1) On boundedness of W_N^1. By (50),

$$\|W_N^1\|_F \leq \lambda^{-1}\cdot 12\beta_N C_3^2\gamma^{2N-3} \leq \gamma,$$

where the last inequality follows from the assumption (79) of λ.

(2) On boundedness of W_i^1, $i=N-1,\dots,2$. By (51),

$$\|W_i^1\|_F \leq \left(1-\frac{\lambda}{\lambda+18\beta_iL_3C_3^3\gamma^{3i-5}}\right)\gamma + \frac{3L_1\beta_iC_3\gamma^{i-2}(4C_3\gamma^{i-1}+L_0\sqrt{nd_i})}{\lambda}.$$

To make $\|W_i^1\|_F \le \gamma$, it requires $\lambda \ge \frac{a_i+\sqrt{a_i^2+4a_ib_i}}{2}$, where $a_i := 3L_1\beta_iC_3\gamma^{i-3}(4C_3\gamma^{i-1} + L_0\sqrt{nd_i})$, and $b_i = 18\beta_iL_3C_3^3\gamma^{3i-5}$. By the assumption (79) of λ, we have

$$\lambda \ge a_i\left(1+\sqrt{\frac{b_i}{a_i}}\right) \ge \frac{a_i+\sqrt{a_i^2+4a_ib_i}}{2}.$$

Thus, $\|W_i^1\|_F \le \gamma$ for $i = 2, \dots, N-1$.

(3) On boundedness of W_1^1. By (52),

$$\|W_1^1\|_F \le \left(1-\frac{\lambda}{\lambda+2\beta_1L_3C_3\|X\|_F^2}\right)\gamma + \frac{L_1\beta_1\|X\|_F(4C_3+L_0\sqrt{nd_1})}{\lambda}.$$

Similarly, by the assumption of λ (79), we can show that if

$$\lambda \ge a_1\left(1+\sqrt{\frac{b_1}{a_1}}\right) \ge \frac{a_1+\sqrt{a_1^2+4a_1b_1}}{2},$$

where $a_1 := L_1\beta_1\|X\|_F(4C_3+L_0\sqrt{nd_1})\gamma^{-1},\ b_1 := 2\beta_1L_3C_3\|X\|_F^2$, then $\|W_1^1\|_F \le \gamma$.

(4) On boundedness of V_j^1, $j = 1, \dots, N-2$. By (57),

$$\|V_j^1\|_F \le \left(1-\frac{\rho_j}{\rho_j+2L_3C_3\gamma^{j+2}}\right)\cdot 3C_3\gamma^{j-1} + (C_3\gamma^{j-1}+L_0\sqrt{nd_j}) + \frac{L_1\gamma(4C_3\gamma^j+L_0\sqrt{nd_{j+1}})}{\rho_j},$$

where $\rho_j := \frac{\beta_j}{\beta_{j+1}}$. To guarantee $\|V_j^1\|_F \le 3C_3\gamma^{j-1}$, it requires

$$\rho_j \ge \frac{\bar{b}_j+\sqrt{\bar{b}_j^2+4\bar{a}_j\bar{c}_j}}{2\bar{a}_j},$$

where $\bar{a}_j = 2 - \frac{L_0\sqrt{nd_j}}{C_3\gamma^{j-1}}$, $\bar{b}_j = 2L_3C_3\gamma^{j+2} + 2L_3\gamma^3L_0\sqrt{nd_j} + 4L_1\gamma^2 + \frac{L_1L_0\sqrt{nd_{j+1}}}{C_3\gamma^{j-2}}$, and $\bar{c}_j = 2L_1L_3\gamma^4(4C_3\gamma^j + L_0\sqrt{nd_{j+1}})$. By the definition of C_3 (38),

$$\bar{a}_j \ge \frac{3}{2},\quad \bar{b}_j \le \left(4.5L_1+3L_3C_3\gamma^j\right)\gamma^2,\quad \bar{c}_j \le 9L_1L_3C_3\gamma^{j+4},$$

where the bound on $\bar{b}_j$ follows from the following facts

$$2L_0\sqrt{nd_j} \le C_3\gamma^{j-1},\quad \frac{L_1L_0\sqrt{nd_{j+1}}}{C_3\gamma^{j-2}} \le \frac{1}{2}L_1\gamma^2,$$

and the bound on $\bar{c}_j$ is due to $L_0\sqrt{nd_{j+1}} \le \frac{1}{2}C_3\gamma^j$.

Thus, it yields

$$\begin{aligned}\frac{\bar{b}_j+\sqrt{\bar{b}_j^2+4\bar{a}_j\bar{c}_j}}{2\bar{a}_j} &\le \frac{1}{3}\bar{b}_j\left(1+\sqrt{1+\frac{6\bar{c}_j}{\bar{b}_j^2}}\right) \le \frac{2}{3}\bar{b}_j + \frac{\sqrt{6\bar{c}_j}}{3}\\ &\le \left(3L_1+\sqrt{6L_1L_3C_3\gamma^j}+2L_3C_3\gamma^j\right)\gamma^2,\end{aligned}$$

where the first inequality holds for $\bar{a}_j \geq \frac{3}{2}$, and the final inequality holds for the upper bounds of $\bar{b}_j$ and $\bar{c}_j$. Thus, we show the boundedness of V_j^1 under our assumptions for any $j = 1, \ldots, N-2$.

(5) On boundedness of V_{N-1}^1. By (56),

$$\begin{aligned}\|V_{N-1}^1\|_F &\leq C_3\gamma^{N-2} + L_0\sqrt{nd_{N-1}} + 4C_3\gamma^N\rho_{N-1}^{-1} \\ &\leq \frac{3}{2}C_3\gamma^{N-2} + 4C_3\gamma^N\rho_{N-1}^{-1} \leq 3C_3\gamma^{N-2}\end{aligned}$$

where the first inequality holds by Assumption 1 and (80), the second inequality by the definition (38) of C_3, and the final inequality is due to (77).

(6) On boundedness of V_N^1. By (55), it shows that

$$\|V_N^1\|_F \leq \frac{3C_3\gamma^{N-1}}{1+\beta_N} + \frac{\beta_N}{1+\beta_N}\gamma \cdot 3C_3\gamma^{N-2} \leq 3C_3\gamma^{N-1}.$$

(7) On boundedness of Λ_N^1. By (47),

$$\|\Lambda_N^1\|_F \leq \|V_N^1\|_F + \|Y\|_F \leq 3C_3\gamma^{N-1} + \|Y\|_F \leq C_3\beta_N\gamma^{N-1},$$

where the final inequality holds by the definition (38) of C_3.

(8) On boundedness of Λ_{N-1}^1. By (48),

$$\|\Lambda_{N-1}^1\|_F \leq 7C_3\beta_N\gamma^N \leq C_3\beta_{N-1}\gamma^{N-2},$$

where the final inequality is due to (77).

(9) On boundedness of Λ_j^1, $j = N-2, \ldots, 1$ by induction. By (49),

$$\begin{aligned}\|\Lambda_j^1\|_F &\leq \beta_{j+1}\gamma\left(2L_1L_0\sqrt{nd_{j+1}} + 7L_1C_3\gamma^j + 12L_3C_3^2\gamma^{2j}\right) \\ &\leq C_3\beta_{j+1}\gamma^{j+1}(8L_1 + 12L_3C_3\gamma^j) \leq C_3\beta_j\gamma^{j-1},\end{aligned}$$

where the second inequality holds for $2L_0\sqrt{nd_{j+1}} \leq C_3\gamma^j$, and the final inequality holds for (78).

Therefore, we have shown that (81)-(83) hold for $k=1$. Similarly, we can show that once (81)-(83) hold for some k, then they will hold for $k+1$. Hence, we can show (81)-(83) hold for any $k \in \mathbb{N}$ recursively. This completes the proof of this lemma. ■

D.1.4 Proof of Lemma 14: Sufficient descent lemma

To prove Lemma 14, we first present a key lemma based on Lemma 18 and Lemma 19. For any $k \geq 2$ and $j = 1, \ldots, N-2$, we denote

$$\begin{aligned}
\mathcal{E}_{1,j}^k &:= (L_1\gamma)^{N-j} L_1^{-1} C_3 \beta_N \gamma^{N-2} \|W_N^k - W_N^{k-1}\|_F \\
&\quad + \sum_{i=j+1}^{N-1} (L_1\gamma)^{i-j} (C_3 \beta_i \gamma^{i-2} + 3L_1^{-1} L_2 C_3^2 \beta_i \gamma^{2i-3}) \|W_i^k - W_i^{k-1}\|_F, \\
\mathcal{E}_{2,j}^k &:= (L_1\gamma)^{N-j}(1+\beta_N) L_1^{-1} \|V_N^k - V_N^{k-1}\|_F + (L_1\gamma)^{N-1-j} \beta_{N-1} \|V_{N-1}^k - V_{N-1}^{k-1}\|_F \\
&\quad + \sum_{i=j+1}^{N-2} (L_1\gamma)^{i-j} \left(\beta_i + (L_1^2 + 2L_3 C_3 \gamma^i)\gamma^2 \beta_{i+1}\right) \|V_i^k - V_i^{k-1}\|_F \\
&\quad + (L_1^2 + 2L_3 C_3 \gamma^j)\gamma^2 \beta_{j+1} \|V_j^k - V_j^{k-1}\|_F,
\end{aligned}$$

and

$$\begin{aligned}
\mathcal{E}_{3,j}^k &:= (L_1\gamma)^{N-j} \beta_N L_1^{-1} \|V_N^{k-1} - V_N^{k-2}\|_F + (L_1\gamma)^{N-1-j} \beta_{N-1} \|V_{N-1}^{k-1} - V_{N-1}^{k-2}\|_F \\
&\quad + \sum_{i=j+1}^{N-2} (L_1\gamma)^{i-j} \left[\beta_i + (L_1^2 + 2L_3 C_3 \gamma^i + L_2 C_3 \gamma^i)\gamma^2 \beta_{i+1}\right] \|V_i^{k-1} - V_i^{k-2}\|_F \\
&\quad + \left(L_1^2 + 2L_3 C_3 \gamma^j + L_2 C_3 \gamma^j\right) \beta_{j+1} \gamma^2 \|V_j^{k-1} - V_j^{k-2}\|_F.
\end{aligned}$$

Lemma 20 *Under assumptions of Lemma 19, for any $k \geq 2$, we have*

$$\begin{aligned}
&\|\Lambda_N^k - \Lambda_N^{k-1}\|_F = \|V_N^k - V_N^{k-1}\|_F, \\
&\|\Lambda_{N-1}^k - \Lambda_{N-1}^{k-1}\|_F \leq C_3 \beta_N \gamma^{N-1} \|W_N^k - W_N^{k-1}\|_F + \gamma(1+\beta_N) \|V_N^k - V_N^{k-1}\|_F \\
&\qquad\qquad + \beta_N \gamma \|V_N^{k-1} - V_N^{k-2}\|_F,
\end{aligned}$$

and for $j = 1, \ldots, N-2$,

$$\|\Lambda_j^k - \Lambda_j^{k-1}\|_F \leq \mathcal{E}_{1,j}^k + \mathcal{E}_{2,j}^k + \mathcal{E}_{3,j}^k.$$

Moreover, the above inequalities imply

$$\sum_{i=1}^N \|\Lambda_i^k - \Lambda_i^{k-1}\|_F^2 \leq \alpha \sum_{i=1}^N \left(\|W_i^k - W_i^{k-1}\|_F^2 + \|V_i^k - V_i^{k-1}\|_F^2 + \|V_i^{k-1} - V_i^{k-2}\|_F^2 \right) \quad (86)$$

for some constant $\alpha > 0$ specified in the proof.

Proof The bounds of $\|\Lambda_N^k - \Lambda_N^{k-1}\|_F$ and $\|\Lambda_{N-1}^k - \Lambda_{N-1}^{k-1}\|_F$ are obvious by Lemma 18 and Lemma 19. For $j = 1, \ldots, N-2$, by Lemma 18 and Lemma 19, it holds

$$\|\Lambda_j^k - \Lambda_j^{k-1}\|_F \leq L_1\gamma \|\Lambda_{j+1}^k - \Lambda_{j+1}^{k-1}\|_F + T_{j+1}^k + I_j^k,$$

where $T_{j+1}^k := (L_1 C_3 \beta_{j+1} \gamma^j + 3C_3^2 L_2 \beta_{j+1} \gamma^{2j}) \|W_{j+1}^k - W_{j+1}^{k-1}\|_F + L_1 \gamma \beta_{j+1} (\|V_{j+1}^k - V_{j+1}^{k-1}\|_F + \|V_{j+1}^{k-1} - V_{j+1}^{k-2}\|_F)$, and

$$I_j^k := (L_1^2 + 2L_3 C_3 \gamma^j) \beta_{j+1} \gamma^2 \|V_j^k - V_j^{k-1}\|_F + \left(L_1^2 + 2L_3 C_3 \gamma^j + L_2 C_3 \gamma^j\right) \beta_{j+1} \gamma^2 \|V_j^{k-1} - V_j^{k-2}\|_F.$$

By the above inequality, we have

$$\|\Lambda_j^k - \Lambda_j^{k-1}\|_F \le (L_1\gamma)^{N-1-j}\|\Lambda_{N-1}^k - \Lambda_{N-1}^{k-1}\|_F + (L_1\gamma)^{N-2-j}T_{N-1}^k + \sum_{i=1}^{N-2-j}(L_1\gamma)^{i-1}\left(T_{j+i}^k + L_1\gamma I_{j+i}^k\right) + I_j^k.$$

Substituting the definitions of T_j^k and I_j^k into this inequality and after some simplifications yields the desired bound for $\|\Lambda_j^k - \Lambda_j^{k-1}\|_F$. Summing up all the above inequalities and using several times of the basic inequality $(\sum_{i=1}^p u_i)^2 \le p\sum_{i=1}^p u_i^2$ for any $u \in \mathbb{R}^p$ yields (86) with some positive constant α. This completes the proof. ■

Based on Lemma 15, Lemma 19 and Lemma 20, we prove Lemma 14 as follows.

Proof [Proof of Lemma 14] By (41) and the definition (38) of C_3 , we have for $j = 1,\dots,N-2$, $\frac{\beta_j}{\beta_{j+1}} \ge f_{\min}^2\gamma^2$, and

$$\beta_j \ge f_{\min}^{2(i-j)}\gamma^{2(i-j)}\beta_i, \quad j < i \le N-1. \tag{87}$$

By (36)-(37) and (43), it holds

$$\alpha_3 \ge 24N + 1. \tag{88}$$

To prove this lemma, we first estimate $\|\Lambda_i^k - \Lambda_i^{k-1}\|_F^2$ for any $i = 1,\dots,N$. By Lemma 20, we get

$$\|\Lambda_N^k - \Lambda_N^{k-1}\|_F^2 = \|V_N^k - V_N^{k-1}\|_F^2, \tag{89}$$

and using the basic inequality $\left(\sum_{i=1}^3 a_i\right)^2 \le 3\sum_{i=1}^3 a_i^2$,

$$\|\Lambda_{N-1}^k - \Lambda_{N-1}^{k-1}\|_F^2 \le 3C_3^2\beta_N^2\gamma^{2(N-1)}\|W_N^k - W_N^{k-1}\|_F^2 + 3\gamma^2(1+\beta_N)^2\|V_N^k - V_N^{k-1}\|_F^2 + 3\beta_N^2\gamma^2\|V_N^{k-1} - V_N^{k-2}\|_F^2, \tag{90}$$

and for $j = 1,\dots,N-2$, using the inequality $(\sum_{i=1}^n a_i)^2 \le n\sum_{i=1}^n a_i^2$,

$$\|\Lambda_j^k - \Lambda_j^{k-1}\|_F^2 \le 2(N-j)\mathcal{T}_{1,j}^k + 4(N-j+1)(\mathcal{T}_{2,j} + \mathcal{T}_{3,j}), \tag{91}$$

where

$$\begin{aligned}\mathcal{T}_{1,j}^k &= (L_1\gamma)^{2(N-j)}L_1^{-2}C_3^2\beta_N^2\gamma^{2(N-2)}\|W_N^k - W_N^{k-1}\|_F^2 \\ &\quad + \sum_{i=j+1}^{N-1}(L_1\gamma)^{2(i-j)}(1 + 3L_1^{-1}L_2C_3\gamma^{i-1})^2C_3^2\beta_i^2\gamma^{2(i-2)}\|W_i^k - W_i^{k-1}\|_F^2, \\ \mathcal{T}_{2,j}^k &= (L_1\gamma)^{2(N-j)}(1+\beta_N)^2L_1^{-2}\|V_N^k - V_N^{k-1}\|_F^2 + (L_1\gamma)^{2(N-1-j)}\beta_{N-1}^2\|V_{N-1}^k - V_{N-1}^{k-1}\|_F^2 \\ &\quad + \sum_{i=j+1}^{N-2}(L_1\gamma)^{2(i-j)}\left[\beta_i + (L_1^2 + 2L_3C_3\gamma^i)\gamma^2\beta_{i+1}\right]^2\|V_i^k - V_i^{k-1}\|_F^2 \\ &\quad + (L_1^2 + 2L_3C_3\gamma^j)^2\gamma^4\beta_{j+1}^2\|V_j^k - V_j^{k-1}\|_F^2,\end{aligned}$$

and

$$\begin{aligned}\mathcal{T}_{3,j}^k &= (L_1\gamma)^{2(N-j)}\beta_N^2 L_1^{-2}\|V_N^{k-1}-V_N^{k-2}\|_F^2 + (L_1\gamma)^{2(N-1-j)}\beta_{N-1}^2\|V_{N-1}^{k-1}-V_{N-1}^{k-2}\|_F^2 \\ &+ \sum_{i=j+1}^{N-2}(L_1\gamma)^{2(i-j)}\left[\beta_i + (L_1^2+2L_3C_3\gamma^i+L_2C_3\gamma^i)\gamma^2\beta_{i+1}\right]^2\|V_i^{k-1}-V_i^{k-2}\|_F^2 \\ &+ \left(L_1^2+2L_3C_3\gamma^j+L_2C_3\gamma^j\right)^2\beta_{j+1}^2\gamma^4\|V_j^{k-1}-V_j^{k-2}\|_F^2.\end{aligned}$$

Substituting (89), (90) and (91) into Lemma 15 and after some simplifications yields

$$\begin{aligned}\mathcal{L}(\mathcal{Q}^k) + \sum_{i=1}^N \xi_i\|V_i^k - V_i^{k-1}\|_F^2 \le \mathcal{L}(\mathcal{Q}^{k-1}) + \sum_{i=1}^N \xi_i\|V_i^{k-1}-V_i^{k-2}\|_F^2 \\ - \sum_{i=1}^N \zeta_i\|W_i^k - W_i^{k-1}\|_F^2 - \sum_{i=1}^N(\eta_i-\xi_i)\|V_i^k-V_i^{k-1}\|_F^2,\end{aligned} \tag{92}$$

where

$$\zeta_N = \frac{\lambda}{2} - 3C_3^2\beta_{N-1}^{-1}\beta_N^2\gamma^{2(N-1)} - 2L_1^{-2}C_3^2\beta_N^2\gamma^{2(N-2)}\sum_{j=1}^{N-2}\beta_j^{-1}(N-j)(L_1\gamma)^{2(N-j)}$$

$$\zeta_i = \frac{\lambda}{2} - 2(1+3L_1^{-1}L_2L_3\gamma^{i-1})^2C_3^2\beta_i^2\gamma^{2(i-2)}\sum_{j=1}^{i-1}\beta_j^{-1}(N-j)(L_1\gamma)^{2(i-j)}, \quad i=2,\ldots,N-1,$$

$$\zeta_1 = \frac{\lambda}{2},$$

$$\eta_N = \frac{1+\beta_N}{2} - \beta_N^{-1} - 3\gamma^2(1+\beta_N)^2\beta_{N-1}^{-1} - \frac{4(1+\beta_N)^2}{L_1^2}\sum_{j=1}^{N-2}\beta_j^{-1}(N-j+1)(L_1\gamma)^{2(N-j)},$$

$$\xi_N = 3\gamma^2\beta_N^2\beta_{N-1}^{-1} + \frac{4\beta_N^2}{L_1^2}\sum_{j=1}^{N-2}\beta_j^{-1}(N-j+1)(L_1\gamma)^{2(N-j)}, \tag{93}$$

$$\eta_{N-1} = \frac{\beta_{N-1}}{2} - 4\beta_{N-1}^2\sum_{j=1}^{N-2}\beta_j^{-1}(N-j+1)(L_1\gamma)^{2(N-1-j)},$$

$$\xi_{N-1} = 4\beta_{N-1}^2\sum_{j=1}^{N-2}\beta_j^{-1}(N-j+1)(L_1\gamma)^{2(N-1-j)}, \tag{94}$$

and for $i = 2, \ldots, N-2$,

$$\begin{aligned}\eta_i = {} & \frac{\beta_i}{2} - 4\left[\beta_i + (L_1^2 + 2L_3C_3\gamma^i)\gamma^2\beta_{i+1}\right]^2 \sum_{j=1}^{i-1} \beta_j^{-1}(N-j+1)(L_1\gamma)^{2(i-j)} \\ & - 4(L_1^2 + 2L_3C_3\gamma^i)^2\gamma^4\beta_{i+1}^2\beta_i^{-1}(N-i+1), \\ \xi_i = {} & 4\left[\beta_i + (L_1^2 + 2L_3C_3\gamma^i + L_2C_3\gamma^i)\gamma^2\beta_{i+1}\right]^2 \sum_{j=1}^{i-1} \beta_j^{-1}(N-j+1)(L_1\gamma)^{2(i-j)} \\ & + 4\left(L_1^2 + 2L_3C_3\gamma^i + L_2C_3\gamma^i\right)^2\gamma^4\beta_{i+1}^2\beta_i^{-1}(N-i+1),\end{aligned} \tag{95}$$

and $\eta_1 = \frac{\beta_1}{2} - 4(L_1^2 + 2L_3C_3\gamma)^2\gamma^4\beta_2^2\beta_1^{-1}N$, and

$$\xi_1 = 4\left(L_1^2 + 2L_3C_3\gamma + L_2C_3\gamma\right)^2\gamma^4\beta_2^2\beta_1^{-1}N. \tag{96}$$

Based on (92), to get (59), we need to show that

$$\zeta_i > 0, \quad \eta_i - \xi_i > 0, \quad i = 1, \ldots, N. \tag{97}$$

Then, let

$$a := \min\{\zeta_i, \eta_i - \xi_i, i = 1, \ldots, N\}, \tag{98}$$

we get (59). In the following, we show (97).

It is obvious that $\zeta_1 = \frac{\lambda}{2} > 0$. For $i = 2, \ldots, N-1$, by (87),

$$\begin{aligned}\zeta_i & \geq \frac{\lambda}{2} - 2C_3^2\beta_i(1 + 3L_1^{-1}L_2L_3\gamma^{i-1})^2\gamma^{2(i-2)}\sum_{j=1}^{i-1}(N-j)\alpha_3^{-(i-j)} \\ & > \frac{\lambda}{2} - 2C_3^2\beta_i(1 + 3L_1^{-1}L_2L_3\gamma^{i-1})^2\gamma^{2(i-2)} \cdot \frac{N}{\alpha_3 - 1} \geq 0,\end{aligned}$$

where the final inequality is due to $\alpha_3 > 24N + 1$ and the assumption of λ, i.e., (42). Similarly, we can show that $\zeta_N > 0$ as follows

$$\begin{aligned}\zeta_N & \geq \frac{\lambda}{2} - \beta_N C_3^2\gamma^{2(N-2)} \cdot \left(\frac{3}{16} + \frac{1}{8}\sum_{j=1}^{N-2}(N-j)\alpha_3^{-(N-j-1)}\right) \\ & > \frac{\lambda}{2} - \beta_N C_3^2\gamma^{2(N-2)} \cdot \left(\frac{3}{16} + \frac{N}{8(\alpha_3 - 1)}\right) \\ & > \frac{\lambda}{2} - \frac{1}{5}\beta_N C_3^2\gamma^{2(N-2)} > 0.\end{aligned}$$

At the end, we show $\eta_i - \xi_i > 0$ for $i = 1, \ldots, N$. Note that

$$\eta_1 - \xi_1 = \frac{\beta_1}{2} - 4\left[(L_1^2 + 2L_3C_3\gamma)^2 + (L_1^2 + 2L_3C_3\gamma + L_2C_3\gamma)^2\right]\gamma^4\beta_2^2\beta_1^{-1}N > 0,$$

where we have used (41) $\frac{\beta_1}{\beta_2} \geq 4\sqrt{N}\left[L_1^2 + (2L_3 + L_2)C_3\gamma\right]\gamma^2$.

For $i = 2, \ldots, N-2$, let $\alpha_1 := (L_1^2 + 2L_3C_3\gamma^i)\gamma^2,\ \alpha_2 := (L_1^2 + 2L_3C_3\gamma^i + L_2C_3\gamma^i)\gamma^2$. Note that

$$\begin{aligned}
\eta_i - \xi_i &= \frac{\beta_i}{2} - 4\left[(\beta_i + \alpha_1\beta_{i+1})^2 + (\beta_i + \alpha_2\beta_{i+1})^2\right]\sum_{j=1}^{i-1}\beta_j^{-1}(N-j+1)(L_1\gamma)^{2(i-j)} \\
&\quad - 4(\alpha_1^2+\alpha_2^2)\beta_{i+1}^2\beta_i^{-1}(N-i+1) \\
&> \frac{\beta_i}{2} - 4\left[(\beta_i + \alpha_1\beta_{i+1})^2 + (\beta_i + \alpha_2\beta_{i+1})^2\right]\sum_{j=1}^{i-1}\beta_i^{-1}(N-j+1)\alpha_3^{-(i-j)} \\
&\quad - 4N(\alpha_1^2+\alpha_2^2)\beta_{i+1}^2\beta_i^{-1} \\
&> \beta_i\left[\frac{1}{2} - \frac{4N}{\alpha_3-1}\left((1+\alpha_1\cdot\frac{\beta_{i+1}}{\beta_i})^2 + (1+\alpha_2\cdot\frac{\beta_{i+1}}{\beta_i})^2\right) - 4N(\alpha_1^2+\alpha_2^2)\left(\frac{\beta_{i+1}}{\beta_i}\right)^2\right] \\
&= \frac{4N}{\alpha_3-1}\beta_i\left[\frac{\alpha_3-1}{8N} - 2 - 2(\alpha_1+\alpha_2)\cdot\frac{\beta_{i+1}}{\beta_i} - \alpha_3(\alpha_1^2+\alpha_2^2)\cdot\left(\frac{\beta_{i+1}}{\beta_i}\right)^2\right] \\
&> \frac{4N}{\alpha_3-1}\beta_i\left[\frac{\alpha_3-1}{8N} - 2 - 2(\alpha_1+\alpha_2)\cdot\frac{\beta_{i+1}}{\beta_i} - \alpha_3(\alpha_1+\alpha_2)^2\cdot\left(\frac{\beta_{i+1}}{\beta_i}\right)^2\right] \\
&\geq 0, \qquad (99)
\end{aligned}$$

where the final inequality follows from (41), $\alpha_3 > 24N+1$, and

$$\frac{1+\sqrt{1+\alpha_3\left(\frac{\alpha_3-1}{8N}-2\right)}}{\frac{\alpha_3-1}{8N}-2} \leq 1 + \sqrt{24N+2} \leq 6\sqrt{N}.$$

Similarly, notice that

$$\begin{aligned}
\eta_{N-1} - \xi_{N-1} &= \frac{\beta_{N-1}}{2} - 8\beta_{N-1}^2\sum_{j=1}^{N-2}\beta_j^{-1}(N-j+1)(L_1\gamma)^{2(N-1-j)} \\
&\geq \frac{\beta_{N-1}}{2} - 8\beta_{N-1}\sum_{j=1}^{N-2}(N-j+1)\alpha_3^{-(N-1-j)} \\
&> \beta_{N-1}\left(\frac{1}{2} - \frac{8N}{\alpha_3-1}\right) \geq \frac{1}{6}\beta_{N-1} > 0.
\end{aligned}$$

Finally, note that

$$\begin{aligned}
\eta_N - \xi_N &= \frac{1+\beta_N}{2} - \beta_N^{-1} - 3\gamma^2\beta_{N-1}^{-1}[(1+\beta_N)^2+\beta_N^2] \\
&\quad - \frac{4}{L_1^2}[(1+\beta_N)^2+\beta_N^2]\sum_{j=1}^{N-2}\beta_j^{-1}(N-j+1)(L_1\gamma)^{2(N-j)} \\
&\geq \frac{1+\beta_N}{2} - \beta_N^{-1} - 3\gamma^2\beta_{N-1}^{-1}[(1+\beta_N)^2+\beta_N^2] \\
&\quad - 4\beta_{N-1}^{-1}\gamma^2[(1+\beta_N)^2+\beta_N^2]\sum_{j=1}^{N-2}(N-j+1)\alpha_3^{-(N-1-j)} \\
&> \frac{\beta_N^2+\beta_N-2}{2\beta_N} - 2(3+\frac{4N}{\alpha_3-1})(\beta_N^2+\beta_N+1)\beta_{N-1}^{-1}\gamma^2 \\
&\geq \frac{\beta_N^2+\beta_N-2}{2\beta_N} - \frac{19}{3}(\beta_N^2+\beta_N+1)\beta_{N-1}^{-1}\gamma^2 \\
&> 0,
\end{aligned}$$

where the final inequality follows from $\beta_N \geq 3.5$, which implies

$$16 > \frac{38}{3}\cdot\frac{\beta_N^2+\beta_N+1}{\beta_N^2+\beta_N-2},$$

and (40). This completes the proof. ∎

D.1.5 Proof of Lemma 21: Relative error lemma

In the following, we provide a lemma to show that the gradients of the augmented Lagrangian and the new Lyapunov function can be bounded by the discrepancy between two successive updates. Such a lemma is important to show the global convergence of a descent sequence by (Attouch et al., 2013, Theorem 2.9).

Lemma 21 *Under conditions of Theorem 7, for any positive $k \geq 2$, there exists some positive constant $\bar{b}$ such that*

$$\|\nabla\mathcal{L}(\mathcal{Q}^k)\|_F \leq \bar{b}\sum_{i=1}^N(\|W_i^k - W_i^{k-1}\|_F + \|V_i^k - V_i^{k-1}\|_F + \|V_i^{k-1} - V_i^{k-2}\|_F), \tag{100}$$

and $\|\nabla\hat{\mathcal{L}}(\hat{\mathcal{Q}}^k)\|_F \leq \hat{b}\|\hat{\mathcal{Q}}^k - \hat{\mathcal{Q}}^{k-1}\|_F$, where $\hat{b} = \sqrt{3N}b$ and $b = \bar{b} + 4\max_{1\leq i\leq N}\xi_i$.

Proof Note that

$$\nabla\mathcal{L}(\mathcal{Q}^k) = \left(\left\{\frac{\partial\mathcal{L}(\mathcal{Q}^k)}{\partial W_i}\right\}_{i=1}^N, \left\{\frac{\partial\mathcal{L}(\mathcal{Q}^k)}{\partial V_i}\right\}_{i=1}^N, \left\{\frac{\partial\mathcal{L}(\mathcal{Q}^k)}{\partial \Lambda_i}\right\}_{i=1}^N\right),$$

then

$$\left\|\nabla\mathcal{L}(\mathcal{Q}^k)\right\|_F \le \sum_{i=1}^N \left(\left\|\frac{\partial\mathcal{L}(\mathcal{Q}^k)}{\partial W_i}\right\|_F + \left\|\frac{\partial\mathcal{L}(\mathcal{Q}^k)}{\partial V_i}\right\|_F + \left\|\frac{\partial\mathcal{L}(\mathcal{Q}^k)}{\partial \Lambda_i}\right\|_F\right). \tag{101}$$

In order to bound $\|\nabla\mathcal{L}(\mathcal{Q}^k)\|_F$, we need to bound each component of $\nabla\mathcal{L}(\mathcal{Q}^k)$.

On $\left\|\frac{\partial\mathcal{L}(\mathcal{Q}^k)}{\partial W_N}\right\|_F$**:** By the optimality condition of (8),

$$\lambda W_N^k + \beta_N(W_N^k V_{N-1}^{k-1} - V_N^{k-1}){V_{N-1}^{k-1}}^T + \Lambda_N^{k-1}{V_{N-1}^{k-1}}^T = 0,$$

which implies

$$\begin{aligned}\frac{\partial\mathcal{L}(\mathcal{Q}^k)}{\partial W_N} &= \lambda W_N^k + \beta_N(W_N^k V_{N-1}^k - V_N^k)(V_{N-1}^k)^T + \Lambda_N^k (V_{N-1}^k)^T\\ &= \beta_N\left[(W_N^kV_{N-1}^k - V_N^k)(V_{N-1}^k - V_{N-1}^{k-1})^T + \left(W_N^k(V_{N-1}^k - V_{N-1}^{k-1}) - (V_N^k - V_N^{k-1})\right){V_{N-1}^{k-1}}^T\right]\\ &+ \Lambda_N^{k-1}(V_{N-1}^k - V_{N-1}^{k-1})^T + (\Lambda_N^k - \Lambda_N^{k-1})(V_{N-1}^k)^T.\end{aligned}$$

By the boundedness of the sequence (81), the above equality yields

$$\left\|\frac{\partial\mathcal{L}(\mathcal{Q}^k)}{\partial W_N}\right\|_F \le 10\beta_N C_3\gamma^{N-1}\|V_{N-1}^k - V_{N-1}^{k-1}\|_F + 3C_3\gamma^{N-2}(\beta_N+1)\|V_N^k - V_N^{k-1}\|_F. \tag{102}$$

On $\left\|\frac{\partial\mathcal{L}(\mathcal{Q}^k)}{\partial W_i}\right\|_F$**:** For $i = 2,\ldots,N-1$, by the optimality condition of (19),

$$\begin{aligned}&\lambda W_i^k + \left((\beta_i\sigma(W_i^{k-1}V_{i-1}^{k-1}) - \beta_iV_i^{k-1} + \Lambda_i^{k-1})\odot\sigma'(W_i^{k-1}V_{i-1}^{k-1})\right){V_{i-1}^{k-1}}^T\\ &+ \frac{\beta_ih_i^k}{2}(W_i^k - W_i^{k-1}){V_{i-1}^{k-1}}^T = 0,\end{aligned}$$

which implies

$$\begin{aligned}\frac{\partial\mathcal{L}(\mathcal{Q}^k)}{\partial W_i} &= \lambda W_i^k + \left((\beta_i\sigma(W_i^kV_{i-1}^k) - \beta_iV_i^k + \Lambda_i^k)\odot\sigma'(W_i^kV_{i-1}^k)\right){V_{i-1}^k}^T\\ &= \left((\beta_i\sigma(W_i^kV_{i-1}^k) - \beta_iV_i^k + \Lambda_i^k)\odot\sigma'(W_i^kV_{i-1}^k)\right){V_{i-1}^k}^T\\ &- \left((\beta_i\sigma(W_i^{k-1}V_{i-1}^{k-1}) - \beta_iV_i^{k-1} + \Lambda_i^{k-1})\odot\sigma'(W_i^{k-1}V_{i-1}^{k-1})\right){V_{i-1}^{k-1}}^T\\ &- \frac{\beta_ih_i^k}{2}(W_i^k - W_i^{k-1}){V_{i-1}^{k-1}}^T\\ &= \Big[\beta_i\left(\sigma(W_i^kV_{i-1}^k) - \sigma(W_i^{k-1}V_{i-1}^k) + \sigma(W_i^{k-1}V_{i-1}^k) - \sigma(W_i^{k-1}V_{i-1}^{k-1})\right)\odot\sigma'(W_i^kV_{i-1}^k)\\ &+ \left(\beta_i(V_i^{k-1} - V_i^k) + (\Lambda_i^k - \Lambda_i^{k-1})\right)\odot\sigma'(W_i^kV_{i-1}^k)\\ &+ \left(\beta_i\sigma(W_i^{k-1}V_{i-1}^{k-1}) - \beta_iV_i^{k-1} + \Lambda_i^{k-1}\right)\odot\left(\sigma'(W_i^kV_{i-1}^k) - \sigma'(W_i^{k-1}V_{i-1}^k)\right)\\ &+ \left(\beta_i\sigma(W_i^{k-1}V_{i-1}^{k-1}) - \beta_iV_i^{k-1} + \Lambda_i^{k-1}\right)\odot\left(\sigma'(W_i^{k-1}V_{i-1}^k) - \sigma'(W_i^{k-1}V_{i-1}^{k-1})\right)\Big]{V_{i-1}^k}^T\\ &+ \left((\beta_i\sigma(W_i^{k-1}V_{i-1}^{k-1}) - \beta_iV_i^{k-1} + \Lambda_i^{k-1})\odot\sigma'(W_i^{k-1}V_{i-1}^{k-1})\right)(V_{i-1}^k - V_{i-1}^{k-1})^T\\ &- \frac{\beta_ih_i^k}{2}(W_i^k - W_i^{k-1}){V_{i-1}^{k-1}}^T.\end{aligned}$$

By Assumption 1 and Lemma 19, the above equality yields

$$\begin{aligned}
&\left\|\frac{\partial \mathcal{L}(\mathcal{Q}^k)}{\partial W_i}\right\|_F \qquad (103)\\
&\leq 3C_3\gamma^{i-2}\left[3\beta_i C_3\gamma^{i-2}\left(L_1^2+L_0L_2\sqrt{nd_i}+4L_2C_3\gamma^{i-1}+\frac{2}{3}L_3\gamma\right)\|W_i^k-W_i^{k-1}\|_F\right.\\
&+\beta_i\left[L_1^2\gamma+(L_1+L_2\gamma)(L_0\sqrt{nd_i}+4C_3\gamma^{i-1})\right]\|V_{i-1}^k-V_{i-1}^{k-1}\|_F\\
&\left.+\beta_i L_1\|V_i^k-V_i^{k-1}\|_F+L_1\|\Lambda_i^k-\Lambda_i^{k-1}\|_F\right].
\end{aligned}$$

On $\left\|\frac{\partial \mathcal{L}(\mathcal{Q}^k)}{\partial W_1}\right\|_F$: Similarly, by the optimality condition of (19) with $i=1$,

$$\lambda W_1^k+\left[\left(\beta_1\sigma(W_1^{k-1}X)-\beta_1V_1^{k-1}+\Lambda_1^{k-1}\right)\odot\sigma'(W_1^{k-1}X)\right]X^T+\frac{\beta_1 h_1^k}{2}(W_1^k-W_1^{k-1})X^T=0,$$

which implies

$$\begin{aligned}
\frac{\partial \mathcal{L}(\mathcal{Q}^k)}{\partial W_1}&=\lambda W_1^k+\left[\left(\beta_1\sigma(W_1^kX)-\beta_1V_1^k+\Lambda_1^k\right)\odot\sigma'(W_1^kX)\right]X^T\\
&=\left[\left(\beta_1(\sigma(W_1^kX)-\sigma(W_1^{k-1}X))-\beta_1(V_1^k-V_1^{k-1})+(\Lambda_1^k-\Lambda_1^{k-1})\right)\odot\sigma'(W_1^kX)\right]X^T\\
&+\left[\left(\beta_1\sigma(W_1^{k-1}X)-\beta_1V_1^{k-1}+\Lambda_1^{k-1}\right)\odot(\sigma'(W_1^kX)-\sigma'(W_1^{k-1}X))\right]X^T\\
&+\frac{\beta_1h_1^k}{2}(W_1^{k-1}-W_1^k)X^T.
\end{aligned}$$

The above inequality yields

$$\begin{aligned}
\left\|\frac{\partial \mathcal{L}(\mathcal{Q}^k)}{\partial W_1}\right\|_F&\leq\beta_1\|X\|_F\left(\|X\|_F\cdot(L_1^2+L_0L_2\sqrt{nd_1}+4L_2C_3)+2L_3C_3\right)\|W_1^k-W_1^{k-1}\|_F\\
&+\beta_1L_1\|X\|_F\|V_1^k-V_1^{k-1}\|_F+L_1\|X\|_F\|\Lambda_1^k-\Lambda_1^{k-1}\|_F. \qquad (104)
\end{aligned}$$

On $\left\|\frac{\partial \mathcal{L}(\mathcal{Q}^k)}{\partial V_j}\right\|_F$ $(1\leq j\leq N-2)$: By the optimality condition of (20),

$$\begin{aligned}
&\beta_j(V_j^k-\sigma(W_j^kV_{j-1}^k))+{W_{j+1}^k}^T\left[\left(\Lambda_{j+1}^{k-1}+\beta_{j+1}(\sigma(W_{j+1}^kV_j^{k-1})-V_{j+1}^{k-1})\right)\odot\sigma'(W_{j+1}^kV_j^{k-1})\right]\\
&-\Lambda_j^{k-1}+\frac{\beta_{j+1}\mu_j^k}{2}{W_{j+1}^k}^TW_{j+1}^k(V_j^k-V_j^{k-1})=0,
\end{aligned}$$

which implies

$$\begin{aligned}
&\frac{\partial \mathcal{L}(\mathcal{Q}^k)}{\partial V_j}\\
&={W_{j+1}^k}^T\left[\left((\Lambda_{j+1}^k-\Lambda_{j+1}^{k-1})+\beta_{j+1}(\sigma(W_{j+1}^kV_j^k)-\sigma(W_{j+1}^kV_j^{k-1}))+\beta_{j+1}(V_{j+1}^{k-1}-V_{j+1}^k)\right)\right.\\
&\left.\odot\sigma'(W_{j+1}^kV_j^k)\right]\\
&+{W_{j+1}^k}^T\left[\left(\Lambda_{j+1}^{k-1}+\beta_{j+1}(\sigma(W_{j+1}^kV_j^{k-1})-V_{j+1}^{k-1})\right)\odot\left(\sigma'(W_{j+1}^kV_j^k)-\sigma'(W_{j+1}^kV_j^{k-1})\right)\right]\\
&+(\Lambda_j^{k-1}-\Lambda_j^k)+\frac{\beta_{j+1}\mu_j^k}{2}{W_{j+1}^k}^TW_{j+1}^k(V_j^{k-1}-V_j^k).
\end{aligned}$$

The above equality yields

$$\left\|\frac{\partial \mathcal{L}(\mathcal{Q}^k)}{\partial V_j}\right\|_F \leq \beta_{j+1}\gamma^2\left(2C_3\gamma^j(2L_2+L_3)+L_0L_2\sqrt{nd_{j+1}}\right)\|V_j^k - V_j^{k-1}\|_F \qquad (105)$$
$$+\beta_{j+1}L_1\gamma\|V_{j+1}^k - V_{j+1}^{k-1}\|_F + \|\Lambda_j^k - \Lambda_j^{k-1}\|_F + L_1\gamma\|\Lambda_{j+1}^k - \Lambda_{j+1}^{k-1}\|_F.$$

On $\left\|\frac{\partial \mathcal{L}(\mathcal{Q}^k)}{\partial V_{N-1}}\right\|_F$: By the optimality condition of (11),

$$\beta_{N-1}(V_N^k - \sigma(W_{N-1}^k V_{N-2}^k)) - \Lambda_{N-1}^{k-1} + {W_N^k}^T\left(\Lambda_N^{k-1} + \beta_N(W_N^k V_{N-1}^k - V_N^{k-1})\right) = 0,$$

which implies

$$\frac{\partial \mathcal{L}(\mathcal{Q}^k)}{\partial V_{N-1}} = \beta_{N-1}(V_N^k - \sigma(W_{N-1}^k V_{N-2}^k)) - \Lambda_{N-1}^k + {W_N^k}^T\left(\Lambda_N^k + \beta_N(W_N^k V_{N-1}^k - V_N^k)\right)$$
$$= \Lambda_{N-1}^{k-1} - \Lambda_{N-1}^k + {W_N^k}^T(\Lambda_N^k - \Lambda_N^{k-1}) + \beta_N {W_N^k}^T(V_N^{k-1} - V_N^k).$$

The above equality implies

$$\left\|\frac{\partial \mathcal{L}(\mathcal{Q}^k)}{\partial V_{N-1}}\right\|_F \leq \beta_N\gamma\|V_N^k - V_N^{k-1}\|_F + \|\Lambda_{N-1}^k - \Lambda_{N-1}^{k-1}\|_F + \gamma\|\Lambda_N^k - \Lambda_N^{k-1}\|_F. \qquad (106)$$

On $\left\|\frac{\partial \mathcal{L}(\mathcal{Q}^k)}{\partial V_N}\right\|_F$: Similarly, by the optimality condition of (12), we get

$$\left\|\frac{\partial \mathcal{L}(\mathcal{Q}^k)}{\partial V_N}\right\|_F = \|\Lambda_N^k - \Lambda_N^{k-1}\|_F. \qquad (107)$$

Moreover, for $i = 1, \ldots, N$, by the update of Λ_i^k, we can easily yield

$$\left\|\frac{\partial \mathcal{L}(\mathcal{Q}^k)}{\partial \Lambda_i}\right\|_F = \beta_i^{-1}\|\Lambda_i^k - \Lambda_i^{k-1}\|_F. \qquad (108)$$

Substituting (102)-(108) into (101), and after some simplifications, we get

$$\left\|\nabla \mathcal{L}(\mathcal{Q}^k)\right\|_F \leq \bar{\alpha}\sum_{i=1}^N(\|W_i^k - W_i^{k-1}\|_F + \|V_i^k - V_i^{k-1}\|_F + \|\Lambda_i^k - \Lambda_i^{k-1}\|_F) \qquad (109)$$

for some $\bar{\alpha} > 0$. By Lemma 20, substituting these upper bounds of $\|\Lambda_i^k - \Lambda_i^{k-1}\|_F$ ($i = 1, \ldots, N$) into (109) and after some simplifications implies (100) for some constant $\bar{b}$.

By (100), it is easy to derive

$$\|\nabla\hat{\mathcal{L}}(\hat{\mathcal{Q}}^k)\|_F \leq \|\nabla\mathcal{L}(\mathcal{Q}^k)\|_F + \sum_{i=1}^N 4\xi_i\|V_i^k - V_i^{k-1}\|_F$$
$$\leq b\sum_{i=1}^N(\|W_i^k - W_i^{k-1}\|_F + \|V_i^k - V_i^{k-1}\|_F + \|V_i^{k-1} - V_i^{k-2}\|_F)$$
$$\leq \hat{b}\|\hat{\mathcal{Q}}^k - \hat{\mathcal{Q}}^{k-1}\|_F,$$

where $b = \bar{b} + 4\max_{1\leq i\leq N}\xi_i$ and $\hat{b} = \sqrt{3N}b$. This completes the proof. ■

D.2 Proof of Theorem 5

Now we provide the detailed proof of Theorem 5 based on the above lemmas.
Proof [Proof of Theorem 5]

(a) By Lemma 19, the boundedness of $\{\mathcal{Q}^k\}$ implies the sequence $\mathcal{L}(\mathcal{Q}^k)$ is lower bounded, and so is $\hat{\mathcal{L}}(\hat{\mathcal{Q}}^k)$ by its definition (29). By Lemma 14, $\hat{\mathcal{L}}(\hat{\mathcal{Q}}^k)$ is monotonically non-increasing, therefore, $\hat{\mathcal{L}}(\hat{\mathcal{Q}}^k)$ is convergent.

(b) Again by Lemma 19, $\hat{\mathcal{Q}}^k$ is bounded, and thus there exists a subsequence $\hat{\mathcal{Q}}^{k_j}$ such that $\hat{\mathcal{Q}}^{k_j} \to \hat{\mathcal{Q}}^*$ as $j \to \infty$. Since $\hat{\mathcal{L}}$ is continuous by Assumption 1, then $\lim_{j\to\infty} \hat{\mathcal{L}}(\hat{\mathcal{Q}}^{k_j}) = \hat{\mathcal{L}}(\hat{\mathcal{Q}}^*)$. This implies the *continuity condition* in the analysis framework formulated in (Attouch et al., 2013) holds. Together with the sufficient descent (Lemma 14), relative error (Lemma 21) and Kurdyka-Łojasiewicz (Lemma 13) properties, the whole sequence convergence to a stationary point is derived via following (Attouch et al., 2013, Theorem 2.9).

(c) The $\mathcal{O}(1/K)$ rate can be easily derived by Lemma 14, Lemma 20 and Lemma 21. Specifically, by Lemma 14, it is easy to show

$$\frac{1}{K}\sum_{k=2}^{K}(\|\mathcal{W}^k - \mathcal{W}^{k-1}\|_F^2 + \|\mathcal{V}^k - \mathcal{V}^{k-1}\|_F^2) \le \frac{\hat{\mathcal{L}}(\hat{\mathcal{Q}}^1) - \hat{\mathcal{L}}(\hat{\mathcal{Q}}^*)}{aK}, \tag{110}$$

which implies

$$\frac{1}{K}\sum_{k=2}^{K}\sum_{i=1}^{N}\|\hat{V}_i^k - \hat{V}^{k-1}\|_F^2 = \frac{1}{K}\sum_{k=1}^{K-1}\|\mathcal{V}^k - \mathcal{V}^{k-1}\|_F^2 \le \frac{a\|\mathcal{V}^1 - \mathcal{V}^0\|_F^2 + (\hat{\mathcal{L}}(\hat{\mathcal{Q}}^1) - \hat{\mathcal{L}}(\hat{\mathcal{Q}}^*))}{aK}. \tag{111}$$

By (86) in Lemma 20, (110) and (111), there holds

$$\frac{1}{K}\sum_{k=2}^{K}\sum_{i=1}^{N}\|\Lambda_i^k - \Lambda_i^{k-1}\|_F^2 \le \bar{C}\cdot\frac{a\|\mathcal{V}^1 - \mathcal{V}^0\|_F^2 + (\hat{\mathcal{L}}(\hat{\mathcal{Q}}^1) - \hat{\mathcal{L}}(\hat{\mathcal{Q}}^*))}{aK}, \tag{112}$$

for some positive constant $\bar{C}$. By (110)–(112), and Lemma 21, it implies

$$\frac{1}{K}\sum_{k=2}^{K}\|\nabla\hat{\mathcal{L}}(\hat{\mathcal{Q}}^k)\|_F^2 \le \hat{C}\cdot\frac{a\|\mathcal{V}^1 - \mathcal{V}^0\|_F^2 + (\hat{\mathcal{L}}(\hat{\mathcal{Q}}^1) - \hat{\mathcal{L}}(\hat{\mathcal{Q}}^*))}{aK},$$

for some positive constant $\hat{C}$. This completes the proof. ■

References

Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parametrization. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, Long Beach, CA, USA, June 2019.

H. Attouch, J. Bolte, and B. F. Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized gauss-seidel methods. *Math. Program.*, 137(1-2):91–129, 2013.

Y. Bengio, . Simard, and P. Simard. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.*, 5(2):157–166, 1994.

J. Bochnak, M. Coste, and M.F. Roy. *Real Algebraic Geometry*, volume 3. Ergeb. Math. Grenzgeb. Springer-Verlag, Berlin, 1998.

J. Bolte, A. Daniilidis, and A. Lewis. The łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM J. Optim.*, 17: 1205–1223, 2007.

S. Boyd, N. Parikh, E. Chu, B. Peleator, and J. Eckstein. Distributed optimization and stastical learning via the alternating direction method of multipliers. *Foundations and Trends ® in Machine Learning*, 3(1):1–122, 2011.

M. Carreira-Perpinan and W. Wang. Distributed optimization of deeply nested systems. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 33, pages 10–19, Reykjavik, Iceland, 2014.

François Chollet et al. Keras. *GitHub*, 2015. URL `https://github.com/fchollet/keras`.

C. K. Chui, X. Li, and H. N. Mhaskar. Neural networks for localized approximation. *Math. Comput.*, 63:607–623, 1994.

C. K. Chui, S. B. Lin, and D. X. Zhou. Construction of neural networks for realization of localized deep learning. *Front. Appl. Math. Statis.*, 4(14), 2018.

C. K. Chui, S. B. Lin, and D. X. Zhou. Deep neural networks for rotation-invariance approximation and learning. *Anal. Appl.*, 17:737–772, 2019.

C. K. Chui, B. Zhang S. B. Lin, and D. X. Zhou. Realization of spatial sparseness by deep relu nets with massive data. *IEEE Trans. Neural Netw. Learn. Syst., in press*, 2020.

G. Cybenko. Approximation by superpositions of sigmoidal function. *Math. Control Signals Syst.*, 2(0):303–314, 1989.

C. Daniel, J. Taylor, and S. Nowozin. Learning step size controllers for robust neural network training. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 1519–1525, Phoenix, USA, February 2016.

J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, and J. Makhoul. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1370–1380, Baltimore, USA, 2014.

S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, Long Beach, CA, USA, June 2019.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011.

R. Eldan and O. Shamir. The power of depth for feedforward neural networks. In *Proceedings of the 29th Annual Conference on Learning Theory*, pages 907–940, New-York City, USA, 2016.

W. Gao, D. Goldfarb, and F. E. Curtis. Admm for multiaffine constrained optimization. *Optim. Methods Softw.*, 35(2):257–303, 2020.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256, Sardinia, Italy, May 2010.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

A. Gotmare, V. Thomas, J. Brea, and M. Jaggi. Decoupling backpropagation using constrained optimization methods. In *Proceedings of 35th International Conference on Machine Learning Workshop on Credit Assignment in Deep Learning and Deep Reinforcement Learning (ICML 2018 ECA)*, 2018.

A. Gotmare, N. S. Keskar, C. Xiong, and R. Socher. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, USA, May 2019.

F. Gu, A. Askari, and L. El Ghaoui. Fenchel lifted networks: a lagrange relaxation of neural network training. *arXiv:1811.08039*, 2018.

Z. C. Guo, L. Shi, and S. B. Lin. Realizing data features by deep nets. *IEEE Trans. Neural Netw. Learn. Syst.*, 31:4036–4048.

Z. Han, S. Yu, S. B. Lin, and D. X. Zhou. Depth selection for deep relu nets in feature extraction and generalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. doi: 10.1109/TPAMI.2020.3032422.

B. Hanin and D. Rolnick. How to start training: The effect of initialization and architecture. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS)*, Montréal, CANADA, December 2018.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1026–1034, Las Condes, Chile, December 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006.

G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, and et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Process. Mag.*, 29(6):82–97, 2012.

M. Hong, Z. Q. Luo, and M. Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM J. Optim.*, 26(1):337–364, 2016.

S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, 2015.

M. Kachuee, S. Fazeli, and M. Sarrafzadeh. Ecg heartbeat classification: a deep transferable representation. In *IEEE International Conference on Healthcare Informatics (ICHI)*, 2018.

F. Kiaee, C. Gagne, and M. Abbasi. Alternating direction method of multipliers for sparse convolutional neural networks. *arXiv:1611.01590*, 2016.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

S. Krantz and H. R. Parks. *A Primer of Real Analytic Functions (2nd ed)*. Birkhäuser, 2002.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS)*, volume 1, pages 1097–1105, Lake Tahoe, Nevada, 2012.

K. Kurdyka. On gradients of functions definable in o-minimal structures. *Annales de l'institut Fourier*, 48(3):769–783, 1998.

T. T. K. Lau, J. Zeng, B. Wu, and Y. Yao. A proximal block coordinate descent algorithm for deep neural network training. *ICLR Workshop*, 2018.

Y. LeCun. A theoretical framework for back-propagation. In *Proceedings of the 1988 Connectionist Models Summer School, CMU*, pages 21–28, Pittsburgh, USA, 1988.

Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller. *Efficient backprop. In Neural Networks: Tricks of the Trade, pages 9-50.* Springer, 1998.

Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

K. C. Lee, J. Ho, and D. J. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27:684–698, 2005.

H. W. Lin, M. Tegmark, and D. Rolnick. Why does deep and cheap learning works so well? *J. Stat. Phys.*, 168:1223–1247, 2017.

S. Lin, J. Zeng, and X. Zhang. Constructive neural network learning. *IEEE Trans. Cybern.*, 49(1):221–232, 2019.

S. B. Lin. Generalization and expressivity for deep nets. *IEEE Trans. Neural Netw. Learn. Syst.*, 30:1392–1406, 2019.

Y. Liu, Y. Xu, and W. Yin. Acceleration of primal-dual methods by preconditioning and simple subproblem procedures. *J Sci. Comput.*, 86, Article No.:21, 2021.

S. Łojasiewicz. *Ensembles Semi-analytiques.* Institut des Hautes Etudes Scientifiques, 1965.

S. Łojasiewicz. Sur la geometrie semi-et sous-analytique. *Annales de l'institut Fourier*, 43 (5):1575–1595, 1993.

Y. Lu, Z. Lai, W. K. Wong, and X. Li. Low-rank discriminative regression learning for image classification. *Neural Netw.*, 125:245–257, 2020.

H. N. Mhaskar. Approximation properties of a multilayered feedforward artificial neural network. *Adv. Comput. Math.*, 1:61–80, 1993.

H. N. Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural Comput.*, 8:164–177, 1996.

B. Mordukhovich. *Variational Analysis and Generalized Differentiation. I. Basic Theorey.* Springer, 2006.

C. Murdock, M. F. Chang, and S. Lucey. Deep component analysis via alternating direction neural networks. In *Proceedings of the 15th European Conference on Computer Vision (ECCV)*, Munich, Germany, 2018.

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 807 –814, Haifa, Israel, 2010.

P. Petersen and F. Voigtlaender. Optimal approximation of piecewise smooth function using deep relu neural networks. *Neural Netw.*, 108:296–330, 2018.

A. Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numer.*, 8: 143–195, 1999.

S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

S. Ruder. An overview of gradient descent optimization algorithms. *ArXiv e-prints, arXiv:1609.04747v2*, 2016.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

I. Safran and O. Shamir. Depth-width tradeoffs in approximating natural functions with neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, Sydney, Australia, August 2017.

T. Sainath, A. R. Mohamed, B. Kingsbury, and B. Ramabhadran. Deep convolutional neural networks for lvcsr. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013.

T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS)*, Barcelona, Spain, December 2016.

A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Banff, Canada, 2014.

C. Schwab and J. Zech. Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in uq. *Anal. Appl.*, 17:19–55, 2019.

A. Senior, G. Heigold, and K. Yang. An empirical study of learning rates in deep neural networks for speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 6724–6728, Vancouver, Canada, May 2013.

U. Shaham, A. Cloninger, and R. R. Coifman. Provable approximation properties for deep neural networks. *Appl. Comput. Harmon. Anal.*, 44:537–557, 2018.

M. Shiota. *Geometry of Subanalytic and Semialgebraic Sets.* Progress in Mathematics 150, Birkhäuser, Boston, 1997.

L. N. Smith and N. Topin. Super-convergence: very fast training of neural networks using large learning rates. In *SPIE 11006, Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, 1100612*, May 2017. doi: 10.1117/12.2520589.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, volume 28, pages 1139–1147, 2013.

G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein. Training neural networks without gradients: A scalable admm approach. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, New York, USA, 2016.

T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by running average of its recent magnitude. *COURSERA Neural networks for machine learning*, 4(2), 2012.

K. Vikraman. A deep neural network to identify foreshocks in real time. *ArXiv e-prints, arXiv:1611.08655*, 2016.

Y. Wang, W. Yin, and J. Zeng. Global convergence of admm in nonconvex nonsmooth optimization. *J. Sci. Comput.*, 78(1):29–63, 2019.

X. Wu, S. S. Du, and R. Ward. Global convergence of adaptive gradient methods for an over-parameterized neural network. *ArXiv e-prints, arXiv:1902.07111*, 2019.

Y. Xu and W. Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM J. Imaging Sci.*, 6:1758–1789, 2013.

Y. Yang, J. Sun, H. Li, and Z. Xu. Deep admm-net for compressive sensing mri. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS)*, Barcelona, Spain, December 2016.

D. Yarotsky. Error bounds for approximation with deep relu networks. *Neural Netw.*, 94: 103–114, 2017.

M. D. Zeiler. Adadelta: An adaptive learning rate method. *ArXiv e-prints, arXiv:1212.5701*, 2012.

J. Zeng, T. T. K. Lau, S. B. Lin, and Y. Yao. Global convergence of block coordinate descent in deep learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, Long Beach, CA, USA, June 2019.

J. Zeng, M. Zhang, and S. B. Lin. Fully-corrective gradient boosting with squared hinge: fast learning rates and early stopping. *arXiv preprint arXiv:2004.00179*, 2020.

Z. Zhang and M. Brand. Convergent block coordinate descent for training tikhonov regularized deep neural networks. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, Long Beach, CA, USA, December 2017.

D. X. Zhou. Deep distributed convolutional neural networks: Universality. *Anal. Appl.*, 16: 895–919, 2018.

D. X. Zhou. Universality of deep convolutional neural networks. *Appl. Comput. Harmonic. Anal.*, 48:787–794, 2020.

Z. H. Zhou, N. V. Chawla, Y. Jin, and G. J. Williams. Big data opportunities and challenges: Discussions from data analytics perspectives. *IEEE Comput. Intell. Mag.*, 9:62–74, 2014.

D. Zou and Q. Gu. An improved analysis of training over-parameterized deep neural networks. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, December 2019.