TRAINING AUTOENCODERS BY ALTERNATING MINIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this work, we present a novel method, DANTE, to train neural networks - in particular, autoencoders - using alternating minimization. This method provides a different perspective in lieu of traditional gradient-based backpropagation commonly used to train deep networks such as autoencoders. DANTE utilizes an adaptation of quasi-convex optimization techniques to cast autoencoder training as a bi-quasi-convex optimization problem. We show that for autoencoder configurations with both differentiable (e.g. sigmoid) and non-differentiable (e.g. ReLU) activation functions, we can perform alternations effectively. In experiments on standard datasets, autoencoders trained using DANTE were found to be very promising when compared to those trained using traditional backpropagation techniques, both in terms of solutions as well as feature extraction and reconstruction performance. We also extended DANTE to multi-layer settings and showed that the proposed method performs promisingly in this setting too.

1 INTRODUCTION

For much of the recent march of deep learning, gradient-based backpropagation methods (e.g. Stochastic Gradient Descent, SGD) 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. On one hand, 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 overshadowed the need to continue to look for options beyond backprogagation to train deep networks. Despite several advancements in deep learning with respect to novel architectures such as encoder-decoder networks and generative adversarial models, the reliance on backpropagation methods remains. While reinforcement learning methods are becoming increasingly used [REF] today, their scope is limited to a particular family of settings such as agent-based systems or reward-based learning. Recent efforts have studied the limitations of SGD-based backpropagation, including parallelization of SGD-based techniques that are inherently serial Taylor et al. (2016), vanishing gradients, especially for certain activation functions Hochreiter & Schmidhuber (1997), convergence of stochastic techniques to local optima Anandkumar & Ge (2016), and many more. For a well-referenced critique of gradient-based methods, we request the reader to refer Taylor et al. (2016).

From another perspective, there has been marked progress in the area of non-convex optimization (beyond deep learning) in recent years, which has resulted in scalable methods such as iterated hard thresholding Blumensath & Davies (2009) and alternating minimization Jain et al. (2013) as methods of choice for solving large-scale sparse recovery, matrix completion, and tensor factorization tasks. Several of these methods not only scale well to large problems, but also offer provably accurate solutions. In this work, we seek to formulate a non-backpropagation strategy to train neural networks, leveraging a recent result in quasi-convex optimization. Our method is called DANTE (Deep AlterNations for Training autoEncoders), an alternating minimization-based technique for training neural networks - in particular, autoencoders.

DANTE is based on our observation that the problem of training a single hidden-layer autoencoder 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 autoencoder, where each step involves relatively simple quasi-convex problems. DANTE then uses efficient solvers for quasi-convex problems including normalized gradient descent Nesterov (1984) and the more recent stochastic normalized gradient descent Hazan et al. (2015) to train an autoencoder. The key contributions of this work are summarized below:

- We introduce DANTE, a new method to efficiently train neural networks using alternating minimization. DANTE trains Autoencoder (AE) networks through alternating optimization of quasi-convex - Strictly-Locally-Quasi-Convex (SLQC), to be specific - problems using an efficient stochastic solver.
- DANTE views each layer of a neural network as a Generalized Linear Model (GLM). While it has been recently shown that a GLM with sigmoid activation function is SLQC Hazan et al. (2015), we introduce a generalized version of the popularly used ReLU activation function, and show that a GLM with such non-differentiable activation functions can also be shown to be SLQC. This allows us to apply our methodology to AEs with both differentiable and non-differentiable activation functions, including ReLUs and sigmoid.
- We show that the Stochastic Normalized Gradient Descent (SNGD) method offers provably more rapid convergence under this newly introduced activation function for an idealized GLM problem than it does for the sigmoidal activation function. A key advantage of our approach is that the theoretical result provides a direction to set the learning rate and batch size to provide an acceptable level of convergence while training the AE.
- We empirically validate DANTE with both generalized ReLU and sigmoid activations and establish that the proposed method provides comparable or better test error, reconstructions and classification performance (with the learned representations), when compared with a regularly used mini-batch SGD setup.
- We also show that the proposed methodology can be extended to train deep AEs, and our results on deep AEs show promise in this setting too.

2 Related Work

Backpropagation-based techniques date back to the early days of neural network research Rumelhart et al. (1986); Chauvin & Rumelhart (1995) 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. Recent years have seen the development of other methods, predominantly based on least-squares approaches, used to train neural networks. Carreira-Perpinan and Wang Carreira-Perpinan & Wang (2014) proposed a least-squares based method to train a neural network. In particular, they introduced the Method of Auxiliary Constraints (MAC), and used quadratic penalties to enforce equality constraints. Patel *et al.* Patel et al. (2015) proposed an Expectation-Maximization (EM) approach derived from a hierarchical generative model called the Deep Rendering Model (DRM), and also used least-squared parameter updates in each of the EM steps. They showed that forward propagation in a convolutional neural network was equivalent to the inference on their DRM. Unfortunately, neither of these methods has publicly available implementations or even published training results to compare against.

More recently, Taylor *et al.* proposed a method to train neural networks using the Alternating Direction Method of Multipliers (ADMM) and Bregman iterations Taylor et al. (2016). The focus of this method, however, was on scaling the training of neural networks to a distributed setting on multiple cores across a computing cluster. Jaderberg also proposed the idea of 'synthetic gradients' in Jaderberg et al. (2016). While this approach is interesting, this work is more towards a more efficient way to carry out gradient-based parameter updates in a neural network. In this work, we focus on a new approach to train neural networks - in particular, autoencoders - using alternating optimization, quasi-convexity and SNGD, and show that this approach shows promising results on the considered datasets. To the best of our knowledge, this is the first such effort in using alternating SNGD to train neural networks with related performance guarantees.

3 DANTE: DEEP ALTERNATIONS FOR TRAINING AUTOENCODERS

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 $\phi_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$. The activations generated by the 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 $\mathbf{a}_l = \phi_l\langle W_l, \mathbf{a}_{l-1}\rangle$, 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:

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

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 some distribution $\mathcal{D}$, the network is trained by tuning the weights $\mathbf{W}$ to minimize a given loss function, J:

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

Note that a multi-layer autoencoder is trained similarly, but with the loss function modified as below:

$$\min_{\mathbf{W}} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[J(f(\mathbf{W}; \mathbf{x}), \mathbf{x})] \tag{3}$$

For purposes of simplicity and convenience, we consider the case of a single-layer autoencoder, 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 describe in a later section on how this idea can be extended to deep multi-layer autoencoders. (Note that our definition of a single-layer autoencoder is equivalent to a two-layer neural network in a classification setting, by nature of the autoencoder.)

A common loss function used to train autoencoders is the squared loss function which, in our simplified setting, yields the following objective.

$$\min_{\mathbf{W}} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\|f(\mathbf{W}; \mathbf{x}) - \mathbf{x}\|_2^2 \tag{4}$$

Now denote

$$\|f(\mathbf{W}; \mathbf{x}) - \mathbf{x}\|_2^2 = \|\phi_2\langle W_2, \phi_1\langle W_1, \mathbf{x}\rangle\rangle - \mathbf{x}\|_2^2 \tag{5}$$

An important observation here is that if we fix W_1, then Eqn (5) turns into a Generalized Linear Model problem with ϕ_2 as the activation function, i.e.

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

where $\mathbf{z} = \phi_1\langle W_1, \mathbf{x}\rangle$. Similarly, inverting the network and fixing W_2 turns the problem into yet another Generalized Linear Model problem, this time with W_1 as the parameter.

$$\min_{W_1} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\|\phi\langle W_1, \mathbf{x}\rangle - \mathbf{x}\|_2^2$$

where $\phi\langle\cdot\rangle = \phi_2\langle W_2, \phi_1\langle\cdot\rangle\rangle$. Thus, a single-layer autoencoder is a combination of two Generalized Linear Model (GLM) problems, and we exploit this key observation in this work. In particular, we leverage a recent result in Hazan et al. (2015) that a GLM with a differentiable activation, such as sigmoid, is Strictly Locally Quasi-Convex (SLQC), which allows us to use SNGD to solve each sub-problem of the alternating setup efficiently (with performance guarantees). We further show that a GLM with a non-differentiable activation – in particular, a generalized Rectified Linear Unit (ReLU) – is also SLQC, thus allowing us to extend the proposed alternating strategy, DANTE, to ReLU-based autoencoders too. We note that while we have developed this idea to train autoencoders in this work (since our approach relates closely to the greedy layerwise training in autoencoders), DANTE can be used to train standard multi-layer neural networks too (discussed in Section 5).

Algorithm 1: Stochastic Normalized Gradient Descent (SNGD)

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

for $t = 1$ *to* T **do**
  Sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^b \sim \mathcal{D}$
  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

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

Algorithm 2: DANTE: Deep AlterNations for Training autoEncoders

Input : Stopping threshold ϵ, Number of iterations of alternating minimization T_{AM}, Number of iterations for each SNGD T_{SNGD}, initial values $\mathbf{w}_1^0, \mathbf{w}_2^0$, 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**
  $W_2^t \leftarrow \arg\min_W \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\|\phi_2\langle W, \phi_1\langle W_1^{t-1}, \mathbf{x}\rangle\rangle - \mathbf{x}\|_2^2$ `//use SNGD`
  $W_1^t \leftarrow \arg\min_W \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\|\phi_2\langle W_2^t, \phi_1\langle W, \mathbf{x}\rangle\rangle - \mathbf{x}\|_2^2$ `//use SNGD`
  $t := t + 1$
end

Output : $\mathbf{w}_1, \mathbf{w}_2$

3.2 METHODOLOGY

We begin our presentation of the proposed method by briefly reviewing the Stochastic Normalized Gradient Descent (SNGD) method, which is used in each step of our alternating strategy. (We explain in the next section, the rationale for the choice of SNGD).

Stochastic Normalized Gradient Descent (SNGD): Normalized Gradient Descent (NGD) is an adaptation of traditional Gradient Descent where the updates in each iteration are purely based on the direction of the gradients, and not the gradients themselves (achieved by normalizing the gradients). SNGD is the stochastic version of NGD, where weight updates are performed at the end of every training sample, instead of the complete set of samples. Mini-batch SNGD updates the parameters at the end of every mini-batch of samples, as in 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 GLM problem.

DANTE: Given this background, Algorithm 2 outlines the proposed method, DANTE. Consider the autoencoder problem below:

$$\min_{\mathbf{W}} \; f(W_1, W_2) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\|\phi_2\langle W_2, \phi_1\langle W_1, \mathbf{x}\rangle\rangle - \mathbf{x}\|_2^2$$

Upon fixing the parameters of the lower layer i.e. W_1, it is easy to see that we are left with a GLM problem:

$$\min_{W} \; \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\|\phi_2\langle W_2, \mathbf{z}\rangle - \mathbf{x}\|_2^2,$$

where $\mathbf{z} = \phi_1\langle W_1, \mathbf{x}\rangle$. DANTE solves this intermediate problem using SNGD steps by sampling several mini-batches of data points and performing updates as dictated by Algorithm 1. Similarly, upon fixing the parameters of the upper layer, i.e. W_2, we are left with another GLM problem:

$$\min_{W} \; \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\|\phi_1\langle W_1, \mathbf{v}\rangle - \mathbf{x}\|_2^2,$$

where $\mathbf{v} = \phi_2\langle W_2, \mathbf{x}\rangle$. This is once again solved by mini-batch SNGD, as before.

3.3 RATIONALE

To describe the motivation for our alternating strategy in DANTE, we first define key terms and results that are essential for our work. We present the notion of a locally quasi-convex function (as introduced in Hazan et al. (2015)) and show that under certain realizability conditions, empirical objective functions induced by Generalized Linear Models (GLMs) are locally quasi-convex. To this end, we introduce a new activation function, the generalized ReLU, and show that the GLM with the generalized ReLU also satisfies this property. We then cite a result that shows that SNGD converges to the optimum solution provably for locally quasi-convex functions, and extend this result to the newly introduced activation function.

Definition 3.1 (***Local-Quasi-Convexity***)**.** Let $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d, \kappa, \epsilon > 0$ and let $f : \mathbb{R}^d \to \mathbb{R}$ 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 $\mathbb{B}(\mathbf{z}, \epsilon/\kappa)$ refers to a ball centered at $\mathbf{z}$ with radius ϵ/κ.

Definition 3.2 (***Idealized and Noisy Generalized Linear Model (GLM)***)**.** Given an (unknown) distribution $\mathcal{D}$ and an activation function $\phi : \mathbb{R} \to \mathbb{R}$, an idealized GLM is defined by the existence of a $\mathbf{w}^* \in \mathbb{R}^d$ such that $y_i = \phi\langle \mathbf{w}^*, \mathbf{x}_i \rangle \forall i \in [m]$ where $\mathbf{w}^*$ is the global minimizer of the error function:

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

Similarly, a noisy GLM is defined by the existence of a $\mathbf{w}^* \in \mathbb{R}^d$, which is the global minimizer of the error function: $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}(y|\mathbf{x}) = \phi(\langle \mathbf{w}^*, \mathbf{x} \rangle)$. the expected error is then:

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

(Hazan et al., 2015, Lemma 3.2) showed that if we draw $m \geq \Omega\left(\frac{\exp(2\|\mathbf{w}^*\|)}{\epsilon^2} \log \frac{1}{\delta}\right)$ samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ from a GLM with the sigmoid activation function, then with probability at least $1 - \delta$, the empirical error function

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

is $(\epsilon, e^{\|\mathbf{w}^*\|_2}, \mathbf{w}^*)$-SLQC in $\mathbf{w}$.

In order to go beyond the sigmoid activation function, we now introduce a new generalized ReLU activation function.

Definition 3.3. ***(Generalized ReLU)*** The generalized ReLU function $f : \mathbb{R} \to \mathbb{R}$, $0 < a < b$, $a, b \in \mathbb{R}$ is defined as:

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

This function is differentiable at every point except 0. Note that this definition also subsumes variants of ReLU such as the leaky ReLU Xu et al. (2015). We define the function g that gives a valid subgradient at all x for the generalized ReLU to be

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

While SLQC is primarily defined for differentiable functions, we now show that with the above definition of the subgradient, the GLM with the generalized ReLU is also SLQC.

Theorem 3.4. *In the idealized GLM with generalized ReLU activation, assuming* $||\mathbf{w}^*|| \leq W$, $\hat{err}(\mathbf{w})$ *is* $\left(\epsilon, \frac{2b^2W}{a}, \mathbf{w}^*\right) - SLQC$ *in* $\mathbf{w}$ *for all* $\boldsymbol{w} \in \mathbb{B}_d(0, W)$.

Proof. Consider $||\mathbf{w}|| \leq W$ 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 $\kappa = \frac{2b^2W}{a}$. 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:

$$
\begin{aligned}
&\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) \left[\langle \mathbf{x}_i, \mathbf{w} - \mathbf{w}^*\rangle + \langle \mathbf{x}_i, \mathbf{w}^* - \mathbf{v}\rangle\right] && \text{(Step 1)} \\
&\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 + \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] \\
& && \text{(Step 2)} \\
&\geq \frac{2}{m}\sum_{i=1}^{m} g\langle \mathbf{w}, \mathbf{x}_i\rangle (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}|| \\
&\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 - |\phi\langle \mathbf{w}, \mathbf{x}_i\rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i\rangle|||\mathbf{x}_i||||\mathbf{w}^* - \mathbf{v}|| && \text{(Step 3)} \\
&\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 - b||\langle \mathbf{w}, \mathbf{x}_i\rangle - \langle \mathbf{w}^*, \mathbf{x}_i\rangle||\frac{\epsilon}{\kappa}||\mathbf{x}_i|| && \text{(Step 4)} \\
&\geq 2ab^{-1}\epsilon - \frac{a\epsilon}{bW}||\langle \mathbf{w}, \mathbf{x}_i\rangle - \langle \mathbf{w}^*, \mathbf{x}_i\rangle||||\mathbf{x}_i|| && \text{(Step 5)} \\
&\geq ab^{-1}\epsilon(2 - \frac{1}{W}||\mathbf{w} - \mathbf{w}^*||||\mathbf{x}_i||^2) \\
&\geq 0 && \square
\end{aligned}
$$

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 ($\mathbf{w} \in \mathbb{B}_d(0, W)$, and $\mathbf{x}_i \in \mathbb{B}_d$, the unit d-dimensional ball, as in Defn 2.6).

We also prove a similar result for the Noisy GLM below.

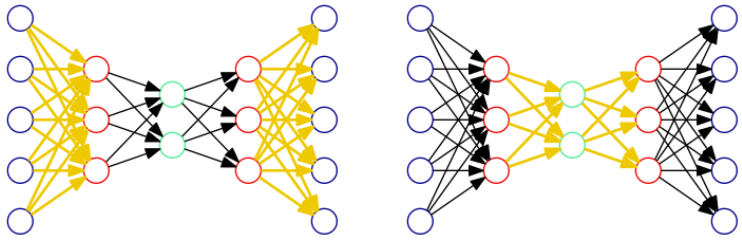
Theorem 3.5. *In the noisy GLM with generalized ReLU activation, assuming $||\mathbf{w}^*|| \leq W$, given $w \in B(0, W)$, then with probability $\geq 1 - \delta$ after $m \geq \frac{4Wb^2}{a\epsilon^2}log(1/\delta)/\epsilon^2$ samples, $\hat{err}(\mathbf{w})$ is $\left(\epsilon, \frac{2b^2W}{a}, \mathbf{w}^*\right) - SLQC$ in $\mathbf{w}$.*

The proof for Theorem 3.5 is included in Appendix A (Supplementary Section).

Now, we review a result from Hazan et al. (2015) which shows that SNGD provably converges to the optimum for SLQC functions, and hence, with very high probability, for empirical objective functions induced by noisy GLM instances too.

Theorem 3.6 (Hazan et al. (2015))**.** *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 $1 - 2\delta$, $f(\mathbf{w}) - f(\mathbf{w}^*) \leq 3\epsilon$.*

Thus, given a single-layer autoencoder with either a sigmoid activation function or a generalized ReLU activation function (since tanh is a rescaled sigmoid and the generalized ReLU includes many variants of ReLUs, this covers a large family of activation functions), DANTE uses SNGD to solve a SLQC problem in each alternating step, which converges to the optima solution in each step with



(a) Phase - 1 (b) Phase - 2

Figure 1: An illustration of the proposed multi-layer DANTE (best viewed in color). In each training phase, the outer pairs of weights (shaded in gold) are treated as a single-layer autoencoder to be trained using single-layer DANTE, followed by the inner single-layer auroencoder (shaded in black). These two phases are followed by a finetuning process that may be empirically determined, similar to standard deep autoencoder training.

Algorithm 3: DANTE for a multi-layer autoencoder

Input : Encoder e with weights $\mathbf{U}$, Decoder d with weights $\mathbf{V}$, Number of hidden layers $2n-1$, Learning rates η, Stopping threshold ϵ, Number of iterations of alternating minimization T_{AM}, initial values $\mathbf{U}^0, \mathbf{V}^0$, minibatch size b

$t := 1$

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

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

    `//Use SNGD for minimizations`

    $$\mathbf{u}_l^t \leftarrow \arg\min_{\mathbf{u}} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left(d(e(\mathbf{x}, \mathbf{U}^t_{[1:l-1]} \cdot \mathbf{u} \cdot \mathbf{U}^{t-1}_{[l+1:n-1]}),\ \mathbf{V}^t_{[1:l-1]} \cdot \mathbf{V}^{t-1}_{[l:n-1]}) - \mathbf{x} \right)^2$$

    $$\mathbf{v}_{n-l}^t \leftarrow \arg\min_{\mathbf{v}} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left(d(e(\mathbf{x}, \mathbf{U}^t_{[1:l]} \cdot \mathbf{U}^{t-1}_{[l+1:n-1]}),\ \mathbf{V}^t_{[1:n-l-1]} \cdot \mathbf{v} \cdot \mathbf{V}^{t-1}_{[n-l+1:n-1]}) - \mathbf{x} \right)^2$$

    $t := t + 1$

  end

end

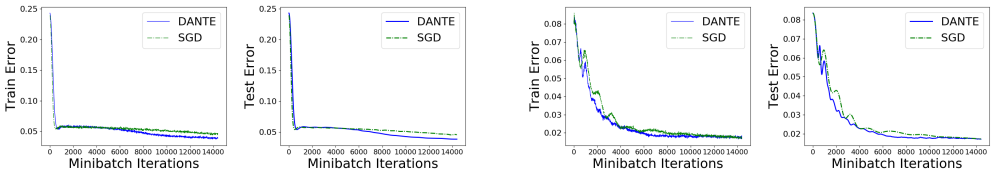
Output : U, V

very high probability, as shown above in Theorem 3.6. Furthermore, while the GLM error function with sigmoid activation has $\kappa = e^W$ Hazan et al. (2015), we obtain $\kappa = \frac{2b^2W}{a}$ (linear in W) for the generalized ReLU setting. This is significant as in Theorem 3.6, the number of iterations T is lower-bounded by an expression with κ in the numerator.

3.4 Extending to a Multi-Layer Autoencoder

In the previous sections, we illustrated how a single hidden-layer autoencoder can be cast as a set of SLQC problems and proposed an alternating minimization method, DANTE. This approach can be generalized to deep autoencoders by considering the greedy layer-wise approach to training a neural network (Bengio et al. (2007)). In this approach, each pair of layers of a deep stacked autoencoder is successively trained in order to obtain the final representation. Each pair of layers considered in this paradigm is a single hidden-layer autoencoder, which can be cast as a pair of SLQC problems that can be trained using DANTE. Therefore, training a deep autoencoder using greedy layer-wise approach can be modeled as a series of SLQC problem pairs. Algorithm 3 summarizes the proposed approach to use DANTE for a deep autoencoder, and Figure 1 illustrates the approach.

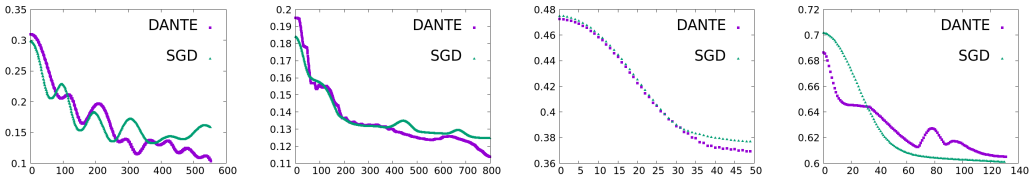
Note that it may be possible to use other schemes to use DANTE for multi-layer autoencoders such as a round-robin scheme, where each layer is trained separately one after the other in the sequence in which the layers appear in the network.



(a) Single-layer autoencoder with Sigmoid activation

(b) Single-layer autoencoder with Leaky ReLU activation

Figure 2: Plots of training (left) and test (right) errors (y-axis) vs training iterations for single hidden-layer autoencoder with Sigmoid (top) and Leaky ReLU (bottom) activations for both DANTE and SGD.



(a) ionosphere dataset (b) satimage dataset (c) svmguide4 dataset (d) USPS dataset

Figure 3: Comparison of proposed DANTE vs SGD on various other datasets. The x-axis on all sub-figures is number of mini-batch iterations and y-axis denotes test error. (Best viewed in color)

4 Experiments and Results

We validated DANTE by training autoencoders on the standard MNIST dataset LeCun et al. (1998) as well as other datasets from the UCI repository.We also conducted experiments with multi-layer autoencoders as well as with single-layer autoencoders with varying number of hidden neurons. Each of these experiments and the corresponding results are described below.

Our first set of experiments were carried out with the standard benchmarking setup of the MNIST dataset[1], with $60,000$ data samples used for training and $10,000$ samples for testing. Experiments were conducted using Torch 7 Collobert et al. (2011).

Autoencoder with Sigmoid Activation: A single-layer autoencoder (equivalent to a neural network with one hidden layer) was trained using DANTE and SGD with 600 hidden units, a sigmoid activation, a learning rate of 0.001, standard Mean-Squared Error loss function and a minibatch size of 500. The results are shown in Figure 2a. While DANTE takes slightly (negligibly) longer to reach a local minimum, it obtains a better solution than SGD.

Autoencoder with ReLU Activation: A single-layer autoencoder was trained using DANTE and SGD with 600 hidden units, a Leaky ReLU activation (with leakiness parameter 0.01), a learning rate of 0.001, Mean-Squared Error loss function and a minibatch size of 500. The results are shown in Figure 2b. The results for ReLU showed an improvement, and DANTE was consistently superior to SGD across the iterations.

In Figure 4, we also show the best reconstructions - both qualitatively and quantitatively - obtained by both trained models (DANTE and SGD) for the autoencoder with the leaky ReLU activation. The model trained using DANTE shows qualitatively better reconstructions, when compared to reconstructions obtained using a model trained by SGD under the same settings. We also conducted experiments to study the effectiveness of the feature representations learned using the models trained using DANTE and SGD in the same setting. After training, we passed the dataset through the autoencoder, extracted the hidden layer representations, and then trained a linear SVM. The

[1] http://yann.lecun.com/exdb/mnist/

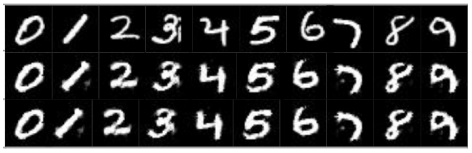


Figure 4: Reconstructions using the autoencoder models with ReLU activation. *Top:* Model trained using SGD; *Bottom:* Model trained using DANTE.

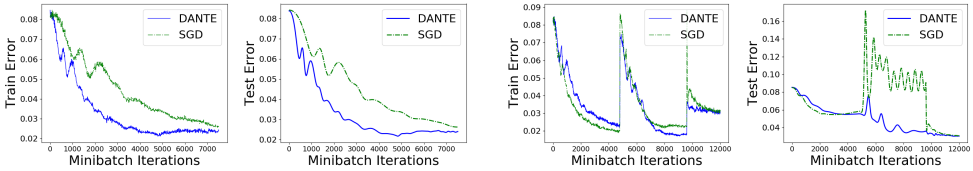
	SGD	DANTE
MNIST	88.65%	89.71%
ionosphere	92.45%	96.22%
satimage	16.50 %	15.70 %
svmguide4	70.37%	87.65%
USPS	89.49%	90.43 %

Table 1: Classification accuracies using ReLU autoencoder features on different datasets

classification accuracy results using the hidden representations on various datasets are given in Table 1.

Varying Number of Hidden Neurons: Given the decomposable nature of the proposed solution to learning autoencoders, we also studied the effect of varying hyperparameters across the layers, in particular, the number of hidden neurons in a single-layer autoencoder. The results of these experiments are shown in Figure 6. The plots firstly show that DANTE performs competitively when compared to SGD. When the number of hidden neurons is relatively lower, both DANTE and SGD show comparable performance (SGD takes longer to reach the same solution, but finds a slightly better solution at the end); however, when the number of hidden neurons increases, DANTE is consistently better. This can be attributed to the fact that the subproblem is relatively more challenging for an alternating optimization setting when the number of hidden neurons is lesser.

Experiments on other datasets: We also studied the performance of proposed method on other standard datasets [2], viz. Ionosphere (34 dimensions, 351 datapoints), Satimage (36 dimensions, 4435 datapoints), SVMGuide4 (10 dimensions, 300 datapoints), and USPS (256 dimensions, 7291 datapoints). Figure 3 shows the performance of the proposed method vs SGD on the above mentioned datasets. It can be clearly seen that the proposed DANTE method demonstrates superior generalization performance on most datasets.



(a) Architecture: 1024->500->500->1024 (b) Architecture: 1024->750->500->750->1024

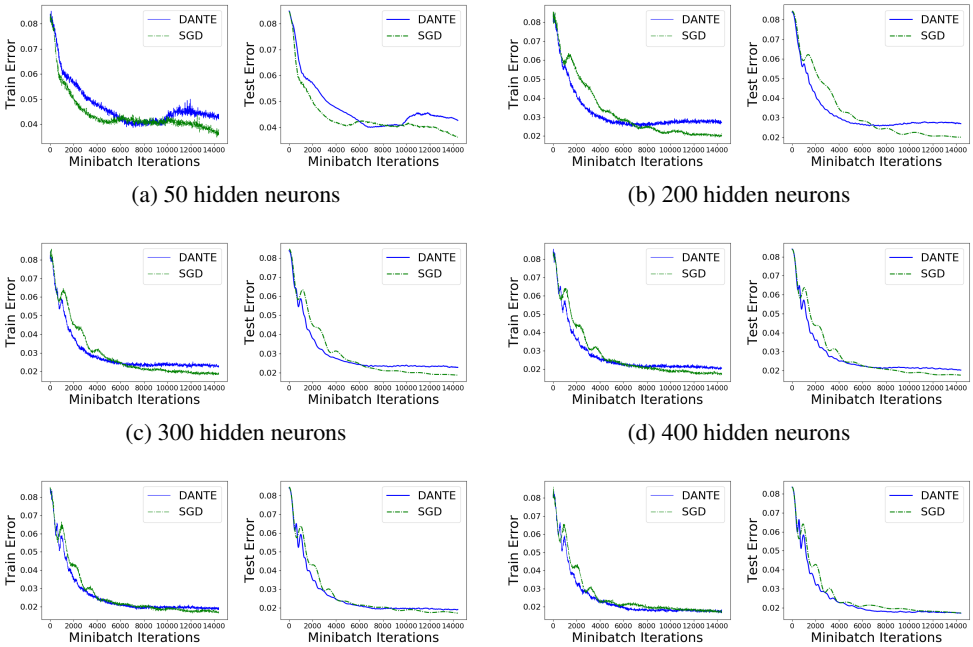
Figure 5: Plots of (a) training error and (b) test error vs training iterations for multi-layer autoencoders with generalized (leaky) ReLU activations for both DANTE and SGD.

Multi-Layer Autoencoder: We also studied the performance of the proposed multi-layer DANTE method (Algorithm 3) for the MNIST dataset. Figure 5 shows the results obtained by stacking two single-layer autoencoders, each with the generalized (leaky) ReLU activation (note that a two single-layer autoencoder corresponds to 4 layers in the overall network, as mentioned in the architecture on the figure). Evidently, the figure shows promising performance for DANTE in this experiment too.

5 Discussions and Conclusion

In this work, we presented a novel methodology, Deep AlterNations for Training autoEncoders (DANTE), to efficiently train autoencoders using a non-backpropagation method, in particular, using alternating minimization. The proposed method formulates training an autoencoder as a set of

[2] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/



(a) 50 hidden neurons (b) 200 hidden neurons

(c) 300 hidden neurons (d) 400 hidden neurons

(e) 500 hidden neurons (f) 600 hidden neurons

Figure 6: Plots of training (left) and test (right) error vs training iterations on a single-layer autoencoder with generalized (leaky) ReLU activation, with a varying number of nodes in the hidden layer.

Generalized Linear Model (GLM) problems, which allows us to use an efficient solver proposed in Hazan et al. (2015) in an alternating fashion to train deep autoencoders. While the earlier work was proposed only for GLMs with sigmoidal activation functions, we propose an extension of local-quasi-convexity to a wider class of activation functions - in particular, the ReLU. We introduce a generalized ReLU activation which allows the Stochastic Normalized Gradient Descent method to have provably faster convergence, when compared to a sigmoidal activation. 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 performs comparably or better than the standard SGD approach to train autoencoders. Our future work will involve a more careful study of the proposed method for deeper autoencoders, as well as in variations of the autoencoder such as the Convolutional Autoencoder Masci et al. (2011).

6 Conclusion

In this work, we have extended the definitions of quasi-convexity to use subgradients in order to enable us to prove that the GLM with generalized ReLU activation is $\left(\epsilon, \frac{b^2 W}{2a}, \mathbf{w}^*\right) - SLQC$, mathematically validating that ReLU is far superior to Sigmoid as an activation function.

Additionally, we have proposed a new algorithm for training autoencoders involving alternating minimization techniques and Stochastic Normal Gradient Descent. We showed experimentally, on using hyperparameters suggested theoretically, our proposed algorithm drastically outperforms traditional methods, with the case of ReLUs showing immense improvement. In this process of experimental validation, we have demonstrated the sensitivity of our algorithm to learning rate and batch size.

REFERENCES

Animashree Anandkumar and Rong Ge. Efficient Approaches for Escaping Higher Order Saddle Points in Non-Convex Optimization. In *29th Conference on Learning Theory (COLT)*, 2016.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pp. 153–160, 2007.

Thomas Blumensath and Mike E. Davies. Iterative Hard Thresholding for Compressed Sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009.

Miguel Carreira-Perpinan and Weiran Wang. Distributed Optimization of Deeply Nested Systems. In *17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 10–19, 2014.

Yves Chauvin and David E Rumelhart. *Backpropagation: Theory, Architectures, and Applications*. Psychology Press, 1995.

R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A Matlab-like Environment for Machine Learning. In *NIPS 2011 Workshop on Algorithms, Systems, and Tools for Learning at Scale (Big Learning)*, 2011.

Elad Hazan, Kfir Y. Levy, and Shai Shalev-Shwartz. Beyond Convexity: Stochastic Quasi-Convex Optimization. In *29th Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 1594–1602, 2015.

Sepp Hochreiter and Jurgen Schmidhuber. Long Short-term Memory. *Neural Computation*, 8(9): 1735–1780, 1997.

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. *arXiv preprint arXiv:1608.05343*, 2016.

Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank Matrix Completion using Alternating Minimization. In *45th Annual ACM Symposium on Theory of Computing (STOC)*, 2013.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. *Artificial Neural Networks and Machine Learning–ICANN 2011*, pp. 52–59, 2011.

Yurii E. Nesterov. Minimization methods for nonsmooth convex and quasiconvex functions. *Matekon*, 29:519–531, 1984.

Ankit B. Patel, Tan Nguyen, and Richard G. Baraniuk. A Probabilistic Theory of Deep Learning. arXiv:1504.00641 [stat.ML], 2015.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Internal Representation by Back-propagating Errors. *Nature*, 323(9):533–536, 1986.

Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training Neural Networks Without Gradients: A Scalable ADMM Approach. In *33rd International Conference on Machine Learning (ICML)*, 2016.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical Evaluation of Rectified Activations in Convolutional Network. arXiv:1505.00853v2 [cs.LG], 2015.

A NOISY GLM WITH GENERALIZED RELU ACTIVATION FUNCTION

The theorem below is a continuation of our discussion from Section **??**.

Theorem A.1. *In the noisy GLM with generalized ReLU activation, assuming* $||\mathbf{w}^*|| \leq W$*, given* $w \in B(0, W)$*, then with probability* $\geq 1 - \delta$ *after* $m \geq \frac{4Wb^2}{a\epsilon^2} log(1/\delta)/\epsilon^2$ *samples,* $\hat{err}(\mathbf{w})$ *is* $\left(\epsilon, \frac{2b^2W}{a}, \mathbf{w}^*\right) - SLQC$ *in* $\mathbf{w}$.

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

$$y_i = \phi\langle \mathbf{w}^*, \mathbf{x}\rangle + \xi_i \tag{6}$$

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 6):

$$\begin{aligned}
\hat{err}_m(\mathbf{w}) &= \frac{1}{m}\sum_{i=1}^{m}(y_i - \phi\langle \mathbf{w}, \mathbf{x}_i\rangle)\rangle)^2 \\
&= \frac{1}{m}\sum_{i=1}^{m}(\phi\langle \mathbf{w}^*, \mathbf{x}_i\rangle - \phi\langle \mathbf{w}, \mathbf{x}_i\rangle)^2 \\
&+ \sum_{i=1}^{m} 2\xi_i(\phi\langle \mathbf{w}^*, \mathbf{x}_i\rangle - \phi\langle \mathbf{w}, \mathbf{x}_i\rangle) + \sum_{i=1}^{m}\xi_i^2
\end{aligned}$$

Therefore, we also have (by definition of noisy GLM in Defn 3.2):

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

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

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

$$
\begin{aligned}
&\geq \frac{2b^{-1}}{m}\sum_{i=1}^{m} g\langle \mathbf{w}, \mathbf{x}_i\rangle(\phi\langle \mathbf{w}^*, \mathbf{x}_i\rangle - \phi\langle \mathbf{w}, \mathbf{x}_i\rangle)^2 \\
&- \frac{2}{m}\sum_{i=1}^{m} g\langle \mathbf{w}, \mathbf{x}_i\rangle \xi_i(\langle \mathbf{w}, \mathbf{x}_i\rangle - \langle \mathbf{w}^*, \mathbf{x}_i\rangle) - 2\frac{\epsilon b}{\kappa}|\mathbf{w} - \mathbf{w^*}| && \text{(Step 3)} \\
&\geq \frac{2b^{-1}}{m}\sum_{i=1}^{m} a\left((\phi\langle \mathbf{w}^*, \mathbf{x}_i\rangle - \phi\langle \mathbf{w}, \mathbf{x}_i\rangle)^2\right. \\
&\qquad -2\xi_i(\phi\langle \mathbf{w}, \mathbf{x}_i\rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i\rangle)) \\
&\qquad + \frac{2}{m}\sum_{i=1}^{m} a\xi_i\left((\langle \mathbf{w}, \mathbf{x}_i\rangle - \langle \mathbf{w}^*, \mathbf{x}_i\rangle)\right. && \text{(Step 4)} \\
&\qquad \left.+2b^{-1}(\phi\langle \mathbf{w}, \mathbf{x}_i\rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i\rangle)\right) - 2\frac{\epsilon b}{\kappa}|\mathbf{w} - \mathbf{w^*}| \\
&\geq 2ab^{-1}\epsilon - 2\frac{\epsilon b}{\kappa}|\mathbf{w} - \mathbf{w^*}| + \frac{1}{m}\sum_{i=1}^{m}\xi\lambda_i(\mathbf{w}) && \text{(Step 5)} \\
&\geq 2ab^{-1}\epsilon - ab^{-1}W^{-1}\epsilon|\mathbf{w} - \mathbf{w^*}| + \frac{1}{m}\sum_{i=1}^{m}\xi_i\lambda_i(\mathbf{w}) && \text{(Step 6)} \\
&\geq 2ab^{-1}\epsilon - ab^{-1}\epsilon + \frac{1}{m}\sum_{i=1}^{m}\xi_i\lambda_i(\mathbf{w}) && \text{(Step 7)} \\
&\geq ab^{-1}\epsilon + \frac{1}{m}\sum_{i=1}^{m}\xi_i\lambda_i(\mathbf{w}) && \text{(Step 8)} \\
& && (7)
\end{aligned}
$$

Here, $\lambda_i(\mathbf{w}) = 2a(\langle \mathbf{w}, \mathbf{x}_i\rangle - \langle \mathbf{w}^*, \mathbf{x}_i\rangle) + 2b^{-1}(\phi\langle \mathbf{w}, \mathbf{x}_i\rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i\rangle)$, and $|\xi_i\lambda_i(\mathbf{w})| \leq 2a(|\langle \mathbf{w}, \mathbf{x}_i\rangle - \langle \mathbf{w}^*, \mathbf{x}_i\rangle| + 2b^{-1}|\phi\langle \mathbf{w}, \mathbf{x}_i\rangle - \phi\langle \mathbf{w}^*, \mathbf{x}_i\rangle|) \leq 2a(2|\langle \mathbf{w}, \mathbf{x}_i\rangle - \langle \mathbf{w}^*, \mathbf{x}_i\rangle|) \leq 2a(4W) = 8aW$

The above proof uses arguments similar to the proof for the idealized GLM (please see the lines after the proof of Theorem 3.4, viz. the b-Lipschitzness of the generalized ReLU, and the problem setup).

Now, when

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

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