Metadata of the chapter that will be visualized in SpringerLink

Book Title	Intelligent Computing
Series Title	
Chapter Title	A Stochastic Modified Limited Memory BFGS for Training Deep Neural Networks
Copyright Year	2022
Copyright HolderName	The Author(s), under exclusive license to Springer Nature Switzerland AG

Author	Family Name	**Yousefi**
	Particle	
	Given Name	**Mahsa**
	Prefix	
	Suffix	
	Role	
	Division	Department of Mathematics and Geosciences
	Organization	University of Trieste
	Address	Trieste, Italy
	Email	mahsa.yousefi@phd.units.it
Corresponding Author	Family Name	**Calomardo**
	Particle	
	Given Name	**Ángeles Martínez**
	Prefix	
	Suffix	
	Role	
	Division	Department of Mathematics and Geosciences
	Organization	University of Trieste
	Address	Trieste, Italy
	Email	amartinez@units.it

Abstract

In this work, we study stochastic quasi-Newton methods for solving the non-linear and non-convex optimization problems arising in the training of deep neural networks. We consider the limited memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) update in the framework of a trust-region approach. We provide an almost comprehensive overview of recent improvements in quasi-Newton based training algorithms, such as accurate selection of the initial Hessian approximation, efficient solution of the trust-region subproblem with a direct method in high accuracy and an overlap sampling strategy to assure stable quasi-Newton updating by computing gradient differences based on this overlap. We provide a comparison of the standard L-BFGS method with a variant of this algorithm based on a modified secant condition which is theoretically shown to provide an increased order of accuracy in the approximation of the curvature of the Hessian. In our experiments, both quasi-Newton updates exhibit comparable performances. Our results show that with a fixed computational time budget the proposed quasi-Newton methods provide comparable or better testing accuracy than the state of the art first-order Adam optimizer.

Keywords (separated by '-')

Quasi-Newton methods - Limited memory BFGS - Trust region - Stochastic optimization - Deep neural networks

A Stochastic Modified Limited Memory BFGS for Training Deep Neural Networks

Mahsa Yousefi and Ángeles Martínez Calomardo(✉)

Department of Mathematics and Geosciences, University of Trieste, Trieste, Italy
mahsa.yousefi@phd.units.it, amartinez@units.it

Abstract. In this work, we study stochastic quasi-Newton methods for solving the non-linear and non-convex optimization problems arising in the training of deep neural networks. We consider the limited memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) update in the framework of a trust-region approach. We provide an almost comprehensive overview of recent improvements in quasi-Newton based training algorithms, such as accurate selection of the initial Hessian approximation, efficient solution of the trust-region subproblem with a direct method in high accuracy and an overlap sampling strategy to assure stable quasi-Newton updating by computing gradient differences based on this overlap. We provide a comparison of the standard L-BFGS method with a variant of this algorithm based on a modified secant condition which is theoretically shown to provide an increased order of accuracy in the approximation of the curvature of the Hessian. In our experiments, both quasi-Newton updates exhibit comparable performances. Our results show that with a fixed computational time budget the proposed quasi-Newton methods provide comparable or better testing accuracy than the state of the art first-order Adam optimizer. AQ1 AQ2

Keywords: Quasi-Newton methods · Limited memory BFGS · Trust region · Stochastic optimization · Deep neural networks

1 Introduction

Deep learning has become the leading technique for solving large-scale machine learning problems. After a prolonged slow start, the advent of higher computational power and the introduction of GPU computing, have made possible the training of neural networks with a high number of layers that have shown impressive efficacy in image classification tasks, natural language processing and text analytic, speech recognition and reinforcement learning among other fields. Deep Learning problems are often posed as highly nonlinear and often non-convex unconstrained optimization problems. For instance, in image classification using a training dataset $\{(x_i, y_i)\}_{i=1}^{N}$ in C classes with input $x_i \in \mathbb{R}^d$ and target $y_i \in \mathbb{R}$, a deep neural network training refers to solving an empirical risk minimization (ERM) problem that can be formulated as follows:

$$\min_{w \in \mathbb{R}^n} F(w) := \frac{1}{N} \sum_{i=1}^{N} f_i(w) \tag{1}$$

where $w \in \mathbb{R}^n$ is the vector of trainable parameters, N is the number of observations in the training dataset and $f_i(w) := f(w; x_i, y_i)$ is a loss function quantifying the prediction error for the ith observation of the training dataset.

Finding an efficient optimization algorithm for (1) has attracted many researchers and a number of algorithms have been proposed both in the machine learning and optimization literature. Since in large-scale machine learning problems (i.e. large n and N) the computation of the loss function $F(w)$ and the gradient $\nabla F(w)$ is expensive and the computation of the true Hessian $\nabla^2 F(w)$ is not practical, stochastic first-order methods have been widely used in many DL applications due to their low per-iteration cost, optimal complexity, easy implementation and proven efficiency in practice. The preferred method is the stochastic gradient descent (SGD) method [6,37], and its variance-reduced [12,20,38] and adaptive [13,21] variants. However, these methods due to the use of only first-order gradient information come with several issues such as relatively-slow convergence, highly sensitivity to the choice of hyper-parameter (e.g., step-length and batch size). First-order methods can also find some difficulties in escaping saddle points [43], and exhibit limited benefits of parallelism due to their usual implementation with small mini-batches [24].

On the other hand, second order methods can often find good minima in fewer steps due to their use of curvature information. The main second order method incorporating the inverse Hessian matrix is Newton's method [34] that computes the next update step by $w_{k+1} = w_k - \eta \nabla^2 F(w_k)^{-1} \nabla F(w_k)$. However, Newton's method presents serious computational and memory usage challenges involved in the computation of the Hessian. Moreover, using exact Hessians will result in algorithms that produce sequences moving towards saddle points, as Newton's method encourages rapid local convergence towards any stationary point regardless of the curvature [11,23].

Quasi-Newton and Hessian-free methods are two techniques aimed at incorporating second order information without computing and storing the true Hessian matrix. Hessian-free methods attempt to find an approximate Newton direction $\nabla^2 F(w_k)^{-1} \nabla F(w_k)$ using conjugate gradient methods [4,28]. Nevertheless, whether true Hessian matrix-vector products or subsampled variants of them, see e.g. [42], are used, the iteration complexity of a (modified) CG algorithm is significantly greater than that of a limited memory quasi-newton method. In fact, quasi-Newton methods and their limited memory variants [34] attempt to combine the speed of Newton's method and the scalability of first-order methods. They construct Hessian approximations using only gradient information and exhibit superlinear convergence.

Quasi-Newton and stochastic quasi-Newton methods to solve large optimization problems arising in machine learning have been recently extensively considered within the context of convex and non-convex optimization. For instance, a stochastic Broyden-Fletcher-Goldfarb-Shanno and its limited memory variant (L-BFGS) were proposed for online convex optimization in [39]. Another stochastic L-BFGS method for solving strongly convex problems was proposed in [9] that uses sampled Hessian-vector products rather than gradient differences, which was proved in [33] to be linearly convergent by incorporating the variance reduction technique (SVRG [20]) to alleviate the effect of noisy gradients. A closely related variance reduced block L-BFGS method was proposed in [19]. A regularized stochastic BFGS method was proposed in [30], and an online L-BFGS method was proposed in [31] for strongly convex problems and extended in [27] to incorporate SVRG variance reduction. For the solution of non-convex optimization problems arising in deep learning, a damped L-BFGS method incorporating SVRG variance reduction was developed and its convergence properties were studied in [40]. Stochastic quasi-Newton methods use a subsampled Hessian approximation or/and subsampled gradient. Some of these stochastic quasi-Newton algorithms employ fixed size batches and compute stochastic gradient differences in a stable way, originally proposed in [39], using the same batch at the beginning and at the end of the iteration. Since this can potentially double the iteration complexity, an overlap batching strategy was proposed to reduce the computational cost in [2] and tested also in [3]. This strategy was further applied in [14,35]. Other stochastic quasi-Newton methods have been considered that employ a progressive batching approach in which the sample size is increased as the iteration progresses, see e.g. [5] and references therein. Recently, in [17] a Kronecker-factored block diagonal BFGS and L-BFGS method was proposed, that takes advantage of the structure of feed-forward DNN training problems.

The main contribution of this work is as follows. As most of the previously cited related works, we consider a limited memory variant of BFGS (L-BFGS), one of the most popular quasi-Newton updates in Broyden's class. We consider a stochastic variant of L-BFGS obtained by fixed-size subsampling. We study also a modified L-BFGS update obtained through a modified secant condition which is theoretically shown to provide an increased order of accuracy in the approximation of the curvature of the Hessian. Both the original and the modified L-BFGS quasi-Newton methods are used in a trust-region framework. We provide an almost comprehensive overview of recent improvements in quasi-Newton based training algorithms, such as accurate selection of the initial Hessian approximation, efficient solution of the trust-region subproblem with a direct method in high accuracy and an overlap sampling strategy to assure stable quasi-Newton updating by computing gradient differences based on this overlap. We examine the behaviour of the studied quasi-Newton methods in the training of deep convolutional neural networks in a supervised learning application, image classification, and provide a comparison with a state of the art first-order method such as Adam [21].

This paper is organized as follows. We provide an overview of the (limited memory) BFGS method in Sect. 2 x. In Sect. 3 we introduce a modified L-BFGS update obtained by imposing a different secant condition. We describe the use of the modified L-BFGS method in a trust-region framework and its stochastic variant in Sects. 4 and 5, respectively. Numerical result are reported in Sect. 6. Finally, some of the conclusions of this study are included in Sect. 7.

2 An Overview on the L-BFGS Update

2.1 The BFGS Update

The BFGS update as Hessian approximation have the following general form

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}, \quad k = 0, 1, \dots, \tag{2}$$

and satisfies the *standard secant condition*

$$B_{k+1} s_k = y_k, \tag{3}$$

where $s_k = p_k$ and $y_k = \nabla F(w_t) - \nabla F(w_k)$. The vector p_k is the search direction at iteration k and can be obtained in many different ways, for instance, using a trust-region framework [10] which proposes a trial point

$$w_t = w_k + p_k. \tag{4}$$

The BFGS updates (2) using only gradient information to incorporate curvature information generate symmetric positive definite matrices, i.e. $B_{k+1} \succ 0$, whenever the initial approximation $B_0 = \gamma_k I$ has the same property and the curvature condition $s_k^T y_k > 0$ holds. In this work, we skip updating B_k if the following curvature condition is not satisfied for some small value of $\epsilon_2 > 0$:

$$s_k^T y_k > \epsilon_2 \|s_k\|^2. \tag{5}$$

2.2 The L-BFGS Update and Its Compact Form

For large-scale optimization problems, the limited-memory BFGS (denoted by L-BFGS) would be more efficient. In fact, for $k \geq r$, the r most recent computed pairs are stored in the following matrices

$$S_k := \left[s_{k-r} \dots s_{k-1} \right], \qquad Y_k := \left[y_{k-r} \dots y_{k-1} \right]. \tag{6}$$

Using (6), the L-BFGS matrix B_k (2) can be represented in the following compact form [34]

$$B_k = B_0 + \Psi_k M_k \Psi_k^T, \quad k = 1, 2, \dots, \tag{7}$$

where $B_0 = \gamma_k I \succ 0$ and

$$\Psi_k = \left[B_0 S_k \; Y_k \right], \qquad M_k = \begin{bmatrix} -S_k^T B_0 S_k & -L_k \\ -L_k^T & D_k \end{bmatrix}^{-1}. \tag{8}$$

In (8), matrices L_k, U_k and D_k are respectively the strictly lower triangular part, the strictly upper triangular part and the diagonal part of the following matrix splitting

$$S_k^T Y_k = L_k + D_k + U_k. \tag{9}$$

2.3 The Initialization of the L-BFGS Update

The initial matrix B_0 is often set to some multiple of the identity matrix. A heuristic and conventional method to choose this multiple is

$$\gamma_k = \frac{y_{k-1}^T y_{k-1}}{y_{k-1}^T s_{k-1}} := \gamma_k^h. \tag{10}$$

The quotient of (10) is an approximation to an eigenvalue of $\nabla^2 F(w_k)$ and appears to be the most successful method, in practice, to generate initial Hessian approximations [34]. However, in a non-convex DL optimization, the choice of γ_k should be carefully operated to avoid the introduction of false negative curvature [14,35]. To this end, an extra condition can be imposed on γ_k to avoid $p_k^T B_k p_k < 0$ while $p_k^T \nabla^2(w_k) p_k > 0$. The hyper-parameter γ_k is selected in $(0, \hat{\lambda})$ where $\hat{\lambda}$ is the smallest eigenvalue of the following generalized eigenvalue problem

$$(L_k + D_k + L_k^T)u = \lambda S_k^T S_k u, \tag{11}$$

with L_k and D_k defined in (9). If $\hat{\lambda} \leq 0$, then γ_k can be set to γ_k^h.

3 A Modified L-BFGS Update

A modified BFGS update, and a consequently modified L-BFGS algorithm, can be proposed by rewriting (3) as a *modified secant condition*

$$B_{k+1} s_k = y_k^*, \tag{12}$$

where (s_k, y_k^*) gives better curvature information than (s_k, y_k) for updating B_{k+1}. Therefore, in a similar fashion as described in the previous section, a modified L-BFGS update can be constructed by using y_k^* in place of y_k.

Let $\psi_k = 2(F_k - F_{k+1}) + (g_k + g_{k+1})^T s_k$. In [41], the vector y_k^* was constructed as

$$y_k^* = y_k + \frac{\psi_k}{\|s_k\|^2} s_k. \tag{13}$$

Definition (13) together with (12) provides more accurate curvature information. In fact, it can be proved that

$$\begin{aligned} s_k^T(\nabla^2 F(w_{k+1})s_k - y_k^*) &= \frac{1}{3} s_k^T (T_{k+1} s_k) s_k + O(\|s_k\|^4), \\ s_k^T(\nabla^2 F(w_{k+1})s_k - y_k) &= \frac{1}{2} s_k^T (T_{k+1} s_k) s_k + O(\|s_k\|^4), \end{aligned} \tag{14}$$

where T_{k+1} is the tensor of the objective function F at w_{k+1} in the Taylor series expansion

$$F_k = F_{k+1} - g_{k+1}^T s_k + \frac{1}{2} s_k^T \nabla^2 F(w_{k+1}) s_k - \frac{1}{6} s_k^T \left(T_{k+1} s_k\right) s_k + O(\|s_k\|^4). \tag{15}$$

In [29], a simple modification of (13) was proposed as $y_k^* = y_k + \text{sign}(\psi_k) \frac{\psi_k}{\|s_k\|^2} s_k$ to handle the case $\psi_k < 0$. We show below that this modification does not provide any improvement.

3.1 Sign Correction

Considering the Eq. in (14) together yields

$$\psi_k = \frac{1}{6} s_k^T (T_{k+1} s_k) s_k + O(\|s_k\|^4). \tag{16}$$

Let $\psi_k < 0$. Therefore, we have $s_k^T y_k^* = s_k^T y_k - \psi_k$ which leads to derive

$$\begin{aligned} s_k^T \nabla^2 F(w_{k+1}) s_k - s_k^T y_k^* &= s_k^T \nabla^2 F(w_{k+1}) s_k - \left(s_k^T y_k + \psi_k\right) + 2\psi_k \\ &= \frac{2}{3} s_k^T \left(T_{k+1} s_k\right) s_k + O(\|s_k\|^4). \end{aligned} \tag{17}$$

Equation (17) shows that the dominant error is even worse than the one in (14). Therefore, we suggest to use y_k whenever $\psi_k < 0$; otherwise we can use y_k^*.

3.2 A New Modified Secant Condition

Writing the Taylor series expansion for g_k and premultiplying it by s_k^T lead to

$$s_k^T g_k = s_k^T g_{k+1} - s_k^T \nabla^2 F(w_{k+1}) s_k + \frac{1}{2} s_k^T \left(T_{k+1} s_k\right) s_k + O(\|s_k\|^4). \tag{18}$$

Combining Eq. (15) and (18) together yields that the third order term disappears and

$$\begin{aligned} s_k^T \nabla^2 F(w_{k+1}) s_k &= 6(F_k - F_{k+1}) + 3 s_k^T \left(g_{k+1} + g_k\right) + s_k^T y_k + O(\|s_k\|^4) \\ &= 3\psi_k + s_k^T y_k + O(\|s_k\|^4), \end{aligned} \tag{19}$$

which suggests the choice of

$$y_k^* = \frac{3\psi_k}{\|s_k\|^2} s_k + y_k. \tag{20}$$

Obviously, the new vector y_k^* in Eq. (20) provides better curvature approximation (the error is of order $O(|s_k|^4)$ instead of $O(|s_k|^3)$) than the one defined in equation (13).

4 The Modified L-BFGS Trust Region Method

Let L-BFGS-TR define the L-BFGS trust region method in which the current parameter vector w_k at iteration k is updated by a search direction obtained using a Hessian approximations B_k for which the standard secant condition (3) holds. In a similar fashion, we describe in this section the modified L-BFGS trust region method (M-LBFGS-TR) in which the Hessian approximations B_k satisfy the modified secant condition (12). In solving (1), both trust-region methods using a Hessian approximation satisfying either the standard (3) or the modified (12) secant condition, generate a sequence of iterates (4) in which p_k is obtained by solving the following trust-region subproblem:

$$p_k = \arg\min_{p\in\mathbb{R}^n} Q_k(p) := \frac{1}{2}p^T B_k p + g_k^T p \quad \text{s.t.} \quad \|p\|_2 \le \delta_k, \tag{21}$$

for some trust-region radius $\delta_k > 0$, where $g_k := \nabla F(w_k)$ and $B_k \approx \nabla^2 F(w_k)$.

The acceptance of the trial (4) is based on the ratio between the actual reduction in the objective function of (1) and the reduction predicted by the quadratic model, that is

$$\rho_k = \frac{F(w_k) - F(w_t)}{Q_k(0) - Q_k(p_k)}. \tag{22}$$

Since the denominator in (22) is nonnegative, if ρ_k is positive, the new iterate w_{k+1} will be computed as in (4) as $w_{k+1} := w_t$; otherwise, $w_{k+1} := w_k$. The process of adjustment of the trust-region radius at each iteration is described in Algorithm 2.

According to [7,8] the subproblem (21) can be efficiently solved if B_k is chosen to be a quasi-Newton matrix. Let B_k be a (modified) L-BFGS Hessian approximation in compact form (7). As described in [15,32], the global solution of (21) is characterized by the following theorem

Theorem 1. *Let δ be a given positive constant. A vector p^* is a global solution of the trust region problem (21) if and only if $\|p^*\|_2 \le \delta$ and there exists a unique $\sigma^* \ge 0$ such that $B_k + \sigma^* I$ is positive semi-definite with*

$$(B_k + \sigma^* I)p^* = -g_k, \qquad \sigma^*(\delta_k - \|p^*\|_2) = 0. \tag{23}$$

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

Following [1,7,35], the solution of the trust-region subproblem (21) can be computed as

$$p^* := p(\sigma^*) = -\frac{1}{\tau_k}\left(I - \Psi_k \left(\tau_k M_k^{-1} + \Psi_k^T \Psi_k\right)^{-1} \Psi_k^T\right) g_k. \tag{24}$$

where $\tau_k = \gamma_k + \sigma^*$. This direct formula can be obtained by exploiting the spectral decomposition of the coefficient matrix $B_k + \sigma^* I$ and its inversion using the Sherman-Morrison-Woodbury formula [34].

Algorithm 1 describes the process of solving the trust-region subproblem. It is based on the strategies described in the subsequent paragraphs. For further details see [1,7,35].

Algorithm 1. Trust-Region Subproblem Solution.

```
1: **Inputs:** Current $\Psi \triangleq \Psi_k$, $M^{-1} \triangleq M_k^{-1}$, $\gamma \triangleq \gamma_k$, $\delta \triangleq \delta_k$ and $g \triangleq g_k$
2: Compute the thin QR factorization of $\Psi$ with factors $Q$ and $R$
3: Compute the spectral decomposition of matrix $RMR^T = U\hat{\Lambda}U^T$
4: Set: the reordered matrix $\hat{\Lambda} = \text{diag}(\hat{\lambda}_1, \dots, \hat{\lambda}_k)$ such that $\hat{\lambda}_1 \leq \dots \leq \hat{\lambda}_k$
5: Compute the spectral decomposition of $B_k$ as $\Lambda_1 = \hat{\Lambda} + \gamma I$
6: Let: $\lambda_{min} = \min\{\lambda_1, \gamma\}$
7: Compute $P_\| = QU$
8: Compute $g_\| = P_\|^T g$
9: Compute $a_j = (g_\|)_j$ and $a_{k+1} = \sqrt{\|g\|_2^2 - \|g_\|\|_2^2}$
10: **if** $\phi(0) \geq 0$ **then**
11:     Set: $\sigma^* = 0$
12:     Compute $p^*$ with (26) as solution of $(B_k + \sigma^* I)p = -g$
13: **else**
14:     Compute a root $\sigma^* \in (0, \infty)$ of (28) by Newton's method
15:     Compute $p^*$ with (26) as solution of $(B_k + \sigma^* I)p = -g$
16: **end if**
```

Algorithm 2. Trust-Region Radius Adjustment

```
1: **Inputs:** Current iteration $k$, $\delta_k$, $\rho_k$, $0 < \tau_2 < 0.5 < \tau_3 < 1$, $0 < \eta_2 \leq 0.5$,
   $0.5 < \eta_3 < 1 < \eta_4$
2: **if** $\rho_k > \tau_3$ **then**
3:     **if** $\|p_k\| \leq \eta_3\delta_k$ **then**
4:         $\delta_{k+1} = \delta_k$
5:     **else**
6:         $\delta_{k+1} = \eta_4\delta_k$
7:     **end if**
8: **else if** $\tau_2 \leq \rho_k \leq \tau_3$ **then**
9:     $\delta_{k+1} = \delta_k$
10: **else**
11:     $\delta_{k+1} = \eta_2\delta_k$
12: **end if**
```

The Spectral Decomposition of Matrix $B_k + \sigma^* I$. Computing the thin QR factorization of matrix Ψ_k, $\Psi_k = Q_k R_k$, where, for $k \geq r$, $Q_k \in \mathbb{R}^{n\times 2r}$ and $R_k \in \mathbb{R}^{2r\times 2r}$, and the cheap spectral decomposition of the $2r \times 2r$ matrix $R_k M_k R_k^T$ as $R_k M_k R_k^T = U_k \hat{\Lambda} U_k^T$, where U_k and $\hat{\Lambda} = \text{diag}(\hat{\lambda}_1, \dots, \hat{\lambda}_{2r})$ are respectively orthogonal and diagonal matrices, leads to

$$B_k = B_0 + Q_k R_k M_k R_k^T Q_k^T = \gamma_k I + Q_k U_k \hat{\Lambda} U_k^T Q_k^T.$$

Now, let $P_\| \triangleq Q_k U_k$ and $P_\perp \triangleq (Q_k U_k)^\perp$ where $(.)^\perp$ denotes orthogonal complement. By Theorem 2.2.1 in [18], we have

$$P^T P = P P^T = I$$

where $P \triangleq \begin{bmatrix} P_{\parallel} & P_{\perp} \end{bmatrix} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix. Therefore, the spectral decomposition of B_k is obtained as

$$B_k = P\varLambda P^T, \qquad \varLambda \triangleq \begin{bmatrix} \varLambda_1 & 0 \\ 0 & \varLambda_2 \end{bmatrix}, \tag{25}$$

where

$$\varLambda_1 = \hat{\varLambda} + \gamma_k I = \mathrm{diag}(\hat{\lambda}_1 + \gamma_k, \hat{\lambda}_2 + \gamma_k, \ldots, \hat{\lambda}_{2k} + \gamma_k),$$
$$\varLambda_2 = \gamma_k I.$$

We assume the eigenvalues are increasingly ordered.

The inversion of $B_k + \sigma^* I$. Let $\tau_k = \gamma_k + \sigma$. Applying the Sherman-Morrison-Woodbury formula [34] to compute the inverse of the coefficient matrix $B_k + \sigma^* I$ leads to

$$p(\sigma) = -(B_k + \sigma I)^{-1} g_k = -\frac{1}{\tau_k}\left(I - \varPsi_k \left(\tau_k M_k^{-1} + \varPsi_k^T \varPsi_k\right)^{-1} \varPsi_k^T\right) g_k. \tag{26}$$

By (25) and (26), we have

$$\|p(\sigma)\| = \sqrt{\left\{\sum_{i=1}^{k} \frac{(g_{\parallel})_i^2}{(\lambda_i + \sigma)^2}\right\} + \frac{\|g_{\perp}\|^2}{(\gamma_k + \sigma)^2}}, \tag{27}$$

where

$$g_{\parallel} = P_{\parallel}^T g,$$
$$\|g_{\perp}\|^2 = \|g\|^2 - \|g_{\parallel}\|^2.$$

Assume $p_u \triangleq p(0)$ is the solution of the first optimality condition $(B_k + \sigma I)p(\sigma) = -g_k$, for which $\sigma = 0$ makes the second optimality condition $\sigma(\delta_k - \|p(\sigma)\|_2) = 0$ holds. If $\|p_u\| \leq \delta$, using (26) we have $(\sigma^*, p^*) = (0, p_u) = (0, p(0))$. If $\|p_u\| > \delta$, then p^* must lie on the boundary of the trust-region to make the second optimality condition hold. To impose this, σ^* must be the root of the following equation:

$$\phi(\sigma) \triangleq \frac{1}{\|p(\sigma)\|} - \frac{1}{\delta} = 0, \tag{28}$$

and can be determined by Newton's method, e.g. the variant proposed in [7]. The global solution of the trust-region subproblem is then $(\sigma^*, p^*) = (\sigma^*, p(\sigma^*))$.

Algorithm 3. Stochastic M-LBFGS-TR

```
1: Inputs: $w_0 \in \mathbb{R}^n$, number of multi-batches $\bar{N}$, number of epochs $epoch_{max}$, overlap set
   size $os$, $r$, $\epsilon_1, \epsilon_2, \tau_1 > 0$, $S_0 = Y_0 = [.]$, $k = 0$, epoch $= 0$
2: while $k \geq 0$ do
3:   if $k = 0$ then
4:     Take subsets $O_{-1}$ and $O_0$ of size $os$ for the initial multi-batch $J_0$
5:     Compute $F_0^{O_{-1}}$, $g_0^{O_{-1}}$ and $F_0^{O_0}$, $g_0^{O_0}$ by (29) and then $F_0^{J_0}$, $g_0^{J_0}$ by (30)
6:   else
7:     Take the second subset $O_k$ of size $os$ for the multi-batch $J_k$
8:     Compute $F_k^{O_k}$, $g_k^{O_k}$ by (29), and then $F_k^{J_k}$, $g_k^{J_k}$ by (30)
9:     if mod$(k+1, \bar{N}) = 0$ then
10:      Shuffle the data without replacement for next epoch and epoch $=$ epoch $+ 1$
11:    end if
12:  end if
13:  if $\|g_k^{J_k}\| \leq \epsilon_1$ or epoch $>$ epoch$_{max}$ then
14:    return
15:  end if
16:  if $k = 0$ or $S_k = [.]$ then
17:    Compute $p_k = -\delta_k \frac{g_k^{J_k}}{\|g_k^{J_k}\|}$
18:  else
19:    Compute $p_k$ using Algorithm 1
20:  end if
21:  Compute trial $w_t = w_k + p_k$ and $F_t^{O_k}$, $g_t^{O_k}$ by (29)
22:  Compute $(s_k, y_k) = (w_t - w_k, g_t^{O_k} - g_k^{O_k})$ and $\rho_k = (F_t^{O_k} - F_k^{O_k})/Q(p_k)$
23:  Compute $\psi_k = (F_k^{O_k} - F_t^{O_k}) + s_k^T(g_k^{O_k} + g_t^{O_k})$
24:  if sign$(\psi_k) > 0$ then
25:    $y_k = y_k + \frac{3\psi_k}{\|s_k\|^2} s_k$
26:  end if
27:  if $\rho_k \geq \tau_1$ then
28:    $w_{k+1} = w_t$
29:  else
30:    $w_{k+1} = w_k$
31:  end if
32:  Update $\delta_k$ using Algorithm 2
33:  if $s_k^T y_k > \epsilon_2 \|s_k\|^2$ then
34:    if $k \leq r$ then
35:      Store: $s_k$ and $y_k$ as new column in $S_{k+1}$ and $Y_{k+1}$
36:    else
37:      Keep: only the $r$ recent $\{s_j, y_j\}_{j=k-l+1}^{k}$ in $S_{k+1}$ and $Y_{k+1}$
38:    end if
39:    Compute the smallest eigenvalue $\hat{\lambda}$ of the problem (11)
40:    if $\hat{\lambda} > 0$ then
41:      $\gamma_{k+1} = \max\{1, 0.9\hat{\lambda}\} \in (0, \hat{\lambda})$
42:    else
43:      Compute $\gamma_k^h$ as $\frac{y_{k-1}^T y_{k-1}}{y_{k-1}^T s_{k-1}}$ and set $\gamma_{k+1} = \max\{1, \gamma_k^h\}$
44:    end if
45:    Compute $\Psi_{k+1}$ and $M_{k+1}^{-1}$ using (8)
46:  else
47:    Set $\gamma_{k+1} = \gamma_k$, $\Psi_{k+1} = \Psi_k$ and $M_{k+1}^{-1} = M_k^{-1}$
48:  end if
49:  $k = k + 1$
50: end while
```

5 Stochastic M-LBFGS-TR

In the stochastic setting, the training set is divided into multiple subsets called batches. The process of selecting a single batch, computing a subsampled gradient and loss for it and then updating the parameters create one single iteration of a stochastic algorithm. This process is repeated for each batch iteratively until one epoch, that is one pass through all data samples, is completed. After each epoch, the dataset is shuffled and new batches are generated.

Let J_k be a random subset of data at iteration k, whose size and index set of the samples included are denoted by $|J_k|$ and J_k^{idx}, respectively. In this work, samples are drawn without replacement for batches with fixed size. The subsampled loss and gradient are computed as follows

$$F_k^{J_k} := F^{J_k}(w_k) = \frac{1}{|J_k|}\sum_{i\in J_k^{idx}} f_i(w_k), \qquad g_k^{J_k} := \nabla F^{J_k}(w_k) = \frac{1}{|J_k|}\sum_{i\in J_k^{idx}} \nabla f_i(w_k). \quad (29)$$

In the stochastic L-BFGS-TR (sL-BFGS-TR) algorithm, when the batch J_k changes from one iteration to the next, the updates might be unstable since different data points are used to evaluate the gradient at the beginning (at w_k) and at the end of the iteration (at w_t), so that the gradient difference employed to update the Hessian approximation is computed as $y_k = g_t^{J_{k+1}} - g_k^{J_k}$. To overcome this problem, a remedy suggested in [39] consists in using the same multi-batch J_k for computing $y_k = g_t^{J_k} - g_k^{J_k}$ which requires double function and gradient evaluations at w_k and w_t. Another sampling strategy was proposed in [2] to compute $y_k = g_t^{O_k} - g_k^{O_k}$ where $O_k = J_k \cap J_{k+1} \neq \emptyset$ such that the overlap set O_k should not be insignificant. Similarly, in the stochastic M-LBFGS-TR (sM-LBFGS-TR) algorithm, when $\psi_k > 0$, the modified vector y_k^* is computed as in (20) with $\psi_k = (F_k^{O_k} - F_t^{O_k}) + (g_k^{O_k} + g_t^{O_k})^T s_k$.

In this work, we take a particular variant of this approach referred as *half overlap sampling* where $J_k = O_{k-1} \cup O_k$ and $|J_k| = 2|O_k|$. With this sampling strategy, the overall loss and gradients in (29) are computed as

$$F_k^{J_k} = \frac{1}{2}(F_k^{O_{k-1}} + F_k^{O_k}), \qquad g_k^{J_k} = \frac{1}{2}(g_k^{O_{k-1}} + g_k^{O_k}). \quad (30)$$

This requires two function and gradient evaluations on the overlap set of the current batch. The stochastic M-LBFGS-TR training algorithm is outlined in Algorithm 3.

Besides the previously indicated function and gradient evaluations, which constitute the predominant cost, the per iteration complexity of both sL-BFGS-TR and sM-LBFGS-TR algorithms consists in $2rn + O(r^3)$ operations needed to update B_k, and in the trust-region framework, $2(4r+1)n + O(r^2)$ flops to compute $Q(p)$ needed for ρ evaluation and to obtain the search direction $p(\sigma)$ using the direct formula described in (24). We also have the cost of computing a QR factorization and a cheap eigenvalue decomposition requiring $O(nr^2)$ and $O(r^3)$ operations, respectively.

Computing the numerator in (22) using subsampled function differences as $F_t^{J_k} - F_k^{J_k}$ requires double function evaluation at the beginning and at the end of the iteration. Experimentally, we examined that using overlap O_k in place of J_k provides a more affordable cost per iteration without any detriment in the attainable training accuracy. In support of this statement we have included Fig. 5.

We note that computing ψ_k in sM-LBFGS-TR does not impose any additional cost because it uses subsampled loss and gradient values corresponding to O_k which have been already evaluated in the previous iteration.

6 Experiments

We summarize in this section the behaviour of the described quasi-Newton optimization algorithms sL-BFGS-TR [35] and sM-LBFGS-TR on the training of two deep neural networks with different architectures for image classification of the benchmark datasets MNIST and CIFAR10 (see [22,25]). We used Glorot (Xavier) approach [16] for initializing the learning parameters. The architecture of the networks, which contain batch normalization layers, is described below.

- **LeNet-5.** A well known convolutional neural network designed for handwritten and machine-printed character recognition [26]. By solving an optimization problem for $w \in \mathbb{R}^{431,080}$, LeNet-5 with the following architecture is trained with the MNIST dataset:
  - Input layer with a $28 \times 28 \times 1$ image.
  - Convolutional layer with 20 filters of 5×5 size, stride 1 followed by ReLU.
  - Max pooling layer with a 2×2 and stride 2.
  - Convolutional layer with 50 filters of 5×5 size, stride 1 followed by ReLU.
  - Max pooling layer with a 2×2 and stride 2.
  - Fully connected layer with 500 neurons followed by ReLU.
  - Fully connected layer with 10 neurons followed by softmax.
- **ConvNet3FC2.** Motivated by [36], we define a CNN with 3 intermediate convolutional networks (ConvNet) and 2 fully connected networks (FC). This network with the structure defined below, is trained with CIFAR10 by solving an optimization problem for $w \in \mathbb{R}^{3,525,162}$:
  - Input layer with a $32 \times 32 \times 3$ image with Z-score normalization[1]
  - Convolutional layer with 32 filters of 5×5 size, stride 1 and padding 2.
  - Batch normalization layer followed by ReLU.
  - Max pooling layer with a 2×2 window and stride 1.
  - Convolutional layer with 32 filters of 5×5 size, stride 1 and padding 2.
  - Batch normalization layer followed by ReLU.
  - Max pooling layer with a 2×2 window and stride 1.
  - Convolutional layer with 64 filters of 5×5 size, stride 1 and padding 2.
  - Batch normalization layer followed by ReLU.

[1] Z-score normalization produces a dataset whose mean and standard deviation is zero and one, respectively.

- Max pooling layer with a 2×2 window and stride 1.
- Fully connected layer with 64 neurons.
- Batch normalization layer followed by ReLU.
- Fully connected layer with 10 neurons followed by softmax.

All experiments were run on an Ubuntu Linux server virtual machine with 32 CPUs and 128GB RAM using MATLAB and its deep learning toolbox. We provide a comparison with the most popular first-order method Adam implemented using the MATLAB built-in function `adamupdate` by a grid search tuning effort on learning rates and batch sizes. The best learning rate for all batch sizes was found to be 10^{-3}. The limited memory parameter for both quasi-Newton methods was set to $r = 20$. We obtained comparable results using different values of $r \in \{5, 10, 15, 20\}$ but we did not include these results here due to space limitation issues. Other hyperparameters for L-BFGS-TR and M-LBFGS-TR algorithms are $\epsilon_1 = 10^{-5}$, $\epsilon_2 = 10^{-2}$, $\gamma_0 = 1$, $\tau_1 = 10^{-6}$, $\tau_2 = 0.1$, $\tau_3 = 0.75$, $\eta_2 = 0.5$, $\eta_3 = 0.8$, $\eta_4 = 2$.

We have investigated the effect of the batch size on the performance of the different training algorithms. The networks were trained for a maximum number of epochs. The program stops before that limit if 100% accuracy has been reached. Figure 1 and 2 show the evolution of loss and accuracy for different batch sizes $|J_k| \in \{100, 500, 2000, 5000\}$ in the classification of MNIST and CIFAR10, respectively. The results corresponding to the smallest batch size for the MNIST dataset are reported within the first epoch only to facilitate the comparison. All the loss and accuracy evolution curves have been filtered by a fixed display frequency. This frequency, when indicated, corresponds to how many iterations per epoch have not been displayed. We observe from Fig. 1 and 2 that, for both problems, both sL-BFGS-TR and sM-LBFGS-TR perform better than *tuned* Adam independently of the batch size. In all the experiments, sM-LBFGS-TR exhibits comparable performance with respect to sL-BFGS-TR. Neither sL-BFGS-TR nor sM-LBFGS-TR are strongly influenced by batch size. Large multi-batch sizes can be employed without a considerable loss of accuracy even though the performance of both methods decreases when larger batch sizes are used, due to the smaller number of iterations per epoch (smaller number of parameters updates). Adam performs very well in both problems providing comparable accuracies to the ones yielded by second-order methods even if it is less accurate when large batch sizes are used.

Figure 4 displays the variability of the obtained test accuracy computed over five runs with random seeds. It can be seen that the results are reliable and that first-order methods exhibit larger variability than the two quasi-Newton algorithms. According to the complexity analysis performed in the former section, we found that the training time of both second-order methods is larger than that of the first-order one (see Table 1). Nevertheless, we underline the fact that, as Fig. 3 illustrates, with a fixed computational time budget the proposed quasi-Newton methods provide comparable or better testing accuracy than the first-order Adam optimizer.

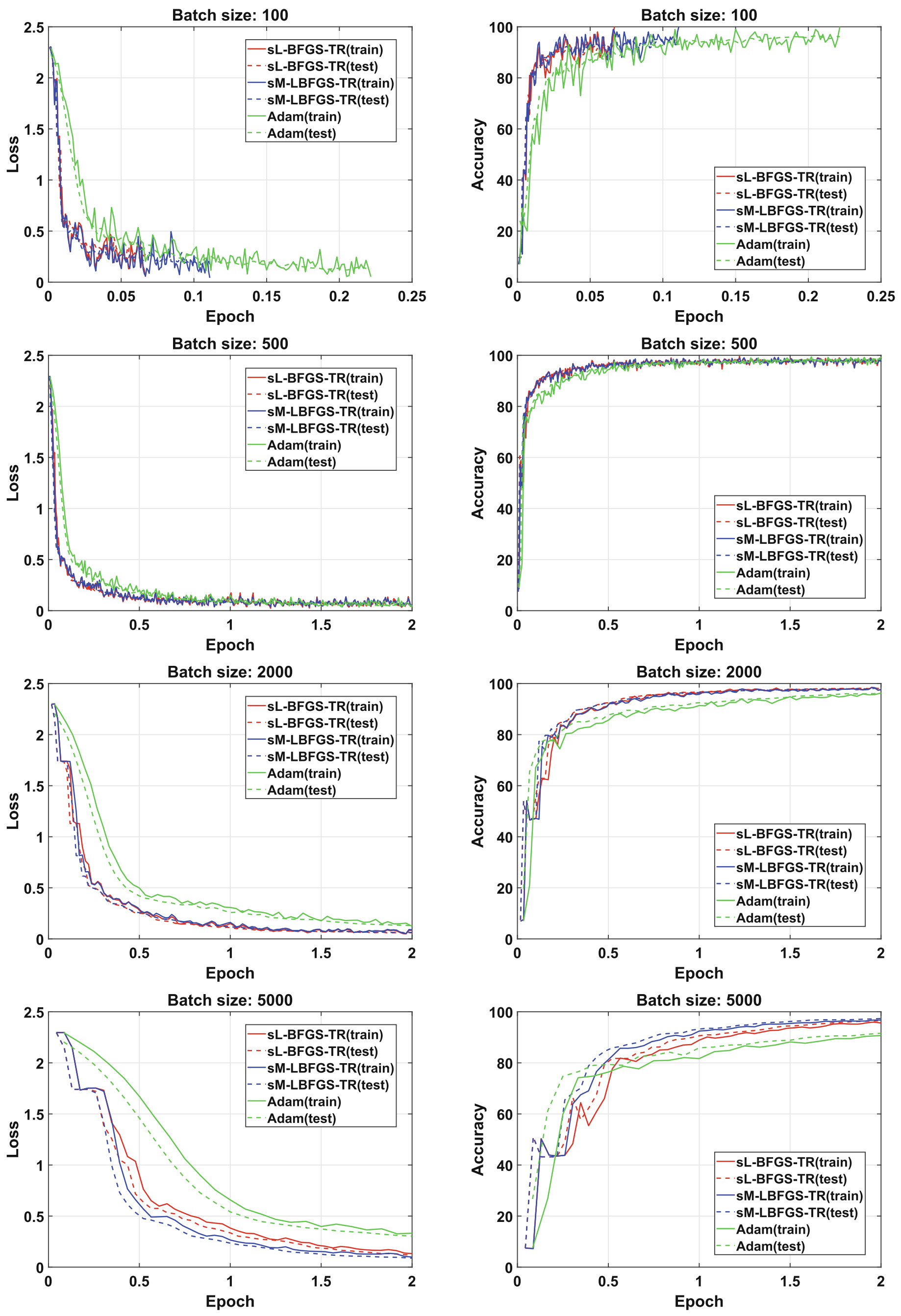

Fig. 1. MNIST: Evolution of the training and testing loss and accuracy using Stochastic Quasi-Newton based methods and *tuned* Adam for different batch sizes.

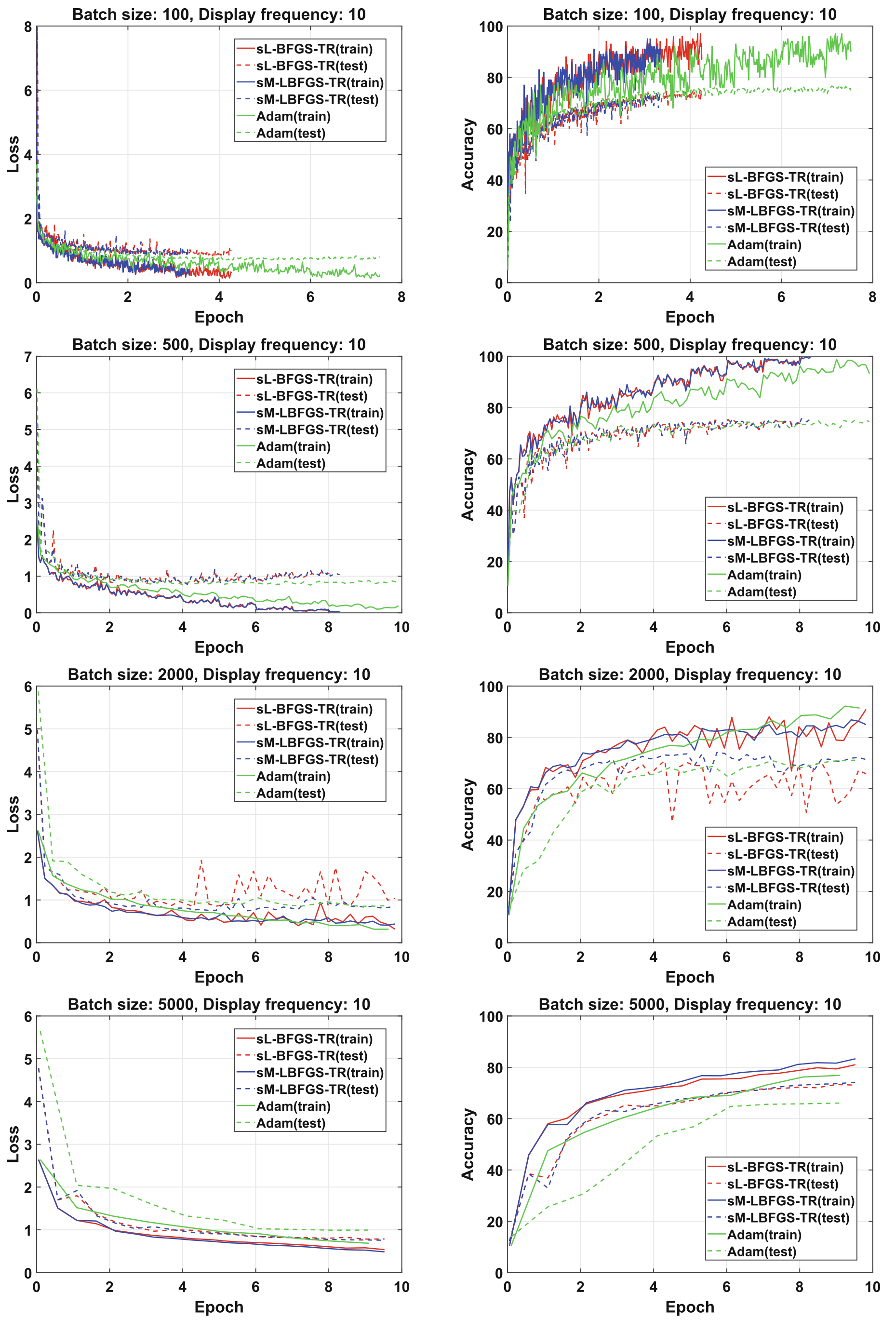

Fig. 2. CIFAR10: Evolution of the Training and Testing Loss and Accuracy using Stochastic Quasi-Newton based Methods and *Tuned* Adam for different Batch Sizes.

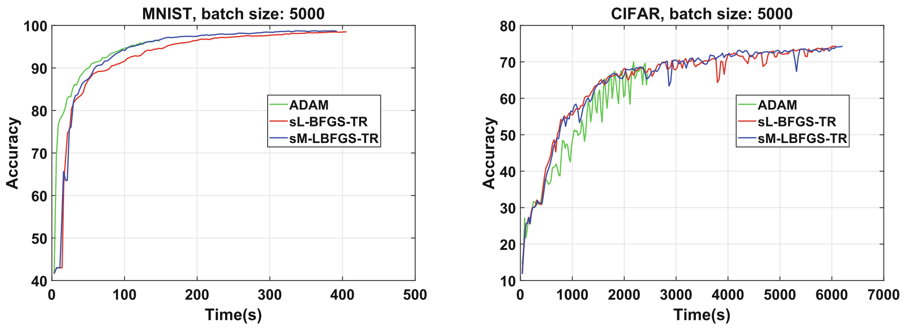

Fig. 3. Testing accuracy of Stochastic Quasi-Based methods and *Tuned* Adam versus training CPU time (in Seconds).

Table 1. Training time of the methods for k_{max} Iterations.

	CIFAR10 ($k_{max} = 100$)		MNIST ($k_{max} = 200$)	
	$bs = 500$	$bs = 5000$	$bs = 500$	$bs = 5000$
Adam	00:18:30	00:41:24	00:03:30	00:07:46
sL-BFGS-TR	00:32:04	00:55:04	00:09:35	00:13:45
sM-LBFGS-TR	00:32:06	00:54:46	00:09:42	00:13:40

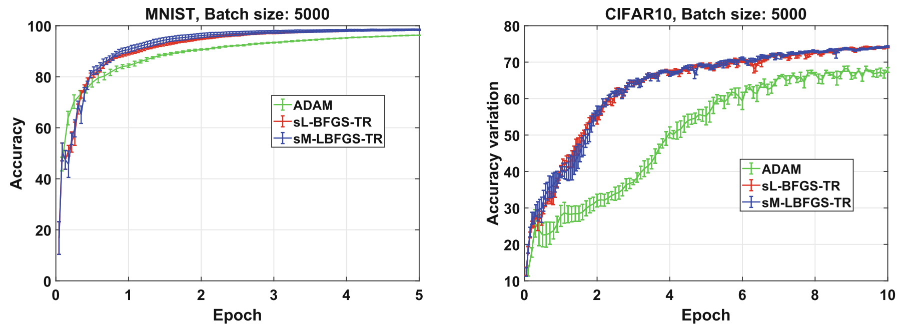

Fig. 4. Error Bars of stochastic Quasi-Based methods and *Tuned* Adam: variability of the test accuracy in the format "mean ± Standard Deviation" computed over five runs with random seeds.

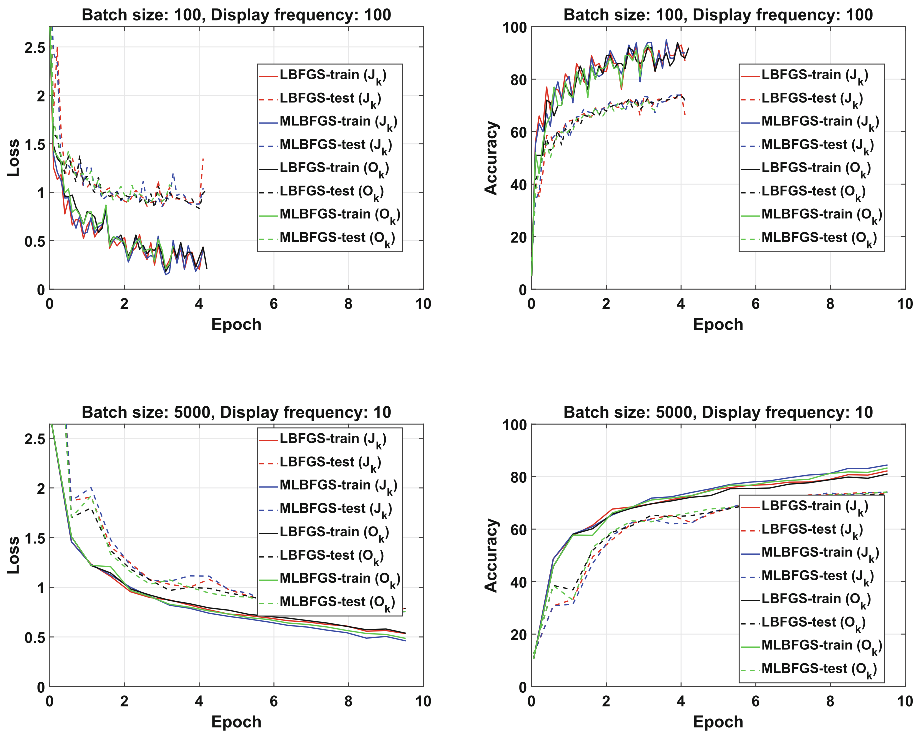

Fig. 5. CIFAR10: Evolution of the Training and Testing Loss and Accuracy using Quasi-Newton Methods with Different Sample Sets (O_k or J_k) to Compute Subsampled Loss Function Differences Needed to Compute ρ_k in (22).

7 Conclusions

In this work, we have considered stochastic limited memory BFGS quasi-Newton methods to solve nonlinear and non-convex optimization problems arising in the training of deep neural networks. Our implementation incorporates an accurate selection of the initial Hessian approximation and stable quasi-Newton updates are obtained by a sampling strategy with overlap. We have provided a comparison of the standard L-BFGS method with a variant of this algorithm based on a modified secant condition which is theoretically shown to provide an increased order of accuracy in the approximation of the curvature of the Hessian. In our experiments, on image classification problems with MNIST and CIFAR10 datasets, both sL-BFGS-TR and sM-LBFGS-TR exhibit comparable performances. Moreover, the results included in this paper illustrate that these methods converge faster than *tuned* Adam and perform better for larger batch sizes which are favorable for parallel computing. Restricted to the experiments with the largest considered batch size, the results show that with a fixed computational time budget the proposed quasi-Newton methods provide comparable or better testing accuracy than the first-order Adam optimizer. Nevertheless, despite their better convergence properties and the advantage of not requiring time-consuming tuning effort needed instead for Adam, the iteration complexity is high, since two loss and gradient evaluations are required at each iteration. Future research will be devoted to devising sampling strategies that reduce the number of loss and gradient evaluations per iteration. Future work will consists also in comparing the efficiency of the proposed stochastic L-BFGS optimizers with the recently introduced Kronecker-factored block diagonal L-BFGS described in [17] for feed-forward networks. Finally, another interesting future line of research we are currently undergoing is the analysis of whether symmetric rank one (SR1) updates, allowing for indefinite Hessian approximations, could potentially outperform L-BFGS in the task of high dimensional optimization in deep neural network training.

References

1. Adhikari, L., DeGuchy, O., Erway, J.B., Lockhart, S., Marcia, R.F.: Limited-memory trust-region methods for sparse relaxation. In: Wavelets and Sparsity XVII, vol. 10394, p. 103940J. International Society for Optics and Photonics (2017)
2. Berahas, A.L., Nocedal, J., Takáč, M.: A multi-batch L-BFGS method for machine learning. In: Advances in Neural Information Processing Systems, pp. 1055–1063 (2016)
3. Berahas, A.S., Takáč, M.: A robust multi-batch L-BFGS method for machine learning. Optim. Methods Softw. **35**(1), 191–219 (2020)
4. Bollapragada, R., Byrd, R.H., Nocedal, J.: Exact and inexact subsampled newton methods for optimization. IMA J. Numer. Anal. **39**(2), 545–578 (2019)
5. Bollapragada, R., Nocedal, J., Mudigere, D., Shi, H.-J., Peter Tang, P.T.: A progressive batching l-bfgs method for machine learning. In: International Conference on Machine Learning, PMLR, pp. 620–629 (2018)
6. Bottou, L., LeCun, Y.: Large scale online learning. Adv. Neural. Inf. Process. Syst. **16**, 217–224 (2004)
7. Brust, J., Erway, J.B., Marcia, R.F.: On solving L-SR1 trust-region subproblems. Comput. Optim. Appl. **66**(2), 245–266 (2016)
8. Burdakov, O., Gong, L., Zikrin, S., Yuan, Y.: On efficiently combining limited-memory and trust-region techniques. Math. Program. Comput. **9**(1), 101–134 (2016)
9. Byrd, R.H., Hansen, S.L., Nocedal, J., Singer, Y.: A stochastic quasi-newton method for large-scale optimization. SIAM J. Optim. **26**(2), 1008–1031 (2016)
10. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust region methods. SIAM (2000)
11. Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., Bengio, Y.: Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In: Advances in Neural Information Processing Systems, vol. 4, pp. 2933–2941 (2014)
12. Defazio, A., Bach, F., Lacoste-Julien, S.: Saga: a fast incremental gradient method with support for non-strongly convex composite objectives. In: Advances in neural information processing systems, pp. 1646–1654 (2014)
13. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. **12**(7), 2121–2159 (2011)

14. Erway, J.B., Griffin, J., Marcia, R.F., Omheni, R.: Trust-region algorithms for training responses: machine learning methods using indefinite hessian approximations. Optim. Methods Softw. **35**(3), 460–487 (2020)
15. Gay, D.M.: Computing optimal locally constrained steps. SIAM J. Sci. Statis. Comput. **2**(2), 186–197 (1981)
16. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference On Artificial Intelligence And Statistics, JMLR Workshop and Conference Proceedings, pp. 249–256 (2010)
17. Goldfarb, D., Ren, Y., Bahamou, A.: Practical quasi-newton methods for training deep neural networks (2020). arXiv preprint, arXiv:2006.08877
18. Golub, G.H., Van Loan, C.F.: Matrix computations, 4th edn. Johns Hopkins University Press (2013)
19. Gower, R., Goldfarb, D., Richtárik, P.: Stochastic block BFGS: squeezing more curvature out of data. In: International Conference on Machine Learning, pp. 1869–1878 (2016)
20. Johnson, R., Zhang, T.: Accelerating stochastic gradient descent using predictive variance reduction. Adv. Neural. Inf. Process. Syst. **26**, 315–323 (2013)
21. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings (2015)
22. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009). https://www.cs.toronto.edu/~kriz/cifar.html
23. Kungurtsevm V., Pevny, T.: Algorithms for solving optimization problems arising from deep neural net models: smooth problems (2018). arXiv preprint, arXiv:1807.00172
24. Kylasa, S., Roosta, F., Mahoney, M.W., Grama, A.: GPU accelerated sub-sampled newton's method for convex classification problems. In: Proceedings of the 2019 SIAM International Conference on Data Mining, pp. 702–710. SIAM (2019)
25. LeCun, Y.: The MNIST database of handwritten digits (1998). http://yann.lecun.com/exdb/mnist/
26. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
27. Lucchi, A., McWilliams, B., Hofmann, T.: A variance reduced stochastic newton method (2015). arXiv preprint, arXiv:1503.08316
28. Martens, J., Sutskever, I.: Training deep and recurrent networks with hessian-free optimization. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 7700, pp. 479–535. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35289-8_27
29. Modarres, F., Malik, A.H., Leong, W.J.: Improved hessian approximation with modified secant equations for symmetric rank-one method. J. Comput. Appli. Math. **235**(8), 2423–2431 (2011)
30. Mokhtari, A., Ribeiro, A.: Res: regularized stochastic BFGS algorithm. IEEE Trans. Signal Process. **62**(23), 6089–6104 (2014)
31. Mokhtari, A., Ribeiro, A.: Global convergence of online limited memory BFGS. J. Mach. Learn. Res. **16**(1), 3151–3181 (2015)
32. Moré, J.J., Sorensen, D.C.: Computing a trust region step. SIAM J. Sci. Stat. Comput. **4**(3), 553–572 (1983)
33. Moritz, P., Nishihara, R., Jordan, M.: A linearly-convergent stochastic L-BFGS algorithm. In: Artificial Intelligence and Statistics, pp. 249–258 (2016)

34. Nocedal, J., Wright, S.: Numerical optimization. Springer Science & Business Media (2006). https://doi.org/10.1007/978-0-387-40065-5
35. Rafati, J., Marcia, R.F.: Improving L-BFGS initialization for trust-region methods in deep learning. In: 2018 17th IEEE International Conference on Machine Learning and Applications, ICMLA, pp. 501–508. IEEE (2018)
36. Ramamurthy, V., Duffy, N.: L-SR1: a second order optimization method for deep learning (2016)
37. Robbins, H., Monro, S.: A stochastic approximation method. Ann. Math. Stat. **22**, 400–407 (1951)
38. Schmidt, M., Le Roux, N., Bach, F.: Minimizing finite sums with the stochastic average gradient. Math. Program. **162**(1–2), 83–112 (2017)
39. Schraudolph, N.N., Yu, J., Günter, S.: A stochastic quasi-Newton method for online convex optimization. In: Artificial intelligence and statistics, PMLR, pp. 436–443 (2007)
40. Wang, X., Ma, S., Goldfarb, D., Liu, W.: Stochastic quasi-newton methods for nonconvex stochastic optimization. SIAM J. Optim. **27**(2), 927–956 (2017)
41. Wei, Z., Li, G., Qi, L.: New quasi-Newton methods for unconstrained optimization problems. Appl. Math. Comput. **175**(2), 1156–1188 (2006)
42. Xu, P., Roosta, F., Mahoney, M.W.: Second-order optimization for non-convex machine learning: an empirical study. In: Proceedings of the 2020 SIAM International Conference on Data Mining, pp. 199–207. SIAM (2020)
43. Ziyin, L., Li, B., Ueda, M.: SGD may never escape saddle points (2021). arXiv preprint, arXiv:2107.11774

Author Queries

Chapter 2

Query Refs.	**Details Required**	**Author's response**
AQ1	As Per Springer style, both city and country names must be present in the affiliations. Accordingly, we have inserted the city names in affiliations. Please check and confirm if the inserted city names are correct. If not, please provide us with the correct city names.	
AQ2	This is to inform you that corresponding author has been identified as per the information available in the Copyright form.	