DANTE: Deep AlterNations for Training nEural networks

Vaibhav B Sinha[a,1,2,*], Sneha Kudugunta[a,1], Adepu Ravi Sankar[a], Surya Teja Chavali[a], Purushottam Kar[b], Vineeth N Balasubramanian[a],

[a]*Department of Computer Science and Engineering, Indian Institute of Technology Hyderabad, India*
[b]*Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, India*

Abstract

We present DANTE, a novel method for training neural networks using the alternating minimization principle. DANTE provides an alternate perspective to traditional gradient-based backpropagation techniques commonly used to train deep networks. It utilizes an adaptation of quasi-convexity to cast training a neural network as a bi-quasi-convex optimization problem. We show that for neural network configurations with both differentiable (e.g. sigmoid) and non-differentiable (e.g. ReLU) activation functions, we can perform the alternations effectively in this formulation. DANTE can also be extended to networks with multiple hidden layers. In experiments on standard datasets, neural networks trained using the proposed method were found to be promising and competitive to traditional backpropagation techniques, both in terms of quality of the solution, as well as training speed.

Keywords: Neural nets, Deep Learning, Backpropagation, Machine Learning.

1. Introduction

For much of the recent march of deep learning, gradient-based backpropagation methods, e.g. Stochastic Gradient Descent (SGD) and its variants, have been the mainstay of practitioners. The use of these methods, especially on vast amounts of data, has led to unprecedented progress in several areas of artificial intelligence in recent years. The intense focus on these techniques has led to an intimate understanding of hardware requirements and code optimizations needed to execute these routines on large datasets in a scalable manner. Today, myriad off-the-shelf and highly optimized packages exist that can churn reasonably large datasets on GPU architectures with relatively mild human involvement and little bootstrap effort.

However, this surge of success of backpropagation-based methods in recent years has somewhat overshadowed the need to continue to look for options beyond backpropagation to train deep networks. Despite several advancements in deep learning with respect to novel architectures such as encoder-decoder networks, generative adversarial models and transformer networks, the reliance on backpropagation methods remains. Several works have studied the limitations of SGD-based backpropagation, whether it be vanishing gradients, especially for certain activation functions [1]; the tendency of SGD to face difficulties with saddle points [2] - even for simple architectures [3]; or even more subtle issues such as significant difference in training time for networks having same expressive power as seen in [4]. Importantly, while existing backpropagation-based methods work, it is essential to continuously look for alternative methods that can help train neural networks effectively.

Complementarily, there has been marked progress in recent years in the broader area of non-convex optimization. Several alternate algorithms with provable guarantees, such as iterative hard thresholding [5], alternating minimization [6], [7] and [8]. In this work, we leverage recent developments in optimization (quasi-convex, to be precise) to propose a non-backpropagation strategy to train neural networks. Our method is called DANTE (Deep AlterNations for Training nEural networks), and it offers an alternating minimization-based technique for training neural networks. There have been a few related efforts of late, which we review in Section 2.

DANTE is based on the simple but useful observation that the problem of training a single hidden-layer neural network can be cast as a bi-quasiconvex optimization problem (described in Section 3.1). This observation allows us to use an alternating optimization strategy to train the neural network, where each step involves solving relatively simpler quasi-convex problems. DANTE then uses efficient solvers for quasi-convex problems such as stochastic normalized gradient descent [9] to train the neural network using alternating minimization. The key original contributions of this work can be summarized as:

- We show that the error in each layer of a neural network can, in fact, be viewed as a quasi-convex function, thus allowing us to treat a single hidden-layer neural network

---

*Corresponding author

Email addresses: `cs15btech11034@iith.ac.in` (Vaibhav B Sinha), `snehakudugunta@google.com` (Sneha Kudugunta), `cs14resch11001@iith.ac.in` (Adepu Ravi Sankar), `chavali2@wisc.edu` (Surya Teja Chavali), `purushot@cse.iitk.ac.in` (Purushottam Kar), `vineethnb@iith.ac.in` (Vineeth N Balasubramanian)

[1]Authors contributed equally

[2]Currently at Department of Computer Science, University of Texas at Austin, US

as a bi-quasi-convex optimization problem. Motivated by recent work [9], this allows us to propose an alternating minimization strategy, DANTE , where each quasi-convex optimization problem can be solved effectively (using Stochastic Normalized Gradient Descent (SNGD)).

- While earlier results on the effectiveness of SNGD for solving a quasi-convex problem was restricted to a simple sigmoid Generalized Linear Model (GLM) with the squared loss, we show that SNGD can converge in high probability to an ϵ-suboptimal solution even in case of layers of a neural network. We also expand the scope to include Rectified Linear Units (ReLU) activation functions and its variants by introducing a Generalized ReLU activation function. We also provide theoretical results in case of networks using Cross Entropy as loss (all of which has not been done before).

- We show DANTE can be extended to train deep neural networks with multiple hidden layers.

- We empirically validate DANTE with both the generalized ReLU and sigmoid activations and establish that DANTE provides competitive or better performance on several standard datasets, when compared to standard mini-batch SGD-based backpropagation.

- While the high level idea of using the definition to prove the quasi-convexity of functions is inspired by [9], we would like to highlight that our proofs are more involved, and use techniques that were not in [9] (to adapt to newer activation functions, handle hidden layers, as well as multiple output neurons).

We now review earlier related efforts, before presenting details of the proposed methodology.

2. Related Work

Backpropagation-based techniques date back to the early days of neural network research [10, 11] but remain to this day, the most commonly used methods for training a variety of neural networks including multi-layer perceptrons, convolutional neural networks, autoencoders, recurrent networks and the like.

In recent years, Taylor *et al.* [12] and Choromanska *et al.* [13] proposed methods to train neural networks which belong to the broad framework of alternating-minimization. Although both of these approaches use alternating-minimization, they are fundamentally different from ours. Both of them use auxiliary variables and the minimization is done on these auxiliary variables too, while our algorithm does not have any auxiliary variables and only minimizes on the weights of the network, giving it significant advantages in space complexity and training-time. Moreover, Taylor's algorithm does not use gradients unlike ours. Jaderberg proposed the idea of 'synthetic gradients' in [14]. Although interesting, this work is more focused towards a more efficient way to carry out gradient-based parameter updates in a neural network. More recently, Jagatap and Hegde [15] proposed a method to train single hidden layer ReLU networks using an alternating minimization technique.

Unlike our method, this method alternates between updating weights, and state variables which indicate which ReLU activations are on, and so is very specific to ReLU activations.

In this work, we focus on an entirely new approach to training neural networks using alternating optimization inspired by quasi-convexity (different from the abovementioned methods), and show that this approach shows promising results to train neural networks of different depths on a range of datasets. Although alternating minimization has found much appeal in areas such as matrix factorization ([6]), to the best of our knowledge, this is the one of the early efforts in using alternating principles to train feedforward neural networks effectively.

Other efforts that are related to this work include target propagation based methods, such as in [16], Difference Target Propagation [17] and target propagation in a Bayesian setting [18]. There are also efforts that use random feedback weights such as feedback-alignment [19] and direct/indirect feedback-alignment[20] where the weights used for propagation need not be symmetric with the weights used for forward propagation. We however do not focus on credit assignment in this work. One could view the proposed method however as carrying out 'implicit' credit assignment using partial derivatives, but there is no defined model for credit assignment which is not the focus of this work. We now describe our methodology.

3. Deep AlterNations for Training nEural networks (DANTE)

We begin by presenting the overall problem formulation.

3.1. Problem Formulation

Consider a neural network with L layers. Each layer $l \in \{1, 2, \ldots, L\}$ has n_l nodes and is characterized by a linear operator $W_l \in \mathbb{R}^{n_{l-1} \times n_l}$ and a non-linear activation function defined as $\phi_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$. The activations generated by layer l are denoted by $\mathbf{a}_l \in \mathbb{R}^{n_l}$. We denote by $\mathbf{a}_0$, the input activations and n_0 to be the number of input activations i.e. $\mathbf{a}_0 \in \mathbb{R}^{n_0}$. Each layer uses activations being fed into it to compute its own activations as $\mathbf{a}_l = \phi_l\langle W_l, \mathbf{a}_{l-1}\rangle \in \mathbb{R}^{n_l}$, where $\phi\langle., .\rangle$ denotes $\phi(\langle., .\rangle)$ for simplicity of notation. A multi-layer neural network is formed by nesting such layers to form a composite function f given as follows:

$$f(\mathbf{W}; \mathbf{x}) = \phi_L\langle W_L, \phi_{L-1}\langle W_{L-1}, \cdots, \phi_1\langle W_1, \mathbf{x}\rangle\rangle \quad (1)$$

where $\mathbf{W} = \{W_l\}$ is the collection of all the weights through the network, and $\mathbf{x} = \mathbf{a}_0$ contains the input activations for each training sample.

Given m data samples $\{(\mathbf{x_i}, y_i)\}_{i=1}^m$ from a distribution $\mathcal{D}$ and a loss function J, the network is trained by tuning the weights $\mathbf{W}$ to minimize the empirical risk associated with:

$$\min_{\mathbf{W}} \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}}[J(f(\mathbf{W}; \mathbf{x}), y)] \quad (2)$$

For purpose of simplicity and convenience, we first consider the case of a single hidden layer neural network, represented as $f(\mathbf{W}; \mathbf{x}) = \phi_2\langle W_2, \phi_1\langle W_1, \mathbf{x}\rangle\rangle$ to describe our methodology. We later describe how this can be extended to multi-layer neural

networks. A common loss function used to train neural networks is the squared loss function which yields the following objective (we later also study changing the loss function in this work):

$$\min_{\mathbf{W}} \ \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}} \|f(\mathbf{W};\mathbf{x}) - \mathbf{y}\|_2^2 \qquad (3)$$

where:

$$\|f(\mathbf{W};\mathbf{x}) - \mathbf{y}\|_2^2 = \|\phi_2\langle W_2, \phi_1\langle W_1, \mathbf{x}\rangle\rangle - \mathbf{y}\|_2^2 \qquad (4)$$

An important observation here is that if we fix W_1, then Eq. (4) turns into a set of Generalized Linear Model (GLM) problems with ϕ_2 as the activation function, i.e.

$$\min_{W_2} \ \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}} \|\phi_2\langle W_2, \mathbf{z}\rangle - \mathbf{y}\|_2^2 \qquad (5)$$

where $\mathbf{z} = \phi_1\langle W_1, \mathbf{x}\rangle$. In particular, a GLM with differentiable activation functions such as sigmoid satisfy a property called *Strict Locally Quasi-Convexity* (SLQC), which allows techniques such as SNGD to solve the GLM problem effectively, as pointed out in [9]. We exploit this observation, and generalize this result in many ways: (i) we firstly show that a GLM with non-differentiable activation functions such as ReLUs (and its variants) also satisfy the SLQC property; (ii) we show that a set of GLMs, such as in a layer of a neural network, also satisfy the SLQC property; and (iii) we leverage these generalizations to develop an alternating minimization methodology to train neural networks; and (iv) we also move away from the restriction of square loss and provide extensions for Cross-Entropy Loss. Sections 3.2 and 3.3 describe these generalizations further.

Optimizing for W_1 in Equation 4, unfortunately, cannot be directly viewed as a set of SLQC GLMs. To this end, we provide a generalization of local quasi-convexity in Section 3.2 and show that fixing W_2 does indeed turn the problem below into yet another SLQC problem, this time with W_1 as the parameter (note that $\phi_{W_2}\langle\cdot\rangle = \phi_2\langle W_2, \phi_1\langle\cdot\rangle\rangle$):

$$\min_{W_1} \ \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}} \|\phi_{W_2}\langle W_1, \mathbf{x}\rangle - \mathbf{y}\|_2^2 \qquad (6)$$

Putting Equations 5 and 6 together gives us a single-layer neural network setup, where each layer is individually SLQC, and can be efficiently solved using SNGD. This allows us to propose our alternating minimization strategy to train a neural network in an effective manner. We now describe in detail each of the above steps, beginning with the background and preliminaries required to set up this problem.

3.2. Background and Preliminaries

Let $\|\cdot\|$ denote the L_2 (Euclidean) norm for vectors, and $\|\cdot\|_F$ denote the Frobenius norm of a matrix. We sometimes drop the subscript F from $\|\cdot\|_F$ for ease of reading (the appropriate norm can be identified from the context). $\mathbb{B}(\mathbf{x}, r)$ denotes a Euclidean ball of radius r with $\mathbf{x}$ as center and $\mathbb{B}$ denotes $\mathbb{B}(0, 1)$. We begin with the formal definitions of Local Quasi-Convexity and Generalized Linear Model (GLM).

Definition 1 (*Local-Quasi-Convexity* [9])**.** Let $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d, \kappa, \epsilon > 0$ and $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function. Then f is said to be $(\epsilon, \kappa, \mathbf{z})$-Strictly-Locally-Quasi-Convex (SLQC) in $\mathbf{x}$, if at least one of the following applies:

1. $f(\mathbf{x}) - f(\mathbf{z}) \leq \epsilon$
2. $\|\nabla f(\mathbf{x})\| > 0$, and $\forall \mathbf{y} \in \mathbb{B}(\mathbf{z}, \epsilon/\kappa), \langle\nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x}\rangle \leq 0$

where $\mathbb{B}(\mathbf{z}, \epsilon/\kappa)$ is a ball centered at $\mathbf{z}$ with radius ϵ/κ.

Definition 2 (*Idealized and Noisy Generalized Linear Model* [9])**.** In the idealized GLM setting, we are given m samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^m \in \mathbb{B} \times [0, 1]$ and an activation function $\phi : \mathbb{R} \to \mathbb{R}$. There exists $\mathbf{w}^* \in \mathbb{R}^d$ such that $y_i = \phi\langle\mathbf{w}^*, \mathbf{x}_i\rangle \forall i \in \{1, \cdots, m\}$ where $\mathbf{w}^*$ is the global minimizer of the empirical error function:

$$\hat{err}(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^m (y_i - \phi(\langle\mathbf{w}, \mathbf{x}_i\rangle))^2$$

In the noisy GLM setting, we are given m samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^m \in \mathbb{B}_d \times [0, 1]$ drawn i.i.d. from an unknown distribution $\mathcal{D}$. There exists a $\mathbf{w}^* \in \mathbb{R}^d$ such that $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[y|\mathbf{x}] = \phi(\langle\mathbf{w}^*, \mathbf{x}\rangle)$, and $\mathbf{w}^*$ is the global minimizer of :

$$err(\mathbf{w}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} (y - \phi(\langle\mathbf{w}, \mathbf{x}\rangle))^2$$

Hazan et al. showed in [9] that the idealized GLM problem with the sigmoid activation function is $(\epsilon, e^{\|\mathbf{w}^*\|}, \mathbf{w}^*)$-SLQC in $\mathbf{w}$, $\forall \mathbf{w} \in \mathbb{B}(0, \|\mathbf{w}^*\|)$ and $\forall \epsilon > 0$; and that if we draw $m \geq \Omega\left(\frac{\exp(2\|\mathbf{w}^*\|)}{\epsilon^2}\log\frac{1}{\delta}\right)$ i.i.d. samples from $\mathcal{D}$, the empirical error function $\hat{err}$ with sigmoid activation is $(\epsilon, e^{\|\mathbf{w}^*\|}, \mathbf{w}^*)$-SLQC in $\mathbf{w}$ for any $\mathbf{w} \in \mathbb{B}(0, \|\mathbf{w}^*\|)$ with probability at least $1 - \delta$. However, these results by themselves are not directly useful, considering they are proved only for a single GLM (which can be viewed as a neural network with no hidden layers and a single output neuron), and which are non-trivial to extend to a traditional multi-layer/feedforward neural network. Besides, their proofs rely on properties of the sigmoid function, which restricts us from using these (and any following) results to contemporary neural networks which use other activation functions such as the ReLU. We overcome all of these restrictions in this work, and provide a new mechanism to use such a theoretical result in practice.

3.2.1. SLQC-ness of a GLM with Non-Linear Activations

Before presenting our approach with multi-layer neural networks, we begin our description of the proposed methodology by showing that a GLM with a ReLU activation function is also SLQC ([9] already showed this for a GLM with sigmoid activation). This will later allow us to seamlessly extend our results to both sigmoid and ReLU multi-layer neural networks. (We note that tanh - which is simply a rescaled sigmoid - is also subsumed in these definitions.) To this end, we introduce a new *generalized ReLU* activation function, defined as follows:

Definition 3. (*Generalized ReLU*) The generalized ReLU function $f : \mathbb{R} \to \mathbb{R}, 0 < a \leq b, a, b \in \mathbb{R}$ is defined as:

$$f(x) = \begin{cases} ax & x \leq 0 \\ bx & x > 0 \end{cases}$$

Note that this definition subsumes variants of ReLU such as the Leaky ReLU [21] or PReLU [22]. This function is differentiable at every point except 0. We define the function g that provides a valid subgradient for the generalized ReLU at all x to be:

$$g(x) = \begin{cases} a & x < 0 \\ b & x \geq 0 \end{cases}$$

3

We now prove our first results using the above definition of the generalized ReLU in idealized and noisy GLMs below.

Theorem 1. *In the idealized GLM with generalized ReLU activation, assuming $\|\mathbf{w}^*\| \leq W$, $e\hat{r}r(\mathbf{w})$ is $\left(\epsilon, \frac{2b^3W}{a}, \mathbf{w}^*\right) - SLQC$ in $\mathbf{w}$, $\forall \mathbf{w} \in \mathbb{B}(0, W)$ and $\forall \epsilon > 0$.*

Proof Sketch. We use Definition 1 to show this result. Consider a point $\mathbf{v}$, ϵ/κ-close to minima $\mathbf{w}^*$ with $\kappa = \frac{2b^3W}{a}$. Throughout the paper, we measure closeness in L2-norm. Let G represent the subgradient of $e\hat{r}r_m(\mathbf{w})$. We show that $\langle G(\mathbf{w}), \mathbf{w} - \mathbf{v} \rangle \geq 0$, which proves the result. To show this inequality, we exploit the Lipschitzness of ReLU function, the bound on its derivative and the fact that $\phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle = y_i$. The complete proof is presented in Section 5.1 for ease of reading further at this time. $\square$

Theorem 2. *In the noisy GLM with generalized ReLU activation, assuming $\|\mathbf{w}^*\| \leq W$, given $\mathbf{w} \in \mathbb{B}(0, W)$, then with probability $\geq 1 - \delta$ after $m \geq O(\log(1/\delta)/\epsilon^2)$ samples, $e\hat{r}r(\mathbf{w})$ is $\left(\epsilon, \frac{2b^3W}{a}, \mathbf{w}^*\right) - SLQC$ in $\mathbf{w}$.*

Proof. Please see Section 5.3 for the proof. $\square$

The above theorems, in combination with the results in [9], allow us to conclude that for a single-output no-hidden-layer neural network with sigmoid or ReLU activation, the error function, $e\hat{r}r$, is SLQC in $\mathbf{w}$. This is however not directly useful for neural networks, as stated earlier. To this end, we propose a new extension of SLQC relevant to a set of GLMs, such as in a layer of a neural network. We note that all of the following sections are novel contributions, which did not exist earlier.

3.2.2. SLQC-ness of a Multi-Output Neural Network with No Hidden Layers

We begin with a revised definition of Local Quasi-Convexity for matrices using the Frobenius inner product.

Definition 4 (*Local Quasi-Convexity for Matrices*). Let $\mathbf{x}, \mathbf{z} \in \mathbb{R}^{d \times d'}$, $\kappa, \epsilon > 0$ and $f : \mathbb{R}^{d \times d'} \to \mathbb{R}$ be a differentiable function. Then f is $(\epsilon, \kappa, \mathbf{z})$-Strictly Locally Quasi-Convex (SLQC) in $\mathbf{x}$, if at least one of the following applies:

1. $f(\mathbf{x}) - f(\mathbf{z}) \leq \epsilon$
2. $\|\nabla f(\mathbf{x})\| > 0$, and $\forall \mathbf{y} \in \mathbb{B}(\mathbf{z}, \epsilon/\kappa)$, $\langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle_F \leq 0$

where $\mathbb{B}(\mathbf{z}, \epsilon/\kappa)$ is a ball centered at $\mathbf{z}$ with radius ϵ/κ. ($\langle \cdot, \cdot \rangle_F$ denotes the Frobenius inner product.)

We now show that the error, $e\hat{r}r(\mathbf{W})$, of a multi-output no-hidden-layer neural network is also SLQC in $\mathbf{W}$. (Note that w.r.t. our problem setup in Equation 4, this is equivalent to showing that the one-hidden layer neural network problem is SLQC in W_2 alone.) The empirical error function is now given by:

$$e\hat{r}r(\mathbf{W}) = \frac{1}{m} \sum_{i=1}^{m} \|\mathbf{y_i} - \phi(\langle \mathbf{W}, \mathbf{x_i} \rangle)\|^2$$

where $\mathbf{x_i} \in \mathbb{R}^d$ is the input, $\mathbf{y_i} \in \mathbb{R}^{d'}$ is the corresponding correct output, $\mathbf{W} \in \mathbb{R}^{d \times d'}$ is the matrix of weights and ϕ is applied element-wise in the multi-output no-hidden-layer neural network. Let the global minimizer of $e\hat{r}r$ be $\mathbf{W}^*$.

Theorem 3. *Let an idealized single-layer multi-output neural network be characterized by a linear operator $\mathbf{W} \in \mathbb{R}^{d \times d'} = [\mathbf{w}_1 \ \mathbf{w}_2 \ \cdots \ \mathbf{w}_{d'}]$ and a generalized ReLU activation function be applied element-wise $\phi : \mathbb{R}^{d'} \to \mathbb{R}^{d'}$. Let the output of the layer be $\phi\langle \mathbf{W}, \mathbf{x} \rangle \in \mathbb{R}^{d'}$ where $\mathbf{x} \in \mathbb{R}^d$ is the input. Assuming $\|\mathbf{W}^*\|_F \leq W$, $e\hat{r}r(\mathbf{W})$ is $\left(\epsilon, \frac{2b^3W}{a}, \mathbf{W}^*\right) - SLQC$ in $\mathbf{W}$ for all $\mathbf{W} \in \mathbb{B}_d(0, W)$ and $\epsilon > 0$.*

Proof Sketch. To show this result, we use Definition 4. Let $\mathbf{V} = [\mathbf{v}_1 \ \mathbf{v}_2 \cdots \ \mathbf{v}_{d'}]$ be a point ϵ/κ-close to minima $\mathbf{W}^*$ with $\kappa = \frac{2b^3W}{a}$. Let $G(\mathbf{W})$ be the subgradient of $e\hat{r}r_m(\mathbf{W})$. Then we show that $\langle G(\mathbf{W}), \mathbf{W} - \mathbf{V} \rangle_F \geq 0$, thus proving the result. Section 5.5 presents the complete proof. $\square$

3.2.3. One Hidden Layer Networks with Single Output Neurons

Taking this further, we next consider a single-hidden-layer neural network. While the outer layer (layer 2) of a single-hidden-layer neural network can be directly viewed as a set of GLMs (see section 5.4), the inner layer cannot be viewed the same way (due to lack of expected outputs in the hidden layer). We hence need to show that given a fixed $\mathbf{w_2}$, the error $e\hat{r}r(\mathbf{W_1}, \mathbf{w_2})$ is also SLQC in $\mathbf{W_1}$. In this case, the empirical error function is:

$$e\hat{r}r(\mathbf{W_1}, \mathbf{w_2}) = \frac{1}{m} \sum_{i=1}^{m} \|y_i - \phi_2\langle \mathbf{w_2}, \phi_1\langle \mathbf{W_1}, \mathbf{x_i} \rangle \rangle\|^2$$

where $\mathbf{x_i} \in \mathbb{R}^d$ is the input; $y_i \in \mathbb{R}$ is the corresponding correct output; and $\mathbf{W_1} \in \mathbb{R}^{d \times d'}$, $\mathbf{w_2} \in \mathbb{R}^{d'}$ are the weights of the inner and outer layers respectively. Let the global minimizer of $e\hat{r}r$ be $(\mathbf{W_1^*}, \mathbf{w_2^*})$. This setting corresponds to the inner layer of single-output single-hidden-layer neural network.

Theorem 4. *Let an idealized two-layer neural network be characterized by linear operators $\mathbf{W_1} \in \mathbb{R}^{d \times d'}$, $\mathbf{w_2} \in \mathbb{R}^{d'}$ and generalized ReLU activation functions $\phi_1 : \mathbb{R}^{d'} \to \mathbb{R}^{d'}$, $\phi_2 : \mathbb{R} \to \mathbb{R}$. Assuming $\|\mathbf{W_1^*}\|_F \leq W_1$, $\|\mathbf{w_2^*}\| \leq W_2$, $e\hat{r}r(\mathbf{W_1}, \mathbf{w_2})$ is $\left(\epsilon, \left(\frac{a}{4b^5W_2^2W_1} - \frac{W_1}{\epsilon}\right)^{-1}, \mathbf{W_1^*}\right) - SLQC$ in $\mathbf{W_1}$, $\forall \mathbf{W_1} \in \mathbb{B}(0, W_1)$ and $\forall \epsilon > 0$.*

Proof Sketch. We again use Definition 4 to prove the result. Let $\mathbf{V_1}$ be a point $\frac{\epsilon}{\kappa}$ close to minima. We show that $\langle \nabla_{\mathbf{W_1}} e\hat{r}r(\mathbf{W_1}, \mathbf{w_2}), \mathbf{W_1} - \mathbf{V_1} \rangle_F \geq 0$. As a consequence of Definition 4, this proves the result. See Section 5.7 for the complete proof. $\square$

3.2.4. One Hidden Layer Networks with Multiple Output Neurons

The above results together postulate that a single-hidden-layer neural network is layer-wise SLQC. We use the above result to show that error $e\hat{r}r(\mathbf{W_1}, \mathbf{W_2})$ is SLQC in $\mathbf{W_1}$ for single-hidden-layer neural network, even with multiple outputs. The empirical error function in this setting is:

$$e\hat{r}r(\mathbf{W_1}, \mathbf{W_2}) = \frac{1}{m} \sum_{i=1}^{m} \|\mathbf{y_i} - \phi_2\langle \mathbf{W_2}, \phi_1\langle \mathbf{W_1}, \mathbf{x_i} \rangle \rangle\|^2$$

4

where $\mathbf{x_i} \in \mathbb{R}^d$ is the input, $\mathbf{y_i} \in \mathbb{R}^{d''}$ is the corresponding correct output, $\mathbf{W_1} \in \mathbb{R}^{d \times d'}$, $\mathbf{W_2} \in \mathbb{R}^{d' \times d''}$ are the weights of the inner and outer layers respectively. Let the global minimizer of $e\hat{r}r$ be $(\mathbf{W_1^*}, \mathbf{W_2^*})$. This setting corresponds to the inner layer of a multi-output single-hidden-layer neural network.

Theorem 5. *Let an idealized two-layer neural network be characterized by linear operators $\mathbf{W_1} \in \mathbb{R}^{d \times d'}$, $\mathbf{W_2} \in \mathbb{R}^{d' \times d''}$ and generalized ReLU activation functions $\phi_1 : \mathbb{R}^{d'} \to \mathbb{R}^{d'}$, $\phi_2 : \mathbb{R}^{d''} \to \mathbb{R}^{d''}$. Assuming $\|\mathbf{W_1^*}\|_F \le W_1$, $\|\mathbf{W_2^*}\|_F \le W_2$, $e\hat{r}r(\mathbf{W_1}, \mathbf{W_2})$ is $\left(\epsilon, \left(\frac{a}{4b^5 W_2^2 W_1} - \frac{W_1}{\epsilon} \right)^{-1}, \mathbf{W_1^*} \right) - SLQC$ in $\mathbf{W_1}, \forall \mathbf{W_1} \in \mathbb{B}(0, W_1)$ and $\forall \epsilon > 0$.*

Proof Sketch. We use Theorem 4 to prove this result. The error $e\hat{r}r(\mathbf{W_1}, \mathbf{W_2})$ of a multi-output single-hidden layer network can be seen as the sum of errors of d'' single-output single-hidden layer networks. This observation combined with Theorem 4 is used to prove the result. See Section 5.8 for the complete proof. □

While the above results have been shown with the generalized ReLU, each of these results also holds for sigmoid activation functions. Moreover, most other widely used error functions such as cross-entropy loss are convex (and thus SLQC) as well as Lipschitz. We believe that our results can be extended to most commonly used error functions, and we present an extension to cross-entropy loss below in Section 3.3.

3.2.5. The ReLU Case

In an earlier subsection, we defined the Generalized ReLU in Defn 3. Note that this definition does not cover the standard ReLU ($a = 0$). We call the standard ReLU as ReLU in this subsection. We hence now provide additional results for networks having ReLU as the activation function. These results are, naturally, quite similar to the ones for Generalized ReLU. Note that for ReLU as the activation function, the subgradient g would be 0 for $x < 0$ and b otherwise.

As in the previous subsection, consider first the case of a network having no hidden layers and only one output neuron. In this case we have the following corollary (to Theorem 1):

Corollary 1. *In the idealized GLM with ReLU activation, assuming $\|\mathbf{w^*}\| \le W$, $e\hat{r}r(\mathbf{w})$ is $\left(\epsilon, 2b^2 W, \mathbf{w^*} \right) - SLQC$ in $\mathbf{w}, \forall \mathbf{w} \in \mathbb{B}(0, W)$ and $\forall \epsilon > 0$.*

Proof Sketch. The proof is similar to the proof of Theorem 1. See Section 5.2 for the complete proof. □

Now consider the case with no hidden layer but multiple output neurons. For this case, we have the following corollary (to Theorem 3).

Corollary 2. *Let an idealized single-layer multi-output neural network be characterized by a linear operator $\mathbf{W} \in \mathbb{R}^{d \times d'} = [\mathbf{w_1} \; \mathbf{w_2} \; \cdots \; \mathbf{w_{d'}}]$ and a standard ReLU activation function applied element-wise $\phi : \mathbb{R}^{d'} \to \mathbb{R}^{d'}$. Let the output of the layer be $\phi\langle W, x \rangle \in \mathbb{R}^{d'}$ where $x \in \mathbb{R}^d$ is the input. Assuming $\|\mathbf{W^*}\|_F \le W$, $e\hat{r}r(\mathbf{W})$ is $\left(\epsilon, 2b^2 W, \mathbf{W^*} \right) - SLQC$ in $\mathbf{W}$ for all $\mathbf{W} \in \mathbb{B}_d(0, W)$ and $\epsilon > 0$.*

Proof Sketch. The proof is similar to the proof of Theorem 3, Section 5.6 presents the complete proof. □

The above results allow us to extend our results to the standard ReLU activation function. We leave the specifics of extending to the one-hidden layer network to the reader. We note, however, that all of our experiments in this work are conducted with the Leaky ReLU which satisfies the Generalized ReLU definition in Defn 3, and hence is in line with the theorems proved in earlier subsections.

3.3. Extension to Cross-Entropy Loss

We now present results for neural networks trained using the Cross Entropy Error, $-\sum_i (y_i \log(p_i))$ (where p_i is the predicted probability for the ith output class from the neural network, obtained using the sigmoid function on a single output neuron, or softmax function when there are multiple output neurons), as loss instead of Mean Square Error. We first consider the case of a network having no hidden layer and only neuron in the output layer. This corresponds to use of J as binary cross-entropy loss in Equation 2, with a sigmoid activation in the output layer. (We wish to highlight that the proof techniques used in the results below are very different from those in [9].)

Theorem 6. *In the idealized GLM like setting with sigmoid activation and binary cross entropy loss, assuming $\|\mathbf{w^*}\| \le W$, $e\hat{r}r(\mathbf{w})$ is $\left(\epsilon, \frac{\epsilon}{(1 - e^{-\epsilon})^2}, \mathbf{w^*} \right) - SLQC$ in $\mathbf{w}, \forall \mathbf{w} \in \mathbb{B}(0, W)$ and $\forall \epsilon > 0$.*

Proof Sketch. To show this, we use Definition 1. With a point $\mathbf{v}$, ϵ / κ close to $\mathbf{w^*}$, we show that $\langle G(\mathbf{w}), \mathbf{w} - \mathbf{v} \rangle \ge 0$. Section 5.9 presents the complete proof. □

We next consider the case of no hidden layers but multiple output neurons. In this case, given cross-entropy error as the loss function, the following result holds:

Theorem 7. *Let an idealized single-layer multi-output neural network be characterized by a linear operator $\mathbf{W} \in \mathbb{R}^{d \times d'} = [\mathbf{w_1} \; \mathbf{w_2} \; \cdots \; \mathbf{w_{d'}}]$ and softmax activation function applied $\phi : \mathbb{R}^{d'} \to \mathbb{R}^{d'}$. Let the output of the layer be $\phi\langle W, x \rangle \in \mathbb{R}^{d'}$ where $x \in \mathbb{R}^d$ is the input and the loss function, $e\hat{r}r$, used is Cross-Entropy Error. Assuming $\|\mathbf{W^*}\|_F \le W$, $e\hat{r}r(\mathbf{W})$ is $\left(\epsilon, \frac{\epsilon d'}{(1 - e^{-\epsilon})^2}, \mathbf{W^*} \right) - SLQC$ in $\mathbf{W}$ for all $\mathbf{W} \in \mathbb{B}_d(0, W)$ and $\epsilon > 0$.*

Proof. See Section 5.10 for the proof. □

The above results allow us to extend our results in Section 3.2 to the cross-entropy loss. We leave the specific statement of the above result for one hidden-layer networks to the reader. We however do experimentally study the use of cross-entropy loss in our methodology, even for deep neural networks, in Section 4. Also, while the cross-entropy loss is more closely related to sigmoid-activated neurons, we empirically also study the use of leaky ReLU as activations in hidden layers in our experiments (Section 4).

Given the above background of results, we now present our methodology to train a multi-layer neural network effectively using these results.

Algorithm 1 Stochastic Normalized Gradient Descent (SNGD)

Input: Number of iterations T, training data $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \in \mathbb{R}^d \times \mathbb{R}$, learning rate η, minibatch size b, Initialization parameters $\mathbf{w}_0$

for $t = 1$ to T **do**

    Select a random mini-batch of training points by sampling $\{(\mathbf{x}_i, y_i)\}_{i=1}^b \sim \text{Uniform}(S)$

    Let $f_t(\mathbf{w}) = \frac{1}{b} \sum_{i=1}^b (y_i - \phi\langle \mathbf{w}, \mathbf{x}_i \rangle)^2$

    Let $\mathbf{g}_t = \nabla f_t(\mathbf{w}_t)$, and $\hat{\mathbf{g}}(t) = \frac{\mathbf{g}_t}{\|\mathbf{g}_t\|}$

    $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \cdot \hat{\mathbf{g}}_t$

end for

Output: Model given by $\mathbf{w}_T$

3.4. Methodology

As stated earlier, [9] showed that Stochastic Normalized Gradient Descent (SNGD) converges with high probability to the optimum for SLQC functions. We leverage this result to arrive at a formal procedure to train neural networks effectively. To this end, we begin by briefly reviewing the Stochastic Normalized Gradient Descent (SNGD) method, and state the relevant result.

3.4.1. Stochastic Normalized Gradient Descent (SNGD)

Normalized Gradient Descent (NGD) is an adaptation of traditional Gradient Descent, where the updates in each iteration are based only on the direction of the gradients. This is achieved by normalizing the gradients. SNGD is the stochastic version of NGD, where weight updates are performed using individual (randomly chosen) training samples, instead of the complete set of samples. Mini-batch SNGD generalizes this by applying updates to the parameters at the end of every mini-batch of samples, as does mini-batch Stochastic Gradient Descent (SGD). In the remainder of this paper, we refer to mini-batch SNGD as SNGD itself, as is common for SGD. Algorithm 1 describes the SNGD methodology for a generic problem.

We now state the result showing the effectiveness of SNGD for SLQC functions.

Theorem 8 ([9]). *Let* $\epsilon, \delta, G, M, \kappa > 0$*, let* $f : \mathbb{R}^d \to \mathbb{R}$ *and* $\mathbf{w}^* = \arg\min_\mathbf{w} f(\mathbf{w})$*. Assume that for* $b \geq b_0(\epsilon, \delta, T)$*, with probability* $\geq 1 - \delta$*,* f_t *defined in Algorithm 1 is* $(\epsilon, \kappa, \mathbf{w}^*)$*-SLQC* $\forall \mathbf{w}$*, and* $|f_t| \leq M \forall t \in \{1, \cdots, T\}$ *. If we run SNGD with* $T \geq \frac{\kappa^2 \|\mathbf{w}_1 - \mathbf{w}^*\|^2}{\epsilon^2}$ *and* $\eta = \frac{\epsilon}{\kappa}$*, and* $b \geq \max\left\{\frac{M^2 log\left(\frac{4T}{\delta}\right)}{2\epsilon^2}, b_0(\epsilon, \delta, T)\right\}$*, with probability* $1 - 2\delta$*,* $f(\mathbf{w}) - f(\mathbf{w}^*) \leq 3\epsilon$[3]*.*

Importantly, note that the convergence rate of SNGD depends on the κ parameter. While the GLM error function with sigmoid activation has $\kappa = e^W$ (stated earlier in the section), the generalized ReLU setting introduced in this work has $\kappa = \frac{2b^3 W}{a}$ (i.e. linear in W) for both GLMs and layers, which is an exponential improvement for the SNGD procedure's effectiveness. This is

significant as the number of iterations T in Theorem 8 depends on κ^2. In other words, SNGD offers accelerated convergence with the proposed generalized ReLU layers as compared to sigmoid GLMs proposed earlier.

3.4.2. DANTE

We have thus far shown that each layer of the considered one-hidden-layer neural network comprises of a set of SLQC problems, each independent in its parameters. Also, SNGD provides an effective method for each such SLQC problem to converge to its respective ϵ-suboptimal solution with high probability, as shown in Theorem 8. This allows us to propose an alternating strategy, DANTE , where each individual SLQC problem is effectively solved (or each individual layer is effectively trained) using SNGD, which we now present. We note that although DANTE uses stochastic gradient-style methods internally (such as SNGD), the overall strategy adopted by DANTE is not necessarily a descent-based strategy, but an alternating-minimization strategy.

Consider the optimization problem below for a single hidden layer network:

$$\min_{\mathbf{W}} \ f(\mathbf{W}_1, \mathbf{W}_2) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \|\phi_2 \langle \mathbf{W}_2, \phi_1 \langle \mathbf{W}_1, \mathbf{x} \rangle \rangle - \mathbf{y}\|_2^2$$

As seen in Section 3.2, on fixing each of W_1 and W_2, we have an SLQC problem. On fixing W_1, we have the SLQC problem:

$$\min_{\mathbf{W}} \ \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \|\phi_2 \langle \mathbf{W}_2, \mathbf{z} \rangle - \mathbf{y}\|_2^2,$$

where $\mathbf{z} = \phi_1 \langle \mathbf{W}_1, \mathbf{x} \rangle$. On fixing $\mathbf{W}_2$, we have the following SLQC problem:

$$\min_{\mathbf{W}} \ \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \|\phi_{\mathbf{W}_2} \langle \mathbf{W}_1, \mathbf{x} \rangle - \mathbf{y}\|_2^2,$$

DANTE optimizes the empirical risk associated with each of these intermediate problems using SNGD steps by sampling several mini-batches of data points and performing updates as in Algorithm 1. Algorithm 2 provides the complete algorithm for the proposed method. Note that the results from the last two subsections hold for any weights and are not limited to the initialized weights. For example if the network is initialized with $\mathbf{W}_1^0$ and $\mathbf{W}_2^0$ and after training the layers once, we obtain weights $\mathbf{W}_1^1$ and $\mathbf{W}_2^1$. The SNGD conditions would still hold for this pair of weights, justifying the applicability of the algorithm.

3.5. Extending to a Multi-Layer Neural Network

In the previous sections, we illustrated how a single hidden-layer neural network can be cast as a set of SLQC problems and proposed an alternating minimization method, DANTE. This approach can be generalized to deep auto-encoders by considering a greedy layer-wise approach to training a neural network [23]. Unlike earlier layer-wise training efforts where such training is used only as a pretraining step, no further finetuning is necessary in our methodology; the layer-wise training directly results in the final model.

We now describe our approach. Consider for example a three-hidden layer autoencoder as pictured in figure 1. Say the

---

[3]Replacing inner product with Frobenius inner product in the proof for this result in [9] allows us to extend this result to our definition of SLQC for matrices.

Algorithm 2 Deep AlterNations for Training nEural networks (DANTE)

Input: Stopping threshold ϵ, Number of iterations of alternating minimization T_{AM}, Number of iterations for SNGD T_{SNGD}, initial values W_1^0, W_2^0, learning rate η, minibatch size b

$t := 1$

while $|f(W_1^t, W_2^t) - f(W_1^{t-1}, W_2^{t-1})| \geq \epsilon$ **or** $t < T_{AM}$ **do**

$\quad W_2^t \leftarrow \underset{W}{\arg\min} \, \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \|\phi_2\langle W, \phi_1\langle W_1^{t-1}, \mathbf{x}\rangle\rangle - \mathbf{y}\|_2^2 \quad$ //use SNGD

$\quad W_1^t \leftarrow \underset{W}{\arg\min} \, \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \|\phi_2\langle W_2^t, \phi_1\langle W, \mathbf{x}\rangle\rangle - \mathbf{y}\|_2^2$ //use SNGD

$\quad t := t + 1$

end while

Output: W_1^{t-1}, W_2^{t-1}

Algorithm 3 DANTE for a multi-layer auto-encoder

Input: Network with $2n - 1$ hidden layers and weights $\mathbf{W}_1, \dots \mathbf{W}_{2n}$

for $l = 1$ to n **do**

$\quad$ Consider the one-hidden layer network formed by $\mathbf{W}_l$ and $\mathbf{W}_{2n-l+1}$.

$\quad$ Train $\mathbf{W}_l$ and $\mathbf{W}_{2n-l+1}$ using Algorithm 2

end for

Output: Trained $\mathbf{W}_1, \dots \mathbf{W}_{2n}$

(a) Phase - 1

(b) Phase - 2

Figure 1: An illustration of the proposed multi-layer DANTE (best viewed in color). In training phase 1, the outer pairs of weights (shaded in gold) are treated as a single-hidden-layer neural network and trained using single-layer DANTE . In phase 2, the inner pair of weights (shaded in gold) are treated as a single-hidden-layer neural network and trained using single-layer DANTE .

is left as an important direction of future work. We focus this work on identifying this alternating minimization procedure, which is derived from a sound understanding of the individual problems underneath (we believe this is a contribution by itself when looking for alternatives to backpropagation), and showcase its empirical effectiveness.

weights in the network are $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$ and $\mathbf{W}_4$ respectively from the leftmost to the rightmost layer. In the first phase we consider the one-hidden layer network obtained by the weights $\mathbf{W}_1$ and $\mathbf{W}_4$ (the network of layer dimensions $5 \rightarrow 3 \rightarrow 5$). We train these two weights using our one-hidden layer DANTE algorithm (section 3.4). Once these layers are trained, in the second phase, we consider the one-hidden layer network obtained by the weights $\mathbf{W}_2$ and $\mathbf{W}_3$ (a network of layer dimensions $3 \rightarrow 2 \rightarrow 3$). We train these weights by the one-hidden layer DANTE algorithm with the input and output being $\phi_1\langle\mathbf{W}_1, \mathbf{x}\rangle$ (the activations of the first hidden layer). This example demonstrated the overall idea behind deep autoencoder training. For a general deep autoencoder, we take pairs of weights symmetric from the center and train them moving from the farthest pair to the one formed by the center layers. Algorithm 3 summarizes the proposed approach to use DANTE for a deep neural network, and Figure 1 illustrates the approach.

Note that it is possible to use other schemes to use DANTE for multi-layer neural networks such as a round-robin scheme, where each layer is trained separately one after the other in the sequence in which the layers appear in the network. Our experiments found that both of these approaches (Algorithm 3 and round-robin scheme) work equally well for autoencoders. To train multi-layer neural networks we use the round-robin scheme.

Following earlier efforts on alternating optimization for neural networks [12][15], we note that proving convergence for alternating minimization methods that train neural networks is not straightforward and a significant effort by itself, and hence

4. Experiments and Results

We validated DANTE by training feedforward neural networks, as well as autoencoders, on standard datasets including MNIST, Kuzushiji-MNIST (KMNIST) [24], SVHN, CIFAR-10 and Tiny ImageNet. The Tiny-Imagenet dataset has 200 classes while the others have 10 classes each. We followed the benchmark training and evaluation protocols established for each of these datasets. We studied the training and test loss as well as the test accuracy on all our experiments. We used vanilla SGD-based backpropagation (henceforth, called SGD in the experiments) as the baseline method. In order to ensure fair comparison between SGD and DANTE , we tried different learning rates and picked the best ones individually for both methods. We show the comparative results with these best learning rates. We also show results later in this section using adaptive learning rate methods on both learning schemes.

Note that in all the presented results (unless explicitly stated otherwise), the X-axis is the number of weights updated. We choose this as the reference instead of number of epochs to be fair to DANTE as it updates fewer weights than SGD in any given epoch (where only one pair of layers is updated). In graphs comparing SGD and DANTE, the blue curve is always DANTE and the green one is SGD.

4.1. Feedforward Neural Networks

This subsection presents the comparative performance of SGD and DANTE on feedforward neural networks. To ensure

an exhaustive comparison, we used multiple datasets and varied network widths and depths in our experiments. Our initial experiments use Leaky ReLU as the activation function, with $a = 0.01$ and $b = 1$, as well as sigmoid activation, and Mean Square Error as the loss function (we later show results with cross-entropy error).

Both MNIST and KMNIST datasets consist of grayscale images of size 28×28. The input layer hence has dimension 784. For both these datasets, we use one-hidden layer networks having 100, 250, 400 and 600 neurons in the hidden layer, as well as a two hidden-layer network ($784 \rightarrow 400 \rightarrow 200 \rightarrow 10$), a three hidden-layer network ($784 \rightarrow 400 \rightarrow 200 \rightarrow 100 \rightarrow 10$) and a five hidden-layer network ($784 \rightarrow 600 \rightarrow 400 \rightarrow 200 \rightarrow 100 \rightarrow 50 \rightarrow 10$). The results are presented in Figure 2.

Both CIFAR-10 and SVHN datasets have colored (3-channel) images of size 3 x 32 x 32, thus the input layer for these is of dimension 3072. For these datasets, we use a one hidden-layer network ($3072 \rightarrow 512 \rightarrow 10$), a two-hidden layer network ($3072 \rightarrow 512 \rightarrow 64 \rightarrow 10$). and a five hidden-layer network ($3072 \rightarrow 1024 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 10$). Figure 3 shows the results.

Tiny-Imagenet is a widely used subset of the original Imagenet dataset having 200 classes, and 500 images of each class in the training set. We train three networks ($12288 \rightarrow 3072 \rightarrow 512 \rightarrow 200$, $12288 \rightarrow 3072 \rightarrow 1024 \rightarrow 512 \rightarrow 200$, and $12288 \rightarrow 3072 \rightarrow 1536 \rightarrow 768 \rightarrow 384 \rightarrow 200$) using DANTE and SGD on this dataset to compare the performance on a more difficult task. Since the labels of the test set are not available, we report the performance on the standard validation set (which has 50 images of each class) in Figure 4. (Both SGD and DANTE do not achieve high accuracies on this dataset as the network considered is a simple MLP. Considering our objective in this work was to prove the feasibility of this approach with MLPs, studying extension of DANTE on convolutional layers, LSTMs and other variants are important directions of our future work.)

The results clearly show the effectiveness of using DANTE for training neural networks - DANTE obtains lower training/test loss and higher test accuracy. Even in cases where the final losses of DANTE and SGD are almost equal, DANTE always minimizes the loss faster than SGD.

Feedforward Neural Networks with Sigmoid Activations. We now present the comparative performance of SGD and DANTE on feedforward neural networks with sigmoid activations and Mean Square Error loss function. We show our results with MNIST and KMNIST datasets. We use the same architectures as in the previous subsection, except that we use sigmoid activation instead of Leaky ReLU. The results are presented in Figure 5. It is apparent from the results that DANTE performs better than SGD in this case too.

Feedforward Neural Networks with Cross-Entropy Loss. To compare DANTE with SGD on networks with cross-entropy loss, we experiment with sigmoid and Leaky ReLU actiavtions on the MNIST dataset with the same network architectures as before, but with cross-entropy as loss function. Figure 6 presents the results of these experiments.

Algorithm	Parameter	Loss
DANTE	LR = 0.001	0.030775
SGD	LR = 0.001	0.031126
SGD+Adam	0.0001	0.021704
SGD+Adagrad	0.0001	0.021896
SGD+RMSProp	0.0001	0.021953
SGD+Momentum	MP=0.9	0.021497
DANTE +Momentum	MP=0.0005	0.020816

Table 1: Loss on using various adaptive learning schemes with DANTE and SGD. (LR = Learning Rate; MP = Momentum Parameter)

Algorithm	Learning Rate	Loss
DANTE	0.001	0.030775
SGD	0.001	0.031126
SGD+AltMin	0.001	0.032912
SGD+AltMin	0.0001	0.044911
SGD+AltMin	0.0005	0.035567

Table 2: Loss on using SGD+AltMin to learn the MNIST dataset.

4.2. Ablation Studies

4.2.1. Impact of Adaptive Learning Rate Methods

As stated earlier, in all of the abovementioned experiments, we chose the best learning rates for both SGD and DANTE in each experiment. To go further, we also studied the use of several adaptive learning schemes with both SGD and DANTE . The results, the final test loss at the end of training, in these studies on the MNIST dataset are presented in Table 1. DANTE with some momentum is able to outperform SGD with all the popular adaptive learning rate schemes.

4.2.2. Using SGD in Alternating Minimization

A natural question one could ask is the relevance of SNGD to train each layer of the proposed methodology. To study this empirically, we compared our algorithm to an analogous algorithm that uses SGD for each inner loop of DANTE . Table 2 presents these results, the test loss at the end of training. Although we allowed different learning rates for the SGD variant, DANTE provides a better performance than any of these variants.

4.2.3. Effect of the T Parameter

DANTE alternatively optimizes over each layer using SNGD. An important parameter for SNGD which can affect performance is the number of epochs for which SNGD algorithm runs for each layer (parameter T in Algorithm 1). We vary T and compare how the performance of DANTE varies when compared to SGD. The results are presented in Figure 7. We observe that DANTE is fairly robust to changes in T. The network used for the presented result was ($784 \rightarrow 100 \rightarrow 10$) with the MNIST dataset. Observing the performance of this experiment, we chose $T = 5$ for all our experiments.
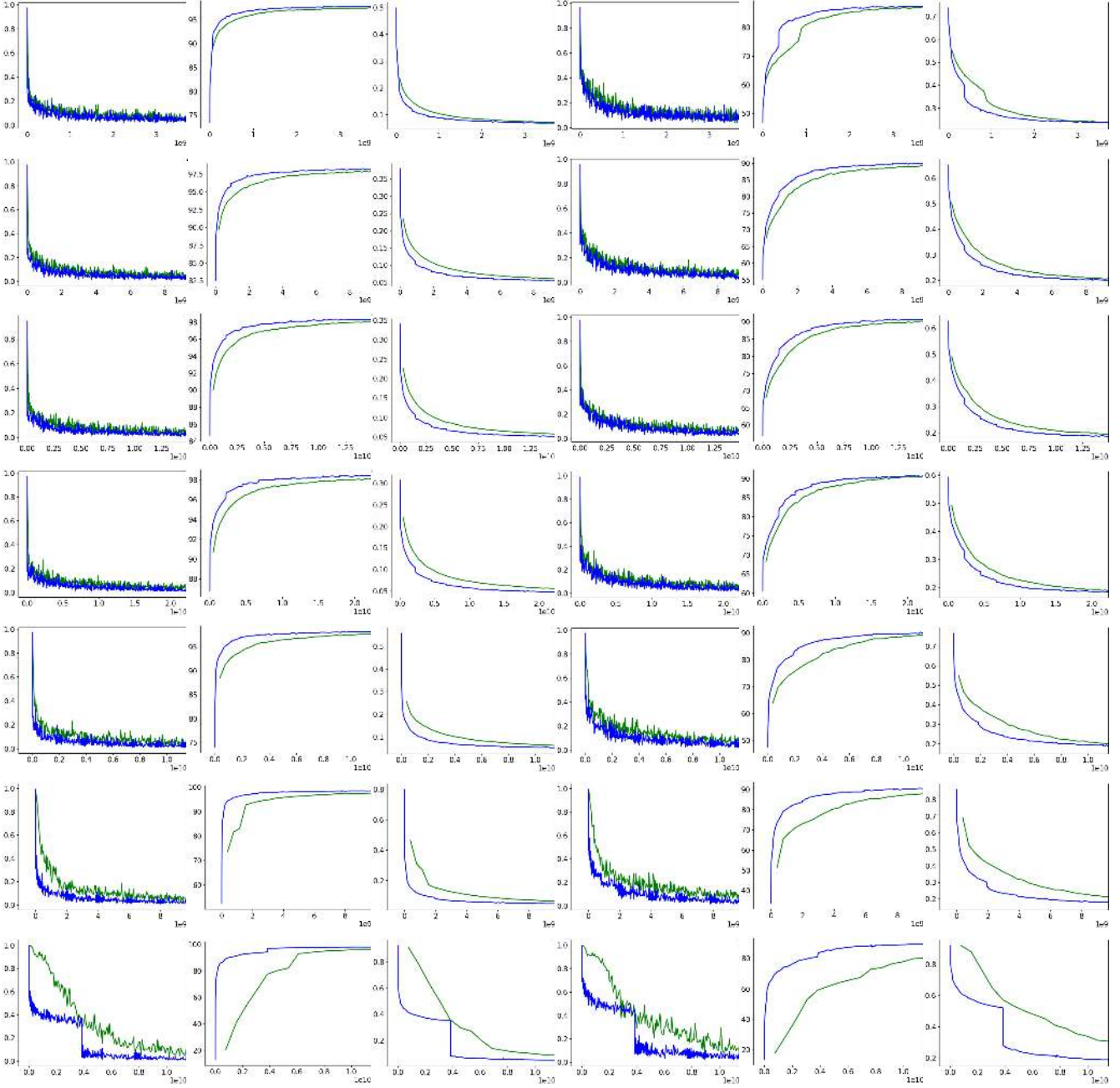
Figure 2: *(Best viewed in color)* Comparative Performance of SGD (Green) and DANTE (Blue) on MNIST and KMNIST datasets. The rows correspond to networks ($784 \rightarrow 100 \rightarrow 10$), ($784 \rightarrow 250 \rightarrow 10$), ($784 \rightarrow 400 \rightarrow 10$), ($784 \rightarrow 600 \rightarrow 10$), ($784 \rightarrow 400 \rightarrow 200 \rightarrow 10$), ($784 \rightarrow 400 \rightarrow 200 \rightarrow 100 \rightarrow 10$) and ($784 \rightarrow 600 \rightarrow 400 \rightarrow 200 \rightarrow 100 \rightarrow 50 \rightarrow 10$) in order from top to bottom, all with Leaky ReLU activations. The first three columns correspond to MNIST, and the last three correspond to KMNIST. The first and fourth columns show training loss; second and fifth columns show test accuracy; third and sixth columns show test loss. For all the plots, X axis is the number of weights updated.
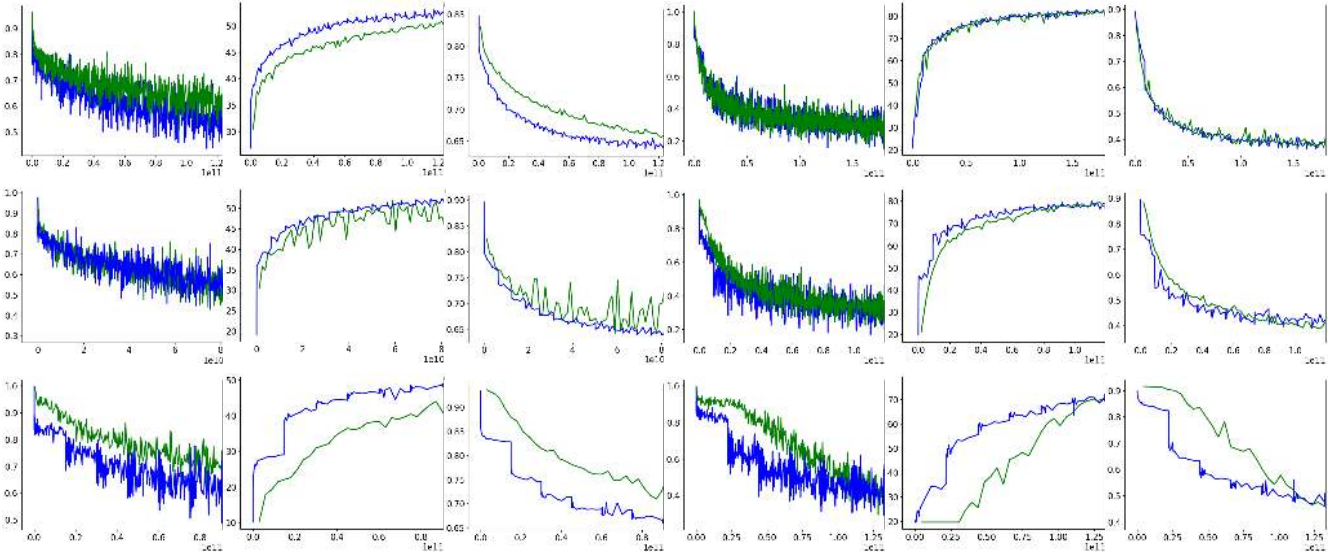
9

Figure 3: *(Best viewed in color)* Comparative performance of SGD (Green) and DANTE (Blue) on CIFAR-10 and SVHN datasets. The rows correspond to networks $(3072 \rightarrow 512 \rightarrow 10)$, $(3072 \rightarrow 512 \rightarrow 64 \rightarrow 10)$ and $(3072 \rightarrow 1024 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 10)$ in order from top to bottom, all with Leaky ReLU activations. The first three columns correspond to CIFAR-10, and the last three correspond to SVHN. The first and fourth columns show training loss; second and fifth columns show test accuracy; third and sixth columns show test loss. For all the plots, X axis is the number of weights updated.

Figure 4: *(Best viewed in color)* Comparative performance of SGD (Green) and DANTE (Blue) on Tiny Imagenet. The rows correspond to networks $(12288 \rightarrow 3072 \rightarrow 512 \rightarrow 200)$, $(12288 \rightarrow 3072 \rightarrow 1024 \rightarrow 512 \rightarrow 200)$ and $(12288 \rightarrow 3072 \rightarrow 1536 \rightarrow 768 \rightarrow 384 \rightarrow 200)$ in order from top to bottom, all with Leaky ReLU activations. The first column shows training loss, second shows validation accuracy and third shows validation loss. For all plots, X-axis is the number of weights updated.

4.3. Other Empirical Studies

4.3.1. Comparative Study

We have compared DANTE to other Alternating-Minimization approaches for training neural networks: Choromanska's [13] AM-Adam (which was their best performing variant) and Taylor's [12] ADMM approach, from the codes provided by the corresponding authors. The results of comparison between DANTE , AM-Adam and ADMM on MNIST are presented in figure 8. Taylor's ADMM algorithm peaked at an accuracy of about 81% for both the networks. Note that Taylor's method seems to show significant instability when trained on well-known datasets over a longer period. Our proposed method does not suffer from this issue. As is clear of the graphs and results, DANTE outperforms both the AM-Adam and Taylor's ADMM algorithm.

4.3.2. Regression Tasks

All the above-mentioned experiments were done for the classification task. We hence also studied the performance of DANTE on standard regression datasets from the UCI repository. Table 3 presents the final test error values at the end of training. We followed The standard benchmark evaluation setup of each of the datasets, as specified in the repository. These results further support the promise of the proposed method.

4.3.3. Training Autoencoder Models

Going further, we conducted experiments to study the effectiveness of the feature representations learned using the autoencoder models trained using DANTE and SGD. After training, we passed the datasets (from UCI repository) through the autoencoder, extracted the hidden layer representations, and then
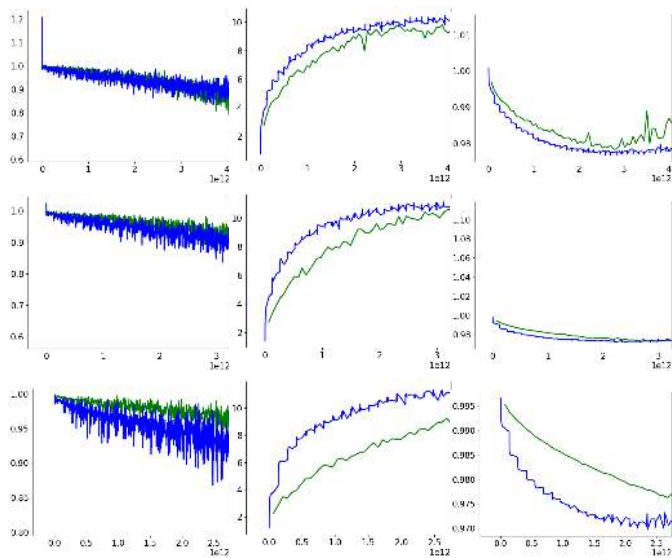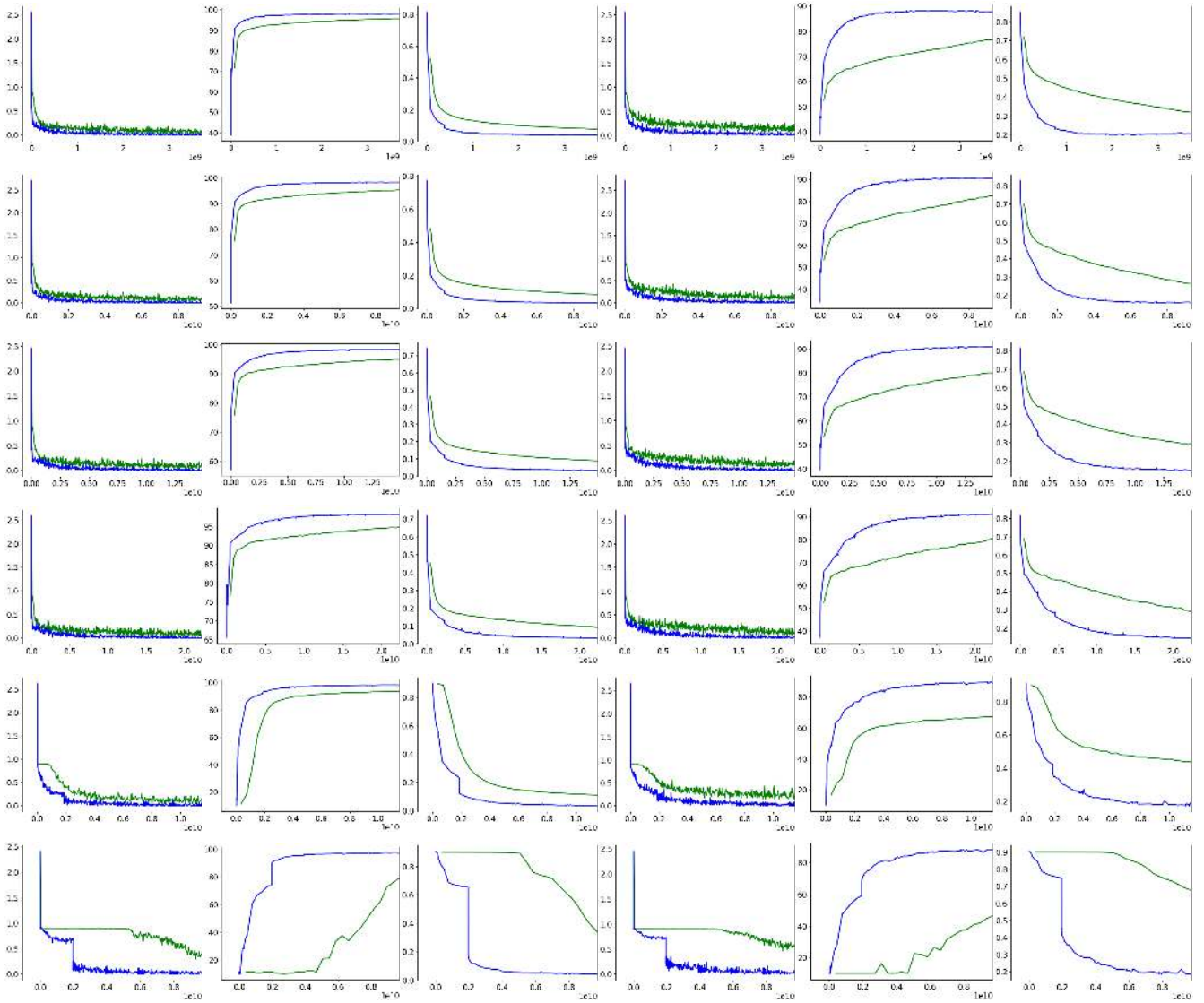
Figure 5: *(Best viewed in color)* Comparative performance of SGD (Green) and DANTE (Blue) on MNIST and KMNIST datasets. The rows correspond to networks (in order) $(784 \rightarrow 100 \rightarrow 10)$, $(784 \rightarrow 250 \rightarrow 10)$, $(784 \rightarrow 400 \rightarrow 10)$, $(784 \rightarrow 600 \rightarrow 10)$, $(784 \rightarrow 400 \rightarrow 200 \rightarrow 10)$ and $(784 \rightarrow 400 \rightarrow 200 \rightarrow 100 \rightarrow 10)$ having sigmoid activations. The first three columns correspond to MNIST, and the last three correspond to KMNIST. The first and fourth columns show training loss; second and fifth columns show test accuracy; third and sixth columns show test loss. For all the plots, X-axis is the number of weights updated.
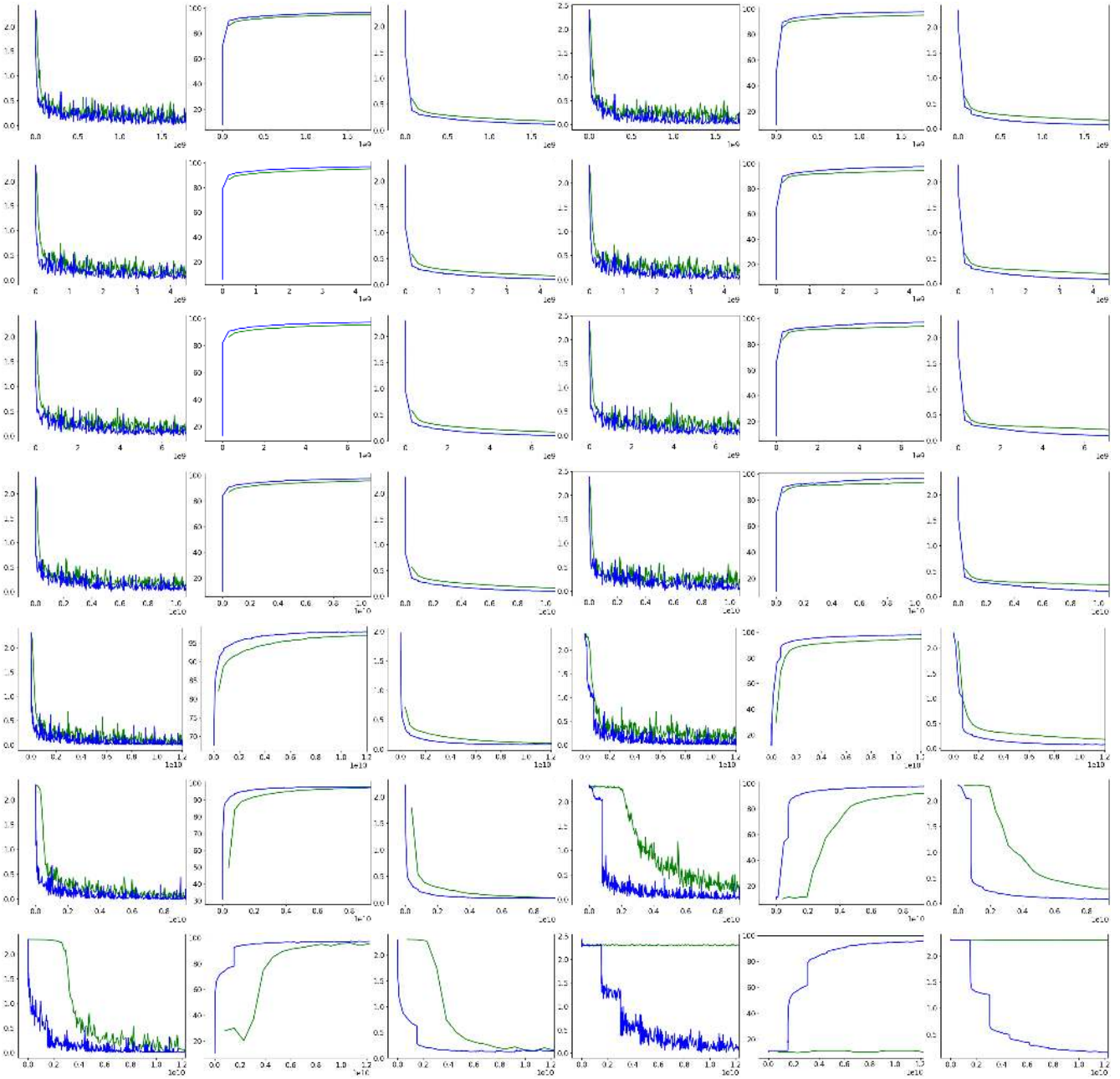
Figure 6: *(Best viewed in color)* Comparative performance of SGD (Green) and DANTE (Blue) on the MNIST dataset with cross-entropy loss. The rows correspond to networks (in order) $(784 \rightarrow 100 \rightarrow 10)$, $(784 \rightarrow 250 \rightarrow 10)$, $(784 \rightarrow 400 \rightarrow 10)$, $(784 \rightarrow 600 \rightarrow 10)$, $(784 \rightarrow 400 \rightarrow 200 \rightarrow 10)$, $(784 \rightarrow 400 \rightarrow 200 \rightarrow 100 \rightarrow 10)$ and $(784 \rightarrow 600 \rightarrow 400 \rightarrow 200 \rightarrow 100 \rightarrow 50 \rightarrow 10)$. The first three columns correspond to networks with Leaky ReLU activations; the last three correspond to networks with sigmoid activations. The first and fourth columns show training loss; second and fifth columns show test accuracy; third and sixth columns show test loss. For all the plots, X-axis is the number of weights updated.

12

	DANTE	SGD
Air-Foil	**0.066852**	0.069338
Fires	**0.024988**	0.029008
CCPP	**0.000283**	0.0003009

Table 3: Test error on UCI regression datasets with DANTE and SGD.

	DANTE	SGD
MNIST	**93.6%**	92.44%
Ionosphere	92.45%	**96.22%**
SVMGuide4	**87.65%**	70.37%
USPS	**90.43%**	89.49%
Vehicle	**77.02%**	74.80%

Table 4: Classification accuracies using ReLU autoencoder features on different datasets.
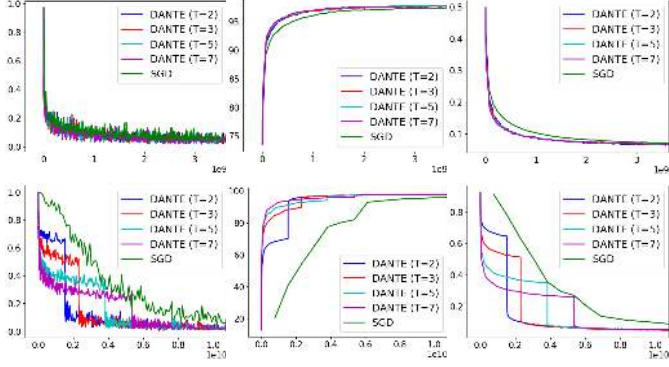

Figure 7: *(Best viewed in color)* Comparative performance of SGD (Green) and DANTE (Blue) with varying parameter T on MNIST dataset. The network used in the top three plots is $(784 \rightarrow 100 \rightarrow 10)$ and the bottom three plots is $(784 \rightarrow 600 \rightarrow 400 \rightarrow 200 \rightarrow 100 \rightarrow 50 \rightarrow 10)$. The first column shows training loss; the second shows test accuracy; and third shows test loss. For all plots, X-axis is the number of weights updated.

trained a linear SVM. The classification accuracy results using the hidden representations are given in Table 4. The table clearly highlights the improved performance of DANTE on this task. In case of the SVMGuide4 dataset, DANTE showed a significant improvement of over 17% on the classification accuracy.

Figure 9 shows some of the best reconstructions obtained by trained models for the autoencoder with the ReLU activation on MNIST in both cases (SGD and DANTE). The model trained using DANTE shows qualitatively better reconstructions, when compared to reconstructions obtained using a model trained by SGD under the same settings.

5. Proofs

5.1. Proof of Theorem 1

Proof. Consider $\mathbf{w} \in \mathbb{B}(0, W)$, $\|\mathbf{w}\| \leq W$ such that $e\hat{r}r_m(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \phi\langle \mathbf{w}, \mathbf{x}_i\rangle)^2 \geq \epsilon$, where m is the total number of samples. Also let $\mathbf{v}$ be a point ϵ/κ-close to minima $\mathbf{w}^*$ with $\kappa = \frac{2b^3W}{a}$. Let g be the subgradient of the generalized ReLU activation and G be the subgradient of $e\hat{r}r_m(\mathbf{w})$. (Note that as before, $g\langle ., .\rangle$ denotes $g(\langle ., .\rangle)$.) Then:

$$\langle G(\mathbf{w}), \mathbf{w} - \mathbf{v}\rangle$$
$$= \frac{2}{m}\sum_{i=1}^{m} g\langle \mathbf{w}, \mathbf{x}_i\rangle \left(\phi\langle \mathbf{w}, \mathbf{x}_i\rangle - y_i\right)\langle \mathbf{x}_i, (\mathbf{w} - \mathbf{v})\rangle$$
$$= \frac{2}{m}\sum_{i=1}^{m} g\langle \mathbf{w}, \mathbf{x}_i\rangle \left(\phi\langle \mathbf{w}, \mathbf{x}_i\rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i\rangle\right) \qquad \text{(Step 1)}$$
$$\left[\langle \mathbf{x}_i, \mathbf{w} - \mathbf{w}^*\rangle + \langle \mathbf{x}_i, \mathbf{w}^* - \mathbf{v}\rangle\right]$$
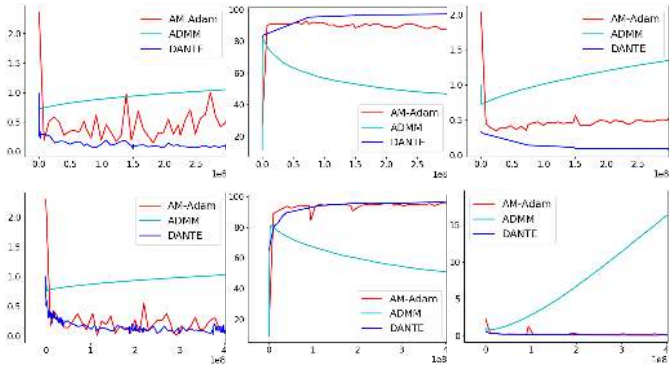

Figure 8: *(Best viewed in color)* Comparative performance of DANTE (Blue), Choromanska's AM-Adam (red) and Taylor's ADMM based algorithm (cyan) on MNIST dataset. The network used in the top three plots is $(784 \rightarrow 100 \rightarrow 10)$ and the bottom three plots is $(784 \rightarrow 100 \rightarrow 100 \rightarrow 100 \rightarrow 10)$. The first column shows training loss; the second shows test accuracy; and third shows test loss. For all plots, X-axis is the number of parameters updated.

Figure 9: Reconstructions using the autoencoder models with ReLU activation. *Top*: Model trained using SGD; *Bottom*: Model trained using DANTE.

$$\geq \frac{2}{m} \sum_{i=1}^{m} g\langle \mathbf{w}, \mathbf{x}_i \rangle \Big[b^{-1} \left(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle \right)^2 \qquad \text{(Step 2)}$$
$$+ \left(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle \right) \langle \mathbf{x}_i, \mathbf{w}^* - \mathbf{v} \rangle \Big]$$

$$\geq \frac{2}{m} \sum_{i=1}^{m} g\langle \mathbf{w}, \mathbf{x}_i \rangle \Big[b^{-1} \left(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle \right)^2$$
$$- |\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle| \|\mathbf{x}_i\| \|\mathbf{w}^* - \mathbf{v}\| \Big]$$

$$\geq \frac{2}{m} \sum_{i=1}^{m} ab^{-1} \left(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle \right)^2 \qquad \text{(Step 3)}$$
$$- \frac{2}{m} \sum_{i=1}^{m} b|\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle| \|\mathbf{x}_i\| \|\mathbf{w}^* - \mathbf{v}\|$$

$$\geq \frac{2}{m} \sum_{i=1}^{m} ab^{-1} \left(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle \right)^2 \qquad \text{(Step 4)}$$
$$- \frac{2}{m} \sum_{i=1}^{m} b^2 \|\langle \mathbf{w}, \mathbf{x}_i \rangle - \langle \mathbf{w}^*, \mathbf{x}_i \rangle\| \frac{\epsilon}{\kappa} \|\mathbf{x}_i\|$$

$$\geq 2ab^{-1}\epsilon - \frac{a\epsilon}{bWm} \sum_{i=1}^{m} \|\langle \mathbf{w}, \mathbf{x}_i \rangle - \langle \mathbf{w}^*, \mathbf{x}_i \rangle\| \|\mathbf{x}_i\|$$

$$\geq 2ab^{-1}\epsilon - \frac{a\epsilon}{bWm} \sum_{i=1}^{m} \|\mathbf{w} - \mathbf{w}^*\| \|\mathbf{x}_i\|^2$$

$$\geq ab^{-1}\epsilon(2 - \frac{1}{W}\|\mathbf{w} - \mathbf{w}^*\|) \qquad \text{(Step 5)}$$

$$\geq 0 \qquad \qquad (7)$$

$\square$

In the above proof, we first use the fact (in Step 1) that in the GLM, there is some $\mathbf{w}^*$ such that $\phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle = y_i$. Then, we use the fact (in Steps 2 and 4) that the generalized ReLU function is b-Lipschitz, and the fact that the minimum value of the quasigradient of g is a (Step 3). Subsequently, in Step 5, we simply use the given bounds on the variables $\mathbf{x}_i, \mathbf{w}, \mathbf{w}^*$ due to the setup of the problem ($\mathbf{w} \in \mathbb{B}(0, W)$, and $\mathbf{x}_i \in \mathbb{B}(0, 1)$, the unit d-dimensional ball, as defined earlier in this section).

5.2. Proof of Corollary 1

Proof. Similar to the previous proof, consider $\mathbf{w} \in \mathbb{B}(0, W)$, $\|\mathbf{w}\| \leq W$ such that $\frac{1}{m} \sum_{i=1,\langle \mathbf{w}, \mathbf{x}_i \rangle > 0}^{m} (y_i - \phi\langle \mathbf{w}, \mathbf{x}_i \rangle)^2 \geq \epsilon$, where m is the total number of samples. Also let $\mathbf{v}$ be a point ϵ/κ-close to minima $\mathbf{w}^*$ with $\kappa = 2b^2 W$. Let g be the subgradient of the generalized ReLU activation and G be the subgradient of $\hat{err}_m(\mathbf{w})$.

Note here that since ϕ is ReLU, if $\langle \mathbf{w}, \mathbf{x}_i \rangle \leq 0$, then $\phi\langle \mathbf{w}, \mathbf{x}_i \rangle = 0$ and $g\langle \mathbf{w}, \mathbf{x}_i \rangle = 0$. Then:

$$\langle G(\mathbf{w}), \mathbf{w} - \mathbf{v} \rangle$$
$$= \frac{2}{m} \sum_{i=1}^{m} g\langle \mathbf{w}, \mathbf{x}_i \rangle \left(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i \right) \langle \mathbf{x}_i, (\mathbf{w} - \mathbf{v}) \rangle$$

$$= \frac{2}{m} \sum_{i=1}^{m} g\langle \mathbf{w}, \mathbf{x}_i \rangle \left(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle \right) \qquad \text{(Step 1)}$$
$$\left[\langle \mathbf{x}_i, \mathbf{w} - \mathbf{w}^* \rangle + \langle \mathbf{x}_i, \mathbf{w}^* - \mathbf{v} \rangle \right]$$

$$\geq \frac{2}{m} \sum_{i=1}^{m} g\langle \mathbf{w}, \mathbf{x}_i \rangle \Big[b^{-1} \left(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle \right)^2 \qquad \text{(Step 2)}$$
$$+ \left(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle \right) \langle \mathbf{x}_i, \mathbf{w}^* - \mathbf{v} \rangle \Big]$$

$$= \frac{2}{m} \sum_{\substack{i=1 \\ \langle \mathbf{w}, \mathbf{x}_i \rangle > 0}}^{m} g\langle \mathbf{w}, \mathbf{x}_i \rangle \Big[b^{-1} \left(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle \right)^2 \qquad \text{(Step 3)}$$
$$+ \left(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle \right) \langle \mathbf{x}_i, \mathbf{w}^* - \mathbf{v} \rangle \Big]$$

$$\geq \frac{2}{m} \sum_{\substack{i=1 \\ \langle \mathbf{w}, \mathbf{x}_i \rangle > 0}}^{m} b \Big[b^{-1} \left(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle \right)^2$$
$$- |\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle| \|\mathbf{x}_i\| \|\mathbf{w}^* - \mathbf{v}\| \Big]$$

$$\geq \frac{2}{m} \sum_{\substack{i=1 \\ \langle \mathbf{w}, \mathbf{x}_i \rangle > 0}}^{m} b \Big[b^{-1} \left(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle \right)^2 \qquad \text{(Step 4)}$$
$$- b\|\langle \mathbf{w}, \mathbf{x}_i \rangle - \langle \mathbf{w}^*, \mathbf{x}_i \rangle\| \frac{\epsilon}{\kappa} \|\mathbf{x}_i\| \Big]$$

$$= \frac{2}{m} \sum_{\substack{i=1 \\ \langle \mathbf{w}, \mathbf{x}_i \rangle > 0}}^{m} \left(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle \right)^2 - \qquad \text{(Step 5)}$$
$$\frac{2}{m} \sum_{\substack{i=1 \\ \langle \mathbf{w}, \mathbf{x}_i \rangle > 0}}^{m} b^2 \|\langle \mathbf{w}, \mathbf{x}_i \rangle - \langle \mathbf{w}^*, \mathbf{x}_i \rangle\| \frac{\epsilon}{\kappa} \|\mathbf{x}_i\|$$

$$\geq 2\epsilon - \frac{2}{m} \sum_{\substack{i=1 \\ \langle \mathbf{w}, \mathbf{x}_i \rangle > 0}}^{m} b^2 \|\mathbf{w} - \mathbf{w}^*\| \frac{\epsilon}{\kappa} \|\mathbf{x}_i\|^2 \qquad \text{(Step 6)}$$

$$= 2\epsilon - \frac{2}{m} mb^2 \|\mathbf{w} - \mathbf{w}^*\| \frac{\epsilon}{\kappa} \qquad \text{(Step 7)}$$

$$\geq \epsilon(2 - \frac{1}{W}\|\mathbf{w} - \mathbf{w}^*\|)$$

$$\geq 0$$

$\square$

The proof uses similar arguments as the proof in Theorem 1. In Step 3, we use the fact that $g\langle \mathbf{w}, \mathbf{x}_i \rangle = 0$ if $\langle \mathbf{w}, \mathbf{x}_i \rangle \leq 0$ and b otherwise. For Step 7, we observe that there at most m i's.

5.3. Proof of Theorem 2
Proof. Here, $\forall i, y_i \in [0, 1]$, the following holds:

$$y_i = \phi\langle \mathbf{w}^*, \mathbf{x} \rangle + \xi_i \qquad (8)$$

where $\{\xi_i\}_{i=1}^{m}$ are zero mean, independent and bounded random variables, i.e. $\forall i \in [m], \|\xi_i\| \leq 1$. Then, $\hat{err}_m(\mathbf{w})$ may be written as follows (expanding y_i as in Eqn 8):

$$\hat{err}_m(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \phi\langle \mathbf{w}, \mathbf{x}_i \rangle)^2$$

14

$$= \frac{1}{m}\Big(\sum_{i=1}^{m} (\phi\langle\mathbf{w}^*,\mathbf{x}_i\rangle - \phi\langle\mathbf{w},\mathbf{x}_i\rangle)^2$$

$$+ \sum_{i=1}^{m} 2\xi_i(\phi\langle\mathbf{w}^*,\mathbf{x}_i\rangle - \phi\langle\mathbf{w},\mathbf{x}_i\rangle) + \sum_{i=1}^{m} \xi_i^2 \Big)$$

Therefore, we also have (by definition of noisy GLM in Defn 2):

$$\hat{err}_m(\mathbf{w}) - \hat{err}_m(\mathbf{w}^*) = \frac{1}{m} \sum_{i=1}^{m} (\phi\langle\mathbf{w}^*,\mathbf{x}_i\rangle - \phi\langle\mathbf{w},\mathbf{x}_i\rangle)^2$$

$$+ \frac{1}{m} \sum_{i=1}^{m} 2\xi_i(\phi\langle\mathbf{w}^*,\mathbf{x}_i\rangle - \phi\langle\mathbf{w},\mathbf{x}_i\rangle)$$

Consider $\|\mathbf{w}\| \le W$ such that $\hat{err}_m(\mathbf{w}) - \hat{err}_m(\mathbf{w}^*) \ge \epsilon$. Also, let $\mathbf{v}$ be a point ϵ/κ-close to minima $\mathbf{w}^*$ with $\kappa = \frac{2b^3 W}{a}$. Let g be the subgradient of the generalized ReLU activation and G be the subgradient of $\hat{err}_m(\mathbf{w})$, as before. Then:

$$\langle G(\mathbf{w}), \mathbf{w} - \mathbf{v}\rangle$$

$$= \frac{2}{m} \sum_{i=1}^{m} g\langle\mathbf{w},\mathbf{x}_i\rangle (\phi\langle\mathbf{w},\mathbf{x}_i\rangle - y_i) \langle\mathbf{x}_i, (\mathbf{w} - \mathbf{v})\rangle$$

$$= \frac{2}{m} \sum_{i=1}^{m} g\langle\mathbf{w},\mathbf{x}_i\rangle (\phi\langle\mathbf{w},\mathbf{x}_i\rangle - \phi\langle\mathbf{w}^*,\mathbf{x}_i\rangle - \xi_i) \qquad \text{(Step 1)}$$

$$[\langle\mathbf{x}_i, \mathbf{w} - \mathbf{w}^*\rangle + \langle\mathbf{x}_i, \mathbf{w}^* - \mathbf{v}\rangle]$$

$$\ge \frac{2b^{-1}}{m} \sum_{i=1}^{m} g\langle\mathbf{w},\mathbf{x}_i\rangle (\phi\langle\mathbf{w}^*,\mathbf{x}_i\rangle - \phi\langle\mathbf{w},\mathbf{x}_i\rangle)^2$$

$$- \frac{2}{m} \sum_{i=1}^{m} g\langle\mathbf{w},\mathbf{x}_i\rangle \xi_i(\langle\mathbf{w},\mathbf{x}_i\rangle - \langle\mathbf{w}^*,\mathbf{x}_i\rangle)$$

$$+ \frac{2}{m} \sum_{i=1}^{m} g\langle\mathbf{w},\mathbf{x}_i\rangle$$

$$\cdot (\phi\langle\mathbf{w},\mathbf{x}_i\rangle - \phi\langle\mathbf{w}^*,\mathbf{x}_i\rangle - \xi_i)\langle\mathbf{w}^* - \mathbf{v},\mathbf{x}_i\rangle \qquad \text{(Step 2)}$$

$$\ge \frac{2b^{-1}}{m} \sum_{i=1}^{m} g\langle\mathbf{w},\mathbf{x}_i\rangle (\phi\langle\mathbf{w}^*,\mathbf{x}_i\rangle - \phi\langle\mathbf{w},\mathbf{x}_i\rangle)^2$$

$$- \frac{2}{m} \sum_{i=1}^{m} g\langle\mathbf{w},\mathbf{x}_i\rangle \xi_i(\langle\mathbf{w},\mathbf{x}_i\rangle - \langle\mathbf{w}^*,\mathbf{x}_i\rangle) \qquad \text{(Step 3)}$$

$$- 2\frac{\epsilon b^2}{\kappa}(\|\mathbf{w} - \mathbf{w}^*\| + \frac{1}{m} \sum_{i=1}^{m} |\xi_i|)$$

$$= \frac{2b^{-1}}{m} \sum_{i=1}^{m} a[(\phi\langle\mathbf{w}^*,\mathbf{x}_i\rangle - \phi\langle\mathbf{w},\mathbf{x}_i\rangle)^2$$

$$- 2\xi_i(\phi\langle\mathbf{w},\mathbf{x}_i\rangle - \phi\langle\mathbf{w}^*,\mathbf{x}_i\rangle)]$$

$$- \frac{2}{m} \sum_{i=1}^{m} [g\langle\mathbf{w},\mathbf{x}_i\rangle (\xi_i(\langle\mathbf{w},\mathbf{x}_i\rangle - \langle\mathbf{w}^*,\mathbf{x}_i\rangle)) \qquad \text{(Step 4)}$$

$$- 2ab^{-1}\xi_i(\phi\langle\mathbf{w},\mathbf{x}_i\rangle - \phi\langle\mathbf{w}^*,\mathbf{x}_i\rangle)]$$

$$- 2\frac{\epsilon b^2}{\kappa}(\|\mathbf{w} - \mathbf{w}^*\| + \frac{1}{m} \sum_{i=1}^{m} |\xi_i|)$$

$$\ge 2ab^{-1}\epsilon - 2\frac{\epsilon b^2}{\kappa}(\|\mathbf{w} - \mathbf{w}^*\| + \frac{1}{m} \sum_{i=1}^{m} |\xi_i|)$$

$$+ \frac{1}{m} \sum_{i=1}^{m} \xi_i\lambda_i(\mathbf{w}) \qquad \text{(Step 5)}$$

$$\ge 2ab^{-1}\epsilon - ab^{-1}W^{-1}\epsilon(\|\mathbf{w} - \mathbf{w}^*\| + \frac{1}{m} \sum_{i=1}^{m} |\xi_i|)$$

$$+ \frac{1}{m} \sum_{i=1}^{m} \xi_i\lambda_i(\mathbf{w}) \qquad \text{(Step 6)}$$

$$\ge 2ab^{-1}\epsilon - ab^{-1}\epsilon(1 + W^{-1}) + \frac{1}{m} \sum_{i=1}^{m} \xi_i\lambda_i(\mathbf{w}) \qquad \text{(Step 7)}$$

$$\ge -ab^{-1}\epsilon W^{-1} + \frac{1}{m} \sum_{i=1}^{m} \xi_i\lambda_i(\mathbf{w}) \qquad \text{(Step 8)}$$

Here, $\lambda_i(\mathbf{w}) = 2g\langle\mathbf{w},\mathbf{x}_i\rangle(\langle\mathbf{w},\mathbf{x}_i\rangle - \langle\mathbf{w}^*,\mathbf{x}_i\rangle) - 4ab^{-1}(\phi\langle\mathbf{w},\mathbf{x}_i\rangle - \phi\langle\mathbf{w}^*,\mathbf{x}_i\rangle)$, and
$|\xi_i\lambda_i(\mathbf{w})| \le 2b(|\langle\mathbf{w},\mathbf{x}_i\rangle - \langle\mathbf{w}^*,\mathbf{x}_i\rangle| + 4ab^{-1}|\phi\langle\mathbf{w},\mathbf{x}_i\rangle - \phi\langle\mathbf{w}^*,\mathbf{x}_i\rangle|) \le 2b(3|\langle\mathbf{w},\mathbf{x}_i\rangle - \langle\mathbf{w}^*,\mathbf{x}_i\rangle|) \le 2b(6W) = 12bW$

The above proof uses arguments similar to the proof for the idealized GLM (please see the lines after the proof of Theorem 1, viz. the b-Lipschitzness of the generalized ReLU, and the problem setup). Now, when

$$\frac{1}{m} \sum_{i=1}^{m} \xi\lambda_i(\mathbf{w}) \ge ab^{-1}W^{-1}\epsilon$$

our model is SLQC. By simply using the Hoeffding's bound, we get that the theorem statement holds for $m \ge \frac{288b^4 W^4}{a^2} log(1/\delta)/\epsilon^2$. $\qquad\square$

5.4. Viewing the Outer Layer of a Neural Network as a Set of GLMs

Given an (unknown) distribution $\mathcal{D}$, let the layer be characterized by a linear operator $W \in \mathbb{R}^{d \times d'}$ and a non-linear activation function defined by $\phi : \mathbb{R} \to \mathbb{R}$. Let the layer output be defined by $\phi\langle W, \mathbf{x}\rangle$, where $\mathbf{x} \in \mathbb{R}^d$ is the input, and ϕ is used element-wise in this function.

Consider the mean squared error loss, commonly used in neural networks, given by:

$$\min_W \quad err(W) = \min_W \quad \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\|\phi\langle W, \mathbf{x}\rangle - \mathbf{y}\|_2^2$$

$$= \min_W \quad \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\|\sum_{i=1}^{d'} \phi\langle W_{:,i}, \mathbf{x}\rangle - \mathbf{y}_i\|_2^2$$

$$= \min_W \quad \sum_{i=1}^{d'} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\|\phi\langle W_{:,i}, \mathbf{x}\rangle - \mathbf{y}_i\|_2^2$$

$$= \sum_{i=1}^{d'} \min_W \quad \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\|\phi\langle W_{:,i}, \mathbf{x}\rangle - \mathbf{y}_i\|_2^2$$

Each of these sub-problems above is a GLM, which can be solved effectively using SNGD as seen in Theorem 8, which we leverage in this work.

5.5. Proof of Theorem 3

Proof. Consider $\mathbf{W} \in \mathbb{B}(0, W)$, $\|\mathbf{W}\| \leq W$ such that $e\hat{r}r_m(\mathbf{W}) = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{y}_i - \phi\langle\mathbf{W}, \mathbf{x}_i\rangle)^2 \geq d'\epsilon$, where m is the total number of samples. Also let $\mathbf{V} = [\mathbf{v}_1 \; \mathbf{v}_2 \cdots \; \mathbf{v}_{d'}]$ be a point ϵ/κ-close to minima $\mathbf{W}^*$ with $\kappa = \frac{2b^3 W}{a}$. Let g be the subgradient of the generalized ReLU activation, $G(\mathbf{W})$ be the subgradient of $e\hat{r}r_m(\mathbf{W})$ and $G(\mathbf{w}_j)$ be the subgradient of $e\hat{r}r_m(\mathbf{w}_j)$. (Note that as before, $g\langle.,.\rangle$ denotes $g(\langle.,.\rangle)$.) Then:

$$\langle G(\mathbf{W}), \mathbf{W} - \mathbf{V}\rangle$$

$$= \sum_{j=1}^{d'} \langle G(\mathbf{w_j}), \mathbf{w_j} - \mathbf{v_j}\rangle_F$$

(By definition of Frobenius inner product)

$$= \frac{2}{m}\sum_{i=1}^{m}\sum_{j=1}^{d'}\left(\phi\langle\mathbf{w_j}, \mathbf{x_i}\rangle - y_{ij}\right)\langle\frac{\partial(\phi\langle\mathbf{w_j}, \mathbf{x_i}\rangle)}{\partial\mathbf{w_j}}, (\mathbf{w_j} - \mathbf{v_j})\rangle$$

(Step 1)

$$= \frac{2}{m}\sum_{i=1}^{m}\sum_{j=1}^{d'} g(\mathbf{w_j}, \mathbf{x}_i)\left(\phi\langle\mathbf{w_j}, \mathbf{x_i}\rangle - y_{ij}\right)$$

(Step 2)

$$[\langle\mathbf{x}_i, \mathbf{w_j} - \mathbf{w_j^*}\rangle + \langle\mathbf{x}_i, \mathbf{w_j^*} - \mathbf{v_j}\rangle]$$

$$\geq \frac{2}{m}\sum_{i=1}^{m}\sum_{j=1}^{d'} g\langle\mathbf{w_j}, \mathbf{x}_i\rangle\Big[b^{-1}\left(\phi\langle\mathbf{w_j}, \mathbf{x_i}\rangle - \phi\langle\mathbf{w_j^*}, \mathbf{x_i}\rangle\right)^2$$

$$+ \left(\phi\langle\mathbf{w_j}, \mathbf{x_i}\rangle - \phi\langle\mathbf{w_j^*}\mathbf{x_i}\rangle\right)\langle\mathbf{x}_i, \mathbf{w_j^*} - \mathbf{v_j}\rangle\Big]$$

(Step 3)

$$\geq \frac{2}{m}\sum_{i=1}^{m}\sum_{j=1}^{d'} g\langle\mathbf{w_j}, \mathbf{x}_i\rangle\Big[b^{-1}\left(\phi\langle\mathbf{w_j}, \mathbf{x_i}\rangle - \phi\langle\mathbf{w_j^*}, \mathbf{x_i}\rangle\right)^2$$

$$- |\phi\langle\mathbf{w_j}, \mathbf{x_i}\rangle - \phi\langle\mathbf{w_j^*}, \mathbf{x_i}\rangle|\|\mathbf{x}_i\|\|\mathbf{w_j^*} - \mathbf{v_j}\|\Big]$$

(Step 4)

$$\geq \frac{2}{m}\sum_{i=1}^{m}\sum_{j=1}^{d'}\Big[ab^{-1}\left(\phi\langle\mathbf{w_j}, \mathbf{x_i}\rangle - \phi\langle\mathbf{w_j^*}, \mathbf{x_i}\rangle\right)^2$$

$$- b|\phi\langle\mathbf{w_j}, \mathbf{x_i}\rangle - \phi\langle\mathbf{w_j^*}, \mathbf{x_i}\rangle|\|\mathbf{x}_i\|\|\mathbf{w_j^*} - \mathbf{v_j}\|\Big]$$

$$\geq \frac{2}{m}\sum_{i=1}^{m}\sum_{j=1}^{d'}\Big[ab^{-1}\left(\phi\langle\mathbf{w_j}, \mathbf{x_i}\rangle - \phi\langle\mathbf{w_j^*}, \mathbf{x_i}\rangle\right)^2$$

(Step 5)

$$- b^2\|\langle\mathbf{w_j}, \mathbf{x_i}\rangle - \langle\mathbf{w_j^*}, \mathbf{x_i}\rangle\|\frac{\epsilon}{\kappa}\|\mathbf{x}_i\|\Big]$$

$$\geq 2ab^{-1}d'\epsilon - \frac{ad'\epsilon}{bWm}\sum_{i=1}^{m}\|\langle\mathbf{w}, \mathbf{x}_i\rangle - \langle\mathbf{w^*}, \mathbf{x_i}\rangle\|\|\mathbf{x}_i\|$$

(Step 6)

$$\geq ab^{-1}d'\epsilon(2 - \frac{1}{Wm}\sum_{i=1}^{m}\|\mathbf{w} - \mathbf{w^*}\|\|\mathbf{x}_i\|^2)$$

(Step 7)

$$\geq ab^{-1}d'\epsilon(2 - \frac{1}{W}\|\mathbf{w} - \mathbf{w^*}\|) \geq 0$$

In Step 6, $\|\langle\mathbf{w}, \mathbf{x}_i\rangle - \langle\mathbf{w^*}, \mathbf{x_i}\rangle\| = \max_j\|\langle\mathbf{w_j}, \mathbf{x}_i\rangle - \langle\mathbf{w_j^*}, \mathbf{x_i}\rangle\|$ To simplify from Step 7 we use the fact that $\|\mathbf{W^*}\| \leq W \implies \|\mathbf{w^*}\| \leq W$. The remainder of the proof proceeds precisely as in Theorem 1. $\square$

5.6. Proof of Corollary 2

Proof. Let all the variables be the same as in the proof for Theorem 3 except that $\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1,\langle\mathbf{w}_j,\mathbf{x}_i\rangle>0}^{d'}(y_{ij} - \phi\langle\mathbf{w}_j, \mathbf{x}_i\rangle)^2 \geq d'\epsilon$ and

$\kappa = 2b^2 W$. Again note here that since ϕ is ReLU, if $\langle\mathbf{w}, \mathbf{x}_i\rangle \leq 0$, then $\phi\langle\mathbf{w}, \mathbf{x}_i\rangle = 0$ and $g\langle\mathbf{w}, \mathbf{x}_i\rangle = 0$. Using the results from previous proof, we continue from Step 4,

$$\langle G(\mathbf{W}), \mathbf{W} - \mathbf{V}\rangle$$

$$\geq \frac{2}{m}\sum_{i=1}^{m}\sum_{j=1}^{d'} g\langle\mathbf{w_j}, \mathbf{x}_i\rangle\Big[b^{-1}\left(\phi\langle\mathbf{w_j}, \mathbf{x_i}\rangle - \phi\langle\mathbf{w_j^*}, \mathbf{x_i}\rangle\right)^2$$

$$- |\phi\langle\mathbf{w_j}, \mathbf{x_i}\rangle - \phi\langle\mathbf{w_j^*}, \mathbf{x_i}\rangle|\|\mathbf{x}_i\|\|\mathbf{w_j^*} - \mathbf{v_j}\|\Big]$$

(Step 4, Borrowed)

$$\geq \frac{2}{m}\sum_{i=1}^{m}\sum_{\substack{j=1 \\ \langle\mathbf{w_j}, \mathbf{x}_i\rangle>0}}^{d'} b\Big[b^{-1}\left(\phi\langle\mathbf{w_j}, \mathbf{x_i}\rangle - \phi\langle\mathbf{w_j^*}, \mathbf{x_i}\rangle\right)^2\Big]$$

$$- \frac{2}{m}\sum_{i=1}^{m}\sum_{j=1}^{d'}\Big[b|\phi\langle\mathbf{w_j}, \mathbf{x_i}\rangle - \phi\langle\mathbf{w_j^*}, \mathbf{x_i}\rangle|\|\mathbf{x}_i\|\|\mathbf{w_j^*} - \mathbf{v_j}\|\Big]$$

(Step 5)

$$\geq d'\epsilon(2 - \frac{1}{W}\|\mathbf{w} - \mathbf{w^*}\|) \geq 0$$

Simplification from Step 5 to last step follows from similar arguments as last proof. $\square$

5.7. Proof of Theorem 4

Proof. In this case, the prediction of the network on $\mathbf{x}$ is $f(\mathbf{W_1}; \mathbf{w_2}; \mathbf{x})$.

Consider $\mathbf{W_1} \in \mathbb{B}(0, W_1)$, $\|\mathbf{W_1}\| \leq W_1$, $\|\mathbf{w_2}\| \leq W_2$ such that $e\hat{r}r(\mathbf{W_1}, \mathbf{w_2}) \geq \epsilon$. Let $\mathbf{V_1}$ be a point $\frac{\epsilon}{\kappa}$ close to minima $\mathbf{W_1}$, where $\kappa = \left(\frac{a}{4b^5 W_2^2 W_1} - \frac{W_1}{\epsilon}\right)^{-1}$.

Let $\|f(\mathbf{W_1}; \mathbf{w_2}; \mathbf{x}) - \mathbf{y}\|_2^2 = \|\phi_2\langle\mathbf{w_2}, \phi_1\langle\mathbf{W_1}, \mathbf{x}\rangle\rangle - \mathbf{y}\|_2^2$ and $\langle\cdot\rangle_F$ be the Frobenius inner product.

$$\langle\nabla_{\mathbf{W_1}} e\hat{r}r(\mathbf{W_1}, \mathbf{w_2}), \mathbf{W_1} - \mathbf{V_1}\rangle_F$$

$$= \frac{2}{m}\sum_{i=1}^{m}(\phi_2\langle\mathbf{w_2}, \phi_1\langle\mathbf{W_1}, \mathbf{x_i}\rangle\rangle - y_i)$$

(Step 1)

$$\langle\frac{\partial(\phi_2\langle\mathbf{w_2}, \phi_1\langle\mathbf{W_1}, \mathbf{x_i}\rangle\rangle)}{\partial\mathbf{W_1}}, (\mathbf{W_1} - \mathbf{V_1})\rangle_F$$

Using chain rule, we can simplify $\frac{\partial(\phi_2\langle\mathbf{w_2}, \phi_1\langle\mathbf{W_1}, \mathbf{x}\rangle\rangle)}{\partial\mathbf{W_1}}$ as

$$\left[\frac{\partial(\phi_2\langle\mathbf{w_2}, \phi_1\langle\mathbf{W_1}, \mathbf{x}\rangle\rangle)}{\partial\mathbf{W_1}}\right]^T = \frac{\partial(\phi_2\langle\mathbf{w_2}, \phi_1\langle\mathbf{W_1}, \mathbf{x}\rangle\rangle)}{\partial\langle\mathbf{w_2}, \phi_1\langle\mathbf{W_1}, \mathbf{x}\rangle\rangle}$$

$$\cdot\left[\frac{\partial\langle\mathbf{w_2}, \phi_1\langle\mathbf{W_1}, \mathbf{x}\rangle\rangle^T}{\partial\phi_1\langle\mathbf{W_1}, \mathbf{x}\rangle} \cdot \frac{\partial\phi_1\langle\mathbf{W_1}, \mathbf{x}\rangle^T}{\partial\langle\mathbf{W_1}, \mathbf{x_i}\rangle}\right]^T \cdot \left[\frac{\partial\langle\mathbf{W_1}, \mathbf{x}\rangle}{\partial\mathbf{W_1}}\right]^T$$

$$= g_2(\mathbf{W_1}, \mathbf{w_2}, x) \cdot g_1(\mathbf{W_1}, x) \cdot \mathbf{w_2} \cdot \mathbf{x}^T$$

(Let)

Continuing from Step 1:

$$= \frac{2}{m}\sum_{i=1}^{m} g_2(\mathbf{W_1}, \mathbf{w_2}, \mathbf{x}_i)(\phi_2\langle\mathbf{w_2}, \phi_1\langle\mathbf{W_1}, \mathbf{x_i}\rangle\rangle - y_i)$$

$$\langle\mathbf{x}_i\mathbf{w_2}^T g_1(\mathbf{W_1}, \mathbf{x_i})^T, (\mathbf{W_1} - \mathbf{V_1})\rangle_F$$

$$= \frac{2}{m} \sum_{i=1}^{m} g_2(\mathbf{W}_1, \mathbf{w}_2, \mathbf{x}_i) \left(\phi_2\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x_i}\rangle\rangle - y_i\right)$$

$$[\langle \mathbf{x}_i \mathbf{w}_2^T g_1(\mathbf{W}_1, \mathbf{x}_i)^T, \mathbf{W_1}\rangle_F$$
$$- \langle \mathbf{x}_i (\mathbf{w}_2^*)^T g_1(\mathbf{W}_1^*, \mathbf{x}_i)^T, \mathbf{W}_1^*\rangle_F$$
$$+ \langle \mathbf{x}_i (\mathbf{w}_2^*)^T g_1(\mathbf{W}_1^*, \mathbf{x}_i)^T, \mathbf{W_1}\rangle_F$$
$$- \langle \mathbf{x}_i \mathbf{w}_2^T g_1(\mathbf{W}_1, \mathbf{x}_i)^T, \mathbf{V_1}\rangle_F]$$

$$= \frac{2}{m} \sum_{i=1}^{m} g_2(\mathbf{W}_1, \mathbf{w}_2, \mathbf{x}_i) \left(\phi_2\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x_i}\rangle\rangle - y_i\right)$$

$$[Tr(g_1(\mathbf{W}_1, \mathbf{x}_i)\mathbf{w}_2\mathbf{x}_i^T \mathbf{W_1}) - Tr(g_1(\mathbf{W}_1^*, \mathbf{x}_i)\mathbf{w}_2^*\mathbf{x}_i^T \mathbf{W_1^*})$$
$$+ Tr(g_1(\mathbf{W}_1^*, \mathbf{x}_i)\mathbf{w}_2^*\mathbf{x}_i^T \mathbf{W_1^*}) - Tr(g_1(\mathbf{W}_1, \mathbf{x}_i)\mathbf{w}_2\mathbf{x}_i^T \mathbf{V_1})]$$
$$\text{(Step 2)}$$

In order to convert the above terms into a more familiar form, we begin with the following observation:

$$\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}, \mathbf{x}\rangle\rangle_F = Tr(g_1(\mathbf{W}, \mathbf{x})\mathbf{w}_2\mathbf{x}^T(\mathbf{W}))$$

Also, note that $g_1(\mathbf{W}, \mathbf{x})$ is a diagonal $d' \times d'$ matrix consisting of a's and b's on the diagonal:

$$\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x}\rangle\rangle - \langle \mathbf{w}_2^*, \phi_1\langle \mathbf{W}, \mathbf{x}\rangle\rangle$$
$$= Tr(g_1(\mathbf{W_1}, \mathbf{x})\mathbf{w}_2\mathbf{x}^T \mathbf{W_1}) - Tr(g_1(\mathbf{W}, \mathbf{x})\mathbf{w}_2^*\mathbf{x}^T \mathbf{W})$$

Therefore, on setting $\mathbf{W} = \mathbf{W_1^*}$ and using the fact that the generalized ReLU is b-Lipschitz and monotonically increasing, we have:

$$(\phi_2\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x}\rangle\rangle - \phi_2\langle \mathbf{w}_2^*, \phi_1\langle \mathbf{W_1^*}, \mathbf{x}\rangle\rangle)^2$$
$$\leq b(\phi_2\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x}\rangle\rangle - \phi_2\langle \mathbf{w}_2^*, \phi_1\langle \mathbf{W_1^*}, \mathbf{x}\rangle\rangle)$$
$$\cdot (\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x}\rangle\rangle - \langle \mathbf{w}_2^*, \phi_1\langle \mathbf{W}_1^*, \mathbf{x}\rangle\rangle)$$
$$= b(\phi_2\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x}\rangle\rangle - \phi_2\langle \mathbf{w}_2^*, \phi_1\langle \mathbf{W_1^*}, \mathbf{x}\rangle\rangle)$$
$$\cdot (Tr(g_1(\mathbf{W_1}, \mathbf{x})\mathbf{w}_2\mathbf{x}^T \mathbf{W_1}) - Tr(g_1(\mathbf{W}_1^*, \mathbf{x})\mathbf{w}_2^*\mathbf{x}^T \mathbf{W_1^*}))$$

Plugging this result into Step 2:

$$\geq \frac{2}{m} \sum_{i=1}^{m} g_2(\mathbf{W}_1, \mathbf{w}_2, \mathbf{x}_i)$$
$$[b^{-1}(\phi_2\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x_i}\rangle\rangle - \phi_2\langle \mathbf{w}_2^*, \phi_1\langle \mathbf{W_1^*}, \mathbf{x_i}\rangle\rangle)^2$$
$$+ (\phi_2\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x_i}\rangle\rangle - y_i)$$
$$\cdot (Tr(g_1(\mathbf{W}_1^*, \mathbf{x}_i)\mathbf{w}_2^*\mathbf{x}_i^T \mathbf{W_1^*}) - Tr(g_1(\mathbf{W}_1, \mathbf{x}_i)\mathbf{w}_2\mathbf{x}_i^T \mathbf{V_1}))]$$
$$\geq 2ab^{-1}\epsilon + \frac{2}{m} \sum_{i=1}^{m} g_2(\mathbf{W}_1, \mathbf{w}_2, \mathbf{x}_i) \left(\phi_2\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x_i}\rangle\rangle - y_i\right)$$
$$\cdot [Tr(g_1(\mathbf{W}_1^*, \mathbf{x}_i)\mathbf{w}_2^*\mathbf{x}_i^T \mathbf{W_1^*}) - Tr(g_1(\mathbf{W}_1, \mathbf{x}_i)\mathbf{w}_2\mathbf{x}_i^T \mathbf{V_1})]$$
$$\geq \frac{2a}{b}\epsilon - \frac{2}{m} \sum_{i=1}^{m} g_2(\mathbf{W}_1, \mathbf{w}_2, \mathbf{x}_i) \cdot |(\phi_2\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x_i}\rangle\rangle - y_i)|$$
$$|Tr(g_1(\mathbf{W}_1^*, \mathbf{x}_i)\mathbf{w}_2^*\mathbf{x}_i^T \mathbf{W}_1^*) - Tr(g_1(\mathbf{W}_1, \mathbf{x}_i)\mathbf{w}_2\mathbf{x}_i^T \mathbf{V_1})|$$
$$\text{(Step 3)}$$

First consider the term $|Tr(g_1(\mathbf{W}_1^*, \mathbf{x}_i)\mathbf{w}_2^*\mathbf{x}_i^T \mathbf{W}_1^*) - Tr(g_1(\mathbf{W}_1, \mathbf{x}_i)\mathbf{w}_2\mathbf{x}_i^T \mathbf{V_1})|$. Note that $\|\mathbf{V}_1 - \mathbf{W}_1^*\| \leq \frac{\epsilon}{\kappa}$. From triangle inequality, $\|\mathbf{V}_1\| \leq \|\mathbf{V}_1 - \mathbf{W}_1^*\| + \|\mathbf{W}_1^*\|$.

$$|Tr(g_1(\mathbf{W}_1^*, \mathbf{x}_i)\mathbf{w}_2^*\mathbf{x}_i^T \mathbf{W}_1^*) - Tr(g_1(\mathbf{W}_1, \mathbf{x}_i)\mathbf{w}_2\mathbf{x}_i^T \mathbf{V_1})|$$
$$\leq |Tr(g_1(\mathbf{W}_1^*, \mathbf{x}_i)\mathbf{w}_2^*\mathbf{x}_i^T \mathbf{W}_1^*)| + |Tr(g_1(\mathbf{W}_1, \mathbf{x}_i)\mathbf{w}_2\mathbf{x}_i^T \mathbf{V_1})|$$
$$\leq b \cdot \|\mathbf{w}_2^*\|\|\mathbf{x}_i\|\|\mathbf{W}_1^*\| + b \cdot \|\mathbf{w}_2\|\|\mathbf{x}_i\|\|\mathbf{V}\|$$
$$\leq 1 \cdot b \cdot W_2[\|\mathbf{W}_1^*\| + \|\mathbf{V}_1\|]$$
$$\leq b \cdot W_2[2 \cdot \|\mathbf{W}_1^*\| + \|\mathbf{V}_1 - \mathbf{W}_1^*\|]$$
$$\leq b \cdot W_2[2 \cdot \|\mathbf{W}_1^*\| + \frac{\epsilon}{\kappa}]$$

Now consider the term $|\phi_2\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x_i}\rangle\rangle - \phi_2\langle \mathbf{w}_2^*, \phi_1\langle \mathbf{W}_1^*, \mathbf{x_i}\rangle\rangle|$ appearing in Step 3. We have,

$$|\phi_2\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x_i}\rangle\rangle - \phi_2\langle \mathbf{w}_2^*, \phi_1\langle \mathbf{W}_1^*, \mathbf{x_i}\rangle\rangle|$$
$$\leq |\phi_2\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x_i}\rangle\rangle| + |\phi_2\langle \mathbf{w}_2^*, \phi_1\langle \mathbf{W}_1^*, \mathbf{x_i}\rangle\rangle|$$
$$\leq b[|\langle \mathbf{w}_2, \phi_1\langle \mathbf{W}_1, \mathbf{x_i}\rangle\rangle| + |\langle \mathbf{w}_2^*, \phi_1\langle \mathbf{W}_1^*, \mathbf{x_i}\rangle\rangle|]$$
$$\leq b[\|\mathbf{w}_2\| \cdot \|\phi_1\langle \mathbf{W}_1, \mathbf{x_i}\rangle\| + \|\mathbf{w}_2^*\| \cdot \|\phi_1\langle \mathbf{W}_1^*, \mathbf{x_i}\rangle\|]$$
$$\leq b \cdot W_2 \cdot [\|\phi_1\langle \mathbf{W}_1, \mathbf{x_i}\rangle\| + \|\phi_1\langle \mathbf{W}_1^*, \mathbf{x_i}\rangle\|]$$
$$\leq b \cdot W_2 \cdot [b\|\langle \mathbf{W}_1, \mathbf{x_i}\rangle\| + b\|\langle \mathbf{W}_1^*, \mathbf{x_i}\rangle\|]$$
$$\leq b^2 \cdot W_2 \cdot [\|\mathbf{W}_1\|\|\mathbf{x_i}\| + \|\mathbf{W}_1^*\|\|\mathbf{x_i}\|]$$
$$\leq b^2 W_2 \cdot [\|\mathbf{W}_1\| + \|\mathbf{W}_1^*\|]$$
$$\leq 2 \cdot b^2 \cdot W_2 \cdot W_1$$

Using these and the fact that $|g_2(\mathbf{W}_1, \mathbf{w}_2, \mathbf{x}_i)| \leq b$ in Step 3

$$\geq \frac{2a}{b}\epsilon - 2b \cdot 2b^2 W_2 W_1 \cdot bW_2[2 \cdot \|\mathbf{W}_1^*\| + \frac{\epsilon}{\kappa}]$$
$$\geq \frac{2a}{b}\epsilon - 4b^4 W_2^2 W_1 \cdot W_1 - 4b^4 W_2^2 W_1 \frac{\epsilon}{\kappa}$$
$$= \frac{2a}{b}\epsilon - 4b^4 W_2^2 W_1 \cdot W_1 - 4b^4 W_2^2 W_1 \epsilon \left(\frac{a}{4b^5 W_2^2 W_1} - \frac{W_1}{\epsilon}\right)$$
$$= \frac{a}{b}\epsilon \geq 0$$

The idea of the proof is similar to that of previous theorems. The proof uses the fact that the minimum value of the quasigradient of g is a. □

5.8. Proof of Theorem 5

Proof. Consider $\mathbf{W_1} \in \mathbb{B}(0, W_1)$, $\|\mathbf{W_1}\| \leq W_1$, $\|\mathbf{W_2}\| \leq W_2$ such that $e\hat{r}r(\mathbf{W_1}, \mathbf{W_2}) \geq \epsilon$. Let $\mathbf{V_1}$ be a point $\frac{\epsilon}{\kappa}$ close to minima $\mathbf{W_1}$, where $\kappa = \left(\frac{a}{4b^5 W_2^2 W_1} - \frac{W_1}{\epsilon}\right)^{-1}$

Let $\mathbf{W_2} \in \mathbb{R}^{d' \times d''} = [\mathbf{w_1^2}, \mathbf{w_2^2}, \ldots \mathbf{w_{d''}^2}]$.

Note here that,

$$\nabla_{\mathbf{W_1}} e\hat{r}r(\mathbf{W_1}, \mathbf{W_2})$$
$$= \nabla_{\mathbf{W_1}} \frac{1}{m} \sum_{i=1}^{m} \|\mathbf{y_i} - \phi_2\langle \mathbf{W_2}, \phi_1\langle \mathbf{W_1}, \mathbf{x_i}\rangle\rangle\|^2$$
$$= \nabla_{\mathbf{W_1}} \frac{1}{m} \sum_{j=1}^{d''} \sum_{i=1}^{m} \|\mathbf{y_{ij}} - \phi_2\langle \mathbf{w_j^2}, \phi_1\langle \mathbf{W_1}, \mathbf{x_i}\rangle\rangle\|^2$$
$$= \sum_{j=1}^{d''} \nabla_{\mathbf{W_1}} \frac{1}{m} \sum_{i=1}^{m} \|\mathbf{y_{ij}} - \phi_2\langle \mathbf{w_j^2}, \phi_1\langle \mathbf{W_1}, \mathbf{x_i}\rangle\rangle\|^2$$

$$= \sum_{j=1}^{d''} \nabla_{\mathbf{W_1}} e\hat{r}r(\mathbf{W_1}, \mathbf{w_j^2})$$

Now,

$$\langle \nabla_{\mathbf{W_1}} e\hat{r}r(\mathbf{W_1}, \mathbf{W_2}), \mathbf{W_1} - \mathbf{V_1} \rangle_F$$

$$= \sum_{j=1}^{d''} \langle \nabla_{\mathbf{W_1}} e\hat{r}r(\mathbf{W_1}, \mathbf{w_j^2}), \mathbf{W_1} - \mathbf{V_1} \rangle_F$$

Observe here that $\|\mathbf{W_2}\| \leq W_2 \implies \|\mathbf{w}_j^2\| \leq W_2 \forall j$. Using this and the result from theorem 4 we get that each term $\langle \nabla_{\mathbf{W_1}} e\hat{r}r(\mathbf{W_1}, \mathbf{w_j^2}), \mathbf{W_1} - \mathbf{V_1} \rangle_F \geq \frac{a}{b}\epsilon$. Hence, we get that:

$$\langle \nabla_{\mathbf{W_1}} e\hat{r}r(\mathbf{W_1}, \mathbf{W_2}), \mathbf{W_1} - \mathbf{V_1} \rangle_F \geq \frac{a}{b}\epsilon d'' \geq 0$$

$\square$

5.9. Proof of Theorem 6

Proof. Consider $\mathbf{w} \in \mathbb{B}(0, W)$, $\|\mathbf{w}\| \leq W$ such that $e\hat{r}r_i(\mathbf{w}) = -(y_i \log(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle)) + (1 - y_i)(1 - \log(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle)) \geq \epsilon$ ($\implies e\hat{r}r_m(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} -(y_i \log(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle) + (1 - y_i)(1 - \log(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle))) \geq \epsilon$, where m is the total number of samples). Also let $\mathbf{v}$ be a point ϵ/κ-close to minima $\mathbf{w}^*$ with $\kappa = \frac{\epsilon}{(1-e^{-\epsilon})^2}$.

Consider the case when $y_i = 1$. In this case $e\hat{r}r_i(\mathbf{w}) = -\log(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle) \geq \epsilon$. Using $-\log p \geq \epsilon \implies (1 - p)^2 \geq (1 - e^{-\epsilon})^2$, we get that $(y_i - \phi\langle \mathbf{w}, \mathbf{x}_i \rangle)^2 \geq (1 - e^{-\epsilon})^2$. In the other case when $y_i = 0$, $e\hat{r}r_i(\mathbf{w}) = -\log(1 - \phi\langle \mathbf{w}, \mathbf{x}_i \rangle) \geq \epsilon$. Here using $-\log(1 - p) \geq \epsilon \implies (p)^2 \geq (1 - e^{-\epsilon})^2$, we get that $\implies (y_i - \phi\langle \mathbf{w}, \mathbf{x}_i \rangle)^2 \geq (1 - e^{-\epsilon})^2$. Combining these we get, $(y_i - \phi\langle \mathbf{w}, \mathbf{x}_i \rangle)^2 \geq (1 - e^{-\epsilon})^2$ for all i. Then:

$$\langle \nabla err(\mathbf{w}), \mathbf{w} - \mathbf{v} \rangle$$

$$= \frac{1}{m}\sum_{i=1}^{m} (\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)\langle \mathbf{x}_i, \mathbf{w} - \mathbf{v} \rangle \qquad \text{(Step 1)}$$

$$= \frac{1}{m}\sum_{i=1}^{m} (\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle)(\langle \mathbf{x}_i, \mathbf{w} - \mathbf{w}^* \rangle + \langle \mathbf{x}_i, \mathbf{w}^* - \mathbf{v} \rangle)$$

$$\text{(Step 2)}$$

$$\geq \frac{1}{m}\sum_{i=1}^{m} (\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle)$$

$$[(\langle \mathbf{w}, \mathbf{x}_i \rangle - \langle \mathbf{w}^*, \mathbf{x}_i \rangle) - \|\mathbf{x}_i\|\|\mathbf{w}^* - \mathbf{v}\|] \qquad \text{(Step 3)}$$

$$\geq \frac{1}{m}\sum_{i=1}^{m} 4(\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle)^2$$

$$- |\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle|\|\mathbf{x}_i\|\|\mathbf{w}^* - \mathbf{v}\| \qquad \text{(Step 4)}$$

$$\geq 4(1 - e^{-\epsilon})^2 - \frac{\epsilon}{\kappa} \qquad \text{(Step 5)}$$

$$= 3(1 - e^{-\epsilon})^2 > 0$$

Step 2 uses the fact that $y_i = \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle$. In Step 4 we use the fact that sigmoid is $\frac{1}{4}$ Lipschitz and so $(\phi(z) - \phi(z'))(z - z') \geq 4(\phi(z) - \phi(z'))^2$. In Step 5 we use $|\phi\langle \mathbf{w}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle| \leq 1$ and $\|\mathbf{w}^* - \mathbf{v}\| \leq \frac{\epsilon}{\kappa}$. $\square$

5.10. Proof of Theorem 7

Proof. Consider $\mathbf{W} \in \mathbb{B}(0, W)$, $\|\mathbf{W}\| \leq W$ such that for all i, $e\hat{r}r_i(\mathbf{W}) = \sum_{j=1}^{d'} -(y_{ij} \log(\phi\langle \mathbf{w}_j, \mathbf{x}_i \rangle)) \geq \epsilon$ ($\implies e\hat{r}r_i(\mathbf{W}) = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{d'} -(y_{ij} \log(\phi\langle \mathbf{w}_j, \mathbf{x}_i \rangle)) \geq \epsilon$, where m is the total number of samples.) Also let $\mathbf{V} = [\mathbf{v}_1 \ \mathbf{v}_2 \cdots \ \mathbf{v}_{d'}]$ be a point ϵ/κ-close to minima $\mathbf{W}^*$ with $\kappa = \frac{\epsilon d'}{(1-e^{-\epsilon})^2}$. Let $G(\mathbf{W})$ be the subgradient of $e\hat{r}r_m(\mathbf{W})$ and $G(\mathbf{w}_j)$ be the subgradient of $e\hat{r}r_m(\mathbf{w}_j)$.

Let for $\mathbf{x}_i$, the correct label be t, then $y_{it} = 1$ and $y_{ij} = 0$, for any $j \neq t$. The error for this one data-point would be $\sum_{j=1}^{d'} -(y_{ij} \log(\phi\langle \mathbf{w}_j, \mathbf{x}_i \rangle)) = -\log(\phi\langle \mathbf{w}_t, \mathbf{x}_i \rangle) \geq \epsilon$. Using $-\log p \geq \epsilon \implies (1 - p)^2 \geq (1 - e^{-\epsilon})^2$, we get that $(y_{it} - \phi\langle \mathbf{w}_t, \mathbf{x}_i \rangle)^2 \geq (1 - e^{-\epsilon})^2$.

Note that for any $\mathbf{x}_i$, $\sum_{j=1}^{d'} \phi\langle \mathbf{w}_j, \mathbf{x}_i \rangle = 1$. Using this we get that $\sum_{j=1, j\neq t}^{d'} \phi\langle \mathbf{w}_j, \mathbf{x}_i \rangle = 1 - \phi\langle \mathbf{w}_t, \mathbf{x}_i \rangle \geq 1 - e^{-\epsilon}$ (The inequality follows as $-\log p \geq \epsilon \implies 1 - p \geq 1 - e^{-\epsilon}$). Now using Cauchy-Schwartz inequality, we get that $\sum_{j=1, j\neq t}^{d'} \phi\langle \mathbf{w}_j, \mathbf{x}_i \rangle^2 \geq (1 - e^{-\epsilon})^2/(d' - 1)$. Adding this with $(y_{it} - \phi\langle \mathbf{w}_t, \mathbf{x}_i \rangle)^2 \geq (1 - e^{-\epsilon})^2$ and recollecting that $y_{it} = 1$ and $y_{ij} = 0$, for any $j \neq t$, we get that $\sum_{j=1}^{d'} (y_{ij} - \phi\langle \mathbf{w}_j, \mathbf{x}_i \rangle)^2 \geq (1 - e^{-\epsilon})^2 \frac{d'}{d'-1}$ Then:

$$\langle G(\mathbf{W}), \mathbf{W} - \mathbf{V} \rangle$$

$$= \sum_{j=1}^{d'} \langle G(\mathbf{w_j}), \mathbf{w_j} - \mathbf{v_j} \rangle_F$$

$$\text{(By definition of Frobenius inner product)}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{d'} \left(\phi\langle \mathbf{w_j}, \mathbf{x}_i \rangle - y_{ij}\right)\langle \mathbf{x}_i, (\mathbf{w_j} - \mathbf{v_j}) \rangle \qquad \text{(Step 1)}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{d'} \left(\phi\langle \mathbf{w_j}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w_j^*}, \mathbf{x}_i \rangle\right)$$

$$[\langle \mathbf{x}_i, \mathbf{w_j} - \mathbf{w_j^*} \rangle + \langle \mathbf{x}_i, \mathbf{w_j^*} - \mathbf{v_j} \rangle] \qquad \text{(Step 2)}$$

$$\geq \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{d'} \Big[2\left(\phi\langle \mathbf{w_j}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w_j^*}, \mathbf{x}_i \rangle\right)^2$$

$$+ \left(\phi\langle \mathbf{w_j}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w_j^*} \mathbf{x}_i \rangle\right)\langle \mathbf{x}_i, \mathbf{w_j^*} - \mathbf{v_j} \rangle\Big] \qquad \text{(Step 3)}$$

$$\geq \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{d'} \Big[2\left(\phi\langle \mathbf{w_j}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w_j^*}, \mathbf{x}_i \rangle\right)^2$$

$$- |\phi\langle \mathbf{w_j}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w_j^*}, \mathbf{x}_i \rangle|\|\mathbf{x}_i\|\|\mathbf{w_j^*} - \mathbf{v_j}\|\Big] \qquad \text{(Step 4)}$$

$$\geq \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{d'} \Big[2\left(\phi\langle \mathbf{w_j}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w_j^*}, \mathbf{x}_i \rangle\right)^2 - \frac{\epsilon}{\kappa}\|\mathbf{x}_i\|\Big] \qquad \text{(Step 5)}$$

$$\geq 2(1 - e^{-\epsilon})^2 \frac{d'}{d'-1} - \frac{d'\epsilon}{\kappa} \qquad \text{(Step 6)}$$

$$\geq (1 - e^{-\epsilon})^2 \frac{d'+1}{d'-1} > 0$$

Step 2 uses the fact that $y_{ij} = \phi\langle \mathbf{w_j^*}, \mathbf{x}_i \rangle$. In Step 3, we use the fact that softmax is $\frac{1}{2}$ Lipschitz and so $(\phi(z) - \phi(z'))(z - z') \geq 2(\phi(z) - \phi(z'))^2$. In Step 5, we use $|\phi\langle \mathbf{w_j}, \mathbf{x}_i \rangle - \phi\langle \mathbf{w_j^*}, \mathbf{x}_i \rangle| \leq 1$ and $\|\mathbf{w_j^*} - \mathbf{v_j}\| \leq \frac{\epsilon}{\kappa}$. $\square$

6. Conclusion and Future Work

In this work, we presented a novel methodology, Deep Alter-Nations for Training nEural networks (DANTE), to effectively train neural networks using alternating minimization, thus providing a competitive alternative to standard backpropagation. We formulated the task of training each layer of a neural network (in particular, an autoencoder without loss of generality) as a Strictly Locally Quasi-Convex (SLQC) problem, and leveraged recent results to use Stochastic Normalized Gradient Descent (SNGD) as an effective method to train each layer of the network. While earlier work [9] simply identified the SLQC nature of sigmoidal GLMs, we introduced a new generalized ReLU activation, and showed that a multi-output layer satisfies this SLQC property, thus allowing us to expand the applicability of the proposed method to networks with both sigmoid and ReLU family of activation functions. In particular, we extended the definitions of local quasi-convexity in order to prove that a one hidden-layer neural network with generalized ReLU activation is $\left(\epsilon, \frac{2b^3W}{a}, \mathbf{W_2^*}\right) - SLQC$ in W_2 (the same result holds for a GLM) and $\left(\epsilon, \left(\frac{a}{4b^5W_2^2W_1} - \frac{W_1}{\epsilon}\right)^{-1}, \mathbf{W_1^*}\right) - SLQC$ in W_1, which improves the convergence bound for SLQC in the GLM with the generalized ReLU (as compared to a GLM with sigmoid). We also showed how DANTE can be extended to train multi-layer neural networks. We empirically validated DANTE with both sigmoidal and ReLU activations on standard datasets as well as in a multi-layer setting, and observed that it provides a competitive alternative to standard backprop-SGD, as evidenced in the experimental results.

Future Work and Extensions

DANTE can not only be used to train multi-layer neural networks from scratch, but can also be combined with back-prop SGD, which can be used to finetune the network end-to-end periodically. Our future work will involve a more careful study of the proposed method for deeper neural networks, as well as in studying convergence guarantees of the proposed alternating minimization strategy. In this paper, we focused on validating the feasibility of DANTE for MLPs; however the ideas should work for more advanced networks too. In our future work, we plan to study the extensions of DANTE to convolutional layers, LSTMs and other architectural variants.

Acknowledgements

References

[1] S. Hochreiter, J. Schmidhuber, Long Short-term Memory, Neural Computation 8 (9) (1997) 1735–1780.

[2] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, Y. Bengio, Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, in: Advances in neural information processing systems, 2014, pp. 2933–2941.

[3] Y. Tian, Symmetry-breaking convergence analysis of certain two-layered neural networks with relu nonlinearity (2016).

[4] S. Shalev-Shwartz, O. Shamir, S. Shammah, Failures of gradient-based deep learning, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 3067–3075.

[5] T. Blumensath, M. E. Davies, Iterative Hard Thresholding for Compressed Sensing, Applied and Computational Harmonic Analysis 27 (3) (2009) 265–274.

[6] P. Jain, P. Netrapalli, S. Sanghavi, Low-rank Matrix Completion using Alternating Minimization, in: 45th Annual ACM Symposium on Theory of Computing (STOC), 2013.

[7] A. Anandkumar, R. Ge, Efficient Approaches for Escaping Higher Order Saddle Points in Non-Convex Optimization, in: 29th Conference on Learning Theory (COLT), 2016.

[8] E. Malach, S. Shalev-Shwartz, A provably correct algorithm for deep learning that actually works, arXiv preprint arXiv:1803.09522 (2018).

[9] E. Hazan, K. Y. Levy, S. Shalev-Shwartz, Beyond Convexity: Stochastic Quasi-Convex Optimization, in: 29th Annual Conference on Neural Information Processing Systems (NIPS), 2015, pp. 1594–1602.

[10] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning Internal Representation by Back-propagating Errors, Nature 323 (9) (1986) 533–536.

[11] Y. Chauvin, D. E. Rumelhart, Backpropagation: Theory, Architectures, and Applications, Psychology Press, 1995.

[12] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, T. Goldstein, Training Neural Networks Without Gradients: A Scalable ADMM Approach, in: 33rd International Conference on Machine Learning (ICML), 2016.

[13] A. Choromanska, B. Cowen, S. Kumaravel, R. Luss, M. Rigotti, I. Rish, P. Diachille, V. Gurev, B. Kingsbury, R. Tejwani, D. Bouneffouf, Beyond backprop: Online alternating minimization with auxiliary variables, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, Vol. 97 of Proceedings of Machine Learning Research, PMLR, Long Beach, California, USA, 2019, pp. 1193–1202.
URL http://proceedings.mlr.press/v97/choromanska19a.html

[14] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, K. Kavukcuoglu, Decoupled neural interfaces using synthetic gradients, arXiv preprint arXiv:1608.05343 (2016).

[15] G. Jagatap, C. Hegde, Learning relu networks via alternating minimization, arXiv preprint arXiv:1806.07863 (2018).

[16] Y. Bengio, How auto-encoders could provide credit assignment in deep networks via target propagation, arXiv preprint arXiv:1407.7906 (2014).

[17] D.-H. Lee, S. Zhang, A. Fischer, Y. Bengio, Difference target propagation, in: Joint european conference on machine learning and knowledge discovery in databases, Springer, 2015, pp. 498–515.

[18] Y. Bengio, D.-H. Lee, J. Bornschein, T. Mesnard, Z. Lin, Towards biologically plausible deep learning, arXiv preprint arXiv:1502.04156 (2015).

[19] T. P. Lillicrap, D. Cownden, D. B. Tweed, C. J. Akerman, Random feedback weights support learning in deep neural networks, arXiv preprint arXiv:1411.0247 (2014).

[20] A. Nøkland, Direct feedback alignment provides learning in deep neural networks, in: Advances in Neural Information Processing Systems, 2016, pp. 1037–1045.

[21] A. L. Maas, A. Y. Hannun, A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: ICML, Vol. 30, 2013, p. 3.

[22] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: ICCV, 2015, pp. 1026–1034.

[23] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in neural information processing systems, 2007, pp. 153–160.

[24] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, D. Ha, Deep learning for classical japanese literature (2018). arXiv:cs.CV/1812.01718.