2018 17th IEEE International Conference on Machine Learning and Applications

Improving L-BFGS Initialization For Trust-Region Methods In Deep Learning

Jacob Rafati
Electrical Engineering and Computer Science
University of California, Merced
Merced, CA 95340 USA
jrafatiheravi@ucmerced.edu

Roummel F. Marcia
Applied Mathematics
University of California, Merced
Merced, CA 95340 USA
rmarcia@ucmerced.edu

***Abstract*—Deep learning algorithms often require solving a highly non-linear and nonconvex unconstrained optimization problem. Generally, methods for solving the optimization problems in machine learning and in deep learning specifically are restricted to the class of first-order algorithms, like stochastic gradient descent (SGD). The major drawback of the SGD methods is that they have the undesirable effect of not escaping saddle-points. Furthermore, these methods require exhaustive trial-and-error to fine-tune many learning parameters. Using the second-order curvature information to find the search direction can help with more robust convergence for the non-convex optimization problem. However, computing the Hessian matrix for the large-scale problems is not computationally practical. Alternatively, quasi-Newton methods construct an approximate of Hessian matrix to build a quadratic model of the objective function. Quasi-Newton methods, like SGD, require only first-order gradient information, but they can result in superlinear convergence, which makes them attractive alternatives. The limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) approach is one of the most popular quasi-Newton methods that construct positive-definite Hessian approximations. Since the true Hessian matrix is not necessarily positive definite, an extra initialization condition is required to be introduced when constructing the L-BFGS matrices to avoid false negative curvature information. In this paper, we propose various choices for initialization methods of the L-BFGS matrices within a trust-region framework. We provide empirical results on the classification task of the MNIST digits dataset to compare the performance of the trust-region algorithm with different L-BFGS initialization methods.**

***Index Terms*—Quasi-Newton Methods, L-BFGS, Trust-Region, Initialization, Deep Learning**

I. Introduction

Deep learning is becoming the leading technique for solving the large-scale machine learning problems, including image classification, natural language processing, and large-scale regression tasks [1]. Deep learning algorithms attempt to train a function approximation (*model*), usually a convolutional neural network (CNN), over a large dataset. In most of deep learning algorithms, solving an unconstrained optimization of a highly nonlinear and non-convex objective function of the form

$$\min_{w\in\mathbb{R}^n} \mathcal{L}(w) \triangleq \frac{1}{N}\sum_{i=1}^{N} \ell_i(w) \qquad (1)$$

is required [2], where $w \in \mathbb{R}^n$ is the vector of trainable parameters of the CNN model, N is the size of dataset, and $\ell_i(w)$ is the error of model's prediction for the ith observation of the *training* dataset.

A. Existing Methods

Finding an efficient optimization algorithm for the large-scale, non-convex problem (1) has attracted many researchers [1]. There are various algorithms proposed in machine learning and optimization literature to solve (1), among those one can name first-order methods such as stochastic gradient descent (SGD) methods [3]–[6], and the quasi-Newton methods [7]–[10], and also Hessian-free methods [11]–[14].

Since, in large-scale machine learning problems usually N and n are very large numbers, the computation of the true gradient $\nabla\mathcal{L}(w)$ is expensive and the computation of the true Hessian $\nabla^2\mathcal{L}(w)$ is not practical. Hence, most of the optimization algorithms in machine learning and deep learning literature are restricted to the variant of first-order gradient descent methods such as SGD methods. SGD methods use a small random sample of data ($\mathcal{S}$) to compute an approximate of the gradient of the objective function, $\nabla\mathcal{L}^{(\mathcal{S})}(w) \approx \nabla\mathcal{L}(w)$. At each iteration of the learning update, the parameters are updated as $w_{k+1} \leftarrow w_k - \eta_k \nabla\mathcal{L}^{(\mathcal{S}_k)}(w_k)$, where η_k is referred to as the *learning rate*.

The computational cost-per-iteration of SGD algorithm is small, making them the most widely used optimization method for vast majority of the deep learning applications. However, these methods require fine-tuning of many hyperparameters, including the learning rates. The learning rates are usually chosen to be very small; therefore, the SGD algorithms require revisiting many epochs of data during the learning process. Indeed, it is unlikely that the SGD methods perform successfully in the first trial, though there are recent works that address tuning these hyperparameters automatically (see e.g., [15], [16]).

Another major drawback of the SGD methods is that they struggle with saddle-points that occur in most of the non-convex optimization problem and has the undesirable effect on the model's generalization of learning. In the other hand, using the second-order curvature information, can help with more robust convergence for the non-convex optimization

This research is supported by National Science Foundation Grants CMMI 1333326, IIS 1741490 and ACI 1429783.

978-1-5386-6805-4/18/$31.00 ©2018 IEEE
DOI 10.1109/ICMLA.2018.00081

problem. The second-order methods like Newton method use the Hessian $\nabla^2\mathcal{L}(w)$ and the gradient to find the search direction, $p_k = -\nabla^2\mathcal{L}(w_k)^{-1}\nabla\mathcal{L}(w_k)$ and then use line-search method to find the step length along the search direction. The main bottleneck in the second-order methods is the serious computational challenges involved in computation of the Hessian $\nabla^2\mathcal{L}(w)$ for large-scale , which is not practical when n is large. The quasi-Newton methods and Hessian-free methods are both using approaches to approximate the Hessian matrix without computing and storing the true Hessian matrix $\nabla^2\mathcal{L}(w)$. Hessian-free methods attempts to find an approximate Newton direction by solving $\nabla^2\mathcal{L}(w_k)p_k = -\nabla\mathcal{L}(w_k)$ without forming the Hessian using the conjugate-gradient methods [11]–[14].

Quasi-Newton methods form an alternative class of first-order methods for solving the large-scale non-convex optimization problem in deep learning. These methods, like SGD, require only computing the first-order gradient of the objective function. By measuring and storing the difference between consecutive gradients, quasi-Newton methods construct *quasi-Newton matrices* $\{B_k\}$ which are low-rank updates to the previous Hessian approximations for estimating $\nabla^2\mathcal{L}(w_k)$ at each iteration. They build a quadratic model of the objective function by using these quasi-Newton matrices and use that model to find a sequence of search directions that can result in superlinear convergence. Since these methods do not require the second-order derivatives, they are more efficient than Newton's method for large-scale optimization problems [17].

There are various quasi-Newton methods proposed in literature. They differ in how they define and construct the quasi-Newton matrices $\{B_k\}$, how the search directions are computed, and how the parameters of the model are updated. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [18]–[21] is considered the most popular quasi-Newton algorithm, which produces positive semidefinite matrix B_k for each iteration. The conventional BFGS minimization employs *line-search*, which first attempts to find the search directions by computing $p_k = -B_k^{-1}\nabla\mathcal{L}(w_k)$ and then decides on the step size $\alpha_k \in (0, 1]$ based on sufficient decrease and curvature conditions [17] for each iteration k and then update the parameters $w_{k+1} = w_k + \alpha_k p_k$. The line-search algorithm first tries the unit step length $\alpha_k = 1$ and if it does not satisfy recursively reduce α_k. There are computational cost regarding the satisfaction of the sufficient decrease and curvature conditions and finding α_k using line-search methods. Also if the curvature condition does not satisfy for $\alpha_k \in (0, 1]$, the BFGS matrix may not stay positive definite and the update will become unstable. On the other hand if the search direction is rejected in order to preserve the positive definiteness of L-BFGS matrices, the progress of learning might stop or become very slow. Finally, solving $B_k p_k = -\nabla\mathcal{L}(w_k)$ can become computationally expensive when B_k becomes a high-rank update.

The *Limited-memory* BFGS (L-BFGS) method constructs a sequence of low-rank updates to the Hessian approximation and consequently solving $p_k = B_k^{-1}\nabla\mathcal{L}(w_k)$ can be done efficiently. Recently, an L-BFGS quasi-Newton method based on *trust-region* methods have been implemented and employed for the classification task in the deep learning framework [22]. Trust-region methods attempt to find the search direction in a region that they trust the accuracy of the quadratic model of the objective function. These methods not only have the benefit of being independent from the fine-tuning of hyperaparameters related to SGD learning methods, but they also improve upon the training performances and the convergence robustness compared to the line-search methods. Furthermore, trust-region L-BFGS methods can easily reject the search directions if the curvature condition does not satisfy to preserve the positive definiteness of the L-BFGS matrices. Based on the distinguishing characteristics of trust-region algorithms, unlike line-search methods, the progress of the learning will not stop or slow down due to the occasional rejection of the undesired search directions.

B. Motivation For This Research

In order to construct the quasi-Newton matrices at each iteration k, it is required to start with an initial matrix B_0 that is often set to some multiple $B_0 = \gamma_k I$ of the identity [17]. Then once B_0 is given, the L-BFGS matrices B_k can be constructed using the L-BFGS compact representation formula [7], [23]. The choice of the initial quasi-Newton matrix B_0 is crucial because it has a direct impact on the quality of the approximation of the Hessian [9], [24] and the quality of the robustness of L-BFGS convergence. L-BFGS matrices are attempting the hard task of approximating the indefinite Hessian matrix with positive definite matrices, which might result in false negative curvature information. This motivates some researchers to prefer indefinite quasi-Newton matrices such as *Limited-memory Symmetric Rank One* (L-SR1) update over the L-BFGS. However, the L-SR1 methods, unlike L-BFGS, do not guarantee a descent direction. We hypothesize that by introducing an extra condition for safe-guarding γ_k, the false negative curvature information can be avoided to some degree when approximating the Hessian matrix in an L-BFGS framework. We note that this work builds upon the results in [9] for defining γ_k.

C. Our Objective

In this paper we discuss the choices for initializing L-BFGS matrices to obtain parameter γ_k that result in better training performance and generalization of learning, without introducing significant computational cost. We define extra conditions that requires solving a general eigenvalue problem of form $A^*z = \lambda B^*z$, where A^* and B^* are obtained from the compact representation of the L-BFGS matrix. Consequently, solving this general eigenvalue problem does not add significant computational cost. We test our hypothesis on a supervised learning problem, namely the classification task of MNIST handwritten digits dataset, in the trust-region L-BFGS framework.

D. Outline

In Section II, we review the L-BFGS trust-region optimization method. First, we introduce the trust-region method and its properties. Second, we introduce the compact representation of the L-BFGS quasi-Newton matrices. Third, we introduce an efficient high accuracy method for solving the trust-region subproblem based on optimality conditions in order to define the search direction. In Section III, we examine three methods for initializing the L-BFGS matrices. In Section IV, we describe our numerical experiments on the classification task of the MNIST digits dataset, using the L-BFGS trust-region optimization method with our proposed initialization methods. In Section V, we present the results of the numerical experimentation and compare the effect of the different initialization methods on the training performance of the L-BFGS trust-region algorithm. In Section VI, we provide the concluding remarks.

II. Methodology

In this section, we give a summary of the trust-region method to solve the unconstrained optimization problem (1), where $\mathcal{L}(w)$ is a continuously differentiable function.

A. Trust-Region Methods

Trust-region methods generate a sequence of iterates $w_{k+1} = w_k + p_k$, where each search step p_k is obtained from solving the following trust-region subproblem:

$$\begin{aligned} p_k \; &= \underset{p\in\mathbb{R}^n}{\arg\min} \quad \mathcal{Q}_k(p) \triangleq g_k^T p + \frac{1}{2} p^T B_k p^T \\ &\text{subject to} \;\; \|p\|_2 \leq \delta_k \end{aligned} \tag{2}$$

where $g_k \triangleq \nabla\mathcal{L}(w_k)$ is the gradient of the objective function, B_k is an approximation to the Hessian $\nabla^2\mathcal{L}(w_k)$, and $\delta_k > 0$ is the trust-region radius. The global solution to the trust-region subproblem (2) can be characterized by the optimality conditions given in the following theorem due to Gay [25] and Moré and Sorensen [26]:

Theorem 1: Let δ_k be a positive constant. A vector p^* is a global solution of the trust-region subproblem (2) if and only if $\|p^*\|_2 \leq \delta_k$ and there exists a unique $\sigma^* \geq 0$ such that $B + \sigma^* I$ is positive semidefinite and

$$(B + \sigma^* I)p^* = -g \quad \text{and} \quad \sigma^*(\delta - \|p^*\|_2) = 0. \tag{3}$$

Moreover, if $B + \sigma^* I$ is positive definite, then the global minimizer is unique.

The computational bottleneck of trust-region methods is the solution of the trust-region subproblem (2). However, recent work (see e.g., [27], [28]) has shown that (2) can be efficiently solved if the Hessian approximation B_k is chosen to be a quasi-Newton matrix, which we describe next. (For further details on trust-region methods, see [29].)

B. Quasi-Newton Methods

For large-scale problems, computing the Hessian $\nabla^2\mathcal{L}(w_k)$ and using it as B_k in the quadratic model in (2) is not practical because of the memory and computational requirements . In contrast, quasi-Newton methods, like gradient descent methods, require only the computation of first-derivative information. They can construct a model of objective function by measuring the changes in the consecutive gradients for estimating the Hessian that can produce a super-linear convergence rate. The most well-known quasi-Newton method is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update, given by

$$B_{k+1} = B_k - \frac{1}{s_k^T B_k s_k} B_k s_k s_k^T B_k + \frac{1}{y_k^T s_k} y_k y_k^T, \tag{4}$$

where

$$s_k = w_{k+1} - w_k, \quad \text{and} \quad y_k = \nabla\mathcal{L}(w_{k+1}) - \nabla\mathcal{L}(w_k).$$

The matrix B_{k+1} is guaranteed to be positive definite if B_k is positive definite and the curvature condition $s_k^T y_k > 0$ is satisfied.

Using this update alone is not efficient for unconstrained optimization problem since it requires solving $B_k p_k = -\nabla\mathcal{L}(w_k)$ for each iteration to find the search step p_k which can be expensive when the rank of B_k is high.

C. Limited-Memory BFGS

For large-scale optimization problems, the *limited-memory BFGS* (L-BFGS) is more efficient (see [30]). In practice, only a limited collection of recent $\{(s_j, y_j)\}$ pairs is stored in memory, say m, where $m \ll n$ (usually $m < 100$). The recently computed pairs $\{(s_j, y_j)\}$ are stored in matrices S_k and Y_k as

$$S_k \triangleq [s_{k-m} \quad \ldots \quad s_{k-1}] \quad \text{and} \quad Y_k \triangleq [y_{k-m} \quad \ldots \quad y_{k-1}].$$

At each iteration, the L-BFGS matrix B_k is then computed recursively using the BFGS rank-2 update rule in (4) with some initial $B_0 = \gamma_k I$. In Section III, we will propose methods to find γ_k.

D. Compact Representation of L-BFGS matrices

Since the BFGS updates are low rank, the L-BFGS matrix B_k can be represented in the compact form

$$B_k = B_0 + \Psi_k M_k \Psi_k^T, \tag{5}$$

where Ψ_k and M_k are defined as

$$\Psi_k = \begin{bmatrix} B_0 S_k & Y_k \end{bmatrix}, \quad M_k = \begin{bmatrix} -S_k^T B_0 S_k & -L_k \\ -L_k^T & D_k \end{bmatrix}^{-1}, \tag{6}$$

and L_k is strictly lower triangular part and D_k is the diagonal part of the matrix $S_k^T Y_k$, i.e.,

$$S_k^T Y_k = L_k + D_k + U_k, \tag{7}$$

where U_k is a strictly upper triangular matrix. (See [23] for further details.)

E. Trust-Region Subproblem Solution

To efficiently solve the trust-region subproblem (2), we exploit the compact representation of the L-BFGS matrix to obtain a global solution based on optimality conditions (3). In particular, we compute the spectral decomposition of B_k using the compact representation of B_k. First, we obtain the QR factorization of $\Psi_k = Q_k R_k$, where Q_k has orthonormal columns and R_k is strictly upper triangular. Then we compute the eigendecomposition of $R_k M_k R_k^T = V_k \hat{\Lambda}_k V_k^T$, so that

$$B_k = B_0 + \Psi_k M_k \Psi_k^T = \gamma_k I + Q_k V_k \hat{\Lambda}_k V_k^T Q_k^T. \tag{8}$$

Note that since V_k is an orthogonal matrix, the matrix $Q_k V_k$ has orthonormal columns. Let $P = [\ Q_k V_k \quad (Q_k V_k)^{\perp}\ ] \in \mathbb{R}^{n \times n}$, where $(Q_k V_k)^{\perp}$ is a matrix whose columns form an orthonormal basis for the orthogonal complement of the range space of $Q_k V_k$, thereby making P an orthonormal matrix. Then

$$B_k = P \begin{bmatrix} \hat{\Lambda} + \gamma_k I & 0 \\ 0 & \gamma_k I \end{bmatrix} P^T. \tag{9}$$

Using this eigendecomposition to change variables and diagonalize the first optimality condition in (3), a closed form expression for the solution p_k^* can be derived.

The general solution for trust-region subproblem using the Sherman-Morrison-Woodbury formula is given by

$$p_k^* = -\frac{1}{\tau^*}\left[I - \Psi_k(\tau^* M_k^{-1} + \Psi_k^T \Psi_k)^{-1} \Psi_k^T\right] g_k, \tag{10}$$

where $\tau^* = \gamma_k + \sigma^*$, and σ^* is optimal Lagrange multiplier in (3) (see [31] for details). The L-BFGS trust-region method is described in Algorithm 1.

III. Proposed Quasi-Newton Matrix Initializations

The most common choice for initializing quasi-Newton methods is a scalar multiple of the identity matrix, i.e., $B_0 = \gamma_k I$ for $\gamma_k > 0$. In this section, we examine three choices for the scalar parameter γ_k. In particular, we label these choices as **Method I**, **Method II**, and **Method III**.

A. Initialization Method I

A conventional method to choose γ_k for L-BFGS is

$$\gamma_k = \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T y_{k-1}}. \tag{11}$$

This choice is proposed for optimal conditioning, and it can be viewed as a spectral estimate for Hessian $\nabla^2 \mathcal{L}(w_k)$. The parameter γ_k is the minimizer of the optimization problem

$$\gamma_k = \arg\min_{\gamma} \|B_0^{-1} y_{k-1} - s_{k-1}\|_2^2, \tag{12}$$

where $B_0^{-1} = \gamma^{-1} I$. We put a lower bound on $\gamma_k = \max(\varepsilon, \gamma_k)$, where $\varepsilon > 0$, to avoid producing sequences of nearly singular quasi-Newton matrices [32]. In our experiments, we used $\varepsilon = 1.0$.

Algorithm 1 Limited-memory BFGS trust-region method.

```
Input: starting point $w_0$, tolerance $\epsilon > 0$, $\delta_0$, $\eta < \frac{1}{4}$
Choose initialization Method I, Method II, or Method III
  for $k = 0, 1, 2, \ldots$ do
      Compute $g_k = \nabla \mathcal{L}(w_k)$
      Compute $\gamma_k$ to initialize $B_0 = \gamma_k I$
      $\gamma_k \leftarrow \max\{\gamma_k, 1\}$
      Compute $\Psi_k$ and $M_k$ from (6)
      Form $B_k$ orthonormal matrices in (9)
      Compute search step $p_k$ by solving TR subproblem (2)
      Compute $s_k = p_k$ and $y_k = \nabla \mathcal{L}(w_k + p_k) - \nabla \mathcal{L}(w_k)$
      if $s_k^T y_k > 0$ then
          Store $\{s_k, y_k\}$ in storage $S_{k+1}$ and $Y_{k+1}$
          Discard $\{s_{k-m}, y_{k-m}\}$ from storage if $k > m$
      end if
      $\rho_k \leftarrow (\mathcal{L}(w_k) - \mathcal{L}(w_k + p_k))/(\mathcal{Q}_k(0) - \mathcal{Q}_k(p_k))$
      if $\rho_k > \eta$ then
          $w_{k+1} = w_k + p_k$
      else
          $w_{k+1} = w_k$
      end if
      Update trust-region radius $\delta_{k+1}$
      if $\|g\|_2 < \epsilon$ or $k$ reached to maximum episodes then
          break
      end if
  end for
```

B. Initialization Method II

The second method for finding γ_k for initialization of $B_0 = \gamma_k I$ requires solving a general eigenvalue problem. This method is inspired by [9] where γ_k is chosen in a way that avoids the false curvature information for limited-memory Symmetric Rank-1 (L-SR1) trust-region method. We summarize the method described in [9] below.

Consider a quadratic objective function of the form $\mathcal{L}(w) = \frac{1}{2} w^T H w + g^T w$, where $H \in \mathbb{R}^{n \times n}$ is symmetric and $w, g \in \mathbb{R}^n$. The true Hessian $\nabla^2 \mathcal{L}(w)$ is equal to matrix H. Note that for this quadratic function, $\nabla \mathcal{L}(w_{k+1}) - \nabla \mathcal{L}(w_k) = H w_{k+1} + g - (H w_k - g) = H(w_{k+1} - w_k)$. Equivalently, $y_k = H s_k$ for all k. Therefore, $H S_k = Y_k$ and consequently, $S_k^T H S_k = S_k^T Y_k$. Using the compact representation of B_k in (5) with $B_0 = \gamma_k I$ for this quadratic function, we obtain

$$S_k^T H S_k - \gamma_k S_k^T S_k = S_k^T \Psi_k M_k \Psi_k^T S_k. \tag{13}$$

Note that if H is not positive definite, then $S_k^T \Psi_k M_k \Psi_k^T S_k$ may not be positive definite either. Therefore, by choosing $\gamma_k > 0$, negative curvature information of H can be captured by $S_k^T \Psi_k M_k \Psi_k^T S_k$. If H is positive definite and γ_k is chosen too big, then false negative curvature information can be produced. To avoid this undesired outcome, we choose $\gamma_k \in (0, \lambda_{\min})$ where $\lambda_{\min}$ is the minimum eigenvalue of the following generalized eigenvalue problem:

$$(L_k + D_k + L_k^T) z = \lambda S_k^T S_k z, \tag{14}$$

TABLE I
SUMMARY OF THE PROPOSED L-BFGS INITIALIZATION METHODS

Initialization	Source	Formula
Method I	Solve the optimization problem (12): $\gamma_k = \arg\min_\gamma \|B_0^{-1} y_{k-1} - s_{k-1}\|_2^2$	$\gamma_k = \max\left\{1, \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T y_{k-1}}\right\}$
Method II	Solve the generalized eigenvalue problem (14): $(L_k + D_k + L_k^T)z = \lambda S_k^T S_k z$	$\gamma_k = \begin{cases} \max\{1, 0.9\lambda_{\min}\} & \text{if } \lambda_{\min} > 0, \\ \text{Use Method I} & \text{if } \lambda_{\min} \leq 0. \end{cases}$
Method III	Solve the generalized eigenvalue problem (17): $A^* z = B^* \lambda z$	$\gamma_k = \begin{cases} \max\{1, 0.9\lambda_{\min}\} & \text{if } \lambda_{\min} > 0, \\ \text{Use Method I} & \text{if } \lambda_{\min} \leq 0. \end{cases}$

where L_k and D_k are defined in Section II. If the smallest eigenvalue in (14) is negative, i.e., $\lambda_{\min} < 0$, we choose γ_k from Method I instead.

C. *Initialization Method III*

We note that in (13), the right-hand side also depends on γ_k because the matrices Ψ_k amd M_k depend on γ_k (see (6)). Yet the generalized eigenvalue value problem (14) for determining bounds on γ_k does not take this into account. In Method III, we attempt to derive the generalized eigenvalue problem with considering the dependency of matrices M_k and Ψ_k on γ_k as defined in (6).

First, we compute the inverse of M_k explicitly using the following block partitioning:

$$M_k = \begin{bmatrix} -\gamma S_k^T S_k & -L_k \\ -L_k^T & D_k \end{bmatrix}^{-1} = \begin{bmatrix} \tilde{A} & \tilde{B} \\ \tilde{B}^T & \tilde{D} \end{bmatrix} \qquad (15)$$

where $\tilde{A}$, $\tilde{B}$, and $\tilde{D}$ are computed as follows:

$$\begin{aligned} \tilde{A} &= -(\gamma S_k^T S_k + L_k D_k^{-1} L_k^T)^{-1} \\ \tilde{B} &= -(\gamma S_k^T S_k + L_k D_k^{-1} L_k^T)^{-1} L_k D_k^{-1} \\ \tilde{D} &= D_k^{-1} - D_k^{-1} L_k^T (\gamma S_k^T S_k + L_k D_k^{-1} L_k^T) L_k D_k^{-1}. \end{aligned}$$

By substituting M_k from (15) and $\Psi_k = [\gamma S_k \; Y_k]$ into (13), we have

$$\begin{aligned} S_k^T H S_k = \gamma S_k^T S_k + S_k^T Y_k \tilde{D} Y_k^T S_k + \gamma^2 \big(S_k^T S_k \tilde{A} S_k^T S_k\big) \\ + \gamma \big(S_k^T S_k \tilde{B} Y_k^T S_k^T + S_k^T Y_k \tilde{B}^T S_k^T S_k\big). \qquad (16) \end{aligned}$$

The last two terms in (16) depend nonlinearly on γ. To find a linear condition for safe-guarding γ_k, we compute these nonlinear terms in (16) using the γ parameter from the previous iteration, i.e. γ_{k-1} (with initial value of $\gamma_0 = 1$). Then, to find an upper bound for γ_k, we solve the following generalized eigenvalue problem:

$$A^* z = \lambda B^* z, \qquad (17)$$

where A^* and B^* is defined as

$$\begin{aligned} A^* &= L_k + D_k + L_k^T - S_k^T Y_k \tilde{D} Y_k^T S_k - \gamma_{k-1}^2 (S_k^T S_k \tilde{A} S_k^T S_k), \\ B^* &= S_k^T S_k + S_k^T S_k \tilde{B} Y_k^T S_k^T + S_k^T Y_k \tilde{B}^T S_k^T S_k. \end{aligned}$$

As in Method II, if $\lambda_{\min} < 0$ in (17), we choose γ_k from Method I.

IV. NUMERICAL EXPERIMENTS

In this section, we test the trust-region L-BFGS optimization algorithm on the image classification task of the MNIST dataset with the three different initialization methods for B_0 discussed in this paper. All simulations were performed on a cluster with 4 NVIDIA Tesla K20m GPU, 256 GB memory, and 20 virtual Intel 1.2 GHz processors.

A. *Supervised Learning Application: Classification Task*

All methods are implemented to train the LeNet-5 convolutional neural network for the image classification task of the MNIST dataset. The MNIST dataset consists of $70,000$ examples of handwritten image of digits 0 to 9, with $N = 60,000$ image training set $\{(x_i, y_i)\}$, and $10,000$ used as the test set. Each image x_i is 28×28 pixel, and each pixel value is between 0 and 255. Each image x_i in training set include a label $y_i \in \{0, \ldots, 9\}$ describing its class. The objective function for the classification task in (1) uses the cross entropy between model prediction and true labels given by

$$\ell_i(w) = -\sum_{j=1}^{J} y_{ij} \log(p_i),$$

where the $p_i(x_i; w) = p_i(y = y_i | x_i; w)$ is the probability distribution of the model, i.e., the likelihood that the image is correctly classified, J is the number of classes ($J = 10$ for MNIST digits dataset) and $y_{ij} = 1$ if $j = y_i$ and $y_{ij} = 0$ if $j \neq y_i$ (see [2] for details).

B. *Convolutional Neural Network Architecture*

We use the convolutional neural network architecture, LeNet-5 for computing the likelihood $p_i(y_i | x_i; w_i)$. The LeNet-5 CNN is mainly used in literature for character and digit recognition tasks [33]. This architecture is given in Table II. The input to the network is 28×28 image and the output is 10 neurons followed by a softmax function attempts to approximate the likelihood probability distribution $p(y_i | x_i; w)$. There are total of $n = 431,080$ trainable parameters (weights) in LeNet-5 CCN.

C. *Computing Gradients*

Computing the gradient $\nabla \mathcal{L}(w)$ can be expensive when the size of data is large. In addition, some of the data points are

TABLE II
LeNet-5 CNN architecture [33].

Layer	Connections
0: input	28×28 image
1	convolutional, 20 5×5 filters (stride = 1), followed by ReLU
2	max pooling, 2×2 window (stride = 2)
3	convolutional, 50 5×5 filters (stride = 1), followed by ReLU
4	max pool, 2×2 window (stride = 2)
5	fully connected, 500 neurons (no dropout) followed by ReLU
6: output	fully connected, 10 neurons followed by softmax (no dropout)

similar, and consequently, usually a smaller random sample $\mathcal{S}$ can be used to estimate the loss and the gradients

$$\mathcal{L}(W) \approx \mathcal{L}^{(\mathcal{S})}(w) = \frac{1}{|\mathcal{S}|}\sum_{i \in \mathcal{S}} \ell_i(w),$$
$$\nabla \mathcal{L}(w) \approx \nabla \mathcal{L}^{(\mathcal{S})}(w) = \frac{1}{|\mathcal{S}|}\sum_{i \in \mathcal{S}} \nabla \ell_i(w)$$

where, $\mathcal{S}$ is a random subset of indices from $\{1, 2, \ldots, N\}$.

D. Multi-batch Sampling

The quality of the gradients directly impacts the quality of the search step and also in approximating the Hessian matrix. We perform our experiments using different data-to-sample ratio $N/|\mathcal{S}| \in$ {1, 2.5, 5, 12.5, 25, 50, 100, 250, 500, 1000}. In particular, the smaller $N/|\mathcal{S}|$ becomes, the larger batch size $|\mathcal{S}|$ becomes. In all simulations, we use the sample that have overlap between consecutive samples $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$. For iteration k, we use $\mathcal{S}_k$ to compute the gradient $g_k = \nabla \mathcal{L}^{(\mathcal{S}_k)}(w_k)$.

E. Computing y_k

Inspired by [34], we use the overlap between the consecutive multi-batch samples $\mathcal{O}_k = \mathcal{S}_k \cap \mathcal{S}_{k+1}$ to compute y_k as

$$y_k = \nabla \mathcal{L}^{(\mathcal{O}_k)}(w_{k+1}) - \nabla \mathcal{L}^{(\mathcal{O}_k)}(w_k). \tag{18}$$

The use of overlap to compute y_k has been shown to result in more robust convergence in L-BFGS since L-BFGS uses gradient differences to update the Hessian approximations (see [9], [34]).

F. Other parameters

We performed the experiments for two choices of the quasi-Newton memory storage $m \in \{10, 20\}$. All simulations stopped after 300 iterations or if the gradient satisfied $\|g_k\|_2 < \epsilon = 10^{-5}$.

V. Results and Discussions

The results of the training the trust-region L-BFGS algorithm with different initialization (**Method I**, **Method II**, and **Method III**), different multi-batch samples ($1 \leq N/|\mathcal{S}| \leq 1000$) and also different memory size m are depicted in Fig. 1. For each simulation, the training and test losses, the training and test accuracy, and the total time of simulation were stored. The minimum losses for both training and test sets for $m = 10$ are plotted in Fig. 1(a), for different sample sizes. Note that $N/|\mathcal{S}|$ increases from left to right, meaning that the multi-batch sample size is becoming smaller. For larger sample sizes (e.g., for smaller values of $N/|\mathcal{S}|$), the initialization Method I, which is commonly used in the literature, performs the best. However, as the mini-batch sample size decreases (e.g., for $N/|\mathcal{S}| > 100$), Methods II and III outperforms Method I. For instance, the minimum training loss for Method II for $N/|\mathcal{S}| = 500$ is $\sim 287\%$ lower than the one for Method I, and the test loss for the same simulation is $\sim 120\%$ lower. For $N/|\mathcal{S}| = 1000$, the training loss is $\sim 286\%$ lower and the test loss is $\sim 79\%$ lower. The training loss for Method III is $\sim 131\%$ lower than the one for Method I for $N/|\mathcal{S}| = 500$, and it is $\sim 450\%$ lower for $N/|\mathcal{S}| = 1000$. The test loss is 58% lower for $N/|\mathcal{S}| = 500$ and $\sim 100\%$ lower for $N/|\mathcal{S}| = 1000$.

Similar phenomena can be observed for the training and test loss for $m = 20$, which is plotted in Fig. 1(b). The minimum training loss for simulations with initialization Method II is $\sim 846\%$ lower for $N/|\mathcal{S}| = 500$, and test loss is $\sim 178\%$ lower. The training loss with Method III is $\sim 279\%$ lower than Method I for $N/|\mathcal{S}| = 250$ and the test loss is $\sim 98\%$ lower. The training loss with Method III is $\sim 426\%$ lower for $N/|\mathcal{S}| = 500$ and the test loss is $\sim 125\%$ lower.

The training and test maximum accuracy for $m = 10$ is reported in Fig. 1(c), and we see similar improvements using Methods II and III for simulations with smaller multi-batch sample sizes ($N/|\mathcal{S}| \geq 100$). Our proposed initialization methods results improves the test accuracy of prediction form 94.3% using Method I to 97.4% using Method II and to 96.3% using Method III, when $N/|\mathcal{S}| = 500$. We saw similar behavior in the maximum train and test accuracy for $m = 20$ which is plotted in Fig. 1(d). The maximum test accuracy improved from 93% using Method I to 97.5% when we used our proposed Method II and we saw improvement to 96.8% when we used the proposed initialization Method III.

The total training time for $m = 10$ is reported in Fig. 1(e). There is only $\sim 1\%$ average increase in training time for simulations using Method II, and $\sim 10\%$ average increase in training time for simulations using Method III, in comparison to Method I. Similarly, for the training run time for simulations with larger storage memory $m = 20$ in Fig. 1(f), there is no significant difference between Method II and Method I, but Method III was about 10% slower than the Method I.

VI. Conclusions

In this paper, we investigated three methods for initializing L-BFGS matrices in a trust-region optimization framework. The L-BFGS quasi-Newton matrix attempts to approximate the curvature informations of the Hessian matrix $\nabla^2 \mathcal{L}(w_k)$ with a positive definite quasi-Newton matrix B_k. In each iteration k, an initial matrix B_0 is required, and the usual choice for B_0 is a non-negative scalar multiple of the identity matrix, i.e., $B_0 = \gamma_k I$ with $\gamma_k > 0$. The Hessian matrix $\nabla^2 \mathcal{L}(w_k)$ can be indefinite if $\mathcal{L}(w)$ is nonconvex, and a careless initialization of the quasi-Newton matrix can have the undesired effect of definiteness mismatch between the true Hessian and the

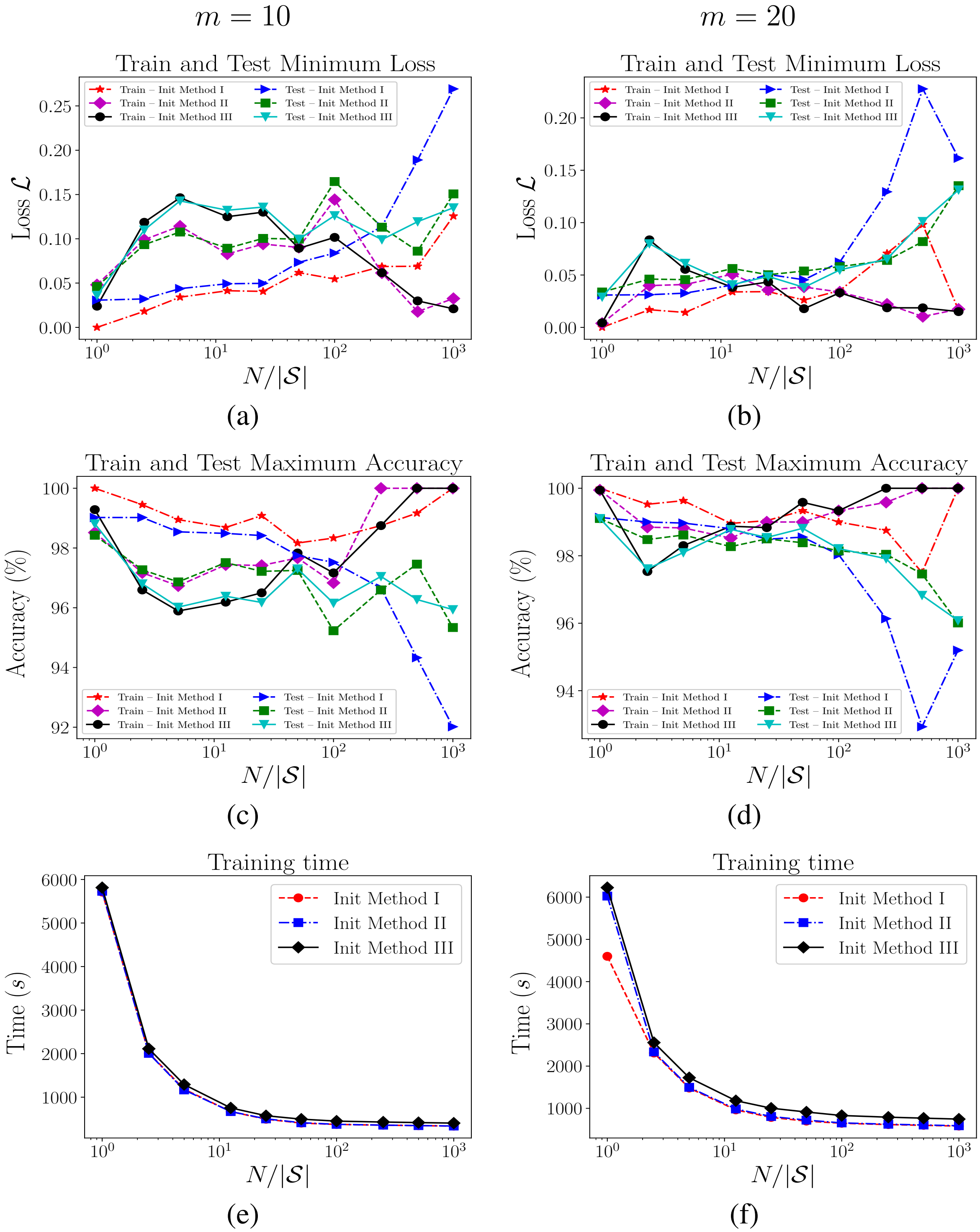

Fig. 1. A trust-region algorithm with different L-BFGS initialization methods is used for training LeNet-5 CNN to learn the task of classification of the MNIST digits set. The performance of learning is depicted for different sample batch sizes $\mathcal{S}$ and different memory storage $m = 10$ and $m = 20$. N is the size of data and $|\mathcal{S}|$ is the size of the sample batch. (a) Train and test minimum loss for $m = 10$. Note that as the batch size becomes smaller, i.e. $N/|\mathcal{S}|$ becomes larger, Methods II and III outperform Method I. (b) Train and test minimum loss for $m = 20$. Similar phenomena occur for most of the smaller batch sizes. (c) Train and test maximum accuracy for $m = 10$. As the batch sizes get smaller, the test accuracy for Method II and III are better than that for Method I. (d) Train and test maximum accuracy for $m = 20$. The test accuracy for Method II and III are better than that for Method I. (e) Training time for $m = 10$. As the batch sizes gets smaller, the training time is also decreases. (f) Training time for $m = 20$. There is no significant difference training time of Method I and Method II. Method III is about %10 slower.

quasi-Newton Hessian approximation. We investigated three methods for the initial matrix $B_0 = \gamma_k I$ (see Table I for a summary of the initialization methods). We also experimented on the effects of initialization on the performance of the training using the LeNet-5 CNN on the classification task of the MNIST handwritten digits dataset.

The initialization Method I given in (11), which is conventionally used in literature [17], [32], is simple to compute and usually is a great choice when the sample size is considerably large, i.e. ($|\mathcal{S}|/N \geq 1\%$ or $N/|\mathcal{S}| \leq 100$). However, once the sample sizes gets smaller, i.e. ($|\mathcal{S}|/N < 1\%$ or $N/|\mathcal{S}| > 100$), the performance of the training drops dramatically.

Our proposed initialization Method II (based on [9]) introduces a new condition on safeguarding γ_k by finding an upper bound which requires solving a low-rank general eigenvalue problem which does not add significant computational cost. For smaller sample sizes ($|\mathcal{S}|/N < 1\%$ or $N/|\mathcal{S}| > 100$), this initialization method outperformed the Method I in all training and testing minimum loss and maximum accuracy performances. However this method does not consider the fact that Ψ_k and M_k is a function of γ_k when constructing the general eigenvalue problem in (14).

Our proposed initialization Method III introduces a more sophisticated condition on safeguarding γ_k. The key difference between this method and Method II is that this method takes into account that Ψ_k and M_k are functions of γ_k when defining the generalized eigenvalue problem in (17). The computation of A^* and B^* adds about 10% to the computational cost. For smaller sample sizes i.e. ($|\mathcal{S}|/N < 1\%$ or $N/|\mathcal{S}| > 100$) this initialization method also outperformed the initialization Method I in all training and testing minimum loss and maximum accuracy performances. There was no significant difference between the performance of the training when using initialization Methods II and III.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press Cambridge, 2016. [Online]. Available: http://www.deeplearningbook.org

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2009. [Online]. Available: http://www-stat.stanford.edu/ tibs/ElemStatLearn/

[3] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[4] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[5] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[6] B. Recht, C. Ré, S. J. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, 2011, pp. 693–701.

[7] L. Adhikari, O. DeGuchy, J. B. Erway, S. Lockhart, and R. F. Marcia, "Limited-memory trust-region methods for sparse relaxation," in *Wavelets and Sparsity XVII*, vol. 10394. International Society for Optics and Photonics, 2017, p. 103940J.

[8] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 2011, pp. 265–272.

[9] J. B. Erway, J. Griffin, R. F. Marcia, and R. Omheni, "Trust-Region Algorithms for Training Responses: Machine Learning Methods Using Indefinite Hessian Approximations," *ArXiv e-prints*, 2018. [Online]. Available: https://arxiv.org/abs/1807.00251

[10] P. Xu, F. Roosta-Khorasan, and M. W. Mahoney, "Second-order optimization for non-convex machine learning: An empirical study," *ArXiv e-prints*, 2017. [Online]. Available: https://arxiv.org/abs/1708.07827

[11] J. Martens, "Deep learning via Hessian-free optimization," in *Proc. of the 27th Intl. Conf. on Machine Learning (ICML)*, 2010, pp. 735–742.

[12] J. Martens and I. Sutskever, "Learning recurrent neural networks with Hessian-free optimization," in *Proc. of the 28th Intl. Conf. on Machine Learning (ICML)*, 2011, pp. 1033–1040.

[13] ——, "Training deep and recurrent networks with Hessian-free optimization," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 479–535.

[14] R. Bollapragada, R. Byrd, and J. Nocedal, "Exact and Inexact Subsampled Newton Methods for Optimization," *ArXiv e-prints*, 2016. [Online]. Available: https://arxiv.org/abs/1609.08502

[15] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012. [Online]. Available: http://arxiv.org/abs/1212.5701

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York: Springer, 2006.

[18] C. G. Broyden, "The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations," *IMA Journal of Applied Mathematics*, vol. 6, no. 1, pp. 76–90, 1970.

[19] R. Fletcher, "A new approach to variable metric algorithms," *The Computer Journal*, vol. 13, no. 3, pp. 317–322, 1970.

[20] D. Goldfarb, "A family of variable-metric methods derived by variational means," *Mathematics of computation*, vol. 24, no. 109, pp. 23–26, 1970.

[21] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Mathematics of computation*, vol. 24, no. 111, pp. 647–656, 1970.

[22] J. Rafati, O. DeGuchy, and R. F. Marcia, "Trust-Region Minimization Algorithm for Training Responses (TRMinATR): The Rise of Machine Learning Techniques," in *26th European Signal Processing Conference, Rome, Italy*, September 2018, Accepted.

[23] R. H. Byrd, J. Nocedal, and R. B. Schnabel, "Representations of quasi-Newton matrices and their use in limited-memory methods," *Math. Program.*, vol. 63, pp. 129–156, 1994.

[24] J. L. Wah and Y. C. Chuei, "A class of diagonal preconditioners for limited memory BFGS method," *Optimization Methods and Software*, vol. 28, no. 2, pp. 379–392, 2013.

[25] D. M. Gay, "Computing optimal locally constrained steps," *SIAM J. Sci. Statist. Comput.*, vol. 2, no. 2, pp. 186–197, 1981.

[26] J. J. Moré and D. C. Sorensen, "Computing a trust region step," *SIAM J. Sci. and Statist. Comput.*, vol. 4, pp. 553–572, 1983.

[27] J. Brust, J. B. Erway, and R. F. Marcia, "On solving l-sr1 trust-region subproblems," *Computational Optimization and Applications*, vol. 66, no. 2, pp. 245–266, 2017.

[28] O. Burdakov, L. Gong, Y.-X. Yuan, and S. Zikrin, "On efficiently combining limited memory and trust-region techniques," *Mathematical Programming Computation*, vol. 9, pp. 101–134, 2016.

[29] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust-Region Methods*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2000.

[30] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, pp. 503–528, 1989.

[31] L. Adhikari, O. DeGuchy, J. B. Erway, S. Lockhart, and R. F. Marcia, "Limited-memory trust-region methods for sparse relaxation," in *Proc.SPIE*, vol. 10394, 2017, pp. 10 394 – 10 394 – 8. [Online]. Available: https://doi.org/10.1117/12.2271369

[32] J. Brust, O. Burdakov, J. B. Erway, and R. F. Marcia, "Dense Initializations for Limited-Memory Quasi-Newton Methods," *ArXiv e-prints*, 2017. [Online]. Available: https://arxiv.org/abs/1710.02396

[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[34] A. S. Berahas, J. Nocedal, and M. Takac, "A multi-batch L-BFGS method for machine learning," in *Advances in Neural Information Processing Systems 29*, 2016, pp. 1055–1063.