SIAM J. OPTIM.
Vol. 30, No. 2, pp. 1693–1723

MULTICOMPOSITE NONCONVEX OPTIMIZATION FOR TRAINING DEEP NEURAL NETWORKS*

YING CUI†, ZIYU HE‡, AND JONG-SHI PANG‡

Abstract. We present in this paper a novel deterministic algorithmic framework that enables the computation of a directional stationary solution of the empirical deep neural network training problem formulated as a multicomposite optimization problem with coupled nonconvexity and nondifferentiability. This is the first time to our knowledge that such a sharp kind of stationary solution is provably computable for a nonsmooth deep neural network. Allowing for arbitrary finite numbers of input samples and training layers, an arbitrary number of neurons within each layer, and arbitrary piecewise activation functions, the proposed approach combines the methods of exact penalization, majorization-minimization, gradient projection with enhancements, and the dual semismooth Newton method, each for a particular purpose in an overall computational scheme. While a routine implementation of the semismooth Newton method would be computationally expensive, we show that careful linear algebraic implementation helps to greatly reduce the computational and storage costs for problems of arbitrary dimensions. Contrary to existing stochastic approaches which provide at best very weak guarantees on the computed solutions obtained in practical implementation, our rigorous deterministic treatment provides guarantee of the stationarity properties of the computed solutions with reference to the optimization problems being solved. Numerical results from a MATLAB implementation demonstrate the effectiveness of the framework for solving reasonably sized networks with a modest number of training samples (in the low thousands).

Key words. deep neural network, nonconvexity, nondifferentiablity, exact penalization, majorization-minimization, semismooth Newton method

AMS subject classifications. 90C26, 49J52

DOI. 10.1137/18M1231559

1. Introduction.

Feed-forward deep neural networks constitute a class of function approximators for which linear combinations of inputs are filtered through activation functions in the hidden layers. Traditional practice employed smooth activation functions in both the hidden layers and the output layers, such as the sigmoid, hyperbolic, or softmax functions [18]. Recent studies show that piecewise affine activation functions, such as the rectified linear unit (ReLU), the leaky ReLU [28], the maxout unit [19], and the hard hyperbolic tangent functions [4], have some advantages over the aforementioned smooth activation functions, most notably because they make the resulting network more sparse and prevent the so-called vanishing gradient phenomenon when stochastic gradient descent type methods are adopted for training the neural networks [16]. One may refer to [23] for numerical comparisons of the deep neural networks with different activation functions. The challenge with piecewise affine activation functions is that the resulting mathematical problem to be solved is a multicomposite optimization problem with coupled nonconvexity and nondifferentiability. The nonconvexity is due to the variable linear transformation from one layer to the next where the transformation matrix is multiplied to the variable input

*Received by the editors December 10, 2018; accepted for publication (in revised form) April 17, 2020; published electronically June 18, 2020.
https://doi.org/10.1137/18M1231559
Funding: This work was based on research supported by the National Science Foundation under grant IIS-1632971 and by the Air Force Office of Scientific Research under grant FA9550-18-1-0382.
†Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis, MN 55414 (yingcui@umn.edu).
‡Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA 90089-0193 (ziyuhe@usc.edu, jongship@usc.edu).

of that layer that is determined from the previous layer. Aggregating the resulting bilinearity sequentially through all the layers leads to a highly nonconvex composite objective function to be minimized. This feature of coupled nonconvexity and nondifferentiablity raises great challenges for the optimization problem to be solved by a mathematically rigorous method with proven guarantee of convergence to a stationary solution of some sort. Indeed, as a stand-alone deterministic optimization problem, there is to date no known algorithm that can accomplish the latter stated goal.

Mathematically, the structure of a deep neural network is specified by its node descriptor, which is the tuple of positive integers $(N_0, N_1, \ldots, N_L)$ with L being the total number of layers (depth of the network) and each N_ℓ being the number of neurons at the ℓth layer. The action of a scalar neuron indexed by (j, ℓ), for $j = 1 \ldots, N_\ell$ and $\ell = 1, \ldots, L$, is described by the relation $\sigma_\ell(W_j^\ell u + w_j^\ell)$, where W_j^ℓ is an $N_{\ell-1}$-dimensional row vector of variable weights to be determined from the computation, $w_j^\ell \in \mathbb{R}$ is an additive bias, $u \in \mathbb{R}^{N_{\ell-1}}$ denotes the multivariate input to the neuron, and $\sigma_\ell : \mathbb{R} \to \mathbb{R}$ is a given activation function (such as a sigmoid or a ReLU or a univariate multipiece affine function). The outputs of layer ℓ are then fed as inputs to the next layer $(\ell+1)$ and so forth till the last layer L, where the output is a scalar; thus $N_L = 1$. Let $N_0 \triangleq d$. The overall action of the full L-layer deep network is as follows: with $u^0 \triangleq x$ being the d-dimensional input, the output is

$$\begin{array}{rcl} \psi(\theta; x) & \triangleq & u^L, \quad \text{where recursively} \\ u^\ell & = & \Xi^\ell \left(W^\ell u^{\ell-1} + w^\ell \right), \quad \ell = 1, \ldots, L. \end{array} \tag{1.1}$$

Here $\theta \triangleq \left\{ (W^\ell, w^\ell) \in \mathbb{R}^{N_\ell \times N_{\ell-1}} \times \mathbb{R}^{N_\ell} \right\}_{\ell=1}^L$, where W^ℓ is the $N_\ell \times N_{\ell-1}$ matrix with rows given by the N_ℓ vectors W_j^ℓ, each of dimension $N_{\ell-1}$, and Ξ^ℓ is a separable vector activation function that aggregates the individual neuron activations in the layer; specifically,

$$\Xi^\ell(u) = \left(\sigma_\ell(u_j) \right)_{j=1}^{N_\ell} \quad \text{for } \ell = 1, \ldots, L,$$

where for simplicity we have taken the activation function to be the same for all neurons within a layer. The successive activations of the layers constitute the composite structure of the deep network and lead to a *multicomposite* feature of the problem. The design task of the deep network consists of the determination of the parameter θ given data $\{(x^s, y_s)\}_{s=1}^S \subset \mathbb{R}^{d+1}$ by solving the empirical risk minimization problem:

$$\underset{\theta}{\textbf{minimize}} \quad \frac{1}{S} \sum_{s=1}^{S} \varphi(\psi(\theta; x^s); y_s), \tag{1.2}$$

where the bivariate function φ is a given loss function measuring the error between the model predicted output $\psi(\theta; x^s)$ of the deep network and the output y_s in the dataset corresponding to the input x^s. For simplicity, we denote $\varphi_s(t) \triangleq \varphi(t; y_s)$ in the rest of the paper.

When the activation function σ_ℓ is a nonsmooth piecewise affine function, the optimization problem in (1.2) is highly nonconvex and nondifferentiable. The stochastic (sub)gradient descent method that originated from [36], together with its innumerable variants, is the main approach for training multilayer deep neural networks in the existing literature and open source solvers, including the Google TensorFlow [1] and the PyTorch library [33]; see [3] for a nice survey on this subject. The (almost sure) convergence and the convergence rate of the stochastic gradient methods are well understood for both the convex and the smooth cases [29, 15]. Recent papers also

consider the cases where the objective function is weakly convex [7] or semialgebraic [8]. Though the overall objective function in (1.2) may satisfy the latter semialgebraic property for many choices of the activation functions, it is challenging to rigorously compute an element from the subdifferential set of the overall objective in (1.2) due to the composition of the nonsmooth functions. Thus, (sub)gradient based methods for solving (1.2) should at best be considered heuristics for treating the problem when the activation function σ_ℓ is piecewise affine. This lack of rigor is one aspect that this paper aims to avoid. One major advantage of existing (sub)gradient based software is that each iteration can be carried out extremely fast, therefore allowing tens of thousands, even more if needed, iterations to be executed using very little time. Nevertheless, as will be seen later, the quality of the objective values fluctuates greatly due to the randomness of the iterates.

There also exists a host of deterministic algorithms proposed for solving the deep neural network, such as the alternating direction method of multipliers (ADMM) [40, 42] and the block coordinate descent method [41, 25]. Numerical results demonstrate that these methods can yield good performance on standard datasets, but their theoretical convergence is either unknown or established based on smooth surrogates of the original problem when the activation function is nonsmooth. For example, in order to adhere to the assumptions in the convergence of the ADMM established in [14], one has to either smooth the original activation functions or relax the piecewise structure through smooth quadratic penalization, each of which does not solve the original problem. For more details, see [14, section 3.5]. Thus, again, there is a lack of guarantees among the deterministic methods to date for solving the problem (1.2) with piecewise affine activation functions that can compute, without smoothing, a solution of the problem with provable properties.

The goal of this paper is to introduce a novel deterministic algorithmic framework that provably computes a directional stationary solution of the empirical deep neural network training problem and to demonstrate the effectiveness of the framework. Along with this goal is the hope that the computed solution from the training problem will generalize well on hidden data for the purpose of prediction. We accomplish the stated goal based on the formulation of the problem as a multicomposite optimization problem with coupled nonconvexity and nondifferentiability to which an assortment of computational tools is applied, each with guaranteed convergence supported by good numerics. Consistent with the research focus of computational optimization, in designing an effective algorithm for solving this problem, we adhere to three basic principles that have long been the unspoken tradition of this field of applied mathematics: (a) There needs to be a sound basis and supporting convergence theory of the developed method. (b) Properties of the computed solution need to be understood. (c) Computation should be efficiently implementable and the solution should be obtainable in a reasonable amount of time without sacrificing the accuracy predicted by the theory. These principles are in contrast to many stochastic gradient based methods employed in the machine learning community where the treatment of the coupled nonconvexity and nondifferentiability properties intrinsic to the deep learning problem with piecewise activation can at best be described as heuristics. In short, our approach is deterministic, the convergence theory is sound, and the computed solution is provably stationary and exhibits reasonable generalization.

With the above guiding principles in mind, this paper contributes to the computational treatment of the deep neural network training problem in three important ways. One, we present a transformation of the original fully connected multilayer neural network into a weakly connected one via a constrained reformulation to which

an exact penalization is applied. Two, we show that the directional stationary point of the penalized problem is also a directional stationary point of the original problem, provided that the penalty parameter is larger than a threshold, the latter being the principle of exact penalization for stationary solutions, as opposed to the traditional results pertaining to minimizers that cannot be computed in practice, due to nonconvexity. Three, we design an iterative computational procedure to solve the penalized problem and establish the subsequential convergence of the generated sequence to a directional stationary point of that problem.

A bird's eye summary of the procedure is as follows. To begin, the procedure itself utilizes several optimization methods each aiming to handle the stepwise computational tasks efficiently. Each iteration of the overall procedure starts with a strongly convex majorization of the penalized objective which we minimize—the *M(ajorization)M(inimization) loop*. In principle, each resulting (convex) MM subproblem—the primal MM problem—can be solved by a host of convex programming methods. However, due to the large number of constraints therein, we adopt a dual approach that solves a simply constrained dual problem with a semismoothly differentiable objective—the *dualization loop*. For each such dual problem, we employ a semismooth Newton (SN) method aided by a gradient projection routine with heuristic enhancements that serve two purposes: for good initialization to achieve fast and accurate convergence and as a back-up procedure to ensure descent—the *SN loop*. Last, each SN step requires solving a system of linear equations, typically of fairly large size and thus demanding considerable care. This linear algebraic step is accomplished by the conjugate gradient (CG) method that is implemented with some careful matrix-theoretic calculations to achieve the much needed efficiency—the *CG loop*. To demonstrate the effectiveness of the overall solution procedure, we have carried out extensive computational experimentation; the results of our numerical findings are reported in the last section of the paper.

2. Reformulation of the multicomposite problem. In this section, we consider a more general form of multicomposite optimization problems that includes the deep neural network problem in (1.2) as a special case. As mentioned in the previous section, the challenge in solving this class of problems is due to the composition of multiple nondifferentiable and nonconvex functions. By introducing proper auxiliary variables, we can decouple the original nested composition of the element functions into weakly connected components via ℓ_1-penalization. Each of the latter components is relatively easy to analyze and manipulate. At this point, we should highlight that the use of the nondifferentiable ℓ_1-penalization (as opposed to perhaps the more common differentiable squared ℓ_2-penalization) is another noteworthy point of our procedure. We postpone the detailed discussion of this point after presenting the penalty formulation; see the paragraph after the display (2.5).

Let L be the number of building functions in the multicomposite problem. For $\ell = 1, \ldots, L$, let $\psi^\ell : \mathbb{R}^{N_{z;\ell}+N_{\ell-1}} \to \mathbb{R}^{N_\ell}$ be a given vector-valued function for some positive integers $\{N_{z;\ell}; N_{\ell-1}\}_{\ell=1}^L$ with $N_0 \triangleq d$; let $\mathcal{Z}^\ell$ be a closed convex subset of $\mathbb{R}^{N_{z;\ell}}$. For a tuple $z \triangleq (z^\ell)_{\ell=1}^L \in \mathcal{Z} \triangleq \prod_{\ell=1}^L \mathcal{Z}^\ell \subseteq \mathbb{R}^{N_{z;L+1}}$, where $N_{z;L+1} \triangleq \sum_{\ell=1}^L N_{z;\ell}$, and a vector $x \in \mathbb{R}^d$, let ψ be the L-fold composite function:

$$\psi(z;x) \triangleq \psi^L(z^L, \bullet) \circ \psi^{L-1}(z^{L-1}, \bullet) \circ \cdots \circ \psi^2(z^2, \bullet) \circ \psi^1(z^1, x). \tag{2.1}$$

In the context of the vanilla deep learning problem (1.2), the constraint set $\mathcal{Z}^\ell$ is the entire Euclidean space, $N_{z;\ell} = N_\ell \times (N_{\ell-1} + 1)$, and z^ℓ is the pair (W^ℓ, w^ℓ).

Constrained formulations of the learning problem arise when there are hard restrictions (such as nonnegativity or sparsity) on the design matrices W^ℓ. Also given is a scalar-valued function $\varphi_s : \mathbb{R}^{N_L} \to \mathbb{R}$. The multicomposite optimization problem is

$$\text{(2.2)} \qquad \underset{z \in \mathcal{Z}}{\textbf{minimize}} \quad \frac{1}{S} \sum_{s=1}^{S} \varphi_s \circ \psi(z; x^s).$$

Writing $u \triangleq (u^s)_{s=1}^S \in \mathbb{R}^{\widehat{n}}$, where $\widehat{n} \triangleq S\, N_{L+1}$ with $N_{L+1} \triangleq \sum_{\ell=1}^{L} N_\ell$, and each

$$u^s \triangleq \left(u^{s;\ell}\right)_{\ell=1}^{L} \in \mathbb{R}^{N_{L+1}} \text{ with } u^{s;\ell} \triangleq \left(u_j^{s;\ell}\right)_{j=1}^{N_\ell} \in \mathbb{R}^{N_\ell},$$

we have the equivalent formulation of (2.2) with additional equality constraints:

$$\text{(2.3)} \qquad \begin{array}{ll} \underset{z \in \mathcal{Z};\, u}{\textbf{minimize}} & \dfrac{1}{S} \displaystyle\sum_{s=1}^{S} \varphi_s(u^{s;L}) \\ \textbf{subject to} & \text{for } s = 1, \ldots, S \text{ and } \ell = 1, \ldots, L: \\ & u^{s;\ell} = \psi^\ell(z^\ell, u^{s;\ell-1}) \quad \text{with } u^{s;0} = x^s. \end{array}$$

By doing this, we transform the multicomposite objective function (2.1) into $L \times S$ equality constraints. However, directly dealing with these nonlinear equality constraints by successive convexification is problematic, even in the case where each component ψ_j^ℓ is a difference-of-convex function. This issue can be explained as follows. Consider a general difference-of-convex equality constraint $g_1(x) - g_2(x) = 0$, where both g_1 and g_2 are convex functions, assumed for simplicity to be differentiable in the following discussion. We write the equation as two inequalities:

$$g_1(x) \le g_2(x) \text{ and } g_2(x) \le g_1(x).$$

Suppose an iterate x^ν is given; we then linearize the function g_2 at x^ν in the first inequality and g_1 in the second, obtaining the following two convex inequalities:

$$\text{(2.4)} \quad g_1(x) \le g_2(x^\nu) + \nabla g_2(x^\nu)^T (x - x^\nu) \text{ and } g_2(x) \le g_1(x^\nu) + \nabla g_1(x^\nu)^T (x - x^\nu).$$

By the gradient inequality of convex functions, it is easy to deduce that any x satisfying the two inequalities (2.4) must satisfy them as equations; hence, there cannot exist a Slater point satisfying these convex inequalities strictly. Consequently, the pointwise Slater constraint qualification introduced in [32] that is needed to understand the tangent cone of the difference-of-convex constraints when these are convexified for solution by a successive convexification scheme fails, and this failure also yields pathological convex subproblems to be solved. The situation becomes more pronounced when the functions involved are nondifferentiable, such as in the case of piecewise activation functions in the deep neural network training problem.

In this work, we employ a penalty approach to deal with the equality constraints in (2.3) and add an ℓ_1-penalty term to the objective function with the penalty parameter $\rho > 0$, obtaining, with $u^{s;0} = x^s$ for all $s = 1, \ldots, L$,

$$\text{(2.5)} \quad \underset{z \in \mathcal{Z};\, u}{\textbf{minimize}} \quad \zeta_\rho(z; u) \triangleq \frac{1}{S} \sum_{s=1}^{S} \left[\varphi_s(u^{s;L}) + \rho \sum_{\ell=1}^{L} \sum_{j=1}^{N_\ell} \left| u_j^{s;\ell} - \psi_j^\ell(z^\ell, u^{s;\ell-1}) \right| \right].$$

The advantage of using the ℓ_1-penalty versus a smooth quadratic penalty is twofold. One, in the deep learning problem, each function ψ_j^ℓ is the composite of an activation function with a quadratic (in fact, bilinear) function of its arguments; the former can be linear, piecewise linear, or general nonlinear. Thus with a quadratic penalty, the resulting penalized objective function will be a nonconvex multivariate quartic polynomial even in the simplest case of an affine activation function. Instead, the penalty term in (2.5) is piecewise quadratic when a piecewise linear activation function is used; this is a significant simplification for the design of practical algorithms for solving the problem (2.2) that will be the focus in our subsequent algorithmic sections. Two, more importantly, we can establish an *exact penalty* result for a finite value of the penalty parameter ρ to recover the exact constraints in (2.3); this is a major advantage over an inexact penalty formulation in which such a parameter is required to tend to infinity for the same recovery of the constraints.

In fact, we can further establish the relationship between the directional stationary points of the penalized problem (2.5) and those of the original problem (2.3). We call a vector $\bar{x}$ a *d(irectional)-stationary point* of a multivariate directionally differentiable function ζ on a convex set $\mathcal{X}$, i.e., of the problem $\text{minimize}_{x\in\mathcal{X}}\,\zeta(x)$, if $\zeta'(\bar{x};x-\bar{x})\geq 0$ for all $x\in\mathcal{X}$, where $\zeta'(\bar{x};v)\triangleq\lim_{\tau\downarrow 0}\frac{\zeta(\bar{x}+\tau v)-\zeta(\bar{x})}{\tau}$ is the directional derivative of ζ at $\bar{x}$ in the direction v. For a bivariate directionally differentiable function $\zeta(x,y)$, we note that $\zeta'((x,y);(0,v))=\zeta(x,\bullet)'(y;v)$. For a positive integer m and a positive scalar r, let $\mathbb{B}^m(r)$ be the closed Euclidean ball in $\mathbb{R}^m$ centered at the origin and radius r.

Before stating and proving the following exact penalty result, we highlight the fact that the result pertains to a directional stationary point; thus we may term this *d-stationary exact penalty result*. This is a departure from many penalty results in the optimization literature (see, e.g., [2]) that pertain to the point in question being a minimizer of some sort, i.e., local or global. There are, however, some exact penalty results that employ directional derivatives of nonsmooth functions, such as in the classic papers [9, 10]. Another noteworthy feature of the formulation (2.5) is that this is a kind of *partial* d-stationary exact penalization [11] wherein only the equality constraints are penalized while the convex constraint $z\in\mathcal{Z}$ is kept intact. The treatment in the latter paper pertains to noncooperative games with convex player optimization problems and thus is not applicable to the pair of problems (2.3) and (2.5). Leveraging the special structure of the objective function in (2.5), we are able to obtain a sharp result that is tailored to the problems on hand. In Theorem 2.1 below, it is assumed that all the functions ψ_j^ℓ are Lipschitz continuous but need not be convex or differentiable. One drawback of the result, however, is that there is a stipulation on the d-stationary point that is the main object being analyzed; see the boundedness assumptions (S1) and (S2). In general, this conceptual requirement is difficult to verify before the penalty problem (2.5) is solved. However, when applied to the deep learning problem (1.2) with piecewise affine activation functions, these conditions (S1) and (S2) can be shown to hold under mild assumptions; see subsection 2.1.

THEOREM 2.1. *Let $\mathcal{Z}^\ell$ be a closed convex subset of $\mathbb{R}^{N_{z;\ell}}$ and let φ_s be a convex function for all $\ell=1,\ldots,L$ and $s=1,\ldots,S$. Suppose that for every $\ell=1,\ldots,L-1$ and every compact set $\mathcal{S}^\ell\subset\mathbb{R}^{N_{z;\ell}}$, the function ψ^ℓ is locally Lipschitz and directionally differentiable on $\mathbb{R}^{N_{z;\ell}+N_{\ell-1}}$; moreover, the directional derivative $\psi^\ell(z^\ell,\bullet)'(u^{\ell-1};\bullet)$ is Lipschitz continuous with a Lipschitz modulus $Lip_\ell>0$ that is independent of the pair $(z^\ell,u^{\ell-1})\in\mathcal{S}^\ell\times\mathbb{R}^{N_{\ell-1}}$; that is, for all such pairs $(z^\ell,u^{\ell-1})$ and vectors $\widehat{v}^{\ell-1}$ and $\widetilde{v}^{\ell-1}$ in $\mathbb{R}^{N_{\ell-1}}$,*

$$\| \psi^\ell(z^\ell, \bullet)'(u^{\ell-1}; \widehat{v}^{\,\ell-1}) - \psi^\ell(z^\ell, \bullet)'(u^{\ell-1}; \widetilde{v}^{\,\ell-1}) \| \le Lip_\ell \| \widehat{v}^{\,\ell-1} - \widetilde{v}^{\,\ell-1} \|.$$

Then for each scalar $r > 0$, there exists a scalar $\rho_r > 0$ such that for every $\rho > \rho_r$, if $(\bar{z}, \bar{u}) \in \mathbb{R}^{N_{z;L+1}+\widehat{n}}$ is a d-stationary point of the problem (2.5) *satisfying*

(S1) $\bar{z}^\ell \in \mathbb{B}^{N_{z;\ell}}(r)$ *for* $\ell = 1, \ldots, L-1$, *and*

(S2) $\bar{u}^L \in \mathbb{B}^{SN_L}(r)$,

then (a) $\bar{u}^{s;\ell} = \psi^\ell(\bar{z}^\ell, \bar{u}^{s;\ell-1})$ for all $s = 1, \ldots, S$ and $\ell = 1, \ldots, L$; (b) $\bar{z}$ is a d-stationary point of the problem (2.2)*; and (c) if $\bar{z}$ is a local minimizer of the problem* (2.5)*, then $\bar{z}$ is a local minimizer of* (2.2)*,*

Proof. Throughout the proof below, the d-stationary pair $(\bar{z}, \bar{u})$ always pertains to a penalty parameter ρ that is understood from the context; the choice of ρ will be specified at the end of the proof; see (2.9). The feasibility assertion (a) is the key to the other two parts. For each pair of indices (s, ℓ), let

$$I_+(s,\ell) \triangleq \left\{ j \mid \bar{u}_j^{s;\ell} > \psi_j^\ell\left(\bar{z}^\ell, \bar{u}^{s;\ell-1}\right) \right\}, \quad I_=(s,\ell) \triangleq \left\{ j \mid \bar{u}_j^{s;\ell} = \psi_j^\ell(\bar{z}^\ell, \bar{u}^{s;\ell-1}) \right\},$$
$$I_-(s,\ell) \triangleq \left\{ j \mid \bar{u}_j^{s;\ell} < \psi_j^\ell(\bar{z}^\ell, \bar{u}^{s;\ell-1}) \right\}.$$

With the above preparation, the rest of the proof is organized in three steps as follows:
1. We first prove by contradiction that there exists a positive scalar ρ_r such that $I_\pm(s, L) = \emptyset$ for all $s = 1, \ldots, S$ and $\rho > \rho_r$.
2. By an induction hypothesis, we establish similar statements for any $\ell = 1, \ldots, L-1$, thus showing the feasibility of $(\bar{z}, \bar{u})$ for all the layers.
3. We finally show the directional stationarity of $\bar{z}$ with respect to the problem (2.2).

Step 1. We notice that for any $(z, u) \in \mathcal{Z} \times \mathbb{R}^{\widehat{n}}$ with $u^{s;0} = x^s$,

(2.6)

$$0 \le \zeta_\rho'((\bar{z};\bar{u});(z-\bar{z};u-\bar{u})) = \frac{1}{S}\sum_{s=1}^S \varphi_s'\left(\bar{u}^{s;L}; u^{s;L} - \bar{u}^{s;L}\right)$$
$$+\frac{\rho}{S}\sum_{s=1}^S\sum_{\ell=1}^L \left[\begin{array}{l} \displaystyle\sum_{j\in I_+(s,\ell)} \left(u_j^{s;\ell} - \bar{u}_j^{s;\ell} - (\psi_j^\ell)'\left(\left(\bar{z}^\ell, \bar{u}^{s;\ell-1}\right); \left(z^\ell - \bar{z}^\ell; u^{s;\ell-1} - \bar{u}^{s;\ell-1}\right)\right)\right) \\ + \displaystyle\sum_{j\in I_=(s,\ell)} \left| u_j^{s;\ell} - \bar{u}_j^{s;\ell} - (\psi_j^\ell)'\left(\left(\bar{z}^\ell, \bar{u}^{s;\ell-1}\right); \left(z^\ell - \bar{z}^\ell; u^{s;\ell-1} - \bar{u}^{s;\ell-1}\right)\right)\right| \\ - \displaystyle\sum_{j\in I_-(s,\ell)} \left(u_j^{s;\ell} - \bar{u}_j^{s;\ell} - (\psi_j^\ell)'\left(\left(\bar{z}^\ell, \bar{u}^{s;\ell-1}\right); \left(z^\ell - \bar{z}^\ell; u^{s;\ell-1} - \bar{u}^{s;\ell-1}\right)\right)\right) \end{array}\right].$$

For a fixed but arbitrary $\bar{s} \in \{1, \ldots, S\}$, let $z^\ell = \bar{z}^\ell$ and $u^{s;\ell-1} = \bar{u}^{s;\ell-1}$ for all $\ell = 1, \ldots, L$ and $s = 1, \ldots, S$; further let $u^{s;L} = \bar{u}^{s;L}$ for all $s \ne \bar{s}$. The above inequality then yields

(2.7)

$$0 \le \varphi_{\bar{s}}'(\bar{u}^{\bar{s};L}; u^{\bar{s};L} - \bar{u}^{\bar{s};L}) + \rho \left[\begin{array}{l} \displaystyle\sum_{j\in I_+(\bar{s},L)} \left(u_j^{\bar{s};L} - \bar{u}_j^{\bar{s};L}\right) \\ + \displaystyle\sum_{j\in I_=(\bar{s},L)} \left| u_j^{\bar{s};L} - \bar{u}_j^{\bar{s};L}\right| - \displaystyle\sum_{j\in I_-(s,L)} \left(u_j^{s;\ell} - \bar{u}_j^{s;L}\right) \end{array}\right]$$
$$= \varphi_{\bar{s}}'\left(\bar{u}^{\bar{s};L}; u^{\bar{s};L} - \bar{u}^{\bar{s};L}\right) + \rho \| \bullet - \psi^L(\bar{z}^L, \bar{u}^{\bar{s};L-1}) \|_1'\left(\bar{u}^{\bar{s};L}; u^{\bar{s};L} - \bar{u}^{\bar{s};L}\right).$$

We next show that there exists a scalar $\rho_r > 0$ such that for all $\rho > \rho_r$, both index sets $I_\pm(s, L)$ are empty for all $s = 1, \ldots, S$. Assume otherwise that for any $\bar{\rho}$ (depending

on r), there exists $\rho > \bar{\rho}$ such that $I_+(s,L) \neq \emptyset$ (one can follow similar arguments for the case that $I_-(s,L) \neq \emptyset$). Then an index $\bar{j} \in I_+(\bar{s},L)$ exists for some $\bar{s} \in \{1,\ldots,S\}$ so that $\bar{u}_{\bar{j}}^{\bar{s};L} > \psi_{\bar{j}}^{L}(\bar{z}^L, \bar{u}^{s;L-1})$. Let $\varepsilon > 0$ be such that the vector $\widehat{u}_j^{\bar{s};L} \triangleq \begin{cases} \bar{u}_{\bar{j}}^{\bar{s};L} - \varepsilon \text{ if } j = \bar{j}, \\ \bar{u}_j^{\bar{s};L} \quad \text{if } j \neq \bar{j} \end{cases}$ belongs to the ball $\mathbb{B}^{N_L}(2r)$. Letting $z^\ell = \bar{z}^\ell$ and $\widehat{u}^{s;\ell-1} = \bar{u}^{s;\ell-1}$ for all $\ell = 1,\ldots,L$ and $s = 1,\ldots,S$ as above, as well as $\widehat{u}^{s;L} = \bar{u}^{s;L}$ for all $s \neq \bar{s}$, we deduce from (2.7),

$$0 \leq \varphi'_{\bar{s}}\left(\bar{u}^{\bar{s};L}; \widehat{u}^{\bar{s};L} - \bar{u}^{\bar{s};L}\right) - \rho\,\varepsilon.$$

Since φ_s is convex for any s, there exists a Lipschitz constant $\mathrm{Lip}_\varphi > 0$ dependent only on the scalar r and independent of s (since there are only finitely many of these integers) and the vector $\bar{u}^{\bar{s};L} \in \mathbb{B}^{N_L}(r)$ (by the compactness of the ball) such that

$$|\,\varphi'_{\bar{s}}(\bar{u}^{\bar{s};L}; v)\,| \leq \mathrm{Lip}_\varphi \,\|\, v \,\| \quad \forall\, v \in \mathbb{R}^{N_L}.$$

Hence provided that $\rho > \mathrm{Lip}_\varphi$, we obtain a contradiction to the existence of the pair of indices $(\bar{s},\bar{j})$ such that $\bar{u}_{\bar{j}}^{\bar{s};L} > \psi_{\bar{j}}^{L}(\bar{z}^L, \bar{u}^{s;L-1})$. Hence $I_\pm(s,L) = \emptyset$ for all $s = 1,\ldots,S$ and $\rho > \mathrm{Lip}_\varphi$.

Step 2. By an induction hypothesis, we may assume that for every $\ell = L, L-1,\ldots,\bar{\ell}$, there exists $\rho_\ell > 0$ such that for all $\rho > \rho_\ell$ and all $s = 1,\ldots,S$, we have $\bar{u}^{s;\ell} = \psi^\ell(\bar{z}^\ell, \bar{u}^{s;\ell-1})$. We next show that this continues to hold for $\ell = \bar{\ell} - 1$ by proceeding as above. The inequality (2.6) can be written as that for any $(z,u) \in \mathcal{Z} \times \mathbb{R}^{\widehat{n}}$ with $u^{s;0} = x^s$,

(2.8)

$$0 \leq \zeta'_\rho((\bar{z};\bar{u});(z-\bar{z};u-\bar{u})) = \frac{1}{S}\sum_{s=1}^{S} \varphi_s'\left(\bar{u}^{s;L}; u^{s;L} - \bar{u}^{s;L}\right)$$

$$+\frac{\rho}{S}\sum_{s=1}^{S}\sum_{\ell=\bar{\ell}}^{L}\sum_{j=1}^{N_\ell}\left|\, u_j^{s;\ell} - \bar{u}_j^{s;\ell} - \left(\psi_j^\ell\right)'\left(\left(\bar{z}^\ell, \bar{u}^{s;\ell-1}\right); \left(z^\ell - \bar{z}^\ell; u^{s;\ell-1} - \bar{u}^{s;\ell-1}\right)\right)\right|$$

$$+\frac{\rho}{S}\sum_{s=1}^{S}\sum_{\ell=1}^{\bar{\ell}-1}\left[\begin{array}{l} \displaystyle\sum_{j\in I_+(s,\ell)}\left(u_j^{s;\ell} - \bar{u}_j^{s;\ell} - (\,\psi_j^\ell)'\left(\left(\bar{z}^\ell, \bar{u}^{s;\ell-1}\right); \left(z^\ell - \bar{z}^\ell; u^{s;\ell-1} - \bar{u}^{s;\ell-1}\right)\right)\right) \\ \displaystyle + \sum_{j\in I_=(s,\ell)}\left|\, u_j^{s;\ell} - \bar{u}_j^{s;\ell} - (\,\psi_j^\ell)'\left(\left(\bar{z}^\ell, \bar{u}^{s;\ell-1}\right); \left(z^\ell - \bar{z}^\ell; u^{s;\ell-1} - \bar{u}^{s;\ell-1}\right)\right)\right| \\ \displaystyle - \sum_{j\in I_-(s,\ell)}\left(u_j^{s;\ell} - \bar{u}_j^{s;\ell} - (\,\psi_j^\ell)'\left(\left(\bar{z}^\ell, \bar{u}^{s;\ell-1}\right); (z^\ell - \bar{z}^\ell; u^{s;\ell-1} - \bar{u}^{s;\ell-1})\right)\right) \end{array}\right].$$

With $z^\ell = \bar{z}^\ell$ and $u^{s;\ell} = \bar{u}^{s;\ell}$ for all $\ell = 1,\ldots,\bar{\ell}-2$ and $s = 1,\ldots,S$ and proper choice of $u^{s;\ell}$ for all $\ell = \bar{\ell}-1,\cdots,L$ and all s, we can similarly establish there exists a constant $\rho_{\bar{\ell}-1} > 0$ such that for all $\rho > \rho_{\bar{\ell}-1}$, $\bar{u}^{\bar{\ell}-1} = \psi^{\bar{\ell}-1}(\bar{z}^{\bar{\ell}-1}; \bar{u}^{\bar{\ell}-2})$. Indeed, suppose that exists a pair $(\bar{s},\bar{j})$ such that $\bar{u}_{\bar{j}}^{\bar{\ell}-1} > \psi_{\bar{j}}^{\bar{\ell}-1}(\bar{z}^{\bar{\ell}-1}; \bar{u}^{\bar{\ell}-2})$. We define a tuple $\widehat{u}$ as follows:

$$\widehat{u}_j^{\bar{s};\bar{\ell}-1} \triangleq \begin{cases} \bar{u}_{\bar{j}}^{\bar{s};\bar{\ell}-1} - \varepsilon \text{ if } j = \bar{j}, \\ \bar{u}_j^{\bar{s};\bar{\ell}-1} \quad \text{if } j \neq \bar{j} \end{cases}$$

for a positive scalar ε, $\widehat{u}^{\,s;\bar{\ell}-1} = \bar{u}^{s;\bar{\ell}-1}$ for all $s \neq \bar{s}$, and recursively,

$$\widehat{u}_j^{s;\ell} = \begin{cases} \bar{u}_j^{\bar{s};\ell} - (\,\psi_j^\ell)'((\bar{z}^\ell, \bar{u}^{\bar{s};\ell-1}); (z^\ell - \bar{z}^\ell; \widehat{u}^{\bar{s};\ell-1} - \bar{u}^{\bar{s};\ell-1}) & \text{if } s = \bar{s}, \\ \bar{u}_j^{s;\ell} & \text{if } s \neq \bar{s} \end{cases}$$

for all $\ell = \bar{\ell}, \ldots, L$ and $j = 1, \ldots, N_\ell$. With this choice of the components of $\widehat{u}$, (2.8) becomes

$$\begin{aligned} 0 &\leq \varphi_{\bar{s}}'\left(\bar{u}^{s;L}; \widehat{u}^{\bar{s};L} - \bar{u}^{\bar{s};L}\right) - \rho\,\varepsilon \leq \mathrm{Lip}_\varphi \left\| \widehat{u}^{\bar{s};L} - \bar{u}^{\bar{s};L} \right\| - \rho\,\varepsilon \\ &= \mathrm{Lip}_\varphi \left\| \psi^L(\bar{z}^L, \bullet)'\left(\bar{u}^{\bar{s};L-1}; \widehat{u}^{\bar{s};L-1} - \bar{u}^{\bar{s};L-1}\right) \right\| - \rho\,\varepsilon \\ &\leq \mathrm{Lip}_\varphi \mathrm{Lip}_L \left\| \widehat{u}^{\bar{s};L-1} - \bar{u}^{\bar{s};L-1} \right\| - \rho\,\varepsilon \\ &= \mathrm{Lip}_\varphi \mathrm{Lip}_L \left\| \psi^{L-1}(\bar{z}^{L-1}, \bullet)'\left(\bar{u}^{\bar{s};L-2}; \widehat{u}^{\bar{s};L-2} - \bar{u}^{\bar{s};L-2}\right) \right\| - \rho\,\varepsilon \\ &\leq \cdots \leq \mathrm{Lip}_\varphi \mathrm{Lip}_L \cdots \mathrm{Lip}_{\bar{\ell}} \left\| \widehat{u}^{\bar{s};\bar{\ell}-1} - \bar{u}^{\bar{s};\bar{\ell}-1} \right\| - \rho\,\varepsilon = \left[\mathrm{Lip}_\varphi \mathrm{Lip}_L \cdots \mathrm{Lip}_{\bar{\ell}} - \rho\right] \varepsilon. \end{aligned}$$

Thus, if $\rho > \rho_{\bar{\ell}-1} \triangleq \mathrm{Lip}_\varphi \mathrm{Lip}_L \cdots \mathrm{Lip}_{\bar{\ell}}$, we deduce a contradiction. In summary, we have therefore shown that for

$$\rho > \rho_r \triangleq \max\left\{ \max(\mathrm{Lip}_\varphi, 1) \max(\mathrm{Lip}_L, 1) \cdots \max(\mathrm{Lip}_1, 1) \right\}, \tag{2.9}$$

then any d-stationary solution $(\bar{z}, \bar{u})$ of the problem (2.5) satisfying conditions (S1) and (S2) must satisfy the constraints of (2.3); i.e., $\bar{u}^{s;\ell} = \psi^\ell(\bar{z}^\ell, \bar{u}^{s;\ell-1})$ for all $s = 1, \ldots, S$ and $\ell = 1, \ldots, L$. This establishes assertion (a).

Step 3. With the above feasibility condition, the stationarity condition (2.6) becomes

$$\begin{aligned} 0 \;\leq\; & \frac{1}{S}\sum_{s=1}^{S} \varphi_s'(\bar{u}^{s;L}, u^{s;L} - \bar{u}^{s;L}) \\ & + \frac{\rho}{S}\sum_{s=1}^{S}\sum_{\ell=1}^{L}\sum_{j=1}^{N_\ell} \left| u_j^{s;\ell} - \bar{u}_j^{s;\ell} - (\,\psi_j^\ell\,)'\left(\left(\bar{z}^\ell, \bar{u}^{s;\ell-1}\right); \left(z^\ell - \bar{z}^\ell; u^{s;\ell-1} - \bar{u}^{s;\ell-1}\right)\right) \right| \end{aligned} \tag{2.10}$$

for all $(z, u) \in \mathcal{Z} \times \mathbb{R}^{\widehat{n}}$ with $u^{s;0} = x^s$ for all $s = 1, \ldots, S$. Starting at $\ell = 1$ and recursively letting

$$u^{s;\ell} \;=\; \bar{u}_j^{s;\ell} - (\,\psi^\ell\,)'\left(\left(\bar{z}^\ell, \bar{u}^{s;\ell-1}\right); \left(z^\ell - \bar{z}^\ell, u^{s;\ell-1} - \bar{u}^{s;\ell-1}\right)\right)$$

for all $\ell = 1, \ldots, L$ and $s = 1, \ldots, S$, we deduce from (2.10) the simple inequality

$$0 \;\leq\; \frac{1}{S}\sum_{s=1}^{S} \varphi_s'(\bar{u}^{s;L}, u^{s;L} - \bar{u}^{s;L}). \tag{2.11}$$

Each difference $u^{s;L} - \bar{u}^{s;L}$ can be substituted in terms of the directional derivatives at the lower layers $\ell = L-1, L-2, \ldots, 1$ successively as follows:

$$\begin{aligned} u^{s;1} - \bar{u}^{s;1} \;&=\; \psi^1(\bullet, x^s)'(\bar{z}^1; z^1 - \bar{z}^1), \\ u^{s;2} - \bar{u}^{s;2} \;&=\; (\,\psi^2\,)'((\bar{z}^2, \bar{u}^{s;1}); (z^2 - \bar{z}^2, u^{s;1} - \bar{u}^{s;1})) \\ &=\; (\,\psi^2\,)'((\bar{z}^2, \psi^1(\bar{z}^1, x^s)); (z^2 - \bar{z}^2, \psi^1(\bullet, x^s)'(\bar{z}^1; z^1 - \bar{z}^1)) \\ &=\; (\,\psi^2(z^2, \bullet) \circ \psi^1(z^1, x^s)\,)'((\bar{z}^2, \bar{z}^1); (z^2 - \bar{z}^2, z^1 - \bar{z}^1)), \end{aligned}$$

where the directional derivative in the last expression is taken with respect to (z^2, z^1) at the pair $(\bar{z}^2, \bar{z}^1)$ along the direction $(z^2 - \bar{z}^2, z^1 - \bar{z}^1)$. Continuing on,

$$\begin{aligned} u^{s;3} - \bar{u}^{s;3} \;&=\; (\,\psi^3\,)'((\bar{z}^3, \bar{u}^{s;2}); (z^3 - \bar{z}^3, u^{s;2} - \bar{u}^{s;2})) \\ &= (\psi^3)'((\bar{z}^3, \psi^2(\bar{z}^2; \psi^1(\bar{z}^1, x^s)); (z^3 - \bar{z}^3, (\psi^2(z^2, \bullet) \circ \psi^1(z^1, x^s))'((\bar{z}^2, \bar{z}^1); \\ &\qquad (z^2 - \bar{z}^2, z^1 - \bar{z}^1))) \\ &=\; (\,\psi(z^3, \bullet) \circ \psi^2(z^2, \bullet) \circ \psi^1(z^1, x^s)\,)'((\bar{z}^3, \bar{z}^2, \bar{z}^1); (z^3 - \bar{z}^3, z^2 - \bar{z}^2, z^1 - \bar{z}^1)), \end{aligned}$$

where the directional derivative in the last expression is taken with respect to the triplet (z^3, z^2, z^1) at $(\bar{z}^3, \bar{z}^2, \bar{z}^1)$ along the direction $(z^3 - \bar{z}^3, z^2 - \bar{z}^2, z^1 - \bar{z}^1)$. Continuing in this fashion, we can show that $u^{s;L} - \bar{u}^{s;L} = \psi(\bullet, x^s)'(\bar{z}; z - \bar{z})$. Thus the inequality (2.11) yields that $\bar{z}$ is a d-stationary point of the problem (2.2). This establishes assertion (b).

If $\bar{z}$ is a local minimizer of (2.5), then it must be d-stationary to the same problem. Hence by (a), $\bar{z}$ is feasible to (2.2). With this feasibility established, the local minimizing property $\bar{z}$ for (2.2) follows readily. ∎

If the sets $\mathcal{Z}^\ell$ are bounded, then condition (S1) holds trivially provided that the scalar r is sufficiently large. Furthermore, if the sets

$$\left\{ u^{s;L} \in \mathbb{R}^{N_L} \mid \varphi_s(u^{s;L}) \leq \varphi_s(\psi^L(\bar{z}^L; \bar{u}^{s;L-1})) \right\} \tag{2.12}$$

are bounded for all s, then condition (S2) holds. Indeed, it follows from (2.7) that

$$\bar{u}^{\bar{s};L} \in \operatorname*{argmin}_{u^L \in \mathbb{R}^{N_L}} \left\{ \varphi_s(u^L) + \rho \, \| \, u^L - \psi^L(\bar{z}^L, \bar{u}^{\bar{s};L-1}) \|_1 \right\},$$

because the minimand is a convex function of u^L. Thus $\varphi(\bar{u}^{\bar{s};L}) \leq \varphi(\psi^L(\bar{z}^L, \bar{u}^{\bar{s};L-1}))$.

We note that in the proof above, the choice of the various tuples $\widehat{u}$ does not involve the variable z. This remark is important because the theorem remains valid when the objective function is augmented by a convex function of the tuple z; such an augmentation will not invalidate the proof.

2.1. Specialization to (1.2) under (1.1). We discuss the specialization of Theorem 2.1 to the case where, with $z^\ell \triangleq (W^\ell, w^\ell)$,

$$\psi_j^\ell(z^\ell, u^{\ell-1}) = \sigma_\ell(W_j^\ell u^{\ell-1} + w_j^\ell), \quad j = 1, \ldots, N_\ell;$$

each (univariate) activation function σ_ℓ is a piecewise affine function (this is the principal class of activation functions to be studied in the rest of this paper). As such, $\psi_j^\ell(z^\ell, \bullet)$ is a piecewise quadratic function in its arguments; moreover

$$\psi_j^\ell(z^\ell, \bullet)'(u^{\ell-1}; v^{\ell-1}) = \sigma_\ell'(u_j^\ell; W_j^\ell v^{\ell-1}), \quad \text{where } u_j^\ell = W_j^\ell u^{\ell-1} + w_j^\ell.$$

Since a piecewise affine function is globally Lipschitz continuous, it follows that the Lipschitz assumption of this directional derivative holds, provided the W^ℓ belongs to a bounded set; indeed, for some constant $\mathrm{Lip}_\psi > 0$, we have

$$\left| \, \psi_j^\ell(z^\ell, \bullet)'(u^{\ell-1}; \widehat{v}^{\ell-1}) - \psi_j^\ell(z^\ell, \bullet)'(u^{\ell-1}; \widetilde{v}^{\ell-1}) \, \right| \leq \mathrm{Lip}_\psi \, \| \, W^\ell \, \| \, \| \, \widehat{v}^{\ell-1} - \widetilde{v}^{\ell-1} \, \| .$$

Moreover, if the range of σ_ℓ is bounded (which holds for, e.g., the hard sigmoid function), then the set (2.12) is contained in the level set $\left\{ u^{s;L} \in \mathbb{R}^{N_L} \mid \varphi_s(u^{s;L}) \leq \eta \right\}$ for some constant η, which is bounded when φ_s has bounded level sets. Summarizing this discussion, we obtain the following corollary of Theorem 2.1 specialized to

$$\begin{aligned} \underset{\{W^\ell, w^\ell\}_{\ell=1}^L}{\textbf{minimize}} \quad & \frac{1}{S} \sum_{s=1}^{S} \varphi(\psi(\theta; x^s); y_s) \\ \text{subject to} \quad & \left(W^\ell, w^\ell \right) \in \mathcal{W}^\ell \times \Omega^\ell, \quad \ell = 1, \ldots, L, \end{aligned} \tag{2.13}$$

when each constraint set $\mathcal{W}^\ell \subseteq \mathbb{R}^{N_\ell \times N_{\ell-1}}$ of the design matrix W^ℓ is bounded and the set $\Omega^\ell \subseteq \mathbb{R}^{N_\ell}$ is a closed convex set. The penalized problem of (2.13) is

$$\begin{aligned} &\textbf{minimize over } \left\{W^\ell,\, w^\ell,\, (u^{s;\ell})_{s=1}^S \right\}_{\ell=1}^L \text{ with } u^{s;0} = x^s,\ s = 1,\dots,S, \\ &\frac{1}{S}\sum_{s=1}^S \left[\varphi(u^{s;L}; y_s) + \rho \sum_{\ell=1}^L \sum_{j=1}^{N_\ell} \left| u_j^{s;\ell} - \sigma_\ell(W_j^\ell u^{s;\ell-1} + w_j^\ell) \right| \right] \\ &\text{subject to} \quad \left(W^\ell, w^\ell\right) \in \mathcal{W}^\ell \times \Omega^\ell,\ \ell = 1,\dots,L. \end{aligned} \tag{2.14}$$

Proof of the following corollary is omitted; it suffices to note that even though the constraint set Ω^ℓ of w^ℓ may not be bounded, the proof of Theorem 2.1 can be followed.

COROLLARY 2.2. *Suppose that* (a) $\mathcal{W}^\ell$ *and* Ω^ℓ *are closed convex sets with* $\mathcal{W}^\ell$ *being bounded,* (b) σ_L *is a bounded piecewise affine function, and* (c) $\varphi(\bullet; y_s)$ *is a convex function with bounded level sets for all* $s = 1,\dots,S$. *Then there exists a scalar* $\bar\rho > 0$ *such that for all* $\rho > \bar\rho$, *if the tuple* $\left\{ \bar W^\ell, \bar w^\ell, (\bar u^{s;\ell})_{s=1}^S \right\}_{\ell=1}^L$ *is a d-stationary point (local minimizer) of* (2.14), *then the tuple* $\left\{ \bar W^\ell, \bar w^\ell \right\}_{\ell=1}^L$ *is a d-stationary solution (local minimizer, resp.) of* (2.13),

We may derive variations of the above corollary. In what follows, we discuss a version of the corollary in which the boundedness of the activation function σ_L and the boundedness of the sets $\mathcal{W}^\ell$ can be removed as a result of an algorithmic design for solving the deep neural network problem. In practical applications, the last layer L typically does not involve activation; i.e., σ_L is the (unbounded) identity function, thus it is important to have a result that handles this important case. Moreover, sparsity control of the design matrices W^ℓ is not uncommon where some norm (such as ℓ_1) is used as the surrogate sparsity measure. Alternatively, some regularization on these matrices is sometimes employed to avoid overfitting. Together, we may consider the following regularized problem:

$$\begin{aligned} &\textbf{minimize over } \left\{W^\ell,\, w^\ell,\, (u^{s;\ell})_{s=1}^S \right\}_{\ell=1}^L \text{ with } u^{s;0} = x^s,\ s = 1,\dots,S: \\ &\frac{1}{S}\sum_{s=1}^S \left[\varphi(u^{s;L}; y_s) + \rho \sum_{\ell=1}^L \sum_{j=1}^{N_\ell} \left| u_j^{s;\ell} - \sigma_\ell\left(W_j^\ell u^{s;\ell-1} + w_j^\ell\right) \right| \right] + \sum_{\ell=1}^L p_\ell(W^\ell) \\ &\text{subject to} \quad \left(W^\ell, w^\ell\right) \in \mathcal{W}^\ell \times \Omega^\ell,\ \ell = 1,\dots,L, \end{aligned} \tag{2.15}$$

where each p_ℓ is a convex coercive function, e.g., a norm. If for each ρ one applies the MM algorithm to be introduced in the next section to the problem (2.15), starting at a tuple $\{\widetilde W^\ell, \widetilde w^\ell, (\widetilde u^{s;\ell})_{s=1}^S\}_{\ell=1}^L$ that satisfies $\widetilde u_j^{s;\ell} = \sigma_\ell(\widetilde W_j^\ell \widetilde u^{s;\ell-1} + \widetilde w_j^\ell)$ for all (s,ℓ,j), then one can obtain (see Theorem 3.1) in the limit of the algorithm a d-stationary tuple $\left\{ \bar W^\ell, \bar w^\ell, (\bar u^{s;\ell})_{s=1}^S \right\}_{\ell=1}^L$ of (2.15) that satisfies

$$\begin{aligned} &\frac{1}{S}\sum_{s=1}^S \varphi(\bar u^{s;L}; y_s) + \sum_{\ell=1}^L p_\ell(\bar W^\ell) \\ &\le \frac{1}{S}\sum_{s=1}^S \left[\varphi(\bar u^{s;L}; y_s) + \rho \sum_{\ell=1}^L \sum_{j=1}^{N_\ell} \left| \bar u_j^{s;\ell} - \sigma_\ell\left(\bar W_j^\ell \bar u^{s;\ell-1} + \bar w_j^\ell\right) \right| \right] + \sum_{\ell=1}^L p_\ell(\bar W^\ell) \\ &\le \frac{1}{S}\sum_{s=1}^S \varphi(\widetilde u^{s;L}; y_s) + \sum_{\ell=1}^L p_\ell(\widetilde W^\ell), \end{aligned}$$

where the right-hand side is a constant dependent on the starting tuple. If each $\varphi(\bullet; y_s)$ is bounded below, then the coercivity of each p_ℓ implies that $\{\bar{W}^\ell\}_{\ell=1}^L$ is bounded; further, if condition (c) in Corollary 2.2 continues to be in place, then we can obtain a similar corollary, but without the boundedness of $\mathcal{W}^\ell$ and that of σ_L, by the same proof as that of Theorem 2.1. Details are omitted.

3. The majorization-minimization algorithm. In this and the subsequent sections, we focus on the computational methods for solving the (unconstrained) penalized version (2.5) of the deep learning problem (1.2) with piecewise affine activation functions, i.e., the variable z^ℓ in (2.5) is given by (W^ℓ, w^ℓ) and the function $\psi_j^\ell(z^\ell, u^{s,\ell-1})$ in (2.5) equals to $\sigma_\ell\left(W_j^\ell u^{s;\ell-1} + w_j^\ell\right)$ for any $s = 1, \ldots, S$, $j = 1, \ldots, N_\ell$, and $l = 1, \ldots, L$, where each σ_ℓ has the form

$$\sigma_\ell(t) \triangleq \max_{1\le i\le k_1} \left(a_{i;\ell}\, t + \alpha_{i;\ell}\right) - \max_{1\le i\le k_2} \left(b_{i;\ell}\, t + \beta_{i;\ell}\right), \quad t \in \mathbb{R},$$

for some given scalars $\{(a_{i;\ell}, \alpha_{i;\ell})_{i=1}^{k_1}, (b_{i;\ell}, \beta_{i;\ell})_{i=1}^{k_2}\}_{\ell=1}^L$. With no loss of generality, the positive integers k_1 and k_2 are taken to be the same for all layers. Based on a result of Scholtes [39], every piecewise affine function can be expressed in the above difference-max form. The overall penalized problem is given by

$$\begin{aligned} \underset{W,\,w,\,u}{\textbf{minimize}} \quad & \Psi_\rho(W, w; u) \\ & \triangleq \frac{1}{S}\sum_{s=1}^{S}\left[\varphi_s(u^{s;L}) + \rho\sum_{\ell=1}^{L}\sum_{j=1}^{N_\ell}\left|\, u_j^{s;\ell} - \sigma_\ell\left(W_j^\ell u^{s;\ell-1} + w_j^\ell\right)\right|\right]. \end{aligned} \tag{3.1}$$

In principle, the derivation in the rest of the paper allows (friendly) constraints on the pair of primary variables (W, w); however, for simplicity, we consider the unconstrained case without any such constraints.

The key to derive a convex function majorizing the objective $\Psi_\rho(W, w; u)$ is via such a majorant for each summand which has the following max form:

$$\left|u_j^{s;\ell} - \sigma_\ell\left(W_j^\ell u^{s;\ell-1} + w_j^\ell\right)\right| = \max\left(u_j^{s;\ell} - \sigma_\ell\left(W_j^\ell u^{s;\ell-1} + w_j^\ell\right),\, \sigma_\ell\left(W_j^\ell u^{s;\ell-1} + w_j^\ell\right) - u_j^{s;\ell}\right).$$

There are several ways to obtain the convex majorant. In what follows, we provide the details of one majorization which is the basis for the overall algorithm that we implement in our numerical study; see section 5. Other majorizations are possible but we have not explored them. In essence, the key thing in all majorizations is the proper handling of the bilinear term $W_j^\ell u^{s;\ell-1}$ in conjunction with the piecewise linearity of σ_ℓ. There are two main steps involved in the majorization presented below. The first step is the treatment of the product term within the activation function, i.e.,

$$\begin{aligned} v_j^{s;\ell} \triangleq\ & W_j^\ell u^{s;\ell-1} + w_j^\ell = \frac{1}{4}\left[\left\|\left(W_j^\ell\right)^T + u^{s;\ell-1}\right\|^2 - \left\|\left(W_j^\ell\right)^T - u^{s;\ell-1}\right\|^2\right] + w_j^\ell \\ =\ & \frac{1}{4}\left[\left\|V_j^{+;s;\ell}\right\|^2 - \left\|V_j^{-;s;\ell}\right\|^2\right] + w_j^\ell, \quad \text{where } V_j^{\pm;s;\ell} \triangleq \left(W_j^\ell\right)^T \pm u^{s;\ell-1}. \end{aligned} \tag{3.2}$$

The auxiliary variables $V_j^{\pm;s;\ell}$ are useful in the description of the algorithm to be described later but will not be used as extra variables in the algorithm; only their values at the current iterate are needed.

The second step is to upper and lower bound the composite activation function,

$$\sigma_\ell\left(W_j^\ell u^{s;\ell-1} + w_j^\ell\right) = \max_{1\le i\le k_1}\left(a_{i;\ell}\, v_j^{s;\ell} + \alpha_{i;\ell}\right) - \max_{1\le i\le k_2}\left(b_{i;\ell}\, v_j^{s;\ell} + \beta_{i;\ell}\right),$$

by deriving upper and lower bounds for both $a_{i;\ell}\, v_j^{s;\ell}$ and $b_{i;\ell}\, v_j^{s;\ell}$. Notice that there are no priori bounds on the auxiliary variables $v_j^{s;\ell}$ which is the sum of w_j^ℓ and the bilinear term $W_j^\ell u^{s;\ell-1}$; since no given bounds are known, one cannot employ the McCormick-type bounds in global optimization [13]. Instead, the derivation below is fairly elementary. In order for our procedure to be applicable to arbitrary piecewise functions σ_ℓ, we allow the coefficients $a_{i;\ell}$ and $b_{i;\ell}$ to be negative. We start by letting $a_{i;\ell} = a_{i;\ell}^+ - a_{i;\ell}^-$ and $b_{i;\ell} = b_{i;\ell}^+ - b_{i;\ell}^-$ be the decomposition of these scalars into the difference of their nonnegative and nonpositive parts, respectively. For a given tuple

$$\left\{ W^{\nu;\ell}; w^{\nu;\ell}; \left(V^{\pm;\nu;s;\ell};\, u^{\nu;s;\ell} \right)_{s=1}^{S} \right\}_{\ell=1}^{L} \tag{3.3}$$

available at the beginning of an iteration $\nu+1$ of the MM algorithm to be described later, we can derive the following desired upper and lower bounds:

$$\begin{aligned}
& a_{i;\ell}\, v_j^{s;\ell} + \alpha_{i;\ell} \\
& \le a_{i;\ell}\, w_j^\ell + \alpha_{i;\ell} + \frac{1}{4}\left[a_{i;\ell}^+ \left\| V_j^{+;s;\ell} \right\|^2 + a_{i;\ell}^- \left\| V_j^{-;s;\ell} \right\|^2 \right] + \frac{1}{4}\left[a_{i;\ell}^+ \left\| V_j^{-;\nu;s;\ell} \right\|^2 \right. \\
& \quad \left. + a_{i;\ell}^- \left\| V_j^{+;\nu;s;\ell} \right\|^2 \right] - \frac{1}{2}\left[a_{i;\ell}^+ \left(V_j^{-;\nu;s;\ell} \right)^T V_j^{-;s;\ell} + a_{i;\ell}^- \left(V_j^{+;\nu;s;\ell} \right)^T V_j^{+;s;\ell} \right]
\end{aligned}$$

and

$$\begin{aligned}
& a_{i;\ell}\, v_j^{s;\ell} + \alpha_{i;\ell} \\
& \ge a_{i;\ell}\, w_j^\ell + \alpha_{i;\ell} - \frac{1}{4}\left[a_{i;\ell}^+ \left\| V_j^{+;\nu;s;\ell} \right\|^2 + a_{i;\ell}^- \left\| V_j^{-;\nu;s;\ell} \right\|^2 \right] - \frac{1}{4}\left[a_{i;\ell}^+ \left\| V_j^{-;s;\ell} \right\|^2 \right. \\
& \quad \left. + a_{i;\ell}^- \left\| V_j^{+;s;\ell} \right\|^2 \right] + \frac{1}{2}\left[a_{i;\ell}^+ \left(V_j^{+;\nu;s;\ell} \right)^T V_j^{+;s;\ell} + a_{i;\ell}^- \left(V_j^{-;\nu;s;\ell} \right)^T V_j^{-;s;\ell} \right].
\end{aligned}$$

With each $b_{i;\ell}\, v_j^{s;\ell} + \beta_{i;\ell}$ being similarly upper and lower bounded, we have

$$\begin{aligned}
& \left| u_j^{s;\ell} - \sigma(W_j^\ell u^{s;\ell-1} + w_j^\ell) \right| \\
& = \max \left(\begin{array}{l} u_j^{s;\ell} - \max\limits_{1\le i\le k_1} \left(a_{i;\ell}\, v_j^{s;\ell} + \alpha_{i;\ell} \right) + \max\limits_{1\le i\le k_2} \left(b_{i;\ell}\, v_j^{s;\ell} + \beta_{i;\ell} \right), \\ \max\limits_{1\le i\le k_1} \left(a_{i;\ell}\, v_j^{s;\ell} + \alpha_{i;\ell} \right) - \max\limits_{1\le i\le k_2} \left(b_{i;\ell}\, v_j^{s;\ell} + \beta_{i;\ell} \right) - u_j^{s;\ell} \end{array} \right) \\
& \le \max \left(\max_{1\le i\le k_2} m_{i;\bar{i}_1;j}^{1;s;\ell},\ \max_{1\le i\le k_1} m_{i;\bar{i}_2;j}^{2;s;\ell} \right) \qquad \begin{array}{l} \text{for all pairs of indices} \\ (\bar{i}_1, \bar{i}_2) \in \{1,\ldots,k_1\} \times \{1,\ldots,k_2\}\ , \end{array}
\end{aligned}$$

where, for any such pair $(\bar{i}_1, \bar{i}_2)$,

$$m_{i;\bar{i}_1;j}^{1;s;\ell} \triangleq \left\{ \begin{array}{l} \dfrac{a_{\bar{i}_1;\ell}^- + b_{i;\ell}^+}{4} \left\| V_j^{+;s;\ell} \right\|^2 + \dfrac{a_{\bar{i}_1;\ell}^+ + b_{i;\ell}^-}{4} \left\| V_j^{-;s;\ell} \right\|^2 + \left(b_{i;\ell} - a_{\bar{i}_1;\ell} \right) w_j^\ell + u_j^{s;\ell} \\ -\dfrac{1}{2}\left[(a_{\bar{i}_1;\ell}^- + b_{i;\ell}^+) \left(V_j^{-;\nu;s;\ell} \right)^T V_j^{-;s;\ell} + (a_{\bar{i}_1;\ell}^+ + b_{i;\ell}^-) \left(V_j^{+;\nu;s;\ell} \right)^T V_j^{+;s;\ell} \right] \\ \underbrace{+\dfrac{a_{\bar{i}_1;\ell}^+ + b_{i;\ell}^-}{4} \left\| V_j^{+;\nu;s;\ell} \right\|^2 + \dfrac{a_{\bar{i}_1;\ell}^- + b_{i;\ell}^+}{4} \left\| V_j^{-;\nu;s;\ell} \right\|^2 + \beta_{i;\ell} - \alpha_{\bar{i}_1;\ell}}_{\text{constant given } V_j^{\pm;\nu;s;\ell};\ \text{denoted } \widehat{c}_{i;\bar{i}_1;j}^{\nu;s;\ell}}, \end{array} \right.$$

$$m_{i;\bar{i}_2;j}^{2;s;\ell} \triangleq \left\{ \begin{array}{l} \frac{a_{i;\ell}^{+} + b_{\bar{i}_2;\ell}^{-}}{4} \left\| V_j^{+;s;\ell} \right\|^2 + \frac{a_{i;\ell}^{-} + b_{\bar{i}_2;\ell}^{+}}{4} \left\| V_j^{-;s;\ell} \right\|^2 + \left(a_{i;\ell} - b_{\bar{i}_2;\ell} \right) w_j^{\ell} - u_j^{s;\ell} \\ -\frac{1}{2} \left[(a_{i;\ell}^{+} + b_{\bar{i}_2;\ell}^{-}) \left(V_j^{-;\nu;s;\ell} \right)^T V_j^{-;s;\ell} + (a_{i;\ell}^{-} + b_{\bar{i}_2;\ell}^{+}) \left(V_j^{+;\nu;s;\ell} \right)^T V_j^{+;s;\ell} \right] \\ \underbrace{+\frac{a_{i;\ell}^{-} + b_{\bar{i}_2;\ell}^{+}}{4} \left\| V_j^{+;\nu;s;\ell} \right\|^2 + \frac{a_{i;\ell}^{+} + b_{\bar{i}_2;\ell}^{-}}{4} \left\| V_j^{-;\nu;s;\ell} \right\|^2 + \alpha_{i;\ell} - \beta_{\bar{i}_2}}_{\text{constant given } V_j^{\pm;\nu;s;\ell}\text{; denoted } \widetilde{c}_{i;\bar{i}_2;j}^{\nu;s;\ell}} . \end{array} \right.$$

In the following formulation of the MM subproblems, the variables $V_j^{\pm;s;\ell}$ are being substituted back by their definitions and therefore do not appear as variables in the subproblems; nevertheless, their values $V_j^{\pm;\nu;s;\ell}$ at the current iterate $(W_j^{\nu;\ell}, u^{\nu;s;\ell-1})$ appear in the constants $\widehat{c}_{i;\bar{i}_2;j}^{\nu;s;\ell}$ and $\widetilde{c}_{i;\bar{i}_2;j}^{\nu;s;\ell}$.

With the above majorization, we introduce the auxiliary variable $m_{\bar{i}_1;\bar{i}_2;j}^{s;\ell}$ for the upper bound $\max(\max_{1\le i\le i_2} m_{i;\bar{i}_1;j}^{1;s;\ell}, \max_{1\le i\le i_1} m_{i;\bar{i}_2;j}^{2;s;\ell})$, which we write as constraints in the following convex, quadratically constrained subproblem, and then solve the regularized MM subproblem with a given proximal scalar $\delta > 0$ and a pair of indices $(\bar{i}_1, \bar{i}_2) \in \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$.

$(\mathrm{P}_{\bar{i}_1,\bar{i}_2})$: **minimize over the variables** $\{W;\, w;\, u;\, m\}$: $\widehat{\Psi}_{\delta}^{\nu+1}(W, w, m, u)$

$$\triangleq \left\{ \begin{array}{l} \frac{1}{S} \sum_{s=1}^{S} \left\{ \varphi_s(u^{s;L}) + \rho \sum_{\ell=1}^{L} \sum_{j=1}^{N_\ell} m_{\bar{i}_1;\bar{i}_2;j}^{s;\ell} \right\} + \frac{\delta}{2} \sum_{s=1}^{S} \sum_{\ell=1}^{L} \left\| u^{s;\ell} - u^{\nu;s;\ell} \right\|^2 \\ + \frac{\delta}{2} \sum_{\ell=1}^{L} \sum_{j=1}^{N_\ell} \left[\sum_{s=1}^{S} \left(m_{\bar{i}_1;\bar{i}_2;j}^{s;\ell} - m_{\bar{i}_1;\bar{i}_2;j}^{\nu;s;\ell} \right)^2 + \left(w_j^{\ell} - w_j^{\nu;\ell} \right)^2 + \left\| W_j^{\ell} - W_j^{\nu;\ell} \right\|^2 \right] \end{array} \right.$$

subject to: for all $s = 1, \ldots, S$; $\ell = 1, \ldots, L$; and $j = 1, \ldots, N_\ell$ with $u^{s;0} = x^s$:

- for all $i = 1, \ldots, k_2$:

$$m_{\bar{i}_1;\bar{i}_2;j}^{s;\ell} \ge \left\{ \begin{array}{l} \frac{a_{\bar{i}_1;\ell}^{-} + b_{i;\ell}^{+}}{4} \left\| (W_j^{\ell})^T + u^{s;\ell-1} \right\|^2 + \frac{a_{\bar{i}_1;\ell}^{+} + b_{i;\ell}^{-}}{4} \left\| (W_j^{\ell})^T - u^{s;\ell-1} \right\|^2 \\ -\frac{1}{2} \left[\begin{array}{l} (a_{\bar{i}_1;\ell}^{-} + b_{i;\ell}^{+}) \left(V_j^{-;\nu;s;\ell} \right)^T \left((W_j^{\ell})^T - u^{s;\ell-1} \right) \\ +(a_{\bar{i}_1;\ell}^{+} + b_{i;\ell}^{-}) \left(V_j^{+;\nu;s;\ell} \right)^T \left((W_j^{\ell})^T + u^{s;\ell-1} \right) \end{array} \right] \\ + \left(b_{i;\ell} - a_{\bar{i}_1;\ell} \right) w_j^{\ell} + u_j^{s;\ell} + \widehat{c}_{i;\bar{i}_1;j}^{\nu;s;\ell}, \end{array} \right.$$

- for all $i = 1, \ldots, k_1$:

$$m_{\bar{i}_1;\bar{i}_2;j}^{s;\ell} \ge \left\{ \begin{array}{l} \frac{a_{i;\ell}^{+} + b_{\bar{i}_2;\ell}^{-}}{4} \left\| (W_j^{\ell})^T + u^{s;\ell-1} \right\|^2 + \frac{a_{i;\ell}^{-} + b_{\bar{i}_2;\ell}^{+}}{4} \left\| (W_j^{\ell})^T - u^{s;\ell-1} \right\|^2 \\ -\frac{1}{2} \left[\begin{array}{l} (a_{i;\ell}^{+} + b_{\bar{i}_2;\ell}^{-}) \left(V_j^{-;\nu;s;\ell} \right)^T \left((W_j^{\ell})^T - u^{s;\ell-1} \right) \\ +(a_{i;\ell}^{-} + b_{\bar{i}_2;\ell}^{+}) \left(V_j^{+;\nu;s;\ell} \right)^T \left((W_j^{\ell})^T + u^{s;\ell-1} \right) \end{array} \right] \\ + \left(a_{i;\ell} - b_{\bar{i}_2;\ell} \right) w_j^{\ell} - u_j^{s;\ell} + \widetilde{c}_{i;\bar{i}_2;j}^{\nu;s;\ell}. \end{array} \right.$$

The MM subproblem $(\mathrm{P}_{\bar{i}_1,\bar{i}_2})$ depends on the pair $(\bar{i}_1, \bar{i}_2) \in \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$. Here we choose the indices corresponding to the ε-maximizing functions in the two

max terms of $m^{s;\ell}_{\bar{i}_1;\bar{i}_2;j}$. That is, given a positive scalar ε and the tuple (3.3), we first choose a pair $(\bar{i}_1, \bar{i}_2)$ that depends on the triple (s, ℓ, j) from the ε-argmax sets

$$\begin{cases} \mathcal{A}^{s;\ell;j}_{1;\varepsilon}(v^{s;\ell}_j) \triangleq \left\{1 \le i \le k_1 \mid a_{\,i;\ell}\, v^{s;\ell}_j + \alpha_{\,i;\ell} \ge \max\limits_{1\le i\le k_1} \left(a_{\,i;\ell}\, v^{s;\ell}_j + \alpha_{\,i;\ell}\right) - \varepsilon\right\}, \\ \mathcal{A}^{s;\ell;j}_{2;\varepsilon}(v^{s;\ell}_j) \triangleq \left\{1 \le i \le k_2 \mid b_{\,i;\ell}\, v^{s;\ell}_j + \beta_{\,i;\ell} \ge \max\limits_{1\le i\le k_2} \left(b_{\,i;\ell}\, v^{s;\ell}_j + \alpha_{\,i;\ell}\right) - \varepsilon\right\}. \end{cases}$$

Originally described in [32] and extended in [5], the algorithm below is an enhanced version of the difference-convex algorithm [34] that corresponds to $\varepsilon = 0$ and computes a weaker kind of stationary solution. The enhancement aims at computing a directional stationary point of the multicomposite penalized problem (2.5). The role of the fixed but arbitrary ε is to ensure that the above index sets at the iterates of the algorithm contain the corresponding argmax sets at the accumulation point in order to ensure its directional stationarity. See the cited reference for details.

Algorithm MM: A majorization-minimization algorithm for (3.1)

Initialization. Given are an initial point $\{W^{\,0;\ell}, w^{\,0;\ell}\}_{\ell=1}^{L}$ and positive scalars ρ, δ, and ε. Let $u^{\,0;\ell}$ be defined by (1.1) recursively with $(W^{\,\ell}, w^{\,\ell}) = (W^{\,0;\ell}, w^{\,0;\ell})$ for $\ell = 1, \ldots, L$ and $u^{\,0;0} = x$. Set $\nu = 0$.

Step 1. Compute the values $\left\{\left(v^{\,\nu;s;\ell}_j\right)_{s=1}^{S}\right\}_{\ell=1}^{L}$ from (3.2).

Step 2. For *every* collection of the indices (I^{ν}_1, I^{ν}_2) in the set

$$\mathcal{A}^{\,\nu}_{\varepsilon} \triangleq \left\{\mathcal{A}^{s;\ell;j}_{1;\varepsilon}(v^{\,\nu;s;\ell}_j) \times \mathcal{A}^{s;\ell;j}_{2;\varepsilon}(v^{\,\nu;s;\ell}_j) \mid j = 1, \ldots, N_l,\ s = 1, \ldots, S,\ \ell = 1, \ldots, L\right\},$$

compute the optimal solution of problem $(\mathrm{P}_{I^{\nu}_1, I^{\nu}_2})$ which is denoted as

$$\left(W^{\,\nu+\frac{1}{2};I^{\nu}_1,I^{\nu}_2}, w^{\,\nu+\frac{1}{2};I^{\nu}_1,I^{\nu}_2}, u^{\,\nu+\frac{1}{2};I^{\nu}_1,I^{\nu}_2}, m^{\,\nu+\frac{1}{2};I^{\nu}_1,I^{\nu}_2}\right).$$

Step 3. Let $(\widehat{I}^{\nu}_1, \widehat{I}^{\nu}_2)$ be a minimizing index in

$$\operatorname{argmin}\left\{\widehat{\Psi}^{\,\nu+1}_{\delta}\left(W^{\,\nu+\frac{1}{2};I^{\nu}_1,I^{\nu}_2}, w^{\,\nu+\frac{1}{2};I^{\nu}_1,I^{\nu}_2}, u^{\,\nu+\frac{1}{2};I^{\nu}_1,I^{\nu}_2}, m^{\,\nu+\frac{1}{2};I^{\nu}_1,I^{\nu}_2}\right) \mid (I^{\nu}_1, I^{\nu}_2) \in \mathcal{A}^{\,\nu}_{\varepsilon}\right\};$$

set $(W^{\,\nu+1}, w^{\,\nu+1}, u^{\,\nu+1}, m^{\,\nu+1}) \triangleq \left(W^{\,\nu+\frac{1}{2};\widehat{I}^{\nu}_1,\widehat{I}^{\nu}_2}, w^{\,\nu+\frac{1}{2};\widehat{I}^{\nu}_1,\widehat{I}^{\nu}_2}, u^{\,\nu+\frac{1}{2};\widehat{I}^{\nu}_1,\widehat{I}^{\nu}_2}, m^{\,\nu+\frac{1}{2};\widehat{I}^{\nu}_1,\widehat{I}^{\nu}_2}\right)$,

Step 4. If $(W^{\,\nu+1}, w^{\,\nu+1}, u^{\,\nu+1}, m^{\,\nu+1})$ satisfies a prescribed stopping rule, terminate; otherwise, return to Step 1 with ν replaced by $\nu + 1$.

The following theorem asserts the subsequential convergence of the sequence generated by the MM algorithm to a d-stationary point of the penalized problem (3.1) for a fixed parameter $\rho > 0$ and arbitrary scalar $\varepsilon > 0$.

THEOREM 3.1 (see [5, Proposition 7]). *Suppose that the objective function* $\Psi_\rho(W, w; u)$ *in* (3.1) *is bounded below. The following two statements hold:*
(a) $\Psi_\rho(W^{\,\nu}, w^{\,\nu}, u^{\,\nu}) \le \Psi_\rho(W^{\,0}, w^{\,0}, u^{\,0})$ *for every* $\nu \ge 1$.
(b) *For any accumulation point* $(W^{\,\infty}, w^{\,\infty}, u^{\,\infty}, m^{\,\infty})$ *of* $\{(W^{\nu}, w^{\,\nu}, u^{\,\nu}, m^{\,\nu})\}$ *generated by the MM algorithm, if it exists, the tuple* $(W^{\infty}, w^{\,\infty}, u^{\,\infty})$ *is a d-stationary point of* (3.1)*,*

In our subsequent implementation of the algorithm, ρ is updated at the end of each MM iteration. The initial value of ρ was set to be 0.1 and was gradually increased by a factor of 1.1 if the average feasibility over the most recent four MM iterations has not reached 10^{-6} and ρ has not been updated within the last two MM iterations. It turns out that with our initial value of ρ the computations can be carried out with ρ being updated only a few times under the stopping criterion to be described in section 5. In spite of this numerical verification of the theory, a systematic way of adjusting ρ in general remains a research topic deserving further investigation.

4. Solving the convex majorizing subproblem via its dual. The MM subproblem $(\mathrm{P}_{\bar{i}_1,\bar{i}_2})$ is the workhorse of the overall algorithm for solving the deep learning problem (1.2). As such it is important to be able to solve it accurately and fast; both accuracy and speed are needed because each such subproblem is embedded in the overall MM iterations, which in turn correspond to a fixed penalty parameter ρ that may need to be iterated in order to recover the feasibility in the problem (2.3). Thus, as an alternative to its solution by a first-order method which typically cannot produce a high-accuracy solution fast, we propose a superlinearly convergent (in a local sense) dual semismooth method that can exploit the separable structures of the problem in order to quickly obtain an approximate solution with high accuracy.

Introducing the nonnegative dual variables $\{(\widehat{\lambda}^{s;\ell;i}_{\bar{i}_1;\bar{i}_2;j})_{i=1}^{k_2}, (\widetilde{\lambda}^{s;\ell;i}_{\bar{i}_1;\bar{i}_2;j})_{i=1}^{k_1}\}$, where $j = 1,\ldots,N_\ell$, $\ell = 1,\ldots,L$, and $s = 1,\ldots,S$, for the quadratic inequality constraints, respectively, we may write the dual program as

$$\underset{\widehat{\lambda};\,\widetilde{\lambda}}{\text{maximize}}\ d_{\bar{i}_1;\bar{i}_2}\left(\widehat{\lambda},\widetilde{\lambda}\right) \quad \text{subject to } \left(\widehat{\lambda};\,\widetilde{\lambda}\right) \geq 0, \tag{4.1}$$

where the dual function $d_{\bar{i}_1;\bar{i}_2}(\widehat{\lambda},\widetilde{\lambda})$ is equal to

minimum over the variables $\{W;\, w;\, u;\, m\}$ with $u^{s;0} = x^s$:

$$\left\{\begin{array}{l} \dfrac{1}{S}\displaystyle\sum_{s=1}^{S}\left\{\varphi_s(u^{s;L}) + \rho\sum_{\ell=1}^{L}\sum_{j=1}^{N_\ell} m^{s;\ell}_{\bar{i}_1;\bar{i}_2;j}\right\} + \dfrac{\delta}{2}\displaystyle\sum_{s=1}^{S}\sum_{\ell=1}^{L}\left\| u^{s;\ell} - u^{\nu;s;\ell}\right\|^2 \\ +\dfrac{\delta}{2}\displaystyle\sum_{\ell=1}^{L}\sum_{j=1}^{N_\ell}\left[\sum_{s=1}^{S}\left(m^{s;\ell}_{\bar{i}_1;\bar{i}_2;j} - m^{\nu;s;\ell}_{\bar{i}_1;\bar{i}_2;j}\right)^2 + \left(w^\ell_j - w^{\nu;\ell}_j\right)^2 + \left\| W^\ell_j - W^{\nu;\ell}_j\right\|^2\right] \end{array}\right\}$$

$$+ \sum_{\substack{s=1,\cdots,S,\\ \ell=1,\cdots L,\\ j=1,\cdots,N_\ell,\\ i=1,\cdots,k_2}} \widehat{\lambda}^{s;\ell;i}_{\bar{i}_1;\bar{i}_2;j} \left\{\begin{array}{l} \dfrac{a^-_{\bar{i}_1;\ell} + b^+_{i;\ell}}{4}\left\|(W^\ell_j)^T + u^{s;\ell-1}\right\|^2 + \dfrac{a^+_{\bar{i}_1;\ell} + b^-_{i;\ell}}{4}\left\|(W^\ell_j)^T - u^{s;\ell-1}\right\|^2 \\ -\frac{1}{2}\left(a^-_{\bar{i}_1;\ell} + b^+_{i;\ell}\right)\left(V^{-;\nu;s;\ell}_j\right)^T\left((W^\ell_j)^T - u^{s;\ell-1}\right) \\ -\frac{1}{2}\left(a^+_{\bar{i}_1;\ell} + b^-_{i;\ell}\right)\left(V^{+;\nu;s;\ell}_j\right)^T\left((W^\ell_j)^T + u^{s;\ell-1}\right) \\ +\left(b_{i;\ell} - a_{\bar{i}_1;\ell}\right) w^\ell_j + u^{s;\ell}_j - m^{s;\ell}_{\bar{i}_1;\bar{i}_2;j} + \widehat{c}^{\,\nu;s;\ell}_{i;\bar{i}_1;j} \end{array}\right\}$$

$$+ \sum_{\substack{s=1,\cdots,S,\\ \ell=1,\cdots L,\\ j=1,\cdots,N_\ell,\\ i=1,\cdots,k_1}} \widetilde{\lambda}^{s;\ell;i}_{\bar{i}_1;\bar{i}_2;j} \left\{\begin{array}{l} \dfrac{a^+_{i;\ell} + b^-_{\bar{i}_2;\ell}}{4}\left\|(W^\ell_j)^T + u^{s;\ell-1}\right\|^2 + \dfrac{a^-_{i;\ell} + b^+_{\bar{i}_2;\ell}}{4}\left\|(W^\ell_j)^T - u^{s;\ell-1}\right\|^2 \\ -\frac{1}{2}\left(a^+_{i;\ell} + b^-_{\bar{i}_2;\ell}\right)\left(V^{-;\nu;s;\ell}_j\right)^T\left((W^\ell_j)^T - u^{s;\ell-1}\right) \\ -\frac{1}{2}\left(a^-_{i;\ell} + b^+_{\bar{i}_2;\ell}\right)\left(V^{+;\nu;s;\ell}_j\right)^T\left((W^\ell_j)^T + u^{s;\ell-1}\right) \\ +\left(a_{i;\ell} - b_{\bar{i}_1;\ell}\right) w^\ell_j - u^{s;\ell}_j - m^{s;\ell}_{\bar{i}_1;\bar{i}_2;j} + \widetilde{c}^{\,\nu;s;\ell}_{i;\bar{i}_2;j} \end{array}\right\}.$$

Our goal is to apply a combination of a gradient projection algorithm (possibly with some heuristics acceleration) and the SN method for solving the dual program (4.1). Notice that the size of the dual variable is $(k_1 + k_2)S \sum_{\ell=1}^{L} N_\ell$, which can be massive when the number of samples is large. We devote the next subsection to discussing how this dual scheme can be carried out in practice. The detailed computation of the Newton direction including the formula of the gradient and generalized Jacobian is analyzed in section 4.2. In order to alleviate the computational burden and storage cost for computing the Newton direction, we provide a low-storage CG scheme in sections 4.3 and 4.4.

4.1. Superlinear computation of the dual problem. Consider the problem

$$\underset{x \in \mathbb{R}^K_+}{\text{minimize}}\ f(x), \tag{4.2}$$

where the function $f : \Xi \subseteq \mathbb{R}^K \to \mathbb{R}$ is a twice continuously differentiable convex function defined on the open set Ξ containing $\mathbb{R}^K_+$. Here x denotes a generic variable not to be confused with the input x in the previous discussions. Notice that the dual program (4.1) is a special case of the above problem where $x = (\widehat{\lambda}, \widetilde{\lambda})$ and $f(x) = d_{\bar{i}_1;\bar{i}_2}(\widehat{\lambda}, \widetilde{\lambda})$. The twice continuity of f is due to the strong convexity of the inner minimization problem in (4.1) and Danskin's theorem; details will be provided in the next subsection. The problem (4.2) is equivalent to its optimality conditions which we write as the nonlinear complementarity problem:

$$0 \le x \perp \nabla f(x) \ge 0,$$

where $\perp$ is the perpendicular notation. In turn, the latter complementarity condition is equivalent to the nonsmooth equation:

$$\Phi(x) \triangleq \min(x, \nabla f(x)) = 0,$$

where min is the componentwise minimum operator. Note that the nonnegativity of the variable x is automatically taken care of by the min-equation. Given a vector x that is not necessarily nonnegative, define three mutually disjoint index sets that partition $\{1, \ldots, K\}$:

$$\alpha(x) \triangleq \left\{ i \mid x_i < \frac{\partial f(x)}{\partial x_i} \right\},\ \beta(x) \triangleq \left\{ i \mid x_i = \frac{\partial f(x)}{\partial x_i} \right\},\ \text{and } \gamma(x) \triangleq \left\{ i \mid x_i > \frac{\partial f(x)}{\partial x_i} \right\}.$$

Described below, the proposed algorithm aims to solve the semismooth equation $\Phi(x) = 0$ by solving at each iteration a system of linear equations to obtain a search direction. A candidate iterate is then tested to determine if there is sufficient decrease in the norm function $\|\Phi(x)\|$ from the iterate x^ν at the current iteration ν. If so, the new iterate is accepted and the iteration proceeds to the next. Otherwise, we resort to the projected gradient iterations that employ the gradient function of f to ensure the required sufficient decrease of $\|\Phi(x)\|$. In the algorithm, the variable x may or may not be nonnegative, depending on whether an immediate acceptance of the new iterate occurs after the equation-solving Step 1. This step is the departure of this algorithm from many existing superlinearly convergent (or second-order) algorithms for solving a nonnegatively constrained twice differentiable program, such as a Newton method for such a problem [31] or the SN with linear search described in [12, Chapter 8] for a nonlinear complementarity problem that employs a complementarity function as the

merit function. Specifically, instead of solving a convex quadratic program or using a complementarity function as the merit function, this step is replaced by the solution of a system of linear equations (4.3) that is of reduced dimension and uses the original objective $f(x)$ as the back-up descent step.

For a given vector $x \in \mathbb{R}^n$, the vector $dx \triangleq [x - \nabla f(x)]_+ - x$, where $a_+ \triangleq \max(0, a)$ is the nonnegative part of the vector a, is the projected gradient direction of f at x. The algorithm below is initiated at a nonnegative iterate x^0; this can be a vector obtained by a preprocessing scheme (such as the gradient projection algorithm with acceleration terminated according to a prescribed inexact criterion), or in the context of the iterative MM algorithm, an iterate from the previous MM iteration. For a given vector x, an index set γ, and two disjoint index sets γ_1 and γ_2, we write

$$\nabla_\gamma f(x) = \left(\frac{\partial f(x)}{\partial x_i}\right)_{i\in\gamma}, \nabla^2_\gamma f(x) = \left[\frac{\partial^2 f(x)}{\partial x_i \partial x_j}\right]_{i,j\in\gamma}, \nabla^2_{\gamma_1\gamma_2} f(x) = \left[\frac{\partial^2 f(x)}{\partial x_i \partial x_j}\right]_{(i,j)\in\gamma_1\times\gamma_2}.$$

Algorithm GP+SN: An enhanced semismooth Newton method for (4.2)

Step 0: Initialization. Let $x^0 \geq 0$ be given. Let $p > 0$ be an arbitrary integer, $\gamma > 0$, $r \in (0,1)$, and $\sigma_1, \sigma_2 \in (0,1)$ be given scalars. Set the outer iteration count $\nu = 0$. (This SN iteration counter is different from the MM iteration counter.)

Step 1: Direction finding. Pick an arbitrary subset β_ν of $\beta(x^\nu)$. Let $\gamma_\nu \triangleq \beta_\nu \cup \gamma(x^\nu)$ and let $\bar{\gamma}_\nu$ be the complement of γ_ν in $\{1, \ldots, K\}$. Solve the following system of linear equations for the subvector $dx^\nu_{\gamma_\nu}$ with the positive parameter $\varepsilon_\nu = \sigma_1 \| \Phi(x^\nu) \|$:

$$\nabla_{\gamma_\nu} f(x^\nu) - \nabla^2_{\gamma_\nu\bar{\gamma}_\nu} f(x^\nu)\, x^\nu_{\bar{\gamma}_\nu} + \left[\, \varepsilon_\nu \mathbf{I}_{|\gamma_\nu|} + \nabla^2_{\gamma_\nu} f(x^\nu) \,\right] dx_{\gamma_\nu} = 0. \tag{4.3}$$

Set $dx^\nu_{\bar{\gamma}_\nu} \triangleq -x^\nu_{\bar{\gamma}_\nu}$.

Step 2: Testing sufficient decrease. If $\|\Phi(x^\nu + dx^\nu)\| \leq \sigma_2 \|\Phi(x^\nu)\|$, let $x^{\nu+1} \triangleq x^\nu + dx^\nu$; go to Step 7. Otherwise set the inner iteration counter $\omega = 0$.

Step 3: Initiation of inner gradient projection scheme. Set the parameter $\beta_{\nu;0} = 1$ and let the initial inner iterate $\widetilde{x}^{\nu;0} = \widehat{x}^{\nu;0}$ be x^ν_+. If $\nabla f(x^\nu_+)^T dx^\nu \leq -\gamma \| dx^\nu \|^p$ is satisfied, set the initial inner search direction $\widetilde{dx}^{\nu;0} = dx^\nu$ and go to Step 5. Otherwise continue.

Step 4: Inner projected gradient direction. Set the inner search direction

$$\widetilde{dx}^{\nu;\omega} \triangleq [\, \widetilde{x}^{\nu;\omega} - \nabla f(\widetilde{x}^{\nu;\omega}) \,]_+ - \widetilde{x}^{\nu;\omega}.$$

Step 5: Armijo line search. Let $m_{\nu;\omega}$ be the smallest nonnegative integer m such that

$$f(\widetilde{x}^{\nu;\omega} + r^m \widetilde{dx}^{\nu;\omega}) - f(\widetilde{x}^{\nu;\omega}) \leq \sigma_2\, r^m\, \nabla f(\widetilde{x}^{\nu;\omega})^T \widetilde{dx}^{\nu;\omega}.$$

Let $\widehat{x}^{\nu;\omega+1} \triangleq \widetilde{x}^{\nu;\omega} + r^{m_{\nu;\omega}} \widetilde{dx}^{\nu;\omega}$.

Step 6: Inner iteration termination check. If $\left\| \Phi(\widehat{x}^{\nu;\omega+1}) \right\| \leq \sigma_2 \|\Phi(x^\nu)\|$, go to Step 7 with $x^{\nu+1} \triangleq \widehat{x}^{\nu;\omega+1}$. Otherwise continue.

Step 7: Overall termination check. If $\left\|\Phi(z^{\nu+1})\right\| \leq$ a prescribed tolerance, stop. Otherwise return to Step 1 with $\nu \leftarrow \nu + 1$.

Note that each iterate $x^{\nu+1}$ obtained from Step 6 must be nonnegative, whereas if such an iterate is directly accepted after a successful Step 1, then its nonnegativity may be violated. In what follows, we show that under a certain nonsingularity assumption

at an accumulation point, the above algorithm will generate a well-defined sequence $\{x^\nu\}$ that converges superlinearly (in a local sense) to an optimal solution of (4.2).

THEOREM 4.1. *Let f be a twice continuously differentiable, convex function defined on an open set containing $\mathbb{R}^K_+$. The following statements hold:*

(a) *If f has bounded level sets on $\mathbb{R}^K_+$, then for every ν, the projected gradient iterations starting at Step 3 with an initial $\widehat{x}^{\nu;0} = x^\nu_+$ will terminate with an $x^{\nu+1}$ at Step 5 in a finite number of iterations. Thus the sequence $\{x^\nu\}$ is well defined.*

(b) *If x^∞ is an accumulation point of the sequence $\{x^\nu\}$ such that the principal submatrix $\nabla^2_{\gamma_\infty} f(x^\infty)$ is nonsingular, where $\gamma_\infty \triangleq \beta(x^\infty) \cup \gamma(x^\infty)$, then x^∞ is the unique global minimizer of f on $\mathbb{R}^K_+$. Moreover in this case,*

(b1) *eventually, the test of sufficient decrease of $\|\Phi(x^\nu + dx^\nu)\|$ in Step 2 is always successful;*

(b2) *eventually, the entire sequence $\{x^\nu\}$ converges superlinearly to x^∞.*

Proof. Statement (a) follows from the well-known convergence result of the gradient projection algorithm because every accumulation point of a sequence of iterates produced by this algorithm must be a stationary point of f and thus is a zero of the function Φ. Hence the test in Step 6 must be satisfied after finitely many gradient projection iterations. Statements (b1) and (b2) follow from similar results for the min based Newton method for the mixed complementarity problem; see, e.g., [12, Theorem 9.1.29]. Finally, by the theory of nonsmooth equations, under the nonsingularity assumption, x^∞ must be an isolated, thus unique, stationary point, thus a global minimizer of the function f on $\mathbb{R}^K_+$. ∎

4.2. Solving a quadratically constrained quadratic program by its dual. When each function φ_s is convex quadratic (such as in the case of a least-squares error function), the MM subprogram is a convex quadratically constrained quadratic program (QCQP) that is potentially of very large size but highly structured. For nonlinear functions φ_s, each MM subproblem is a nonquadratic program with convex quadratic constraints. In this subsection, we use the squared loss

$$\varphi_s(u^{s;L}) \;=\; \tfrac{1}{2}\,(\,y_s - u^{s;L}\,)^2, \quad s \;=\; 1,\ldots,S, \tag{4.4}$$

as an example to show how to apply the GP+SN method to solve the corresponding dual problem. For such a case, the MM subproblem essentially has the form

$$\begin{array}{ll} \underset{z\in\mathbb{R}^{N_z}}{\text{minimize}} & q_0(z) \;\triangleq\; \frac{1}{2}\, z^T Q^{\,0} z + z^T r^{\,0} \\ \text{subject to} & q_k(z) \;\triangleq\; \frac{1}{2}\, z^T Q^{\,k} z + z^T r^{\,i} + c_k \;\le\; 0, \quad k \;=\; 1,\ldots,K, \end{array} \tag{4.5}$$

where each $Q^{\,k}$ for $k = 0, 1, 2, \ldots, K$ is an $N_z \times N_z$ symmetric positive semidefinite matrix with $Q^{\,0}$ being positive definite; each $r^{\,k}$ for $k = 0, 1, 2, \ldots, K$ is an N_z-vector, and each c_k is a scalar. Write

$$\mathcal{Q}(\xi) \;\triangleq\; Q^{\,0} + \sum_{k=1}^{K} \xi_k\, Q^{\,k} \quad \text{and} \quad r(\xi) \;\triangleq\; r^{\,0} + \sum_{k=1}^{K} \xi_k\, r^{\,k}.$$

Since $Q^{\,0}$ is positive definite by assumption, the matrix $\mathcal{Q}(\xi)$ remains positive definite for all ξ in a sufficiently small open neighborhood of the nonnegative orthant $\mathbb{R}^K_+$. We assume that (4.5) satisfies the Slater constraint qualification so that the dual program

$$\underset{\xi\in\mathbb{R}^K_+}{\text{maximize}}\; \chi(\xi) \;\triangleq\; \underset{z\in\mathbb{R}^{N_z}}{\text{minimum}}\; \tfrac{1}{2}\, z^T \mathcal{Q}(\xi) z + z^T r(\xi) + \sum_{k=1}^{K} \xi_k\, c_k \tag{4.6}$$

is well defined and there is no duality gap between (4.5) and (4.6). Notice that the Slater constraint qualification for every MM subproblem $(\mathrm{P}_{\bar{i}_1,\bar{i}_2})$ is automatically satisfied since one can choose sufficiently large $m^{s;\ell}_{\bar{i}_1;\bar{i}_2;j}$ such that all quadratic constraints hold strictly. Before describing the details of the combined GP+SN method, we present some key derivative formulas for the dual function. Let $R \in \mathbb{R}^{K\times N_z}$ be the matrix whose kth row is equal to $(r^{\,k})^T$ for $k = 1,\ldots,K$. The unique global minimizer in $\chi(\xi)$ is given by the solution of the nonsingular equation:

$$\mathcal{Q}(\xi)z(\xi) + r(\xi) \;=\; 0.$$

Differentiating this equation with respect to ξ_k, we deduce

$$\mathcal{Q}(\xi)\frac{\partial z(\xi)}{\partial \xi_k} + \underbrace{r^k + Q^k z(\xi)}_{\nabla q_k(z(\xi))} \;=\; 0, \quad k \;=\; 1,\ldots,K.$$

To proceed, we write $q(z) \in \mathbb{R}^K$ as the K-vector with $q_k(z)$ in (4.5) as its components for $k = 1,\ldots,K$. We also denote $Jq(z)$ as a $K \times N_z$ Jacobian matrix whose kth row is $\nabla q_k(z)^T$ for $k = 1,\ldots,K$. The above equation can be written in the matrix form

$$\mathcal{Q}(\xi)Jz(\xi) + Jq(z(\xi))^T \;=\; 0,$$

which yields

$$Jz(\xi) \;=\; -\mathcal{Q}(\xi)^{-1}Jq(z(\xi))^T \;\in\; \mathbb{R}^{N_z\times K}.$$

By the well-known Danskin's theorem (see, e.g., [12, Theorem 10.2.1]), it follows that

$$\frac{\partial \chi(\xi)}{\partial \xi_k} \;=\; c_k + z(\xi)^T r^k + \tfrac{1}{2}\, z(\xi)^T Q^{\,k} z(\xi) \;=\; \nabla q_k(z(\xi)) \quad \text{for } k = 1,\ldots,K.$$

Hence

$$\nabla\left(\frac{\partial \chi(\xi)}{\partial \xi_k}\right) \;=\; Jz(\xi)^T\left[\, r^k + Q^k z(\xi)\,\right] \;\in\; \mathbb{R}^K, \quad k \;=\; 1,\ldots,K.$$

In summary, letting $\mathrm{vct}[a_k]$ be the vector with components a_k, we obtain

$$\begin{cases} z(\xi) \;=\; -\mathcal{Q}(\xi)^{-1}r(\xi) \;\in\; \mathbb{R}^{N_z}, \\ \chi(\xi) \;=\; c^T\xi - \frac{1}{2}\, r(\xi)^T\mathcal{Q}(\xi)^{-1}r(\xi), \\ \nabla\chi(\xi) \;=\; c + Rz(\xi) + \frac{1}{2}\,\mathrm{vct}\left[\,\left(\, z(\xi)^T Q^{\,k} z(\xi)\,\right)_{k=1}^K\right] \;=\; \left(\,\nabla q_k(z(\xi))\,\right)_{i=1}^K \;\in\; \mathbb{R}^K, \\ \nabla^2\chi(\xi) \;=\; -Jq(z(\xi))\mathcal{Q}(\xi)^{-1}Jq(z(\xi))^T \;\in\; \mathbb{R}^{K\times K}. \end{cases}$$

We notice that the principal submatrices of the Hessian matrix $\nabla^2\chi(\xi)$ have the same structure as the full Hessian. Namely, for any subset γ of $\{1,\ldots,K\}$, we have

$$\nabla^2\chi(\xi)_{\gamma\gamma} \;=\; Jq_\gamma(z(\xi))\mathcal{Q}(\xi)^{-1}Jq_\gamma(z(\xi))^T. \tag{4.7}$$

It is important to point out that in the algorithm described in the next subsection, there is no need to form the full Hessian $\nabla^2\chi(\xi)$ or its principal submatrices explicitly; only the product form of these matrices suffices. In turn, the knowledge of the matrices $\{Q^{\,i}\}_{i=0}^K$ and vectors $\{r^{\,k}\}_{k=1}^K$ is sufficient to implement the algorithm for solving (4.6), and thus (4.5), hence, the MM dual (4.1), and finally, the MM subproblem $(\mathrm{P}_{\bar{i}_1,\bar{i}_2})$. Clearly, the bulk of the computation in Steps 1 through 8 of the above algorithm lies

in the solution of the system of linear equations (4.3) in Step 1. In the context of the dual function $\chi(\xi)$ of the QCQP (4.5), this step amounts to solving the system of linear equations for the search vector $d\xi_\gamma \in \mathbb{R}^{|\gamma|}$,

$$\nabla_\gamma \chi(\xi) - \nabla^2_{\gamma\bar{\gamma}} \chi(\xi)\, \xi_{\bar{\gamma}} + \left[\, \varepsilon\, \mathbf{I}_{|\gamma|} + \nabla^2_\gamma \chi(\xi) \,\right]\, d\xi_\gamma \;=\; 0, \tag{4.8}$$

corresponding to a given iterate ξ and a subset γ of $\{1, \ldots, K\}$ with complement $\bar{\gamma}$. Taking advantage of the product form of the principal submatrix $\nabla^2_\gamma \chi(\xi)$, we describe in subsection 4.3 how this step can be carried out by the CG method. Here, we note that by the expression (4.7), the nonsingularity condition of $\nabla^2_{\gamma_\infty} f(x^\infty)$ required for the superlinear convergence in Theorem 4.1 amounts to the full column rank of $Jq_{\gamma_\infty}(z(\xi^\infty))^T$, where γ_∞ is the index set of the active constraints of the QCQP (4.5) at the optimal solution $z^\infty \triangleq z(\xi^\infty)$; in turn, this full-rank condition is exactly the classical linear independence constraint qualification of the active constraint gradients in nonlinear programming theory.

4.3. Solving (4.8) by conjugate gradient: Vector-matrix multiplication. In applying the renowned CG method [17] for solving a system of linear equations $Ax = b$ with a symmetric positive definite matrix A, the main computational effort lies in the vector-matrix multiplication of the matrix A by an iterate. In this subsection, we focus on this key step in preparation for its application to the MM subproblem $(\mathrm{P}_{\bar{i}_1,\bar{i}_2})$ to be discussed in the next section. We refer the reader to the cited reference for the details of the CG method.

In the context of (4.8), we have

$$A = \varepsilon_\nu\, \mathbf{I}_{|\gamma_\nu|} + Jq_\gamma(z(\xi))\mathcal{Q}(\xi)^{-1} Jq_\gamma(z(\xi))^T,$$

where $\varepsilon_\nu = \sigma_1 \,\|\, \Phi(x^\nu) \,\|$ is given in Step 1 of Algorithm GP+SN. Thus, omitting the identity matrix in the discussion, the key computational effort per CG iteration is the vector-matrix multiplication to yield the vector and inner products

$$\widehat{\eta} \;\triangleq\; Jq(z)\mathcal{Q}(\xi)^{-1} Jq(z)^T\, \eta \quad \text{and} \quad \varsigma \;\triangleq\; \eta^T \widehat{\eta} \tag{4.9}$$

for a given vector $\eta \in \mathbb{R}^K$, where for simplicity of notation, we have dropped the index set γ and use a general vector z for $z(\xi)$. With a proper subset γ, we simply restrict the computation to the constraint functions $q_k(z)$ for $k \in \gamma$.

The computation of the vector $\widehat{\eta}$ and scalar ς can be accomplished in several steps. To begin, we note that the matrix $\mathcal{Q}(\xi)$ remains the same throughout the CG iterations. Thus, we factor it at the outset before starting such iterations:

$$\mathcal{Q}(\xi) \;=\; \mathcal{L}\, \mathcal{D}\, \mathcal{L}^T, \tag{4.10}$$

where $\mathcal{L}$ is a unit lower triangular matrix and $\mathcal{D}$ is a positive diagonal matrix; the dependence of these factor matrices on ξ is omitted. Note that

$$Jq(z)^T\, \eta \;=\; \sum_{k=1}^{K} \eta_k\, \nabla q_k(z) \;=\; \sum_{k=1}^{K} \eta_k\, \left(r^k + Q^k z \right).$$

Moreover, for a vector $\bar{z} \in \mathbb{R}^{N_z}$, we have $Jq(z)\bar{z}$ is the K-vector whose kth component is equal to the inner product $\nabla q_k(z)^T \bar{z}$. With the $LDL^\top$ factorization (4.10) available, the following is the step-by-step formation of the product vector $\widehat{\eta}$ defined by (4.9).

Computing the vector $\widehat{\eta}$ of (4.9) given z and η

Step 1. Compute for each $k = 1, \ldots, K$, the vector $\bar{z}^k \triangleq \nabla q_k(z) = r^k + Q^k z$.

Step 2. Solve the lower triangular system for $\bar{z}^{1/2}$: $\mathcal{L}\,\bar{z}^{1/2} = \sum_{k=1}^{K} \eta_k\,\bar{z}^k$.

Step 3. Solve the upper triangular system of linear equations for $\bar{z}$: $\mathcal{L}^T\,\bar{z} = \mathcal{D}^{-1}\,\bar{z}^{1/2}$,
Step 4, Compute the products $\nabla q_k(z)^T \bar{z}$ for $k = 1, \ldots, K$ and output these products as the components of $\widehat{\eta}$. Finally, $\varsigma = (\bar{z}^{1/2})^T\,\mathcal{D}^{-1}\,\bar{z}^{1/2}$.

4.4. The $LDL^\top$ factorization of (W, u) diagonal blocks. Finally, we need to derive the $LDL^\top$ factorization of the matrix $\delta\,\mathbf{I}_{N^\ell_{Wu}} + \widehat{Q}^\ell(\widehat{\lambda}^\ell) + \widetilde{Q}^\ell(\widetilde{\lambda}^\ell)$ for $\ell = 1, \ldots, L$, where the first term (δ-multiple of the identity matrix) comes from the objective function of the MM subproblem $(\mathrm{P}_{\bar{i}_1,\bar{i}_2})$ pertaining to the second group of variables, i.e., W and u except $u^{s;L}$. In fact, such a factorization needs to be computed whenever a new gradient of the dual objective $\chi(\xi)$ is needed for solving the MM subproblem. This can be seen from the formula of $\nabla\chi(\xi)$ in subsection 4.2. We can accomplish the factorization very easily using the $LDL^\top$ factorization of the matrix

$$\delta\,\mathbf{I}_{N_\ell+S} + \Lambda^\ell_{\bar{i}_1;\bar{i}_2}(\lambda^\ell) = \underbrace{\mathcal{L}^\ell_{\bar{i}_1;\bar{i}_2}(\lambda^\ell)}_{\substack{\text{unit lower triangular matrix}\\ \text{of order } (S+N_\ell)\times(S+N_\ell)}} \quad \underbrace{\mathcal{D}^\ell_{\bar{i}_1;\bar{i}_2}(\lambda^\ell)}_{\substack{\text{positive diagonal matrix}\\ \text{of order } (S+N_\ell)}} \quad \mathcal{L}^\ell_{\bar{i}_1;\bar{i}_2}(\lambda^\ell)^T.$$

Indeed, with the latter factorization, we have

$$\begin{aligned}&\delta\,\mathbf{I}_{N^\ell_{Wu}} + \widehat{Q}^\ell(\widehat{\lambda}^\ell) + \widetilde{Q}^\ell(\widetilde{\lambda}^\ell)\\ &= \left(\underbrace{\mathcal{L}^\ell_{\bar{i}_1;\bar{i}_2}(\lambda^\ell) \otimes \mathbf{I}_{N_{\ell-1}}}_{\text{block unit lower triangular}}\right)\left(\underbrace{\mathcal{D}^\ell_{\bar{i}_1;\bar{i}_2}(\lambda^\ell) \otimes \mathbf{I}_{N_{\ell-1}}}_{\text{diagonal}}\right)\left(\mathcal{L}^\ell_{\bar{i}_1;\bar{i}_2}(\lambda^\ell) \otimes \mathbf{I}_{N_{\ell-1}}\right)^T.\end{aligned}$$

Thus, the solution of a system of linear equations defined by the block unit lower (upper) triangular matrix $\mathcal{L}^\ell_{\bar{i}_1;\bar{i}_2}(\lambda^\ell) \otimes \mathbf{I}_{N_{\ell-1}}$ (and its transpose, resp.) can be accomplished by a block forward (backward, resp.) substitution process; each substitution step aims to compute a vector of dimension $N_{\ell-1}$ and there are $S + N_\ell$ steps of substitution.

Finally, since typically the number of neurons N_ℓ in each layer is much smaller than the number of samples S, an efficient way to obtain the $LDL^\top$ factorization of the $(S + N_\ell) \times (S + N_\ell)$ matrix

$$\delta\,\mathbf{I}_{S+N_\ell} + \Lambda^\ell_{\bar{i}_1;\bar{i}_2}(\lambda^\ell) = \begin{bmatrix} \delta\,\mathbf{I}_S + \Lambda^{u;\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell) & \left(\Lambda^{Wu;\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell)\right)^T \\ \Lambda^{Wu;\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell) & \delta\,\mathbf{I}_{N_\ell} + \Lambda^{W;\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell) \end{bmatrix}$$

is to use the Schur complement approach, taking advantage of the diagonal property of the two diagonal blocks. Specifically, obtain the $LDL^\top$ factorization of the $N_\ell \times N_\ell$ matrix

$$\left[\delta\,\mathbf{I}_{N_\ell}+\Lambda^{W;\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell)\right]-\Lambda^{Wu;\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell)\,\left[\delta\,\mathbf{I}_S+\Lambda^{u;\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell)\right]^{-1}\,\left(\Lambda^{Wu;\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell)\right)^T$$
$$= \underbrace{\mathcal{L}^{\mathrm{Schur;W};\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell)}_{\substack{\text{unit lower triangular matrix}\\ \text{of order } N_\ell\times N_\ell}}\qquad \underbrace{\mathcal{D}^{\mathrm{Schur;W};\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell)}_{\substack{\text{positive diagonal matrix}\\ \text{of order } N_\ell\times N_\ell}}\qquad \left(\mathcal{L}^{\mathrm{Schur;W};\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell)\right)^T.$$

With the latter factorization, we then have

$$\mathcal{L}^{\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell) = \begin{bmatrix} \mathbf{I}_S & 0 \\ \Lambda^{Wu;\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell)\,\left[\delta\,\mathbf{I}_S+\Lambda^{u;\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell)\right]^{-1} & \mathcal{L}^{\mathrm{Schur;W};\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell) \end{bmatrix},$$

$$\mathcal{D}^{\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell) = \begin{bmatrix} \delta\,\mathbf{I}_S+\Lambda^{u;\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell) & 0 \\ 0 & \mathcal{D}^{\mathrm{Schur;W};\ell}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell) \end{bmatrix}.$$

With the identity matrix $\mathbf{I}_S$ appearing in the upper left diagonal block of $\mathcal{L}_{\bar{i}_1;\bar{i}_2}(\lambda^\ell)$, we see that these S steps of substitution can be trivially implemented; the remaining N_ℓ steps of substitution require very little work when the number of neurons within each layer is not large. Below we summarize the computational cost for computing the gradient $\nabla\chi(\xi)$ at a given vector $\xi\in\mathbb{R}^{(k_1+k_2)SN_{\ell+1}}$ and the matrix-vector product of $\nabla^2\chi(\xi)$ with a given vector $d\in\mathbb{R}^{(k_1+k_2)SN_{\ell+1}}$; the latter is approximately the computational cost of one CG step for solving the SN equation (4.8). The integers k_1 and k_2 are typically very small; therefore they are considered constants in the following expressions of the computational costs.

- Compute $\mathcal{Q}(\xi)$ and $r(\xi)$: $O\left(S\sum_{\ell=1}^{L}N_{\ell-1}N_\ell\right)$;
- Compute LDL^T of $\mathcal{Q}(\xi)$ via Schur complement: $O\left(S\sum_{\ell=1}^{L}N_\ell^2+\sum_{\ell=1}^{L}N_\ell^3\right)$;
- Compute $z(\xi)$ via the backward- and forward-substitution: $O\left(\sum_{\ell=1}^{L}N_{\ell-1}N_\ell^2\right)$;
- Final computation given $z(\xi)$: $O\left(S\sum_{\ell=1}^{L}N_{\ell-1}N_\ell\right)$;

Total computational cost for $\nabla\chi(\xi)$: $O\left(\sum_{\ell=1}^{L}\left[\,S\left(N_\ell^2+N_{\ell-1}N_\ell\right)+N_{\ell-1}N_\ell^2+N_\ell^3\,\right]\right)$.

- Given $d\in\mathbb{R}^{(k_1+k_2)SN_{\ell+1}}$, compute $Jq(z(\xi))^Td$: $O\left(S\sum_{\ell=1}^{L}N_{\ell-1}N_\ell\right)$;
- Compute $\widehat{d}(\xi)\triangleq\mathcal{Q}(\xi)^{-1}Jq(z(\xi))^Td$ via backward- and forward-substitution: $O\left(\sum_{\ell=1}^{L}N_{\ell-1}N_\ell^2\right)$;
- Final computation given $\widehat{d}(\xi)$: $O\left(S\sum_{\ell=1}^{L}N_{\ell-1}N_\ell\right)$;

Total computational cost for $\nabla^2\chi(\xi)d$: $O\left(\sum_{\ell=1}^{L}\left(SN_{\ell-1}N_\ell+N_{\ell-1}N_\ell^2\right)\right)$.

5. Numerical experiments. In this section, we use two representative piecewise affine activation functions, the difference-of-convex hard sigmoid activation and the convex leaky ReLU activation, to demonstrate the effectiveness of our proposed method. It is worth mentioning that our purpose here is not to compare different activation functions but to illustrate our method's universal applicability to all networks with any kind of piecewise affine activation.

5.1. Design of experiments. All our computations are implemented in MATLAB R2017b. The numerical experiments are conducted on a Mac OS X with an 2.3 GHz Intel Core i7 and 8 GB RAM.

For each choice of the piecewise affine activation function $\{\sigma_\ell\}_{\ell=1}^L$, we generate the output data $\{y_s\}_{s=1}^S$ (for training and testing) from a three-layer neural network under the regression setting:

$$y = \sigma_3\left(W^3\,\sigma_2\left(W^2\,\sigma_1\left(W^1 x + w^1\right) + w^2\right) + w^3\right) + \varepsilon,$$

where $x \overset{\text{iid}}{\sim} \mathcal{N}(\mu, \Sigma)$ and $\varepsilon \overset{\text{iid}}{\sim} \varepsilon_0\,\mathcal{N}(0,1)$. The mean μ of x is generated by spherical Gaussian and the covariance Σ is set to be $\Sigma_0^T\Sigma_0$, where Σ_0 is a random matrix with independent and identically distributed standard Gaussian components. The network architecture has two hidden layers with 20 and 5 neurons, respectively. We terminate the MM algorithm if the difference of objective values given by two consecutive MM iterates is less than 10^{-4} and

$$\begin{aligned}&\text{average feasibility of the current iterate}\\ &\triangleq \left(S\sum_{\ell=1}^{L} N_\ell\right)^{-1}\left[\sum_{s=1}^{S}\sum_{\ell=1}^{L}\sum_{j=1}^{N_\ell}\left|\,u_j^{\nu;s;\ell} - \sigma_\ell\left(W_j^{\nu;\ell}u^{\nu;s;\ell-1} + w_j^{\nu;\ell}\right)\right|\right] \le 10^{-6}.\end{aligned}$$

These double stopping criteria provide approximations of feasibility recovery and stationarity of the problem (2.3); these criteria are in contrast to the hard iteration limit derived from the nonasymptotic rates of simulation based or stochastic gradient schemes for smooth problems. For convenience of the implementation, we take all the activation functions to be the same across all the layers in the network being trained.

We randomly initialize the variable (W, w) and compute the initial value of u based on (1.1) recursively. Starting from the second MM iteration, we warm start the dual subproblem (3.1) by the final dual solution of the previous MM iteration. Throughout our implementation, we also include the extrapolation step of the projected gradient method introduced by Nesterov [30] for faster convergence (henceforth we refer to the resulting implementation as accelerated projected gradient (APG)). Each subproblem is solved by the GP+SN method until $\|\Phi(\bullet)\|$, the norm of the residual function, is less than 10^{-8}; see section 4.1 for the definition of $\Phi(\bullet)$. The proximal parameter δ was set to be 10^{-3} for the numerical experiments in subsection 5.2 and 10^{-5} in subsection 5.3, respectively.

5.2. Numerical results and discussion. Table 1 shows the numerical results of our overall algorithm for training the three-layer neural network with the hard sigmoid activation function (averaged over 100 simulations). The results indicate that our method can effectively bring down the training error and test error of problem (2.5) to a satisfactory level in a little more than a dozen MM iterations with accurate feasibility. A typical performance of the MM algorithm is demonstrated in Figure 1, where we see that the trajectories of training error and objective are basically identical; this demonstrates in particular that feasibility is recovered quite consistently. We also

TABLE 1
Results for the hard sigmoid activation function.

S	d	ε_0	Training error	Test error	Feasibility error	MM iters	SN iters	CG iters	Time (mm:ss)
500	5	5%	2.85-3	2.90-3	3.43-7	10	42	3650	0:35
500	5	10%	6.14-3	6.14-3	2.88-7	12	53	5198	0:51
500	10	5%	3.52-3	3.39-3	3.45-7	11	45	3820	0:41
500	10	10%	6.64-3	6.60-3	2.33-7	13	55	5790	0:55
1000	10	5%	3.63-3	3.65-3	3.50-7	12	50	5187	1:30
1000	10	10%	6.65-3	6.65-3	2.27-7	14	61	6699	1:58
1000	20	5%	3.56-3	3.72-3	3.39-7	12	51	5240	2:01
1000	20	10%	6.56-3	6.44-3	2.21-7	14	62	7145	2:47
2000	20	5%	3.90-3	3.97-3	3.45-7	13	57	8105	6:48
2000	20	10%	7.67-3	7.82-3	2.60-7	14	67	10127	6:52
2000	30	5%	3.50-3	3.51-3	3.59-7	13	52	7878	7:46
2000	30	10%	7.61-3	7.65-3	2.83-7	15	72	10452	8:21

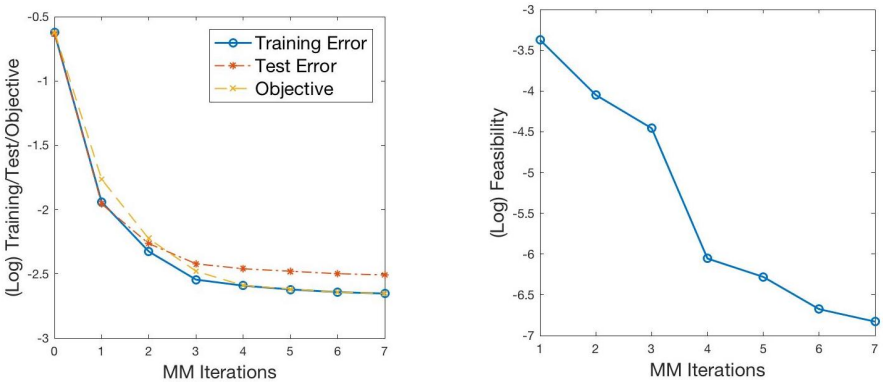

FIG. 1. *A typical performance of the MM algorithm.*

draw a scatter plot for the distribution of the attainable d-stationary points obtained from 100 different random initial points (uniformly distributed between -5 and 5) for a fixed set of training data in Figure 2. One can see from the figure that a high percentage of the initial points leads to an objective value that clusters near the smallest. In order to better understand the scalability of our method in terms of sample size, number of features, and network architectures, we further run our algorithm with a fixed noise level of $\varepsilon_0 = 5\%$ and with (a) a three-layer neural network with $d = 5$ and $(N_1, N_2, N_3) = (20, 5, 1)$ for different sample sizes, and (b) a training dataset with $S = 100$ for different input dimensions d and architectures of the neural networks up to four layers. The corresponding results are reported in Tables 2 and 3. The third column of each table is the average number of APG iterations for each MM subproblem, and the fifth column represents the average number of CG iterations for finding each SN direction.

We also examine the performance of our GP+SN method for solving each dual MM subproblem. The necessity of employing the second-order SN method can be seen from Figure 3 (for the case where $S = 1000, d = 10$, and $\alpha = 0.1$). The first-order APG method decreases fast at the beginning and becomes stagnant after reaching the accuracy around 10^{-3}, whereas it only takes four SN iterations to reach the prescribed accuracy (10^{-8}) if we switch the subproblem solver at the point where the APG stagnates. It is worth emphasizing that highly accurate solutions of the MM subproblems are essential for the overall effectiveness of the MM procedure, which is demonstrated

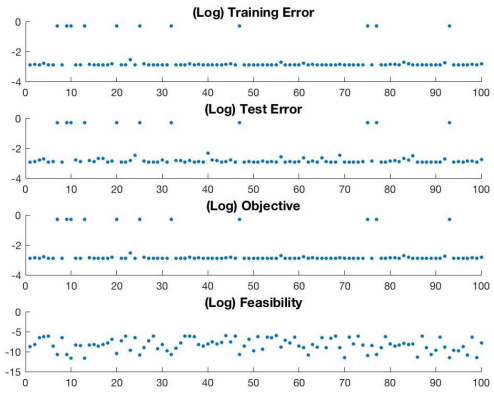

FIG. 2. *Solutions under different initializations.*

TABLE 2
Results for a three-layer neural network with $d = 5$ and different sample sizes.

S	MM iters	APG/MM iters	SN iters	CG/SN iters	APG time (mm:ss)	SN time (mm:ss)	Time (mm:ss)
500	10	552	56	110	0:15	0:19	0:34
1000	11	947	61	166	0:41	0:52	1:33
2000	12	1064	62	149	1:42	1:38	3:20
3000	13	1365	62	187	3:39	3:26	7:05
4000	15	2523	68	195	9:41	5:45	16:28

TABLE 3
Results for different d and network architectures with $S = 100$.

$\sum_{\ell=1}^{L} N_\ell$	d	N_1	N_2	N_3	N_4	MM iters	APG/MM iters	SN iters	CG/SN iters	APG time (mm:ss)	SN time (mm:ss)	time (mm:ss)
11	5	10	1	0	0	12	393	57	35	0:04	0:03	0:07
	5	5	5	1	0	6	372	30	20	0:03	0:02	0:05
	5	4	3	3	1	3	234	13	9	0:01	0:01	0:02
21	8	20	1	0	0	22	480	108	53	0:09	0:08	0:18
	8	10	10	1	0	12	488	62	22	0:07	0:06	0:13
	8	7	7	6	1	4	352	18	96	0:02	0:01	0:04
41	10	40	1	0	0	28	621	141	96	0:23	0:22	0:45
	10	20	20	1	0	28	527	145	84	0:23	0:31	0:55
	10	14	13	13	1	10	643	51	52	0:11	0:11	0:22
101	15	100	1	0	0	18	892	92	61	0:29	0:21	0:50
	15	50	50	1	0	25	984	126	81	1:15	1:15	2:32
	15	34	33	33	1	25	944	121	117	1:10	1:20	2:30
201	20	200	1	0	0	18	896	92	88	2:00	0:42	2:44
	20	100	100	1	0	16	893	80	149	1:44	1:09	2:54
	20	67	67	66	1	15	1257	71	173	1:29	1:09	2:38
301	25	300	1	0	0	16	879	89	82	1:56	0:46	2:44
	25	150	150	1	0	16	2002	78	147	3:12	1:30	4:44
	25	100	100	100	1	18	1447	87	175	3:12	2:20	5:35
401	30	400	1	0	0	12	1010	65	67	2:03	0:41	2:46
	30	200	200	1	0	12	2053	58	144	8:51	2:39	11:33
	30	134	133	133	1	12	1943	62	161	4:08	2:08	6:19
501	35	500	1	0	0	15	1226	77	61	3:48	1:05	4:56
	35	250	250	1	0	13	2582	65	146	14:57	3:46	18:48
	35	167	167	166	1	14	1866	67	162	14:09	4:49	19:04

in Figure 4. For this figure, we take 50 samples and compare the training/test errors under two different stopping criteria for the MM subproblems ($\|\Phi(\bullet)\| < 10^{-8}$ versus $\|\Phi(\bullet)\| < 10^{-3}$). Obviously having a more accurate subproblem solution can make a significant difference in terms of both training and test errors at least for this kind of small datasets. Figure 5 shows another example for the consequence of accurate

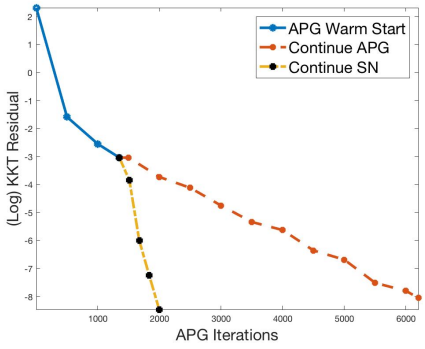

FIG. 3. *Hybrid SN versus pure APG.*

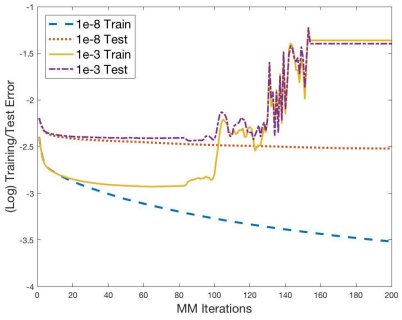

FIG. 4. *High versus low accuracy for* $S = 50$.

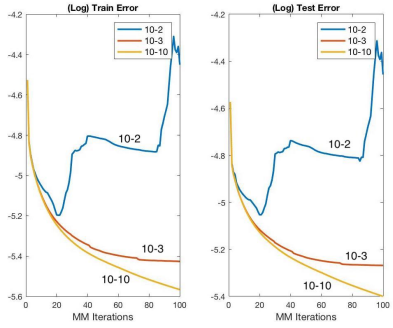

FIG. 5. *High versus low accuracy near convergence for* $S = 250$.

subproblem solutions when we are close to convergence. However, when we conduct the same experiment on a relatively large dataset (e.g., $S = 1000$), we observe that solving the MM subproblems to the accuracy $\|\Phi(\bullet)\| < 10^{-3}$ can yield results almost as good as those based on the accuracy $\|\Phi(\bullet)\| < 10^{-8}$; see Figure 6. This does not necessarily nullify the need for a more accurate MM solution; on the contrary, the results from the latter termination serve as a certificate for us to accept inaccurate MM solutions in practice for faster computations.

5.3. Comparisons with SGD and Adam. We also compare our method with two first-order heuristic methods: the vanilla SGD and the Adam [24]. All the numerical tests of SGD and Adam, regardless of lacking the theoretical guarantees due to the composition of nondifferentiable activation functions, are conducted via Keras (available at https://keras.io) using TensorFlow [1] backend in Python with Glorot uniform initialization and batch-size 10 (step-sizes are automatically tuned).

When the activation function is taken to be the hard sigmoid function, the SGD typically stagnates at some suboptimal points and ceases to make substantial progress. In contrast, starting from the same points, our method provides better solutions with smaller objective values. We summarize this phenomenon in Figures 7 and 8, which correspond to instances with $S = 500$, $d = 5$, and $\varepsilon_0 = 0.05$. For Figure 7, we first run 200 epochs of the SGD, for which it cannot make substantial progress after the 50th epoch. If our method is started from the solution given by the last SGD iteration, it

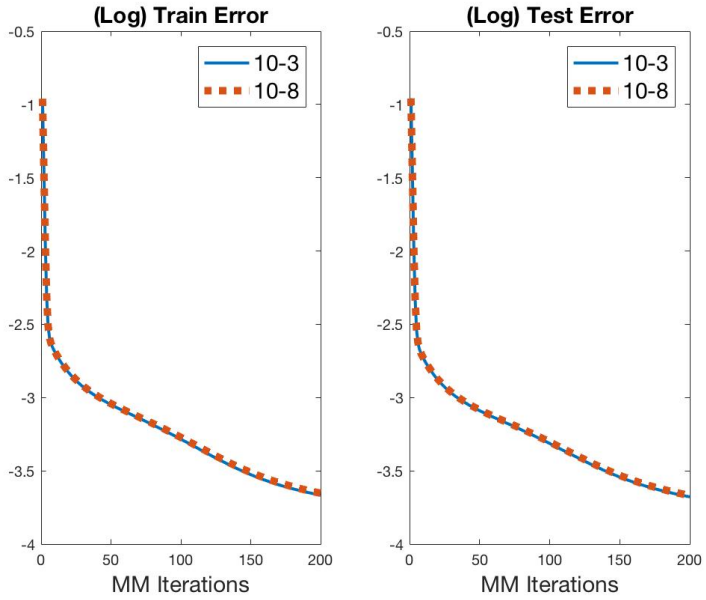

FIG. 6. *High versus low accuracy near convergence for* $S = 1000$.

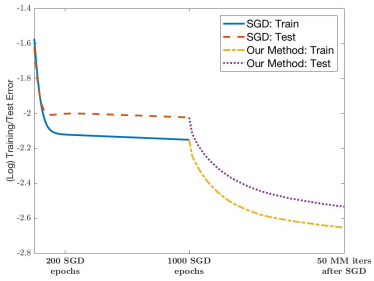

FIG. 7. *Our method can escape the region where SGD gets stuck.*

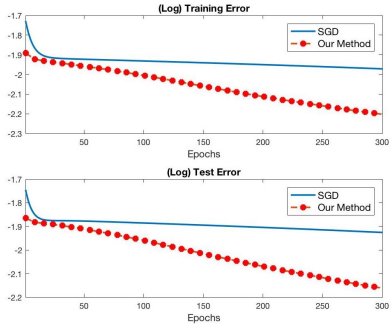

FIG. 8. *Our method versus SGD with the same initialization.*

moves out of the region immediately and eventually converges to a point with much better training and test errors after 100 MM iterations (notice that the MM iterations in our method and epochs in the SGD are of a different nature; here we put them together with proper ratio of scales for convenience of visualization). In Figure 8, we compare our method with the SGD using the same initial point. We run SGD for 300 epochs and our method for 40 MMs simultaneously since the computation time is similar for such numbers of iterations. Obviously our method is faster than the vanilla SGD for this particular case.

When taking the leaky ReLU as the activation function, we observe that the realization of SGD on Keras could crash numerically (i.e., ends up with "NaN" in training/test error) quite often, while our algorithm always works steadily. Table 4 reports the ratio of the crashed SGD cases over 100 test cases under three different settings. Such robustness of our method can be expected, since the MM algorithm

TABLE 4
Vanilla SGD can work poorly for the leaky ReLu activation function.

Cases (S, d, ε_0)	(500, 5, 5%)	(1000, 10, 5%)	(2000, 20, 10%)
SGD crashed/total	20/100	41/100	39/100
Our method crashed/total	0/100	0/100	0/100

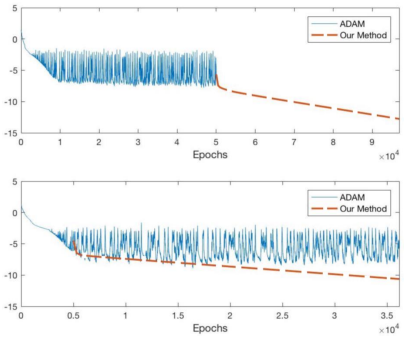

FIG. 9. *Log scaled training error for* $S = 30$.

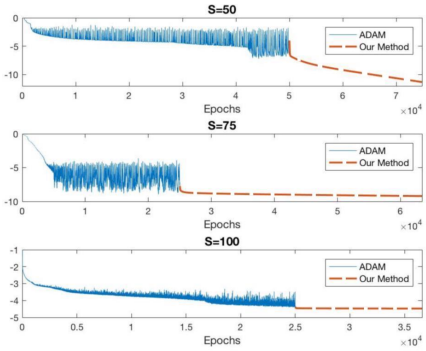

FIG. 10. *Log scaled training error for larger* S.

is proved to converge to a d-stationary point of the penalized problem (2.5) and a d-stationary point of the original problem (2.3) when the feasibility is achieved.

If we employ leaky ReLU as activations, our method can obtain better solutions compared with Adam when we are working with relatively small datasets. Figure 9 shows two results for the case where $S = 30$, $d = 5$, and $\varepsilon_0 = 0.05$. In both results, we first ran Adam for certain numbers of epochs. In the first result, Adam made no improvement after 50,000 epochs. At that point, we stopped Adam and ran 400 MM iterations. From the figure we see that the training error goes down. In the second subfigure, Adam started to oscillate after 5000 epochs. At that point, we let Adam continue to run and we independently ran 400 MM iterations. We can see that the MM algorithm can decrease the training error whereas Adam continues to oscillate without decreasing the error. A similar behavior can also be observed from the first subplot of Figure 10, where $S = 50$, $d = 5$, and $\varepsilon_0 = 0.05$. However, when we increased the number of samples to $S = 75$ or $S = 100$, as shown in the second and third subplots of Figure 10, our method with 800 and 200 MM iterations respectively did not decrease the training error significantly after we stopped Adam. In all cases, our method can serve as a theoretical support that Adam produces approximate d-stationary solutions.

In summary, we have sound evidence that our method, when implemented properly, can outperform the vanilla SGD with better and more robust solutions. On the other hand, even if it is overall difficult at this time for us to surpass Adam's practical performance, by bringing theoretical certificates to empirical heuristics, we can still benefit from our method with its rigorous analysis, which after all is a key contribution of our paper.

Acknowledgments. The authors are grateful to Defeng Sun at the Hong Kong Polytechnic University and Kim-Chuan Toh at the National University of Singapore for discussions on this paper. They are particularly indebted to Dr. Toh for his review of our MATLAB codes for accuracy and possible improvements. We also thank two referees for constructive comments that have improved the presentation of the paper.

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, et al., TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, tensorflow.org, 2015.

[2] D.P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1999.

[3] L. Bottou, F. Curtis, and J. Nocedal, *Optimization methods for large-scale machine learning*, SIAM Rev., 60 (2018), pp. 223–311.

[4] R. Collobert and S. Bengio, *Links between perceptrons, MLPs and SVMs*, in Proceedings of the 21st International Conference on Machine learning, 2004.

[5] Y. Cui, J.S. Pang, and B. Sen, *Composite difference-max programs for modern statistical estimation problems*, SIAM J. Optim., 28 (2018), pp. 3344–3374.

[6] Y. Cui, D. Sun, and K.C. Toh, *Computing the best approximation over the intersection of a polyhedral set and the doubly nonnegative cone*, SIAM J. Optim., 29 (2019), pp. 2785–2813.

[7] D. Davis and D. Drusvyatskiy, *Stochastic model-based minimization of weakly convex functions*, SIAM J. Optim., 29 (2019), pp. 207–239.

[8] D. Davis, D. Drusvyatskiy, S. Kakade, and J. Lee, *Stochastic subgradient method converges on tame functions*, Found. Comput. Math., 20 (2020), pp. 119–154.

[9] V.F. Demyanov, G. Di Pillo, and F. Facchinei, *Exact penalization via Dini and Hadamard conditional derivatives*, Optim. Methods Softw., 9 (1998), pp. 19–36.

[10] G. Di Pillo and F. Facchinei, *Exact penalty functions for nondifferentiable programming problems*, in Nonsmooth Optimization and Related Topics, Springer, New York, 1989, pp. 89–107.

[11] F. Facchinei and L. Lampariello, *Partial penalization for the solution of generalized Nash equilibrium problems*, J. Global Optim., 50 (2011), pp. 39–50.

[12] F. Facchinei and J.S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems*, Springer, New York, 2003.

[13] C.A. Floudas, *Deterministic Global Optimization: Theory, Methods, and Applications*, Nonconvex Optim. Appl. 37, Springer, New York, 2000.

[14] W. Gao, D. Goldfarb, and F. Curtis, *ADMM for multiaffine constrained optimization*, Optim. Methods Softw., 35 (2020), pp. 257–303.

[15] S. Ghadimi and G. Lan, *Stochastic first- and zeroth-order methods for nonconvex stochastic programming*, SIAM J. Optim., 23 (2013), pp. 2341–2368.

[16] X. Glorot, A. Bordes, and Y. Bengio, *Deep sparse rectifier neural networks*, in Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.

[17] G.H. Golub and C.F. van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 2013.

[18] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, Vol. 1, MIT Press, Cambridge, MA, 2016.

[19] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, *Maxout networks*, in Proceedings of the 30th International Conference on Machine Learning, 2013, pp. 1319–1327.

[20] G. E. Hinton, S. Osindero, and Y.-W. Teh, *A fast learning algorithm for deep belief nets*, Neural Comput., 18 (2006), pp. 1527–1554.

[21] R.A. Horn and C.R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1991.

[22] K. Hornik, M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators*, Neural Networks, 2 (1989), pp. 359–366.

[23] K. Jarrett, K. Kavukcuoglu, and Y. LeCun, *What is the best multi-stage architecture for object recognition?*, in Proceedings of the IEEE 12th International Conference on Computer Vision, (2009), pp. 2146–2153.

[24] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in Proceedings of the International Conference on Learning Representations, Vol. 5, 2015.

[25] T. Lau, J. Zeng, B. Wu, and Y. Yao, *A proximal block coordinate descent algorithm for deep neural network training*, in Proceedings of the Workshop of the 6th International Conference on Learning Representations, 2018.

[26] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature, 521 (2015), pp. 436–444.

[27] M. Leshno, V.Y. Lin, A. Pinkus, and S. Schocken, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*, Neural Networks, 6 (1993), pp. 861–867.

[28] A.L. Maas, A.Y. Hannun, and A.Y. Ng, *Rectifier nonlinearities improve neural network acoustic models*, in Proceedings of the 30th International Conference on Machine Learning, 2013.

[29] A. Nemirovsky and D. Yudin, *Problem Complexity and Method Efficiency in Optimization*, John Wiley & Sons, New York, 1983.
[30] Y. Nesterov, *A method of solving a convex programming problem with convergence rate* $O(1/k^2)$, in Soviet Math. Dokl., 27 (1983), pp. 372–376.
[31] J.S. Pang and L. Qi, *A globally convergent Newton method for convex* SC^1 *minimization problems*, J. Optim. Theory Appl., 85 (1995), pp. 633–648.
[32] J.S. Pang, M. Razaviyayn, and A. Alvarado, *Computing B-stationary points of nonsmooth DC programs*, Math. Oper. Res., 42 (2016), pp. 95–118.
[33] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. Devito, Z. Lin, A. Desmaison, L. Antiga, and A. Lever, *Automatic Differentiation in PyTorch*, https://pytorch.org, 2017,
[34] D.T. Pham and H.W. Le Thi, *Convex analysis approach to DC programming: Theory, algorithm and applications*, Acta Math. Vietnam., 22 (1997), pp. 289–355.
[35] J.R. Quinlan, *Combining instance-based and model-based learning*, in Proceedings of the 10th International Conference on Machine Learning, 1993, pp. 236–243.
[36] H. Robbins and S. Monro, *A stochastic approximation method*, Ann. Math. Statist., 22 (1951), pp. 400–407.
[37] K. Schäcke, *On the Kronecker Product*, https://www.math.uwaterloo.ca/~hwolkowi/henry/reports/kronthesisschaecke04.pdf, 2013.
[38] J. Schmidhuber, *Deep learning in neural networks: An overview*, Neural Networks, 61 (2015), pp. 85–117.
[39] S. Scholtes, *Introduction to Piecewise Differentiable Equations*, Springer Briefs Optim., Springer, New York, 2002.
[40] 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, 2016, pp. 2722–2731.
[41] Z. Zhang and B. Matthew, *Convergent block coordinate descent for training Tikhonov regularized deep neural networks*, in Proceedings of the 31st Conference on Neural Information Processing System, 2017, pp. 1721–1730.
[42] Z. Zhang, Y. Chen, and V. Saligrama, *Efficient training of very deep neural networks for supervised hashing*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1487–1495.