A stochastic nonmonotone trust-region training algorithm for image classification

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

Ángeles Martínez Calomardo
Department of Mathematics and Geosciences
University of Trieste
Trieste, Italy
amartinez@units.it

***Abstract*—In this work, we consider the target of solving the nonlinear and nonconvex optimization problems arising in the training of deep neural networks. To this aim we propose a nonmonotone trust-region (NTR) approach in a stochastic setting under inexact function and gradient approximations. We use the limited memory SR1 (L-SR1) updates as Hessian approximations when the curvature information is obtained by several different strategies. We provide results showing the performance of the proposed optimizer in the training of residual networks for image classification. Our results show that the proposed algorithm provides comparable or better testing accuracy than standard stochastic trust-region depending on the adopted curvature computing strategy and outperforms the well-known Adam optimizer.**

***Index Terms*—nonlinear optimization, stochastic quasi-Newton methods, SR1, deep neural networks, nonmonotone trust-region**

I. Introduction

In supervised learning problems, such as, image classification, the goal is to minimize the empirical risk of the model, e.g. a Deep Neural Network (DNN), by finding an optimal parametric mapping function $h(\cdot; w) : \mathbb{R}^n \longrightarrow \mathbb{R}$

$$\min_{w\in\mathbb{R}^n} F(w) \triangleq \frac{1}{N}\sum_{i=1}^{N} L(y_i, h(x_i; w)) \triangleq \frac{1}{N}\sum_{i=1}^{N} L_i(w), \quad (1)$$

where $w \in \mathbb{R}^n$ is the vector of trainable parameters of the model and (x_i, y_i) denotes the ith sample pair in an available C-class training dataset $\{(x_i, y_i)\}_{i=1}^N$ with an image x_i and its target y_i. In image classification tasks, usually the softmax cross-entropy function $L_i(w) = -\sum_{j=1}^{C}(y_i)_j \log(h(x_i; w))_j$, is used to measure the prediction error between the model computed value $h(x_i; w)$ and the target y_i. For solving the (nonconvex) nonlinear problem (1), applying traditional optimization algorithms is ineffective. Moreover, in most applications, both N and n can be very large, and thus the required computations can be too costly. For this reason, the most popular approaches for solving (1) are stochastic approximation methods in which only a small randomly-chosen sample set (i.e., mini-batch) from the training data is needed at each iteration to update the DNN parameters. Stochastic first-order methods (e.g. [1], [2]) have been widely used due to their low per-iteration cost and proven efficiency in practice. On the other hand, second-order methods can often find good minima in fewer steps due to their use of curvature information. There have been many efforts to develop methods including second-order information such as quasi-Newton (QN) [3] and Hessian-free (HF) Newton methods [4]. In this work, we focus on a QN-based algorithm; we consider the limited memory symmetric rank one (L-SR1) method (see, e.g. [3]) to generate Hessian approximations. L-SR1 methods in a stochastic trust-region approach for training neural networks were proposed in various works such as [5]–[7]. The main differences of these methods are based on the sampling strategy used to approximate the objective function and its gradient (progressive sampling with overlap batches [5], progressive without overlapping [6], [8], or fixed-size batches with overlapping [7]). In [6], [8] progressive sampling strategies are used where the size of the mini-batches increases at each iteration while in [5] it is increased only at specific iterations under certain conditions. In this work, we consider a stochastic quasi-Newton L-SR1 method with a nonmonotone Trust-region (TR) approach using fixed-size mini-batches without overlapping. The potential usefulness of nonmonotonicity may be traced back to [9] where a nonmonotone line-search technique was proposed for Newton's method to solve $\min_{w\in\mathbb{R}^n} F(w)$ where $F(w)$ is any twice continuously differentiable function. All modifications of Newton's method to ensure global convergence towards local minima require the use of a line-search technique which guarantees a monotonical decrease. This, however, may slow down the rate of convergence. This led to nonmonotone line-search in order to relax some standard line-search conditions, in the sense that it allows a local increase in the function values without affecting the convergence properties. The idea of using nonmonotonicity in trust-region could be dated back to [10]. Later, various nonmonotone trust-region methods were proposed for solving unconstrained optimization problems; see e.g., [11]–[13].

For solving (1), one can refer to the aforementioned nonmonotone techniques whether in line-search or trust-region regime. However, since in deep learning applications the sample size N and the number of parameters n of the model are huge, applying these techniques can be very costly.

In [14], a class of algorithms was proposed that use nonmonotone line-search rules fitting a variable sample scheme at each iteration. The main contribution of this work is the analysis for the first time of a nonmonotone stochastic trust-region method. Our method relies on inexact (subsampled) function, gradient and Hessian approximations obtained using fixed-size batches. We leave the study of different sampling strategies, like progressive or adaptive ones, for future research.

The paper is organized as follows: section II provides a general overview of the L-SR1 method within a trust-region framework (L-SR1-TR) for solving problem (1). In section III, we introduce a new training algorithm named L-SR1-NTR featuring a new nonmonotone trust-region technique and describe several strategies for computing curvature information to construct the L-SR1 approximated Hessian matrix. Our experimental results are presented in section IV. Finally, some concluding remarks are given in section V.

II. A brief review of the stochastic L-SR1-TR algorithm

A trust-region (TR) method [15] can iteratively solve the optimization problem (1) by replacing its objective function by a quadratic model in a region which is usually a ball of radius $\delta_k > 0$ around the current iterate w_k. This requires dynamically adjusting δ_k, and solving a quadratic (nonconvex) constrained optimization problem (the TR subproblem). Let $f_k := f_k^{J_k}$, $g_k := g_k^{J_k}$ be, respectively, the subsampled function and the subsampled gradient with respect to a mini-batch J_k of size bs whose samples are randomly selected at each iteration k, i.e.,

$$\begin{aligned} f_k^{J_k} &= \frac{1}{bs}\sum_{i=1}^{bs} L_i(w_k), \\ g_k^{J_k} &= \frac{1}{bs}\sum_{i=1}^{bs} \nabla L_i(w_k), \end{aligned} \tag{2}$$

and B_k any approximation of the true Hessian of the objective. Then the stochastic TR subproblem can be written as

$$p_k = \arg\min_{p\in\mathbb{R}^n} q_k(p) \triangleq \frac{1}{2}p^T B_k p + g_k^T p \quad \text{s.t.} \quad \|p\|_2 \le \delta_k. \tag{3}$$

After solving (3) a (stochastic) TR method proposes a trial iterate $w_t = w_k + p_k$. Let $f_t := f_t^{J_k}$ as defined in (2); then accepting w_t is subject to the value of the ratio between the actual and predicted reduction

$$\rho_k^{TR} := \rho_k = \frac{f_t - f_k}{q_k(p_k)}. \tag{4}$$

If $\rho_k \ge \bar{\eta} > 0$, then $w_{k+1} \triangleq w_t$; otherwise, $w_{k+1} \triangleq w_k$. Moreover, it makes sense to adjust the region by increasing δ_k when ρ_k is sufficiently large and decreasing δ_k when ρ_k is small.

A quasi-Newton (QN) method (see, e.g. [3]) can be used to construct matrix B_k in (3). In our work, we consider the limited memory SR1, L-SR1, method. Given an initial Hessian approximation $B_0 = \gamma_k I$, and curvature vectors pair (s_k, y_k) provided that $(y_k - B_k s_k)^T s_k \ne 0$, then the SR1 updates are defined as

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}, \quad k = 0, 1, \ldots. \tag{5}$$

Using two storage matrices S_k and Y_k with at most l columns where $l \ll n$ for storing the most recent l pairs $\{s_j, y_j\}$, the compact form of L-SR1 updates can be written as

$$B_k = B_0 + \Psi_k M_k \Psi_k^T, \quad k = 1, 2, , \ldots, \tag{6}$$

where

$$\begin{aligned} \Psi_k &= Y_k - B_0 S_k, \\ M_k &= (D_k + L_k + L_k^T - S_k^T B_0 S_k)^{-1}, \end{aligned}$$

with matrices L_k and D_k which are, respectively, the strictly lower triangular part and the diagonal part of $S_k^T Y_k$.

In the next section, we introduce a stochastic L-SR1 method using a nonmonotone TR method. Function and gradient are computed approximately using fixed-size sampling without overlapping. Regardless of the sampling strategy and the type of TR ratio adjustment, the solution of the TR subproblem (3) is the computational bottleneck of most TR methods. In our work, we use an efficient algorithm called OBS [16], exploiting the structure of the L-SR1 matrix to obtain global solutions. Regarding the selection of γ_k for the initial Hessian approximation $B_0 = \gamma_k I$, we refer to the strategy proposed in [5]. Our training algorithm uses Jacobian regularization [17] to penalize large changes in predictions with respect to small changes in inputs in order to make the network more robust. The regularized problem is formulated as:

$$F(w) \triangleq \frac{1}{N}\sum_{i=1}^{N}\left(L_i(w) + \frac{\lambda}{2}\|J_{x_i}\|_F^2\right), \tag{7}$$

where $\lambda > 0$ is a regularization parameter and $\|J_{x_i}\|_F$ stands for the Frobenius norm of the input-output Jacobian matrix $J := J_{x_i} \in \mathbb{R}^{C\times n}$ w.r.t. the ith input image x_i. Given $v \in \mathbb{R}^C, v \sim N(0,1)$, an approximation of the Frobenius norm is obtained by the Hutchinson's trick, i.e., $\mathbb{E}_v[v^T J J^T v] = \text{trace}(JJ^T)$, so that we end up with

$$\|J_{x_i}\|_F^2 \approx C\frac{1}{m_p}\sum_{j=1}^{m_p}\left\|\frac{\partial(v_{[j]}^T \hat{y})}{\partial x}\right\|_2^2, \tag{8}$$

where $\hat{y} = h(x_i; w) \in \mathbb{R}^C$ is the output of the network and $v_{[j]}, j = 1, \ldots, m_p$ are random vectors drawn from a standard normal distribution. In practice m_p can be considered as unit; see [17] for more details on the effective better performance of Jacobian regularization to increase the generalization capability with respect to L2. Equation (8) shows that the regularization term is equivalent in cost to one gradient evaluation w.r.t inputs when $m_p = 1$.

III. The new training algorithm

A. Nonmonotone trust-region

Monotone techniques require the value of the function to be decreased at each iteration which may reduce the speed of convergence for some problems, especially in the presence of narrow curved valley. To improve the efficiency of trust region methods, a variety of nonmonotone techniques have been proposed. In [9] a technique called nonmonotone line-search was introduced to relax some standard conditions by allowing the objective function value to increase in some iterations and accepting the step-length α_k whenever

$$F(w_k + \alpha_k p_k) \leq F_{m(k)} + c_1 \alpha_k \nabla F(w_k)^T p_k, \tag{9}$$

where $c_1 \in (0, 0.5)$ and $F_{m(k)}$ is the maximum function value of a prefixed number (M) of previous iterates, i.e.,

$$F_{m(k)} = \max_{0 \leq j \leq m(k)} \{F(w_{k-j})\}, \quad k = 0, 1, \ldots$$

in which $m(0) = 0$, $0 \leq m(k) \leq \min\{m(k-1)+1, M\}$ for all $k \geq 1$, and $M \geq 0$.

Considering that the TR method is a generalization of the Armijo line-search approach [18], a nonmonotone trust-region (NTR) was also proposed [10]. The most common NTR ratio is defined as

$$\bar{\rho}_k^{NTR} = \frac{F(w_k + p_k) - F_{m(k)}}{Q_k(p_k)}, \tag{10}$$

where $Q_k(p)$ is a quadratic model of F around the current iterate w_k and is similarly defined as $q_k(p)$ in (3). There are also other variants to (10) where the nonmonotone term $F_{m(k)}$ is replaced with c_k (see e.g. [12]) or r_k (see e.g. [11]), where c_k is defined as a weighted moving average of objective function values and r_k is a convex combination of $F_{m(k)}$ and the latest objective function value.

As mentioned in section I, the sample size N and the number of parameters n are usually large numbers in deep learning applications; therefore, applying these techniques can be costly. In [14], a class of algorithms was proposed that use nonmonotone line-search rules fitting a variable sample scheme in which the sample size may increase or decrease at each iteration. In this work we present a nonmonotone trust-region algorithm using subsampled function and gradient obtained with fixed-size batches.

B. Stochastic L-SR1 nonmonotone trust-region

In this section, we describe our stochastic algorithm by exploiting a nonmonotone term r_k similar to the one proposed in [11]. Given the inexact evaluated quantities at iteration k (i.e. f_k, g_k, and B_k), the solution p_k obtained by solving (3), and the subsampled objective function evaluated at the trial parameter $w_t = w_k + p_k$, f_t, we define a stochastic NTR ratio as follows

$$\rho_k^{NTR} = \frac{f_t - r_k}{q_k(p_k)}, \tag{11}$$

where $r_k = \tau_k f_{m_k} + (1-\tau_k) f_k$ in which $\tau_k \in [\tau_{min}, \tau_{max}]$ with $\tau_{min} \in [0, 1)$, $\tau_{max} \in [\tau_{min}, 1]$, and f_{m_k} is the nonmonotone term. The parameter τ_k determining the level of monotonicity can be updated by the following formula

$$\tau_k = \begin{cases} \frac{\tau_0}{2}, & \text{if } k = 1, \\ \frac{\tau_{k-1} + \tau_{k-2}}{2}, & \text{if } k \geq 2. \end{cases} \tag{12}$$

The definition of f_{m_k} in the stochastic context is what differentiates our proposed nonmonotone trust-region ratio from the one used in [11]. We consider the following three options:

1) The first trivial option is to compute the nonmonotone term r_k using the subsampled function values rather than the *true* function value. In our algorithm, the quantity f_{m_k} in r_k can be the maximum value of the (at most) M recent subsampled function values; i.e., for $k = 0, 1, \ldots$
$$f_{m_k} = \max\{f_j^{J_j} \,|\, k - M + 1 \leq j \leq k\}. \tag{13}$$
2) It may be possible to reduce the stochasticity of the first choice by taking f_{m_k} as a reference point which is updated at specific iterations. For this reason, our second choice for f_{m_k} is the maximum value of the most recent M subsampled function evaluations which is computed every M iterations. In other words, (13) is computed only if $k = 0$ or $\mathrm{mod}(k, M) = 0$; otherwise $f_{m_k} = f_{m_{k-1}}$.
3) Since f_t and r_k in ρ_k^{NTR} are evaluated w.r.t different mini-batches, the ratio ρ_k^{NTR} may suffer from some noise resulting from function differences in the numerator. Therefore, another natural but costly option can be computing all functions involved in the numerator w.r.t to the same (current) mini-batch. In this option, f_{m_k} is the maximum value of the M most recent subsampled functions evaluated w.r.t J_k; i.e., for $k = 0, 1, \ldots$
$$f_{m_k} = \max\{f_j^{J_k} \,|\, k - M + 1 \leq j \leq k\}. \tag{14}$$

Experimentally, we found that the second choice performed slightly better than the others, even if all the choices behave identically, in the sense that they allow for almost the same final testing accuracy. The stochastic nonmonotone trust-region ratio is then used to adjust the trust-region radius δ_k and accepting the trial point as explained in section II. The stochastic L-SR1 nonmonotone trust-region method (denoted by sL-SR1-NTR from now on) is outlined in Algorithm 1.

C. Curvature computing strategies

In this section, we describe different strategies for computing the curvature vector y_k needed to update the L-SR1 matrix B_k (6). The L-SR1 update (5) is originally obtained by defining the iterate and gradient displacements so that it satisfies the secant equation $y_k = B_{k+1} s_k$, meaning that a second-order Taylor expansion is satisfied along the most recent direction s_k [3], i.e.,

$$s_k = p_k, \qquad y_k = g_t - g_k, \tag{15}$$

where $g_t := g_t^{J_k} = \frac{1}{bs}\sum_{i=1}^{bs} \nabla L_i(w_t)$. However, quasi-Newton updating is inherently an overwriting process rather

Algorithm 1 sL-SR1-NTR

```
1: Inputs: $k=0,\ w_0,\ S=Y=[\,]$
2: for $epoch = 1,2,\ldots,$ do
3:     Shuffle $N$ samples for randomly creating $Nb$ mini-batches
4:     for $iter = 1,2,\cdots,Nb$ do
5:         Compute $f_k$ and $g_k$ w.r.t given $J_k$ of size $bs$ at $w_k$
6:         if $\|g_k\|_2 < \eta_0$ or other stopping conditions then
7:             Stop training
8:         end if
9:         if $k=0$ or $S=[\,]$ then
10:            Set $B_k = \gamma_0 I$, and compute $p_k = \frac{-\delta_k g_k}{\|g_k\|}$
11:        else
12:            Compute $B_k = \gamma_k I + \Psi_k M_k^{-1} \Psi_k^T$, and $p_k$ by OBS
13:        end if
14:        Set $w_t = w_k + p_k$
15:        Compute $f_t$ w.r.t $J_k$ at $w_t$
16:        Compute $\rho_k$ NTR ratio (11) using $r_k$ with $\tau_k$ defined in (12)
17:        if $\rho_k > \eta_1$ then
18:            Compute $s_k = p_k$, and $y_k$ by one of the strategies in Table I
19:            Construct a new well-defined $B_{k+1}$ by (6)
20:            Set $w_{k+1} = w_t$
21:        else
22:            Find $\alpha_k$ by a stochastic nonmonotone line-search (NLS), i.e.,
               $f^{J_k}(w_k + \alpha_k p_k) \le r_k + c_1 \alpha_k g_k^T p_k$
23:            if NLS succeeds then
24:                Set $w_t = w_k + \alpha_k p_k$
25:                Compute $s_k = \alpha_k p_k$, and $y_k$ by one of the strategies in
                   Table I
26:                Construct a new well-defined $B_{k+1}$ by (6)
27:                Set $w_{k+1} = w_t$
28:            else
29:                Skip updating $B_k$ and $w_k$
30:            end if
31:        end if
32:        if $\rho_k > \eta_2$ then
33:            $\delta_{k+1} = \min\{2\delta_k, 10\}$
34:        else if $\eta_1 \le \rho_k \le \eta_2$ then
35:            $\delta_{k+1} = \delta_k$
36:        else
37:            $\delta_{k+1} = \eta_3 \delta_k$
38:        end if
39:    end for
40:    $k = k+1$
41: end for
```

than an averaging process; therefore, a single poor update might have long-lasting effects on several next iterations. This can cause a detrimental effect in a stochastic setting. Indeed, the curvature estimate y_k must reproduce the action of the Hessian of the entire objective function in (1) while this is not achieved by subsampled gradient differences based on a limited number of samples. An effective approach to achieving a more stable Hessian approximation is to decouple the calculations of stochastic gradient done for updating parameters and computing y_k [19], [20]. In this manner, one can employ a different and larger random mini-batch, if necessary, for both gradients involved in y_k (15).

Considering the first-order Taylor expansion for approximating gradient difference, an alternative strategy is a subsampled Hessian matrix-vector product which might better represent the action of the true Hessian even with a high variance subsampled gradient. To do so, a different and large enough random mini-batch J^{H_k} of size Bs is considered for computing y_k such that

$$s_k = p_k, \qquad y_k = \tilde{H}_k^{J_{H_k}} s_k, \tag{16}$$

where $\tilde{H}_k^{J_{H_k}} = \frac{1}{Bs}\sum_{i=1}^{Bs} \nabla^2 L_i(w_k)$. Regardless of the definition of y_k whether in (15) or (16), the cost of computing y_k with respect to a larger set of samples is high. To address this issue, the curvature estimate vector y_k can be computed periodically, where the quasi-Newton approximation (L-SR1) is updated after a sequence of iterations (say L iterations) and is kept fixed within these iterations. Note that the subsampled Hessian-vector product $\tilde{H}_k^{J_{H_k}} s_k$ in (16) can be coded directly in practice, without explicitly constructing $\tilde{H}_k^{J_{H_k}}$. In our work, we use the Hessian automatic differentiation technique provided by MATLAB software. This technique performs one forward-backward process in order to retain trace of the first order derivatives in $g_k^{J_{H_k}}$. Subsequently, given vector s_k, it calculates the gradient of ${g_k^{J_{H_k}}}^T s_k$ by one more forward-backward process. More precisely,

$$\frac{\partial (g^{J_{H_k}})^T s_k}{\partial w} = \frac{\partial (g^{J_{H_k}})^T}{\partial w} s_k + (g^{J_{H_k}})^T \frac{\partial s_k}{\partial w} = \frac{\partial (g^{J_{H_k}})^T}{\partial w} s_k, \tag{17}$$

where the first equality comes from applying the chain rule, the second equality is because of $\frac{\partial s_k}{\partial w} = 0$. The last expression in (17) produces the subsampled Hessian matrix-vector product at w_k, i.e., $\tilde{H}_k^{J_{H_k}} s_k$. This shows that $\tilde{H}_k^{J_{H_k}} s_k$ can be obtained at a computational cost of no more than two gradient evaluations.

Note that the computation of y_k by (16) or (15) increases the per iteration cost with respect to first-order stochastic methods which require only one subsampled gradient estimate per iteration. As mentioned in [19], this is not too serious as long as the per-iteration improvement outweighs the extra per-iteration cost. Nevertheless, we consider a third strategy for computing y_k using a variant of the empirical Fisher Information Matrix (eFIM), see e.g. [21], which is called accumulated eFIM (aeFIM). Given a memory budget of L_f, we use aeFIM-vector products for curvature computations as

$$s_k = p_k, \qquad y_k = \frac{1}{\gamma} \sum_{j=k-\gamma+1}^{k} g_j g_j^T s_k, \tag{18}$$

where $g_j := g^{J_j}(w_j)$ and $\gamma = \min\{k, L_f\}$. Obviously, this can only come at the expense of one storage matrix to store L_f computed stochastic gradients. Although aeFIM can reduce the computational cost considerably, using old noisy gradient estimates for computing y_k can produce poor curvature information. The choice of the curvature matrix for the computation of y_k must address the trade-off between providing informative curvature information and the computational and storage expenses.

IV. Numerical Experiments

We provide in this section some experimental results[1] to illustrate the performance of our nonmonotone trust-region method on image classification of the CIFAR10 benchmark[2] dataset on ResNet-20, a deep residual network with around

[1]Codes available at: https://github.com/MATHinDL/sL_QN_TR/
[2]Available at: https://www.cs.toronto.edu/~kriz/cifar.html

TABLE I
EXPERIMENTAL CONFIGURATION

Task	
Cifar10 images classification	
Approach	
ResNet-20 with 273,258 parameters [22]	
Problem	
Solving (7) where $\lambda = 1$	
Training algorithms	
sL-SR1-NTR	Algorithm 1 using NTR ratio (11)
sL-SR1-TR	Algorithm 1 using TR ratio (4)
Adam	[2]
Curvature strategies for computing y_k	
Gd:	$bs = 1000$
Fv:	$bs = 1000$
Hv-T0:	$J_k = J_{H_k}$ and $bs = 1000$, $Bs = bs$
pHv-T0:	$J_k = J_{H_k}$ and $bs = 1000$, $Bs = bs$
Hv-T1:	$J_k \neq J_{H_k}$ and $bs = 1000$, $Bs = bs$
pHv-T1:	$J_k \neq J_{H_k}$ and $bs = 1000$, $Bs = bs$
pHv-T2:	$J_k \neq J_{H_k}$ and $bs = 1000$, $Bs = 3bs$
pHv-T0 + Fv:	$J_k = J_{H_k}$ and $bs = 1000$, $Bs = bs$
pHv-T1 + Fv:	$J_k \neq J_{H_k}$ and $bs = 1000$, $Bs = bs$
pHv-T2 + Fv:	$J_k \neq J_{H_k}$ and $bs = 1000$, $Bs = 3bs$
Training time	
90 minutes (GPU Time)	
Hyper-parameters	
$\delta_0 = 1$, $\gamma_0 = 1$, $\eta_0 = 10^{-4}$, $\eta_1 = 0.1$, $\eta_2 = 0.75$, $\eta_3 = 0.5$, $l = 30$, $L = 5$, $L_f = 100$, $M = 10$, $\tau_0 = 0.25$, $\tau_1 = \frac{\tau_0}{2}$	

27k parameters [22]. To this goal, 10000 images (out of 60000 images) are set aside as testing set during the training phase.

Table I describes the configuration of our experiments. All of them were performed with the MATLAB DL toolbox on a Ubuntu 20.04.4 LTS (64-bit) Linux server VMware with 20GB memory using a VGPU NVIDIA A100D-20C. We have used the same initial parameter $w_0 \in \mathbb{R}^n$ by specifying the same seed to the MATLAB random number generator. Due to the employment of the Jacobian regularization technique, whose benefit can be observed in Fig. 1, we solved the optimization problem of minimizing (7) rather than solving the original problem (1). Note that our algorithm uses subsampled loss which includes the regularization term for network training while the quantity (denoted as **Loss**) which is displayed in the training progress plots reported in Fig. 1 does not include the regularization term. In Fig. 2–5 we show the performance of the algorithms in terms of the testing accuracy of the classification model versus training time or iteration number of the training process. Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions. We consider several different strategies for computing vector y_k in both sL-SR1-TR and sL-SR1-NTR which have been described in subsection III-C, that is, gradient differences (15), accumulated empirical Fisher matrix-vector product (18) and Hessian matrix-vector product (16) (respectively denoted as **Gd**, **Fv** and **Hv** in Table I). The latter approach can be implemented in many different settings depending on the sampling strategy used and the frequency in

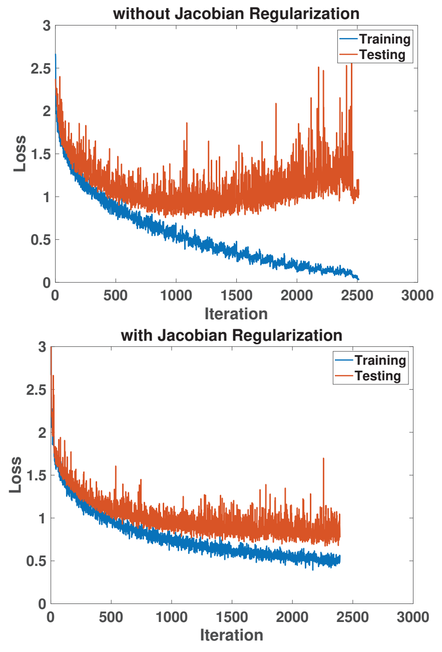

Fig. 1. The effect of regularization over testing accuracy for sL-SR1-NTR with $bs = 500$ and the Fv approach.

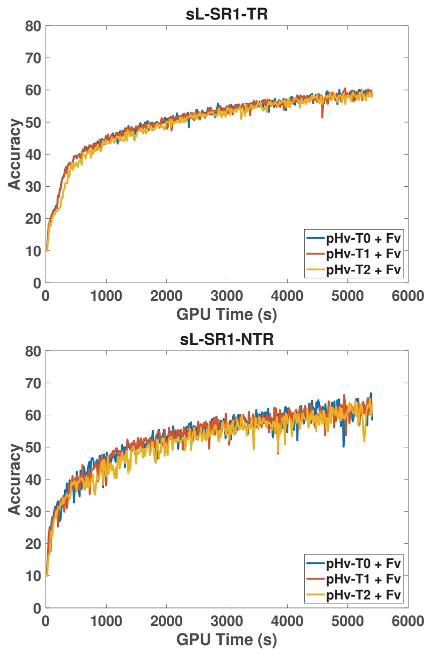

Fig. 2. Testing accuracy provided by sL-SR1-TR (left) and sL-SR1-NTR (right) using different **pHv + Fv** approaches.

which this computation is performed. We have studied also different sampling techniques for computing y_k, i.e. using the same mini-batch, using the same batch size but different samples or using different mini-batches with different sizes for computing the subsampled gradient and subsampled Hessian approximations (respectively denoted as **T0**, **T1** and **T2** in Table I). We have also investigated the periodic computation of y_k and thus the decoupling of the parameters' update from the quasi-Newton update. In Table I, the letter **p** stands for **p**eriodic. Finally, we have also experimented to see wether in the periodic setting keeping the QN Hessian approximation B_k fixed between curvature updates is beneficial or better results can be obtained when computing y_k in the intermediate iterations by eaFIM (e.g. **pHv-T2+Fv**).

In Fig. 2 and 3, we report the results of the experiments comparing the testing accuracy reached with the different curvature computing strategies when using sL-SR1-TR or sL-SR1-NTR. Fig. 3 shows that computing y_k using Hessian matrix-vector products (**Hv**) by the **T0** strategy leads to a more oscillatory testing curve than when using **T1** for both algorithms sL-SR1-TR and sL-SR1-NTR. On the other hand, periodically computing y_k produces smoother testing curves, in general. Nevertheless, there are some strong instabilities in the testing accuracy of sL-SR1-NTR when using **pHv-T0** while they are damped by **pHv-T1**. This result shows that using different samples for computing y_k with respect to the ones used for computing the subsampled gradient is more important in sL-SR1-NTR than in sL-SR1-TR.

Fig. 3 shows that using a hybrid approach for computing y_k leads to more stable testing accuracy for both sL-SR1-TR and sL-SR1- NTR even though it produces lower testing accuracy for sL-SR1-NTR. Fig. 2 illustrate that the periodical approaches behave similarly both in sL-SR1-TR or sL-SR1-NTR. Therefore, **pHv-T1 + Fv** from Fig. 2 is selected as the better curvature computation strategy in the NTR framework. Since the **pHv-T1 + Fv** approach produces more stable testing accuracy than **pHv-T1**, we have considered this approach as the best **Hv**-based strategy for computing y_k, and compared it with other strategies in Fig. 4. Finally, the most important conclusion that can be extracted from Fig. 3 is that computing the curvature at each iteration by accumulated empirical Fisher matrix-vector product allows for faster training and higher final testing accuracy in all cases, but the improvement is greater when using NTR.

Fig. 4 shows the testing accuracy versus iterations of sL-SR1-TR and sL-SR1-NTR using three different curvature strategies, i.e., from left to right **Fv**, **Gd** and **pHv-T1 + Fv**. The results reported in this figure show that the best and worst performances of our proposed algorithm are obtained with **Fv**- and **Gd**-based curvature computing strategies, respectively. We include also for comparison the testing accuracies provided by the state-of-the-art Adam optimizer [2]. Adam is run for the same amount of time than the stochastic QN algorithms with learning rate $\alpha_k = 10^{-3}$ experimentally found to be the optimal one. It can be observed that sL-SR1-NTR provides a comparable or a better testing accuracy than Adam in the same fixed amount of training time.

Fig. 5 shows the results obtained in minimizing function (7) with sL-SR1-TR and sL-SR1-NTR using the periodic hybrid strategy **pHv + Fv** for curvature computation and different values bs of the batch-size J_k. This time we allowed the code to run for three hours to show that the final testing accuracy increases with respect to the previous experiments in which the training process was stopepd after 90 minutes. The results reported in Fig. 5 include both training and testing accuracies versus time and show that the best approach reveals always sL-SR1-NTR.

Finally, in Fig. 6 we complete the study performed in Fig. 5 including also for comparison the training and testing accuracies provided by Adam.

V. Conclusion

In this work, we have studied a stochastic limited memory SR1 quasi-Newton method in a nonmonotone trust-region framework for solving the optimization problems arising in image classification tasks on deep neural networks. As far as we know this is the first attempt to analyze a nonmonotone trust-region method in a stochastic setting. We have also experimented with three different strategies for computing the curvature vector y_k required for SR1 updates of the Hessian approximation. We have found that using accumulated empirical Fisher matrix-vector products produces better training than when curvature is obtained by subsampled gradient differences or subsampled Hessian matrix-vector products.

Our experiments show that our proposed algorithm (sL-SR1-NTR) provides better testing accuracy when using the accumulated empirical Fisher matrix-vector products to compute y_k and its performance is comparable or superior than sL-SR1-TR and the state-of-the-art Adam optimizer.

Future work we are currently undergoing regards the convergence analysis of nonmonotone trust-region under inexact objective, gradient and Hessian information.

Acknowledgment

We acknowledge the support of the INdAM - GNCS Project CUP_E55F22000270001.

References

[1] L. Bottou and Y. Cun, "Large scale online learning," *Advances in neural information processing systems*, vol. 16, 2003.

[2] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[3] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[4] J. Martens *et al.*, "Deep learning via hessian-free optimization." in *ICML*, vol. 27, 2010, pp. 735–742.

[5] J. B. Erway, J. Griffin, R. F. Marcia, and R. Omheni, "Trust-region algorithms for training responses: machine learning methods using indefinite hessian approximations," *Optimization Methods and Software*, vol. 35, no. 3, pp. 460–487, 2020.

[6] X. Wang and Y.-x. Yuan, "Stochastic trust-region methods with trust-region radius depending on probabilistic models," *Journal of Computational Mathematics*, vol. 40, no. 2, pp. 294–334, 2022.

[7] M. Yousefi and A. Martínez, "On the efficiency of stochastic quasi-Newton methods for deep learning," *arXiv preprint arXiv:2205.09121*, 2022.

[8] R. Chen, M. Menickelly, and K. Scheinberg, "Stochastic optimization using a trust-region method and random models," *Mathematical Programming*, vol. 169, no. 2, pp. 447–487, 2018.

[9] L. Grippo, F. Lampariello, and S. Lucidi, "A nonmonotone line search technique for Newton's method," *SIAM journal on Numerical Analysis*, vol. 23, no. 4, pp. 707–716, 1986.

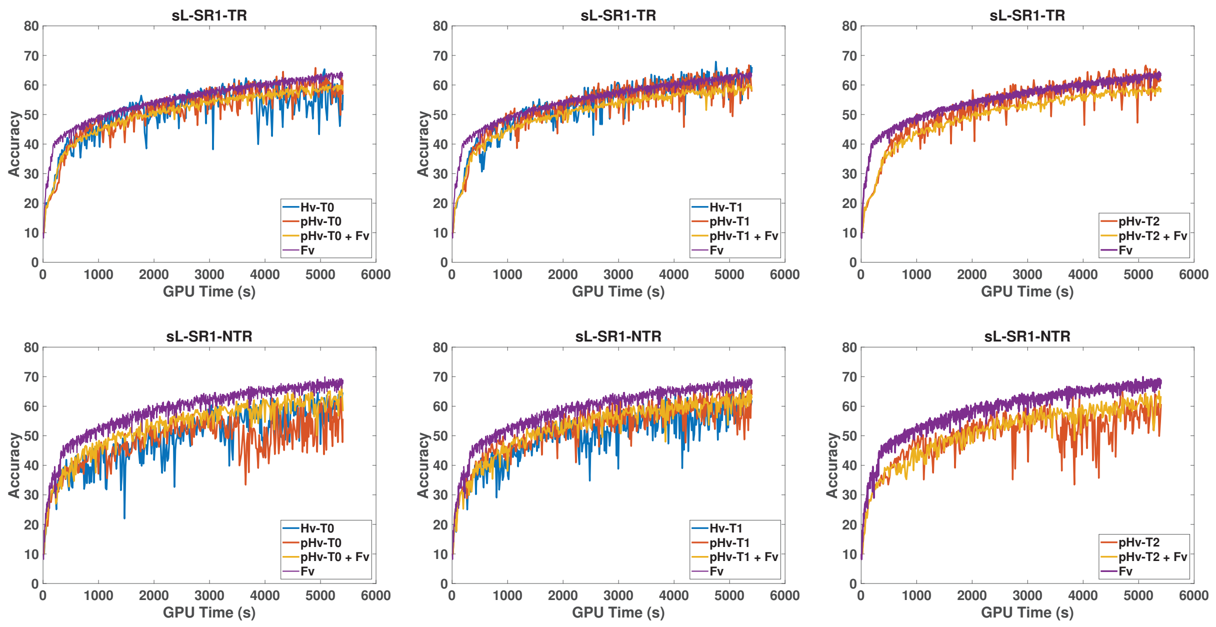

Fig. 3. Testing accuracy with sL-SR1-TR (up) and sL-SR1-NTR (down) using different curvature computing strategies.

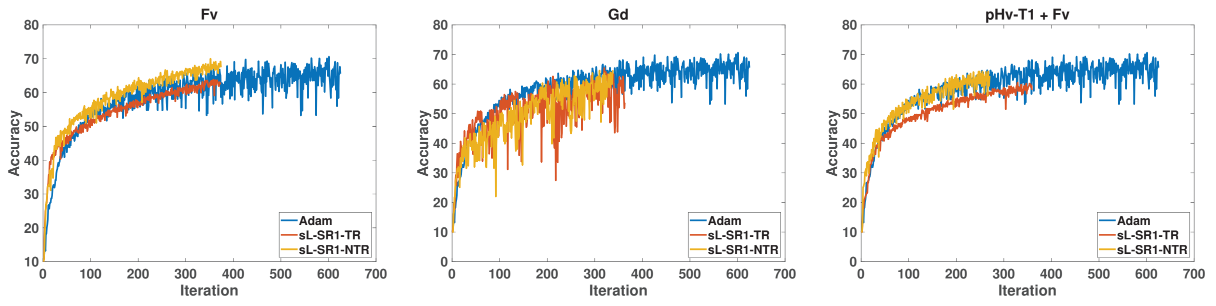

Fig. 4. Testing accuracy provided by sL-SR1-TR and sL-SR1-NTR for different curvature computing strategies **Fv** (left), **Gd** (middle) and **pHv-T1 + Fv** (right) and Adam for 90 minutes training time.

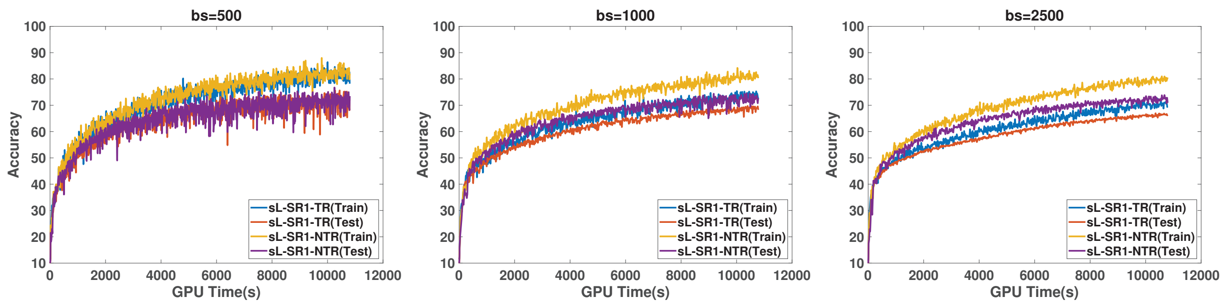

Fig. 5. Training and testing accuracy provided by sL-SR1-TR and sL-SR1-NTR using the Fv approach for different batch-sizes within 3 hours of training from scratch.

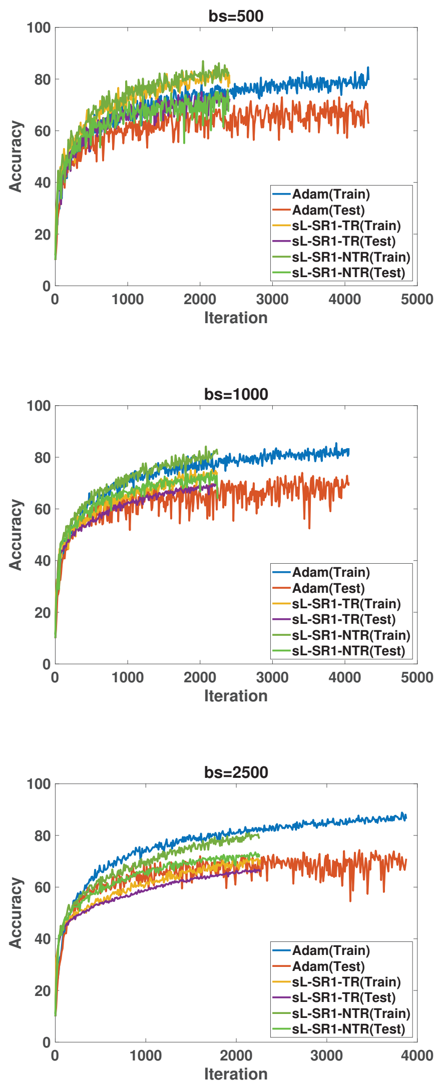

Fig. 6. Training and testing accuracy provided by sL-SR1-TR and sL-SR1-NTR using the Fv approach for different batch-sizes and the tuned Adam optimizer within 3 hours of training from scratch.

[10] N. Deng, Y. Xiao, and F. Zhou, "Nonmonotonic trust region algorithm," *Journal of optimization theory and applications*, vol. 76, no. 2, pp. 259–285, 1993.

[11] M. Ahookhosh, K. Amini, and M. R. Peyghami, "A nonmonotone trust-region line search method for large-scale unconstrained optimization," *Applied Mathematical Modelling*, vol. 36, no. 1, pp. 478–487, 2012.

[12] Z. Cui, B. Wu, and S. Qu, "Combining nonmonotone conic trust region and line search techniques for unconstrained optimization," *Journal of computational and applied mathematics*, vol. 235, no. 8, pp. 2432–2441, 2011.

[13] A. Kamandi and K. Amini, "A new nonmonotone adaptive trust region algorithm." *Applications of Mathematics*, vol. 67, pp. 233–250, 2022.

[14] N. Krejić and N. Krklec Jerinkić, "Nonmonotone line search methods with variable sample size," *Numerical Algorithms*, vol. 68, no. 4, pp. 711–739, 2015.

[15] A. R. Conn, N. I. Gould, and P. L. Toint, *Trust region methods*. SIAM, 2000.

[16] 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.

[17] J. Hoffman, D. A. Roberts, and S. Yaida, "Robust learning with jacobian regularization," *arXiv preprint arXiv:1908.02729*, 2019.

[18] W. Sun, "Nonmonotone trust region method for solving optimization problems," *Applied Mathematics and Computation*, vol. 156, no. 1, pp. 159–174, 2004.

[19] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[20] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, "A stochastic quasi-Newton method for large-scale optimization," *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 1008–1031, 2016.

[21] J. Martens, "New insights and perspectives on the natural gradient method," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5776–5851, 2020.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.