A non-monotone trust-region method with noisy oracles and additional sampling

Nataša Krejić[1], Nataša Krklec Jerinkić[1],
Ángeles Martínez[2], Mahsa Yousefi[2*]

[1]Department of Mathematics and Informatics, University of Novi Sad, Trg Dositeja Obradovića 4, Novi Sad, 21000, Serbia.
[2*]Department of Mathematics, Informatics, and Geosciences, University of Trieste, Via Alfonso Valerio 12/1, Trieste, 34127, Italy.

*Corresponding author: mahsa.yousefi@phd.units.it;
Contributing authors: natasak@uns.ac.rs; natasa.krklec@dmi.uns.ac.rs;
amartinez@units.it;

Abstract

In this work, we introduce a novel stochastic second-order method, within the framework of a non-monotone trust-region approach, for solving the unconstrained, nonlinear, and non-convex optimization problems arising in the training of deep neural networks. The proposed algorithm makes use of subsampling strategies that yield noisy approximations of the finite sum objective function and its gradient. We introduce an adaptive sample size strategy based on inexpensive additional sampling to control the resulting approximation error. Depending on the estimated progress of the algorithm, this can yield sample size scenarios ranging from mini-batch to full sample functions. We provide convergence analysis for all possible scenarios and show that the proposed method achieves almost sure convergence under standard assumptions for the trust-region framework. We report numerical experiments showing that the proposed algorithm outperforms its state-of-the-art counterpart in deep neural network training for image classification and regression tasks while requiring a significantly smaller number of gradient evaluations.

Keywords: Stochastic Optimization, Second-order Methods, Non-monotone Trust-Region, Quasi-Newton, Deep Neural Networks Training, Adaptive Sampling

MSC Classification: 90C30 , 90C06 , 90C53 , 90C90 , 65K05

1 Introduction

Deep learning (DL) as a leading technique of machine learning (ML) has attracted much attention and become one of the most popular directions of research. DL approaches have been applied to solve many large-scale problems in different fields by training deep neural networks (DNNs) over large available datasets. Let $\mathcal{N} = \{1, 2, \ldots, N\}$ be the index set of the training dataset $\{(x_i, y_i)\}_{i=1}^N$ with $N = |\mathcal{N}|$ sample pairs including input $x_i \in \mathbb{R}^d$ and target $y_i \in \mathbb{R}^C$. DL problems are often formulated as unconstrained optimization problems with an empirical risk function, in which a parametric function $\hat{h}(x_i; \cdot) : \mathbb{R}^d \longrightarrow \mathbb{R}^C$ is found such that the prediction errors are minimized. More precisely, we obtain the following problem

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

where $w \in \mathbb{R}^n$ is the vector of trainable parameters and $f_i(w) \triangleq L(y_i, \hat{h}(x_i; w))$ with a relevant loss function $L(\cdot)$ measuring the prediction error between the target y_i and the network's output $\hat{h}(x_i; w)$. The DL problem (1) is large-scale, highly nonlinear, and often non-convex, and thus it is not straightforward to apply traditional (deterministic) optimization algorithms like steepest descent or Newton-type methods. Recently, much effort has been devoted to the development of DL optimization algorithms. Popular DL optimization methods can be divided into two *general* categories, *first-order* methods using gradient information, e.g. steepest gradient descent, and *(higher-) second-order* methods using also curvature information, e.g. Newton methods [1]. On the other hand, since the full (training) sample size N in (1) is usually excessively large for a deterministic approach, these optimizers are further adapted to use subsampling strategies that aim to reduce computational costs. Subsampling strategies employ sample average approximations of the function and its gradient as follows

$$f_{\mathcal{N}_k}(w) = \frac{1}{N_k} \sum_{i \in \mathcal{N}_k} f_i(w), \quad \nabla f_{\mathcal{N}_k}(w) = \frac{1}{N_k} \sum_{i \in \mathcal{N}_k} \nabla f_i(w), \tag{2}$$

where $\mathcal{N}_k \subseteq \mathcal{N}$ represents a subset of the full (training) sample set at iteration k and N_k is the subsample size, i.e., $N_k = |\mathcal{N}_k|$.

In this work, we propose a second-order trust-region (TR) algorithm [2] adapted to the stochastic framework where the step and the candidate point for the next iterate are obtained using subsampled function values and subsampled gradients (2). The quadratic TR models are constructed by using Hessian approximations, without imposing a positive definiteness assumption, as the true Hessian in DL problems may not be positive definite due to their non-convex nature. Moreover, having in mind that we work with noisy approximations (2), imposing a strict decrease might be unnecessary. Thus, we employ a non-monotone trust-region (NTR) approach; see e.g. [3] or references therein. Unlike the classical TR, our decision on acceptance of the trial point is not based only on the agreement between the model and the approximate

objective function decrease, but on the independent subsampled function. This "control" function which is formed through *additional sampling*, similar to one proposed in [4] for the line search framework, also has a role in controlling the sample average approximation error by adaptively choosing the sample size. Depending on the estimated progress of our algorithm, this can yield sample size scenarios ranging from mini-batch to full sample functions. We provide convergence analysis for all possible scenarios and show that the proposed method achieves almost sure convergence under standard assumptions for the TR framework such as Lipschitz-continuous gradients and bounded Hessian approximations.

Literature review. Stochastic first-order methods such as stochastic gradient descent (SGD) method [5, 6], and its variance-reduced [7–10] and adaptive [11, 12] variants have been widely used in many ML and DL applications likely because of their proven efficiency in practice. However, due to the use of only gradient information, these methods come with several issues like, for instance, relatively slow convergence, high sensitivity to the hyper-parameters, stagnation at high training loss [13], difficulty in escaping saddle points, and suffering from ill-conditioning [14]. To cope with some of these issues, there are some attractive alternatives as well as their stochastic variants aimed at incorporating second-order information, e.g. Hessian-Free methods [15–20] which find an estimation of the Newton direction by (subsampled) Hessian-vector products without directly calculating the (subsampled) Hessian matrix, and limited memory Quasi-Newton methods which construct some approximations of the true Hessian only by using gradient information. Furthermore, algorithms based on Quasi-Newton methods have been the subject of many research efforts both in convex (see e.g. [21, 22] and references therein) and non-convex settings (see e.g. [23–25] and references therein), or [26, 27] where the advantage of modern computational architectures and parallelization for evaluating the full objective function and its derivatives is employed. In almost all these articles, the Quasi-Newton Hessian approximation used is BFGS or limited memory BFGS (L-BFGS) with positive definiteness property, which is often considered in the line-search framework except e.g. [28, 29]. A disadvantage of using BFGS may occur when it tries to approximate the true Hessian of a non-convex objective function in DL optimization. We refer to [30] as one of the earliest works in which the limited memory Quasi-Newton update SR1 (L-SR1) allowing for indefinite Hessian approximation was used in a trust-region framework with a periodical progressive overlap batching.

The potential usefulness of non-monotonicity may be traced back to [31] where a non-monotone line-search technique was proposed for the Newton method to relax some standard line-search conditions and to avoid slow convergence of a deterministic method. Similarly, the idea of non-monotonicity exploited for trust-region could be dated back to [32] and later e.g. [3, 33] for a general unconstrained minimization problem. This idea was also used for solving problems such as (1) in a stochastic setting; in [34], a class of algorithms was proposed that uses non-monotone line-search rules fitting a variable sample scheme at each iteration. In [35], a non-monotone trust-region algorithm using fixed-size subsampling batches was proposed for solving (1). Recently, in [36] a noise-tolerant TR algorithm has been proposed at the time of writing this paper, in which both the numerator and the denominator of the TR reduction ratio

are relaxed. The convergence analysis presented in [36] is based on the assumption that errors in function and gradient evaluations are bounded and do not diminish as the iterates approach the solution. In [37] the authors derive high probability complexity bounds for first- and second-order trust-region methods with noisy oracles, where the targeted vicinity of the solution depends on the quality of the stochastic estimates of the objective function and its gradient. They analyze modified trust-region algorithms that utilize stochastic zeroth-order oracles both in the bounded noise case and the independent subexponential noise case. Inspired by [35], in this work, we introduce a new second-order method in a subsampled non-monotone trust-region (NTR) approach, which works well with any Hessian approximation and employs an adaptive subsampling scheme. The foundation of our method differs from that of [36, 37]. Our method is based on an additional sampling strategy helping to control the non-martingale error due to the dependence between the non-monotone TR radius and the search direction. To the best of our knowledge, there are only a few approaches using additional sampling; see [4, 38, 39]. The rule that we apply in our method mainly corresponds to that presented in [4] where it is used in a line-search framework and plays a role in deciding whether to switch from the line-search to a predefined step size sequence or not. We adapted this strategy to the TR framework and used it to control the sample size in our method. It is worth pointing out that the additional sampling can be arbitrarily small, i.e. the sample size can even be 1, and hence, it does not increase the computational cost significantly. Adaptive sample size strategies can be also found in other works, for instance, a type of adaptive subsampling strategy was applied for the STORM algorithm [40, 41] which is also a second-order method in a standard TR framework. A different strategy using inexact restoration was proposed in [42] for a first-order standard TR approach. The variable size subsampling is not restricted to TR frameworks, see e.g. [25] where a progressive subsampling was considered for a line-search method.

Notation. Throughout this paper, vectors and matrices are respectively written in lowercase and uppercase letters unless otherwise specified in the context. The symbol $\triangleq$ is used to define a new variable. $\mathbb{N}$ and $\mathbb{R}^n$ denote the set of natural numbers and the real coordinate space of dimension n, respectively. The set of positive real numbers and non-negative integers are denoted by $\mathbb{R}_+$ and $\mathbb{N}_0$, respectively. Subscripts indicate the elements of a sequence. For a random variable X, the mathematical expectation of X and the conditional expectation of X given the σ-algebra $\mathcal{F}$ are respectively denoted by $\mathbb{E}(X)$ and $\mathbb{E}(X|\mathcal{F})$. The Euclidean vector norm and the corresponding matrix norm are denoted by $\|.\|$, while the cardinality of a set or the absolute value of a number is indicated by $|.|$. Finally, "a.s" abbreviates the expression "almost surely".

Outline of the paper. In Sect. 2, we describe the algorithm and all the necessary ingredients. In Sect. 3, we state the assumptions and provide an almost sure convergence analysis of the proposed method. Section 4 is devoted to the numerical evaluation of a specific version of the proposed algorithm that makes use of an L-SR1 update and a simple sampling rule; we make a comparison with the state-of-the-art method STORM [40, 41] to show the effectiveness of the proposed method for training DNNs in classification and regression tasks. A comparison with the popular first-order method ADAM [12] is also presented. Some conclusions are drawn in Sect. 5.

2 The algorithm

Within this section, we describe the proposed method called ASNTR (Adaptive Subsample Non-monotone Trust-Region). At iteration k, given the current iteration w_k, we form a quadratic model based on the subsampled function (2), with $g_k \triangleq \nabla f_{\mathcal{N}_k}(w_k)$ and an arbitrary Hessian approximation B_k satisfying

$$\|B_k\| \leq L, \tag{3}$$

for some $L > 0$ and solve the common TR subproblem to obtain the relevant direction

$$p_k \triangleq \arg\min_{p \in \mathbb{R}^n} Q_k(p) \triangleq \frac{1}{2} p^T B_k p + g_k^T p \quad \text{s.t.} \quad \|p\|_2 \leq \delta_k, \tag{4}$$

for some TR radius $\delta_k > 0$. We assume that at least some fraction of the Cauchy decrease is obtained, i.e., the direction satisfies

$$Q_k(p_k) \leq -\frac{c}{2} \|g_k\| \min\{\delta_k, \frac{\|g_k\|}{\|B_k\|}\}, \tag{5}$$

for some $c \in (0, 1]$. This is a standard assumption in TR and it is not restrictive even in the stochastic framework. In the classical deterministic TR approach, the trial point $w_t = w_k + p_k$ acceptance is based on the agreement between the decrease in the function and that of its quadratic model. However, since we are dealing with noisy approximations (2), we modify the acceptance strategy as follows. Motivated by the study in [35], we propose a non-monotone TR (NTR) framework instead of the standard TR one because we do not want to impose a strict decrease in the approximate function. Therefore, we define the relevant ratio as follows

$$\rho_{\mathcal{N}_k} \triangleq \frac{f_{\mathcal{N}_k}(w_t) - r_{\mathcal{N}_k}}{Q_k(p_k)}, \tag{6}$$

where

$$r_{\mathcal{N}_k} \triangleq f_{\mathcal{N}_k}(w_k) + t_k \delta_k, \quad t_k > 0, \tag{7}$$

and

$$\sum_{k=0}^{\infty} t_k \leq t < \infty. \tag{8}$$

The sequence $\{t_k\}$ is a summable sequence of positive numbers and one can define it in different ways to control the level of non-monotonicity in each iteration. Different choices of $\{t_k\}$ lead to different upper bounds of its sum which we denote by t. We allow $\mathcal{N}_k$ to be chosen arbitrarily in ASNTR. However, even if we impose uniform sampling with replacement to $\mathcal{N}$ such that it yields an unbiased estimator $f_{\mathcal{N}_k}(w_k)$ of the objective function at w_k, the dependence between $\mathcal{N}_k$ and w_t may produce a biased estimator $f_{\mathcal{N}_k}(w_t)$ of $f(w_t)$. This is because $\mathcal{N}_k$ directly affects the TR model $Q_k(p)$ through the approximate gradient $g_k = \nabla f_{\mathcal{N}_k}(w_k)$ and thus p_k also depends on the choice of $\mathcal{N}_k$. Thus, $w_t = w_k + p_k$ is also dependent on $\mathcal{N}_k$. To overcome this difficulty,

we apply an additional sampling strategy [4]. To this end, at every iteration at which $N_k < N$, we choose another independent subsample set represented by the index set $\mathcal{D}_k \subset \mathcal{N}$ of size $D_k = |\mathcal{D}_k| < N$ and calculate $f_{\mathcal{D}_k}(w_k), f_{\mathcal{D}_k}(w_t)$ and $\bar{g}_k \triangleq \nabla f_{\mathcal{D}_k}(w_k)$ (see lines 5-6 of the ASNTR algorithm). There are no theoretical requirements on the size of $\mathcal{D}_k$, and hence, the additional sampling might be done cheaply, i.e. with a modest number of additional samples. In fact, in our experiments, we set $D_k = 1$ for all k. Furthermore, in the spirit of TR, we define a linear model as $L_k(v) \triangleq v^T \bar{g}_k$, and consider another agreement measure defined as follows

$$\rho_{\mathcal{D}_k} \triangleq \frac{f_{\mathcal{D}_k}(w_t) - r_{\mathcal{D}_k}}{L_k(-\bar{g}_k)}, \tag{9}$$

where

$$r_{\mathcal{D}_k} \triangleq f_{\mathcal{D}_k}(w_k) + \delta_k \tilde{t}_k, \quad \tilde{t}_k > 0, \tag{10}$$

and

$$\sum_{k=0}^{\infty} \tilde{t}_k \leq \tilde{t} < \infty. \tag{11}$$

We assume that $\{\tilde{t}_k\}$ is a summable sequence of positive numbers and that $\tilde{t}$ is an upper bound of the sum of $\{\tilde{t}_k\}$. Therefore, $\tilde{t}$ is controllable through the choice of $\{\tilde{t}_k\}$. Notice that choosing a greater $\tilde{t}_k$ yields more chances for the trial point w_t to be accepted. The denominator in (9) is the linear model computed along the negative gradient $\bar{g}_k$ and thus it is negative. Therefore, the condition $\rho_{\mathcal{D}_k} \geq \nu$ corresponds to Armijo-like condition for the function $f_{\mathcal{D}_k}$, similar to one in [4] i.e., $f_{\mathcal{D}_k}(w_t) \leq f_{\mathcal{D}_k}(w_k) - \nu\|\nabla f_{\mathcal{D}_k}(w_k)\|^2 + \delta_k \tilde{t}_k$. If $N_k < N$, the trial point is accepted only if both $\rho_{\mathcal{N}_k}$ and $\rho_{\mathcal{D}_k}$ are bounded away from zero; otherwise, if the full sample is used, the decision is made by $\rho_{\mathcal{N}_k}$ solely as in deterministic NTR (see lines 23-35 of the ASNTR algorithm). The rationale behind this is the following: we double-checked that the trial point obtained employing $f_{\mathcal{N}_k}$ is acceptable also with respect to another approximation of the objective function $f_{\mathcal{D}_k}$. Notice that $\mathcal{D}_k$ is chosen with replacement, uniformly and randomly from $\mathcal{N}$, independently from the choice of $\mathcal{N}_k$, and after the trial point w_t is already determined. Therefore, conditionally on σ-algebra generated by all the previous choices of $\mathcal{N}_j, j = 1, ..., k$ and $\mathcal{D}_j, j = 1, ..., k-1$, the approximation $f_{\mathcal{D}_k}(w_t)$ is an unbiased estimator of $f(w_t)$.

Another role of $\rho_{\mathcal{D}_k}$ is to control the sample size. If the obtained trial point w_t yields an uncontrolled increase in $f_{\mathcal{D}_k}$ in a sense that $\rho_{\mathcal{D}_k} < \nu$, i.e., $f_{\mathcal{D}_k}(w_t) > f_{\mathcal{D}_k}(w_k) - \nu\|\nabla f_{\mathcal{D}_k}(w_k)\|^2 + \delta_k \tilde{t}_k$, we conclude that we need a better approximation of the objective function and we increase the sample size N_k by choosing a new sample set $\mathcal{N}_k$ for the next iteration. Roughly speaking, an uncontrolled increase in $f_{\mathcal{D}_k}$ is possible if the approximate function $f_{\mathcal{D}_k}$ is very different from $f_{\mathcal{N}_k}$ given that the search direction is computed for $f_{\mathcal{N}_k}$. The sample can also be increased if we are too close to a stationary point of the approximate function $f_{\mathcal{N}_k}$. This is stated in line 7 of ASNTR, where h represents an SAA error estimate given by

$$h(N_k) \triangleq \frac{N - N_k}{N}.$$

Algorithm 1 ASNTR

Initialization: Choose $\mathcal{N}_0 \subseteq \mathcal{N}$. Set $k = 0$, $\{t_k\} \in \mathbb{R}_+^\infty$ satisfying (8), $\{\tilde{t}_k\} \in \mathbb{R}_+^\infty$ satisfying (11), δ_0 and $\delta_{max} \in (0, \infty)$, $\epsilon \in [0, \frac{1}{2})$, $\nu \in (0, 1/4)$, $0 < \tau_1 \leq 0.5 < \tau_2 < 1 < \tau_3$, $0 < \eta < \eta_2 \leq 3/4$, and $\eta_1 \in (\eta, \eta_2)$.
Given $f_{\mathcal{N}_k}(w_k)$, g_k and B_k satisfying (3), solve (4) for p_k such that (5) holds, and define the trial iterate $w_t = w_k + p_k$.
Given $f_{\mathcal{N}_k}(w_t)$, compute $\rho_{\mathcal{N}_k}$ using (6).
if $N_k < N$ **then**
    Choose $\mathcal{D}_k \subset \mathcal{N}$ randomly and uniformly, with replacement.
    Given $f_{\mathcal{D}_k}(w_k)$, $\nabla f_{\mathcal{D}_k}(w_k)$, and $f_{\mathcal{D}_k}(w_t)$, compute $\rho_{\mathcal{D}_k}$ using (9).
    if $\|g_k\| < \epsilon h(N_k)$ **then**
        Increase N_k to N_{k+1} and choose $\mathcal{N}_{k+1}$.
    else
        if $\rho_{\mathcal{D}_k} < \nu$ **then**
            Increase N_k to N_{k+1} and choose $\mathcal{N}_{k+1}$.
        else
            if $\rho_{\mathcal{N}_k} < \eta$ **then**
                Set $N_{k+1} = N_k$ and $\mathcal{N}_{k+1} = \mathcal{N}_k$.
            else
                Set $N_{k+1} = N_k$ and choose $\mathcal{N}_{k+1}$.
            end if
        end if
    end if
else
    $N_{k+1} = N$
end if
if $N_k < N$ **then**
    if $\rho_{\mathcal{N}_k} \geq \eta$ and $\rho_{\mathcal{D}_k} \geq \nu$ **then**
        $w_{k+1} = w_t$.
    else
        $w_{k+1} = w_k$.
    end if
else
    if $\rho_{\mathcal{N}_k} \geq \eta$ **then**
        $w_{k+1} = w_t$.
    else
        $w_{k+1} = w_k$.
    end if
end if
if $\rho_{\mathcal{N}_k} < \eta_1$, **then**
    $\delta_{k+1} = \tau_1 \delta_k$
else if $\rho_{\mathcal{N}_k} > \eta_2$ and $\|p_k\| \geq \tau_2 \delta_k$, **then**
    $\delta_{k+1} = min\{\tau_3 \delta_k, \delta_{max}\}$,
else
    $\delta_{k+1} = \delta_k$.
end if
if Some stopping conditions hold **then**
    Stop training
else
    Set $k = k + 1$ and go to step 2.
end if

The algorithm can also keep the same sample size (see lines 14 and 16 of the ASNTR algorithm). Keeping the same sample $\mathcal{N}_k$ in line 14 corresponds to the case where the trial point is acceptable with respect to $f_{\mathcal{D}_k}$, but we do not have a decrease in $f_{\mathcal{N}_k}$. In that case, the (non-monotone) TR radius (δ_k) is decreased (see line 37 of ASNTR). Otherwise, we allow the algorithm in line 16 to choose a new sample of the current size and exploit some new data points. The strategy for updating the sample size is described in lines 7-19 of the ASNTR algorithm.

Notice that the sample size cannot be decreased, and if the full sample is reached it is kept until the end of the procedure. Moreover, it should be noted that ASNTR provides complete freedom in terms of the increment in the sample size as well as the choice of samples $\mathcal{N}_k$. This leaves enough space for different sampling strategies within the algorithm. As we already mentioned, mostly depending on the problem, ASNTR can result in a mini-batch strategy, but it can also yield an increasing sample size procedure.

The TR radius is updated within lines 36-42 of ASNTR. We follow a common update strategy for TR approaches. It is completely based on $f_{\mathcal{N}_k}$ since it is related to the error of the quadratic model, and not to the SAA error which we control by additional sampling. Thus, if the $\rho_{\mathcal{N}_k}$ is small we decrease the trust-region size (see lines 36-37 of ASNTR). Otherwise, the trust-region is either enlarged or kept the same (see lines 38-42 of ASNTR). Notice that the additional condition that relates the norm of the search direction p_k and the current trust-region size δ_k does not play any role in the theoretical analysis but it is important in practical implementation. We need some predetermined hyper-parameters for ASNTR, which are established in the algorithm's initial line according to ones outlined in relevant references (e.g. [1, 30]), and to meet the assumptions underlying the convergence analysis.

3 Convergence analysis

We make the following standard assumption for the TR framework needed to prove the a.s. convergence result of ASNTR.

Assumption 1. *The functions $f_i, i = 1, ..., N$ are bounded from below and twice continuously-differentiable with L-Lipschitz-continuous gradients.*

First, we prove an important lemma that will help us prove the main result, the a.s. convergence of ASNTR. There are two possible scenarios depending on the size of the sample sequence: 1) mini-batch scenario - where the subsampling is employed in each iteration; 2) deterministic scenario - where the full sample is reached eventually. We start the analysis by considering the first, mini-batch scenario. In Lemma 1, we show that in this case there holds $\rho_{\mathcal{D}_k} \geq \nu$ for any realization of $\mathcal{D}_k$ and all k sufficiently large.

Let us denote by $\mathcal{D}_k^+$ the subset of all possible outcomes of $\mathcal{D}_k$ at iteration k that satisfy $\rho_{\mathcal{D}_k} \geq \nu$, i.e.,

$$\mathcal{D}_k^+ = \{\mathcal{D}_k \subset \mathcal{N} \mid f_{\mathcal{D}_k}(w_t) \leq f_{\mathcal{D}_k}(w_k) - \nu\|\nabla f_{\mathcal{D}_k}(w_k)\|^2 + \delta_k \tilde{t}_k\}. \tag{12}$$

Notice that if $\mathcal{D}_k \in \mathcal{D}_k^+$ and $\rho_{\mathcal{N}_k} \geq \eta$ then $w_{k+1} = w_t$. On the other hand, if $\mathcal{D}_k \in \mathcal{D}_k^+$ and $\rho_{\mathcal{N}_k} < \eta$ then $w_{k+1} = w_k$. Finally, if $\mathcal{D}_k \in \mathcal{D}_k^-$, where

$$\mathcal{D}_k^- = \{\mathcal{D}_k \subset \mathcal{N} \mid f_{\mathcal{D}_k}(w_t) > f_{\mathcal{D}_k}(w_k) - \nu\|\nabla f_{\mathcal{D}_k}(w_k)\|^2 + \delta_k \tilde{t}_k\}, \tag{13}$$

we have again $w_{k+1} = w_k$. Notice that $\mathcal{D}_k^- = \emptyset$ if and only if $\rho_{\mathcal{D}_k} \geq \nu$ for all possible realizations of $\mathcal{D}_k$.

Lemma 1. *Suppose that Assumption 1 holds. If $N_k < N$ for all $k \in \mathbb{N}$, then a.s. there exists $k_1 \in \mathbb{N}$ such that $\rho_{\mathcal{D}_k} \geq \nu$ for all $k \geq k_1$ and for all possible realizations $\mathcal{D}_k$.*

Proof. Assume that $N_k < N$ for all $k \in \mathbb{N}$. So, there exists some $\bar{N} < N$ and $k_0 \in \mathbb{N}$ such that $N_k = \bar{N}$ for all $k \geq k_0$. Now, let us use the notation as in (12)-(13) and assume that there exists an infinite subsequence of iterations $K \subseteq \mathbb{N}$, such that $\mathcal{D}_k^- \neq \emptyset$ for all $k \in K$. Since $\mathcal{D}_k$ is chosen randomly and uniformly with replacement from the finite set $\mathcal{N}$ and $D_k \leq N - 1$, we know that each $\mathcal{D}_k$ has only finitely many possible outcomes. More precisely, we conclude that $S(D_k) \leq \bar{S} := (2N-2)!/((N-1)!)^2$ where the upper bound comes from the combinatorics of unordered sampling with replacement[1]. Therefore, there exists $\underline{p} \in (0,1)$ such that $P(\mathcal{D}_k \in \mathcal{D}_k^-) \geq \underline{p}$, i.e., $P(\mathcal{D}_k \in \mathcal{D}_k^+) \leq 1 - \underline{p} =: \bar{p} < 1$ for all $k \in K$. So,

$$P(\mathcal{D}_k \in \mathcal{D}_k^+, k \in K) \leq \prod_{k \in K} \bar{p} = 0.$$

Therefore, a.s. there exists an iteration $k \geq k_1$ such that $\rho_{\mathcal{D}_k} < \nu$ and the sample size is increased, which is in contradiction with $N_k = \bar{N}$ for all k large enough. This completes the proof.

□

Next, we prove that the mini-batch scenario implies a non-monotone type of decrease for all k large enough.

Lemma 2. *Suppose that Assumption 1 is satisfied and $N_k < N$ for all $k \in \mathbb{N}$. Then a.s.*

$$f(w_t) \leq f(w_k) - \nu\|\nabla f(w_k)\|^2 + \delta_k \tilde{t}_k,$$

holds for all $k \geq k_1$, where k_1 is as in Lemma 1.

Proof. Lemma 1 implies that $\rho_{\mathcal{D}_k} \geq \nu$, i.e.,

$$f_{\mathcal{D}_k}(w_t) \leq f_{\mathcal{D}_k}(w_k) - \nu\|\nabla f_{\mathcal{D}_k}(w_k)\|^2 + \delta_k \tilde{t}_k, \tag{14}$$

for all $k \geq k_1$ and for all possible realizations of $\mathcal{D}_k$ a.s.. Since the sampling of $\mathcal{D}_k$ is with replacement, notice that this further yields

$$f_i(w_t) \leq f_i(w_k) - \nu\|\nabla f_i(w_k)\|^2 + \delta_k \tilde{t}_k, \tag{15}$$

[1] When $D_k = N - 1$, which is its maximal size, we choose $N - 1$ element from the set of N numbers with replacement. Then, there are $\binom{N-1+N-1}{N-1}$ different unordered subsamples, i.e., $(2N-2)!/((N-1)!)^2$ possible choices for $\mathcal{D}_k$.

for all $i \in \mathcal{N}$ and all $k \geq k_1$ a.s.. Indeed, if there exists $i \in \mathcal{N}$ such that (15) is violated, then (14) is violated for at least one possible realization of $\mathcal{D}_k$, namely, for $\mathcal{D}_k = \{i, i, ..., i\}$. Thus, a.s. for all $k \geq k_1$ there holds

$$\begin{aligned} f(w_t) = \frac{1}{N}\sum_{i=1}^{N} f_i(w_t) &\leq \frac{1}{N}\sum_{i=1}^{N}(f_i(w_k) - \nu\|\nabla f_i(w_k)\|^2 + \delta_k\tilde{t}_k) \\ &= f(w_k) - \nu\frac{1}{N}\sum_{i=1}^{N}\|\nabla f_i(w_k)\|^2 + \delta_k\tilde{t}_k \\ &\leq f(w_k) - \nu\|\nabla f(w_k)\|^2 + \delta_k\tilde{t}_k, \end{aligned} \tag{16}$$

where the last inequality comes from the fact that $\|\cdot\|^2$ is convex and therefore

$$\|\nabla f(w_k)\|^2 = \|\frac{1}{N}\sum_{i=1}^{N}\nabla f_i(w_k)\|^2 \leq \frac{1}{N}\sum_{i=1}^{N}\|\nabla f_i(w_k)\|^2.$$

□

Now, let us prove that starting from some finite but random iteration, the sequence of iterates generated by ASNTR belongs to a level set of the original objective function f a.s.. This level set is also random since it depends on the sample path[2].

Lemma 3. *Suppose that Assumption 1 holds. Then a.s.*

$$f(w_{\tilde{k}+k}) \leq f(w_{\tilde{k}}) + \delta_{max}\max\{t, \tilde{t}\}, \quad k = 0, 1, \ldots$$

where $\tilde{k}$ is some finite random number, t and $\tilde{t}$ correspond to those in (8) and (11) respectively.

Proof. Let us consider both scenarios, mini-batch and deterministic separately. In the mini-batch case, when $N_k < N$ for each k, Lemma 2 implies that a.s.

$$f(w_t) \leq f(w_k) - \nu\|\nabla f(w_k)\|^2 + \delta_k\tilde{t}_k \leq f(w_k) + \delta_k\tilde{t}_k,$$

holds for all $k \geq k_1$, where k_1 is as in Lemma 1. Since $w_{k+1} = w_t$ or $w_{k+1} = w_k$, there a.s.

$$f(w_{k+1}) \leq f(w_k) + \delta_k\tilde{t}_k,$$

holds for all $k \geq k_1$. So, the summability of $\{\tilde{t}_k\}$ (11) and the fact that $\delta_k \leq \delta_{max}$ for all k together imply that the statement of this lemma holds with $\tilde{k} = k_1$.

Assume now that N is achieved at some finite iteration, i.e., there exists a finite iteration k_2 such that $N_k = N$ for all $k \geq k_2$. Thus, the trial point for all $k \geq k_2$

[2]The sample path or, more precisely, the realization of the sequence of iterates generated by the algorithm, $\{w_k\}_{k\in\mathbb{N}}$, depends on realization of stochastic factors within the algorithm. In our case, stochasticity comes from the random choices of subsamples $\mathcal{N}_k$ and $\mathcal{D}_k$ in each particular iteration k.

is accepted if and only if $\rho_{\mathcal{N}_k} \geq \eta$. This implies that for all $k \geq k_2$ we either have $f(w_{k+1}) = f(w_k)$ or

$$f(w_{k+1}) \leq f(w_k) + \delta_k t_k - \frac{\eta c}{2}\|g_k\| \min\{\delta_k, \frac{\|g_k\|}{\|B_k\|}\} \leq f(w_k) + \delta_{max} t_k, \tag{17}$$

where $\|g_k\| = \|\nabla f(w_k)\|$ in this scenario. Thus, using the summability of $\{t_k\}$ in (8) we obtain the result with $\tilde{k} = k_2$. □

In order to prove the main convergence result, we assume that the expected value of $f(w_{\tilde{k}})$ is uniformly bounded.

Assumption 2. *There exists a constant $C > 0$ such that $\mathbb{E}(|f(w_{\tilde{k}})|) \leq C$, where $\tilde{k}$ is as in Lemma 3 and the expectation is taken over all possible sample paths.*

This assumption, together with the result of Lemma 3, implies that a.s. the sequence $\{f(w_k)\}_{k \geq \tilde{k}}$ is uniformly bounded in expectation. Moreover, the Assumption 2 also implies that $\mathbb{E}(|f(w_{\tilde{k}})| \mid A) \leq C_1$, where A represents the subset of all possible outcomes (sample paths) such that the full sample is reached eventually and C_1 is some positive constant. To see this, observe that the following holds

$$P(A)\mathbb{E}(|f(w_{\tilde{k}})| \mid A) \leq P(A)\mathbb{E}(|f(w_{\tilde{k}})| \mid A) + P(\bar{A})\mathbb{E}(|f(w_{\tilde{k}})| \mid \bar{A}) = \mathbb{E}(|f(w_{\tilde{k}})|) \leq C,$$

where $\bar{A}$ represents all possible sample paths that remain in the mini-batch scenario. Thus, we obtain

$$\mathbb{E}_A(|f(w_{\tilde{k}})|) := \mathbb{E}(|f(w_{\tilde{k}})| \mid A) \leq C/P(A) =: C_1.$$

Similarly, there exists a constant $C_2 > 0$ such that

$$\mathbb{E}_{\bar{A}}(|f(w_{\tilde{k}})|) := \mathbb{E}(|f(w_{\tilde{k}})| \mid \bar{A}) \leq C/P(\bar{A}) =: C_2.$$

Notice that Assumption 2 is satisfied under the assumption of bounded iterates ($\{w_k\}_{k\in\mathbb{N}} \subset \bar{G}$, where $\bar{G}$ is a compact subset of $\mathbb{R}^n$) which is fairly common in a stochastic optimization framework.

Theorem 1. *Suppose that Assumption 1 and Assumption 2 hold. Then the sequence $\{w_k\}$ generated by ASNTR algorithm satisfies*

$$\liminf_{k\to\infty} \|\nabla f(w_k)\| = 0 \quad a.s.$$

Proof. Let us consider two different scenarios, namely, $N_k = N$ for all k large enough, and $N_k < N$ for all k. Let us start with the first one in which $N_k = N$ for all $k \geq \tilde{k}$, where $\tilde{k}$ is random but finite. In this case, ASNTR reduces to the non-monotone *deterministic* trust-region algorithm applied on f. By L-Lipschitz continuity of ∇f, we obtain

$$|f(w_k + p_k) - f(w_k) - \nabla^T f(w_k) p_k| \leq \frac{L}{2}\|p_k\|^2. \tag{18}$$

Now, let us denote $\rho_k \triangleq \rho_{\mathcal{N}_k}$ and assume that $\|\nabla f(w_k)\| \geq \varepsilon > 0$ for all $k \geq \tilde{k}$. Then,

$$
\begin{aligned}
|\rho_k - 1| &= \frac{|f(w_k + p_k) - f(w_k) - \delta_k t_k - \nabla^T f(w_k) p_k - 0.5 p_k^T B_k p_k|}{|Q_k(p_k)|} \\
&\leq \frac{0.5L\|p_k\|^2 + \delta_k t_k + 0.5L\|p_k\|^2}{0.5c\|g_k\| \min\{\delta_k, \frac{\|g_k\|}{\|B_k\|}\}} \\
&\leq \frac{L\|p_k\|^2 + \delta_k t_k}{0.5c\varepsilon \min\{\delta_k, \frac{\varepsilon}{L}\}},
\end{aligned} \tag{19}
$$

where $\|g_k\| = \|\nabla f(w_k)\|$ in this scenario. Let us define $\tilde{\delta} = \frac{\varepsilon c}{20L}$. Without loss of generality, given that the sequence $\{t_k\}$ is summable and hence $t_k \to 0$ (see (8)), we can assume that $t_k \leq L\tilde{\delta}$ for all $k \geq \tilde{k}$. Observe now the iterations k for $k > \tilde{k}$. If $\delta_k \leq \tilde{\delta}$, then $\delta_{k+1} \geq \delta_k$. This is due to the fact that

$$
|\rho_k - 1| \leq \frac{L\delta_k^2 + \delta_k t_k}{0.5c\varepsilon\delta_k} \leq \frac{2L}{0.5c\varepsilon}\tilde{\delta} = \frac{2L}{0.5c\varepsilon}\frac{\varepsilon c}{20L} = \frac{1}{5} < \frac{1}{4}, \tag{20}
$$

i.e., $\rho_k > \frac{3}{4} \geq \eta_2$ which implies that the NTR radius is not decreased; see lines 36-42 in ASNTR. Further, there exists $\hat{\delta} > 0$ such that $\delta_k \geq \hat{\delta}$ for all $k \geq \tilde{k}$. Moreover, for all $k \geq \tilde{k}$, the assumption $\rho_k < \eta$, where $\eta < \eta_1$, would yield a contradiction since it would imply $\lim_{k\to\infty} \delta_k = 0$. Therefore, there must exist an infinite set $K \subseteq \mathbb{N}$ as $K = \{k \geq \tilde{k} \,|\, \rho_k \geq \eta\}$. For all $k \in K$, we have

$$
f(w_{k+1}) \leq f(w_k) + \delta_k t_k - \frac{c}{8}\|g_k\| \min\{\delta_k, \frac{\|g_k\|}{\|B_k\|}\} \leq f(w_k) + \delta_{max} t_k - \frac{c}{8}\varepsilon \min\{\hat{\delta}, \frac{\varepsilon}{L}\}.
$$

Now, let $\bar{c} \triangleq \frac{c}{8}\varepsilon \min\{\hat{\delta}, \frac{\varepsilon}{L}\}$. Since t_k tends to zero, we have $\delta_{max} t_k \leq \frac{\bar{c}}{2}$ for all k large enough. Without loss of generality, we can say that this holds for all $k \in K$, rewriting $K = \{k_j\}_{j\in\mathbb{N}_0}$, we have

$$
f(w_{k_j+1}) \leq f(w_{k_j}) - \frac{\bar{c}}{2}.
$$

Since $w_{k+1} = w_k$ for all $k \geq \tilde{k}$ and $k \notin K$, i.e. when $\rho_k < \eta$, we obtain

$$
f(w_{k_{j+1}}) \leq f(w_{k_j}) - \frac{\bar{c}}{2}.
$$

Thus we obtain for all $j \in \mathbb{N}_0$

$$
f(w_{k_j}) \leq f(w_{k_0}) - j\frac{\bar{c}}{2} \leq f(w_{\tilde{k}}) + \delta_{max} \max\{t, \tilde{t}\} - j\frac{\bar{c}}{2}, \tag{21}
$$

where the last inequality is due to Lemma 3. Furthermore, applying the expectation to both sides of (21) and using Assumption 2 we get

$$
\mathbb{E}_A(f(w_{k_j})) \leq C_1 + \delta_{max} \max\{t, \tilde{t}\} - j\frac{\bar{c}}{2}. \tag{22}
$$

Letting j tend to infinity in (22), we obtain $\lim_{j\to\infty} \mathbb{E}_A(f(w_{k_j})) = -\infty$, which is in contradiction with the assumption of f being bounded from below. Therefore, $\|\nabla f(w_k)\| \geq \varepsilon > 0$ for all $k \geq \tilde{k}$ can not be true, so we conclude that

$$\liminf_{k\to\infty} \|\nabla f(w_k)\| = 0.$$

Now let us consider the mini-batch scenario, i.e., $N_k < N$ for all k, i.e., the sample size is increased only finitely many times. According to Lemma 1 and lines 7-8 of the algorithm, the currently considered scenario implies the existence of a finite $\tilde{k}_1$ such that

$$N_k = \tilde{N}, \quad \|g_k\| \geq \epsilon h(N_k) \triangleq \epsilon_{\tilde{N}} > 0 \quad \text{and} \quad \rho_{\mathcal{D}_k} \geq \nu, \tag{23}$$

for all $k \geq \tilde{k}_1$ and some $\tilde{N} < N$ a.s. Now, let us prove that there exists an infinite subset of iterations $\tilde{K} \subseteq \mathbb{N}$ such that $\rho_k \geq \eta$ for all $k \in \tilde{K}$, i.e., the trial point w_t is accepted infinitely many times. Assume a contrary, i.e., there exists some finite $\tilde{k}_2$ such that $\rho_k < \eta$ for all $k \geq \tilde{k}_2$. Since $\eta < \eta_1$, this further implies that $\lim_{k\to\infty} \delta_k = 0$; see lines 36 and 37 in Algorithm 1. Moreover, line 13 of Algorithm 1 implies that the sample does not change, meaning that there exists $\tilde{\mathcal{N}} \subset \mathcal{N}$ such that $\mathcal{N}_k = \tilde{\mathcal{N}}$ for all $k \geq \tilde{k}_3 \triangleq \max\{\tilde{k}_1, \tilde{k}_2\}$. By L-Lipschitz continuity of $\nabla f_{\tilde{\mathcal{N}}}$, we obtain

$$|f_{\tilde{\mathcal{N}}}(w_k + p_k) - f_{\tilde{\mathcal{N}}}(w_k) - \nabla^T f_{\tilde{\mathcal{N}}}(w_k) p_k| \leq \frac{L}{2}\|p_k\|^2. \tag{24}$$

For every $k \geq \tilde{k}_3$, then we have

$$\begin{aligned} |\rho_k - 1| &= \frac{|f_{\tilde{\mathcal{N}}}(w_k + p_k) - f_{\tilde{\mathcal{N}}}(w_k) - \delta_k t_k - \nabla^T f_{\tilde{\mathcal{N}}}(w_k) p_k - 0.5 p_k^T B_k p_k|}{|Q_k(p_k)|} \\ &\leq \frac{0.5L\|p_k\|^2 + \delta_k t_k + 0.5L\|p_k\|^2}{0.5c\|g_k\| \min\{\delta_k, \frac{\|g_k\|}{\|B_k\|}\}} \\ &\leq \frac{L\delta_k^2 + \delta_k t_k}{0.5c\epsilon_{\tilde{N}} \min\{\delta_k, \frac{\varepsilon}{L}\}}. \end{aligned} \tag{25}$$

Since $\lim_{k\to\infty} \delta_k = 0$, there exists $\tilde{k}_4$ such that for all $k \geq \tilde{k}_4$ we obtain

$$|\rho_k - 1| \leq \frac{L\delta_k^2 + \delta_k t_k}{0.5c\epsilon_{\tilde{N}}\delta_k} = \frac{L\delta_k + t_k}{0.5c\epsilon_{\tilde{N}}}.$$

Due to the fact that t_k tends to zero, we obtain that $\lim_{k\to\infty} \rho_k = 1$ which is in contradiction with $\rho_k < \eta < 1/4$. Thus, we conclude that there must exist $\tilde{K} \subseteq \mathbb{N}$ such that $\rho_k \geq \eta$ for all $k \in \tilde{K}$. Let us define $K_1 \triangleq \tilde{K} \cap \{\tilde{k}_1, \tilde{k}_1 + 1, \ldots\}$. Notice that we have $\rho_{\mathcal{D}_k} \geq \nu$ and $\rho_k \geq \eta$ for all $k \in K_1$. Thus, due to Lemma 2, a.s. the following holds for all $k \in K_1$

$$\begin{aligned} f(w_{k+1}) = f(w_t) &\leq f(w_k) - \nu\|\nabla f(w_k)\|^2 + \delta_k \tilde{t}_k, \\ &\leq f(w_k) - \nu\|\nabla f(w_k)\|^2 + \delta_{max}\tilde{t}_k. \end{aligned} \tag{26}$$

Notice that $w_{k+1} = w_k$ in all other iterations $k \geq \tilde{k}_1$ and $k \notin K_1$. Rewriting $K_1 = \{k_j\}_{j\in\mathbb{N}_0}$, for all $j \in \mathbb{N}_0$, we obtain

$$f(w_{k_{j+1}}) = f(w_{k_j+1}) \leq f(w_{k_j}) - \nu\|\nabla f(w_{k_j})\|^2 + \delta_{max}\tilde{t}_{k_j}.$$

Then, due to the summability of the sequeence $\{\tilde{t}_k\}$ in (11), and Lemma 3, the following holds a.s. for any $s \in \mathbb{N}_0$

$$\begin{aligned} f(w_{k_{s+1}}) &\leq f(w_{k_0}) - \nu\sum_{j=0}^{s}\|\nabla f(w_{k_j})\|^2 + \delta_{max}\tilde{t} \\ &\leq f(w_{\tilde{k}}) + \delta_{max}\max\{t,\tilde{t}\} - \nu\sum_{j=0}^{s}\|\nabla f(w_{k_j})\|^2 + \delta_{max}\tilde{t}. \end{aligned} \tag{27}$$

Now, applying the expectation to both sides of the second inequality in (27), using Assumption 2 which implies $\mathbb{E}_{\bar{A}}(|f(w_{\tilde{k}})|) \leq C_2$, and the boundedness of f from below, letting s tend to infinity yields

$$\sum_{j=0}^{\infty}\mathbb{E}_{\bar{A}}(\|\nabla f(w_{k_j})\|^2) < \infty.$$

Moreover, the following holds as a consequence of an extended version of Markov's inequality with the specific choice of the nonnegative function $\Phi(y) = y^2$ nondecreasing on $y \geq 0$: for any $\epsilon > 0$

$$P_{\bar{A}}(\|\nabla f(w_{k_j})\| \geq \epsilon) \leq \frac{\mathbb{E}_{\bar{A}}(\Phi(\|\nabla f(w_{k_j})\|))}{\Phi(\epsilon)} = \frac{\mathbb{E}_{\bar{A}}(\|\nabla f(w_{k_j})\|^2)}{\epsilon^2}.$$

Therefore, we have

$$\sum_{j=0}^{\infty}P_{\bar{A}}(\|\nabla f(w_{k_j})\| \geq \epsilon) < \infty.$$

Finally, Borel-Cantelli Lemma implies that (in the current scenario) almost surely $\lim_{j\to\infty}\|\nabla f(w_{k_j})\| = 0$. Combining all together, the result follows as in both scenarios we have at least

$$\liminf_{k\to\infty}\|\nabla f(w_k)\| = 0 \quad \text{a.s.}$$

□

Finally, we analyze the convergence rate of the proposed algorithm for a restricted class of problems stated in the following theorem.

Theorem 2. *Suppose that the assumptions of Theorem 1 hold and that $\nu < L/2$. Moreover, suppose that f is m-strongly convex and the sequence $\{\tilde{t}_k\}$ converges to zero R-linearly. If $N_k < N$ for all $k \in \mathbb{N}$, then the sequence $\{w_k\}_{k\in K_1}$ with K_1 as in (26) converges to the unique minimizer w^* of f R-linearly in the mean squared sense.*

Proof. Considering the mini-batch scenario in the proof of the previous theorem, we obtain

$$f(w_{k_{j+1}}) \leq f(w_{k_j}) - \nu \|\nabla f(w_{k_j})\|^2 + \delta_{max} \tilde{t}_{k_j},$$

for all $j \in \mathbb{N}$ a.s.. Moreover, the strong convexity of f implies

$$f(w_{k_{j+1}}) - f(w^*) \leq f(w_{k_j}) - f(w^*) - \nu \frac{2}{L} (f(w_{k_j}) - f(w^*)) + \delta_{max} \tilde{t}_{k_j},$$

for all $j \in \mathbb{N}$ a.s.. Now, applying the expectation and defining $e_j := \mathbb{E}_{\bar{A}}(f(w_{k_j}) - f(w^*))$, from the previous inequality we obtain

$$e_{j+1} \leq \theta e_j + \varepsilon_j,$$

where $\varepsilon_j := \delta_{max} \tilde{t}_{k_j}$ and $\theta := 1 - 2\nu/L \in (0, 1)$ according to the assumption on ν that is $\nu \in (0, 1/4)$. Since the previous inequality holds for each j, we conclude via induction that for all $j \in \mathbb{N}$ there holds

$$e_j \leq \theta^j e_0 + s_j,$$

where $s_j := \delta_{max} \sum_{i=1}^{j} \theta^{i-1} \varepsilon_{j-i}$. Notice that $\{\varepsilon_j\}$ converges to zero R-linearly according to the assumption on $\tilde{t}_k$. This further implies that s_j converges to zero R-linearly (see Lemma 4.2 in [34]). Thus, we conclude that $\{e_j\}$ converges to zero R-linearly. Finally, since the strong convexity also implies

$$\frac{m}{2} \mathbb{E}_{\bar{A}}(\|w_{k_j} - w^*\|^2) \leq e_j,$$

we obtain the result. □

4 Numerical experiments

In this section, we provide some experimental results to make a comparison between ASNTR and STORM (as Algorithm 5 in [41]). We examine the performance of both algorithms for training DNNs in two types of problems: (i) Regression with synthetic `DIGITS` dataset and (ii) Classification with `MNIST` and `CIFAR10` datasets. [3]

We also provide additional results to give more insights into the behavior of the ASNTR algorithm, especially concerning the sampling strategy. All experiments were conducted with the MATLAB DL toolbox on an Ubuntu 20.04.4 LTS (64-bit) Linux server VMware with 20GB memory using a VGPU NVIDIA A100D-20C.

4.1 Experimental configuration

All three image datasets are divided into training and testing sets including N and $\hat{N}$ samples, respectively, and whose pixels in the range $[0, 255]$ are divided by 255 so that the pixel intensity range is bounded in $[0, 1]$ (zero-one rescaling). To define the

[3] Available online: https://www.kaggle.com/datasets/hojjatk/mnist-dataset (`MNIST`) https://www.mathworks.com/help/deeplearning/ug/data-sets-for-deep-learning.html (`DIGITS` and `CIFAR10`)

initialized networks and training loops of both algorithms, we have applied `dlarray` and `dlnetwork` MATLAB objects.[4] The networks' parameters in $w \in \mathbb{R}^n$ are initialized by the Glorot (Xavier) initializer [43] and zeros for respectively weights and biases of convolutional layers as well as ones and zeros respectively for scale and offset variables of batch normalization layers. Table 1 describes the hyper-parameters applied in both algorithms.

Since ASNTR and STORM allow the use of any Hessian approximations, the underlying quadratic model can be constructed by exploiting a Quasi-Newton update for B_k. Quasi-Newton updates aim at constructing Hessian approximations using only gradient information and thus incorporating second-order information without computing and storing the true Hessian matrix. We have considered the **L**imited **M**emory **S**ymmetric **R**ank **one** (L-SR1) update formula to generate B_k rather than other widely used alternatives such as limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS). The L-SR1 updates might better navigate the pathological saddle points present in the non-convex optimization found in DL applications. Given $B_0 = \gamma_k I$ at iteration k, and curvature pair $(s_k, y_k) = (w_t - w_k, g_t - g_k)$ where $g_k \triangleq \nabla f_{\mathcal{N}_k}(w_k)$ and $g_t \triangleq \nabla f_{\mathcal{N}_k}(w_t)$ provided that $(y_k - B_k s_k)^T s_k \neq 0$, the SR1 updates are obtained as follows

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}, \quad k = 0, 1, \ldots. \tag{28}$$

Using two limited memory storage matrices S_k and Y_k with at most $l \ll n$ columns for storing the recent l pairs $\{s_j, y_j\} \quad j \in \{1, \ldots, l\}$, the compact form of L-SR1 updates can be written as

$$B_k = B_0 + \Psi_k M_k \Psi_k^T, \quad k = 1, 2, , \ldots, \tag{29}$$

where

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

with matrices L_k and D_k which respectively are the strictly lower triangular part and the diagonal part of $S_k^T Y_k$; see [1]. Regarding the selection of the variable multiplier γ_k in B_0, we refer to the initialization strategy proposed in [30]. Given the quadratic model $Q_k(p)$ using L-SR1 in ASNTR and STORM, we have used an efficient algorithm called OBS [44], exploiting the structure of the L-SR1 matrix (29) and the Sherman-Morrison-Woodbury formula for inversions to obtain p_k.

We have randomly (without replacement) chosen the index subset $\mathcal{N}_k \subseteq \{1, 2, \ldots, N\}$ to generate a mini-batch of size N_k for computing the required quantities, i.e., subsampled functions and gradients. Given $N_0 = d + 1$ where d is the dimension of a single input sample $x_i \in \mathbb{R}^d$, we have adopted the linearly increased sampling rule that $N_k = \min(N, \max(100k + N_0, \lceil \frac{1}{\delta_k{}^2} \rceil))$ for STORM as in [41] while we have exploited the simple following sampling rule for ASNTR when the increase is necessary

$$N_{k+1} = \lceil 1.01 N_k \rceil, \tag{30}$$

otherwise $N_{k+1} = N_k$. Using the non-monotone TR framework in our algorithm, we set $t_k = \frac{C_1}{(k+1)^{1.1}}$ and $\tilde{t}_k = \frac{C_2}{(k+1)^{1.1}}$ for some $C_1, C_2 > 0$ in (7) and (10), respectively. We have also selected $\mathcal{D}_k$ with cardinality 1 at every single iteration in ASNTR.

[4] A MATLAB-based tutorial on implementing custom loops for training a deep neural network is available here: http://doi.org/10.13140/RG.2.2.33008.94720/2

In our implementations, each algorithm was run with 5 different initial random seeds. The criteria of both algorithms' performance (accuracy and loss) are compared against the number of gradient calls (N_g) during the training phase. The algorithms were terminated when N_g reached the determined budget of the gradient evaluations ($N_g^{\max}$). Each network is trained by ASNTR and STORM; then the trained network is used for the prediction of the testing dataset. Notice that the training loss and accuracy are the subsampled function's value and the number of correct predictions in percentage with respect to the given mini-batch.

Table 1: Hyper-parameters

STORM
$\delta_0 = 1,\ \delta_{max} = 10,\ l = 30,\ \eta_1 = 10^{-4},\ \eta_2 = 10^{-3},\ \gamma = 2$
ASNTR
$\delta_0 = 1,\ \delta_{max} = 10,\ l = 30,\ \eta = \nu = 10^{-4},\ \eta_1 = 0.1,\ \eta_2 = 0.75,\ \tau_1 = 0.5,\ \tau_2 = 0.8, \tau_3 = 2$

4.2 Classification problems

To solve an image classification problem for images with unknown classes/labels, we need to seek an optimal classification model by using a C-class training dataset $\{(x_i, y_i)_{i=1}^N\}$ with an image $x_i \in \mathbb{R}^d$ and its one-hot encoded label $y_i \in \mathbb{R}^C$. To this end, the generic DL problem (1) is minimized, where its single loss function $f_i = L(y_i, h(x_i; .))$ is *softmax cross-entropy* as follows

$$f_i(w) = -\sum_{k=1}^{C} (y_i)_k \log(h(x_i; w))_k, \quad i = 1, \ldots, N. \tag{31}$$

In (31), the output $h(x_i; w)$ is a prediction provided by a DNN whose last layer includes the softmax function. For this classification task, we have considered two types of networks, the `LeNet-like` network with a shallow structure inspired by `LeNet-5` [48], and the modern residual network `ResNet-20` [46] with a deep structure. See Table 2 for the network's architectures. We have also considered the two most popular benchmarks; the `MNIST` dataset [49] with 7×10^4 samples of handwritten gray-scale digit images of 28×28 pixels and the `CIFAR10` dataset [50] with 6×10^4 RGB images of 32×32 pixels, both in 10 categories. Every single image of `MNIST` and `CIFAR10` datasets is defined as a 3-D numeric array $x_i \in \mathbb{R}^d$ where $d = 28 \times 28 \times 1$ and $d = 32 \times 32 \times 3$, respectively. Moreover, every single label y_i must be converted into a one-hot encoded label as $y_i \in \mathbb{R}^C$, where $C = 10$. In both datasets, $\hat{N} = 10^4$ images are set aside as testing sets, and the remaining N images are set as training sets. Besides the zero-one rescaling, we implemented z-score normalization to have zero mean and unit variance. Precisely, all N training images undergo normalization by subtracting the mean (an $h \times w \times c$ array) and dividing by the standard deviation (an $h \times w \times c$ array) of training images

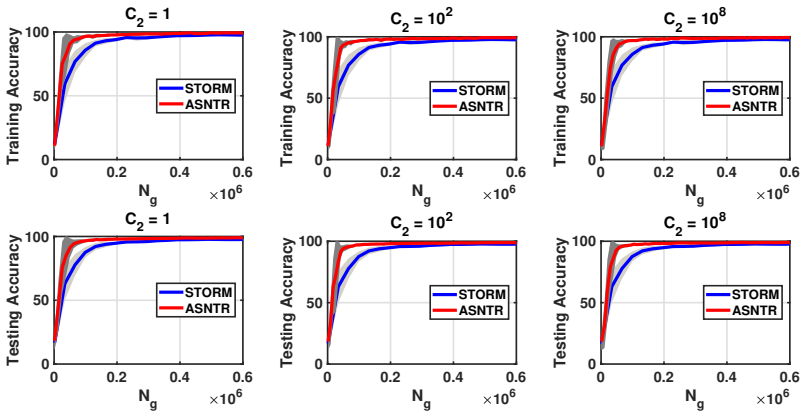


Fig. 1: The accuracy variations of STORM and ASNTR on `MNIST`.

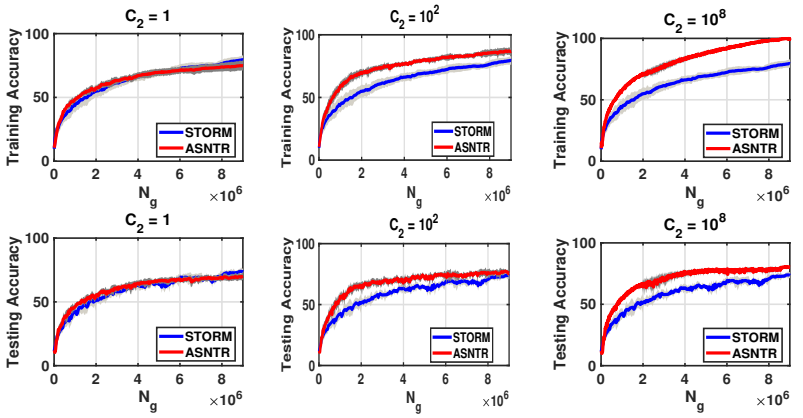


Fig. 2: The accuracy variations of STORM and ASNTR on `CIFAR10`.

as an array of size $h \times w \times c \times N$. Here, h, w, and c denote the height, width, and number of channels of the images, respectively, while N represents the total number of images. Test data are also normalized using the same parameters as in the training data.[5]

Figures 1 to 4 show the variations, the mean and standard error obtained from 5 separate runs, of the aforementioned measures for both train and test datasets of ASNTR with $C_2 = 1, 10^2, 10^8$ in $\rho_{\mathcal{D}_k}$, and $C_1 = 1$ in $\rho_{\mathcal{N}_k}$ within fixed budgets of gradient evaluations $N_g^{\max} = 6 \times 10^5$ for `MNIST` and $N_g^{\max} = 9 \times 10^6$ for `CIFAR10`. These figures demonstrate that ASNTR achieves higher training and testing accuracy than STORM in all considered values of C_2 except in Fig. 2 with $C_2 = 1$ by which ASNTR can be comparable with STORM. Nevertheless, ASNTR is capable of achieving higher

[5]Mathematically, we have already denoted the $i-$th image of dimension d as $x_i \in \mathbb{R}^d$ where $d = h \times w \times c$.

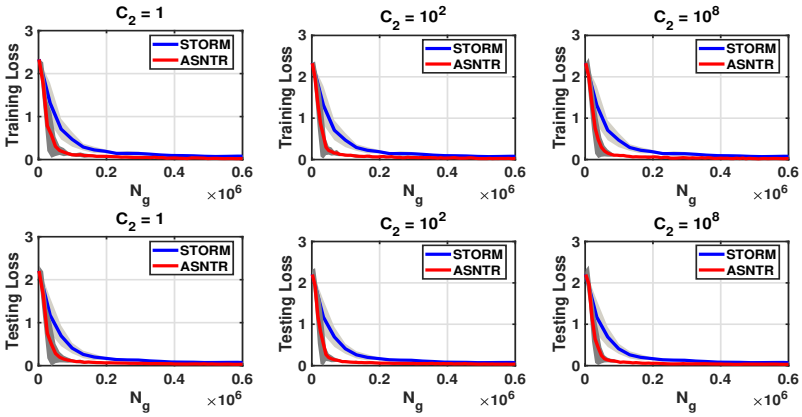


Fig. 3: The loss variation of STORM and ASNTR on `MNIST`.

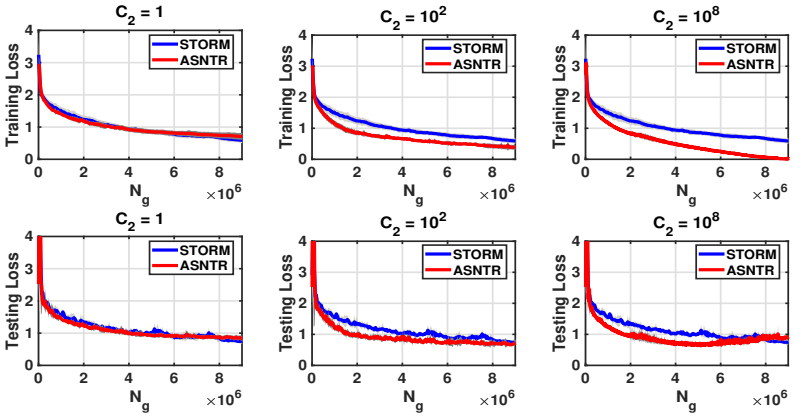


Fig. 4: The loss variation of STORM and ASNTR on `CIFAR10`.

accuracy (lower loss) with fewer N_g in comparison with STORM. Moreover, these figures indicate the robustness of ASNTR for larger values of C_2 meaning higher rates of non-monotonicity.

4.3 Regression problem

This example shows how to fit a regression model using a neural network to be able to predict the angles of rotation of handwritten digits which is useful for optical character recognition. To find an optimal regression model, the generic problem (1) is minimized, where $f_i = L(y_i, h(x_i; .))$ with a predicted output $h(x_i; w)$ is *half-mean-squared error* as follows

$$f_i(w) = -\frac{1}{2}(y_i - h(x_i; w))^2, \quad i = 1, \ldots, N. \tag{32}$$

In this regression example, we have considered a convolutional neural network (CNN) with an architecture named `CNN-Rn` as indicated in Table 2. We have also used the `DIGITS` dataset containing 10×10^3 synthetic images with 28×28 pixels as well as their angles (in degrees) by which each image is rotated. Every single image is defined as a 3-D numeric array $x_i \in \mathbb{R}^d$ where $d = 28 \times 28 \times 1$. Moreover, the response y_i (the rotation angle in degrees) is approximately uniformly distributed between $-45°$ and $45°$. Each training and testing dataset has the same number of images ($N = \hat{N} = 5 \times 10^3$). Besides zero-one rescaling, we have also applied zero-center normalization to have zero mean. The problem (1) with single loss function (32) is solved for $w \in \mathbb{R}^n$ where $n = 16,881$ using `CNN-Rn` for `DIGITS`. In this experiment, the accuracy is the number of predictions in percentage within an acceptable error margin (threshold) that we have set to be 10 degrees.

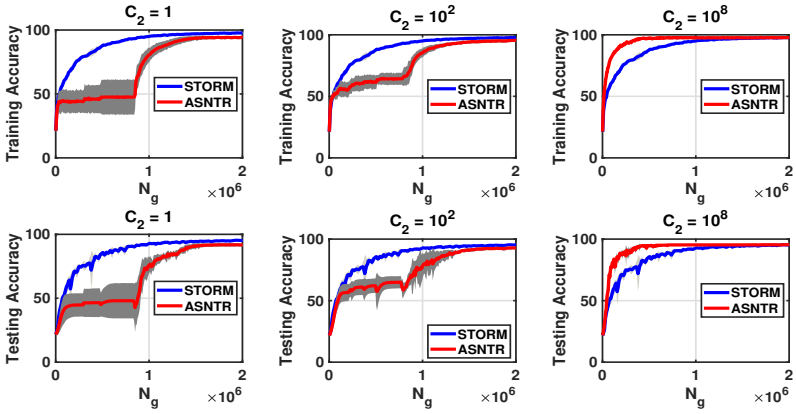


Fig. 5: The accuracy variations of STORM and ASNTR on `DIGITS`.

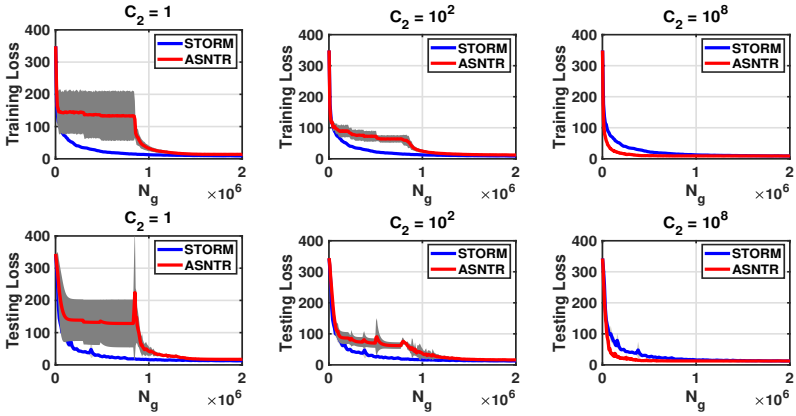


Fig. 6: The loss variation of STORM and ASNTR on `DIGITS`.

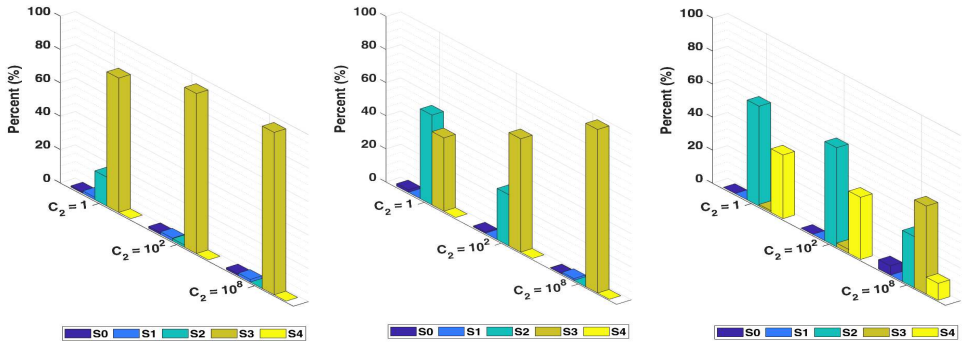
Figures 5 and 6 show training accuracy and loss, and testing accuracy and loss variations of ASNTR for `DIGITS` dataset with three values of C_2 within a fixed budget of gradient evaluations $N_g^{\max} = 2 \times 10^6$. These figures also illuminate how resilient ASNTR is for the highest value of C_2. Despite several challenges in the early stages of the training phase with $C_2 = 1$ and $C_2 = 10^2$, ASNTR can overcome them and achieve accuracy levels comparable to those of the STORM algorithm.

4.4 Additional results

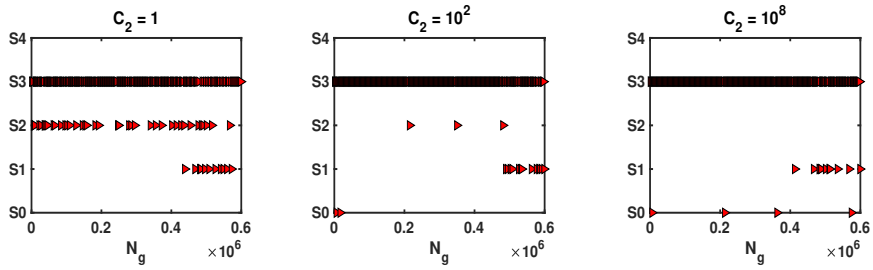
We present two additional figures (Figs. 7 and 8) containing further details regarding our proposed algorithm, ASNTR, with $C_2 = 1, 10^2, 10^8$ in $\rho_{\mathcal{D}_k}$ and $C_1 = 1$ in $\rho_{\mathcal{N}_k}$. More specifically, these results aim to give useful insights concerning the sampling behavior of ASNTR. Let S1 and S2 indicate the iterations of ASNTR at which steps 7 and 10, respectively, are executed using the increasing sampling rule (30). When the cardinality of the sample set is not changed, i.e., $N_{k+1} = N_k$, let S3 and S0 show the iterations at which new samples through step 15 and current samples through step 13 are selected. We also define variable S4 representing the iteration of ASNTR at which all available samples ($N_k = N$) are needed for computing the required quantities.

Figure 7a shows the (average) contributions of the aforementioned sampling types in ASNTR running with five different initial random generators for `MNIST`, `CIFAR10`, and `DIGITS` with respectively predetermined $N_g^{\max} = 0.6 \times 10^6$, 9×10^6 and 2×10^6; however, considering only a specific initial seed, e.g. `rng(42)`, Figs. 7b, 7c and 7d indicate when/where each of these sampling types is utilized in ASNTR. Obviously, the contribution rate of $S3$ is influenced by $S2$, where ASNTR has to increase the batch size if $\rho_{\mathcal{D}_k} < \nu$. In fact, the larger C_2 in ρ_{D_k} results in the higher portion of $S3$ and the smaller portion of $S2$. Moreover, using a large value of C_2, there is no need to increase the current batch size unless the current iterate approaches a stationary point of the current approximate function. This, in turn, leads to increasing the portion of $S1$, which usually happens at the end of the training stage. In addition, we have also found that the value of C_2 in $\tilde{t}_k$ plays an important role in the robustness of our proposed algorithm as observed in Figs. 1 to 6; in other words, the higher the C_2, the more robust ASNTR is. These mentioned observations, more specifically, can be followed for every single dataset as below:

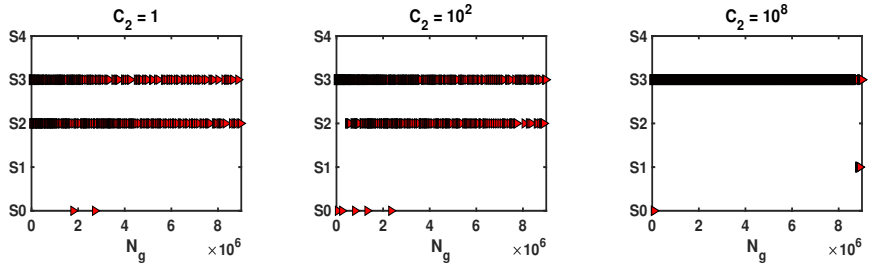
- `MNIST`: according to Fig. 7a, the portion of the sampling type $S3$ is larger than the others. This means many new batches without an increase in the batch size are applied in ASNTR during training; i.e., the proposed algorithm can train a network with fewer samples, and thus fewer gradient evaluations. Nevertheless, ASNTR with different values of C_2 in $\rho_{\mathcal{D}_k}$ increases the size of the mini-batches in some of its iterations; see the portions of $S1$ and $S2$ in Fig. 7a or their scatters in Fig. 7b. We should note that the sampling type $S1$ occurs almost at the end of the training phase where the algorithm tends to be close to a stationary point of the current approximate function; Fig. 7b shows this fact.
- `CIFAR10`: according to Fig. 7a, the portion of the sampling type $S3$ is still large. Unlike `MNIST`, the sampling type $S1$ rarely occurs during the training phase. On the other hand, a large portion of the increase of the sample size through $S2$ may com-



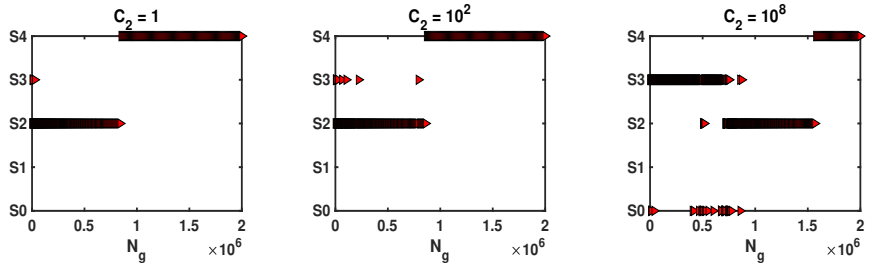
(a) Sampling behaviour of ASNTR on `MNIST` (left), `CIFAR10` (middle), `DIGITS` (right)



(b) Sampling behaviour with initial `rng(42)` on `MNIST`



(c) Sampling behaviour with initial `rng(42)` on `CIFAR10`



(d) Sampling behaviour with initial `rng(42)` on `DIGITS`

Fig. 7: Tracking subsampling in ASNTR.

pensate for the lack of sufficiently accurate functions and gradients required in ASNTR. These points are also illustrated in Fig. 7c, which shows how ASNTR successfully trained the `ResNet-20` model without frequently enlarging the sample sizes. For both the `MNIST` and `CIFAR10` problems, $S3$ as the predominant type corresponds to $C_2 = 10^8$.

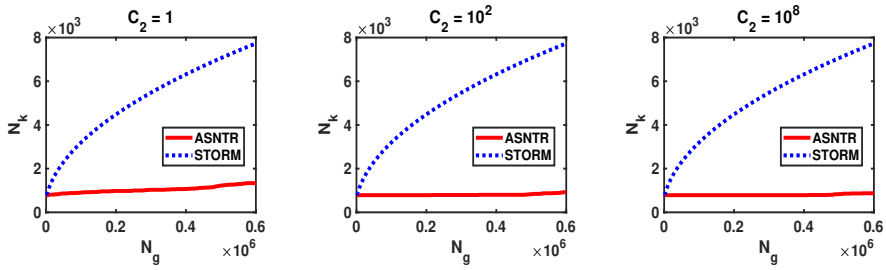
- `DIGITS`: according to Fig. 7a, we observe that the main sampling types are $S2$ and $S4$. As the portion of $S2$ increases, the portion of $S3$ decreases and the highest portion of $S3$ corresponds to the largest value of C_2. This pattern is similar to what is seen in the `MNIST` and `CIFAR10` datasets. However, in the case of `DIGITS`, the portion of $S4$ is higher. This higher portion of $S4$ in `DIGITS` may be attributed to the smaller number of samples in this dataset ($N = 5000$), which causes ASNTR to quickly encompass all the samples after a few iterations. Notably, the sampling type $S4$ occurs towards the end of the training phase, as shown in Fig. 7d.

Figure 8 compares the progression of batch size growth in both ASNTR and STORM. In contrast to the STORM algorithm, ASNTR increases the batch size only when necessary, which can reduce the computational costs of gradient evaluations. This is considered a significant advantage of ASNTR over STORM. However, according to this figure, the proposed algorithm needs fewer samples than STORM during the initial phase of the training task, but it requires more samples toward the end. Nevertheless, we should notice that the increase in batch size that happened at the end of the training phase is determined either by $S1$ or by $S4$ (see Figs. 7b and 7d). In our experiments, we have observed that ASNTR does not use very large $N_g^{\max}$, as it typically achieves the required training accuracy.

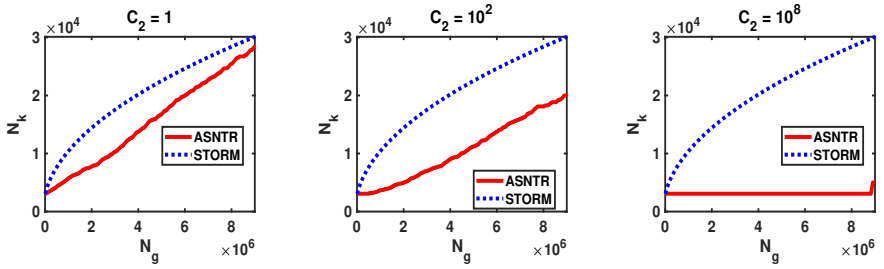
4.5 Comparison with ADAM

We have compared the proposed stochastic second-order method (ASNTR) with a popular efficient first-order optimizer, i.e., Adaptive Moment Estimation (ADAM) [12] used in DL. We have implemented ADAM using MATLAB built-in function `adamupdate` in customized training loops. It's widely recognized that this optimizer is highly sensitive to the value of its hyper-parameters including the learning rate, α_k, the gradient decay factor, β_1, the squared gradient decay factor, β_2, a small constant to prevent division by zero, ϵ, and batch size. Users should be aware of the hyper-parameter choices and invest time in tuning them using techniques such as grid search. In our experiments, we set $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$, respectively, and focus our tuning effort on learning rate and batch size. The specified choices for tuning the learning rate were $\alpha_k = \alpha$ with α taking values from the set $\{10^{-4}, 10^{-3}, 10^{-2}\}$, and the corresponding values for batch size were $N_k = bs$ where bs varied within $\{128, 256, 784\}$ for both `MNIST` and `DIGITS`, and within $\{128, 256, 3072\}$ for `CIFAR10`.

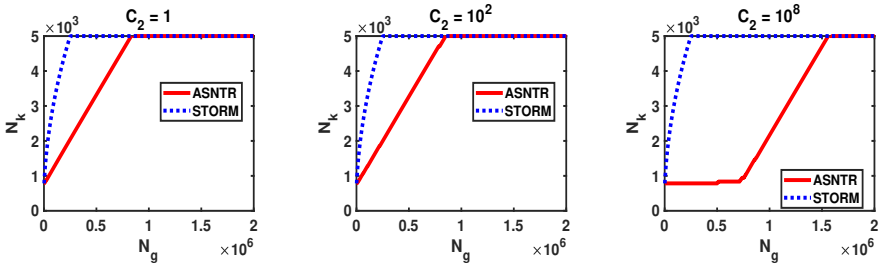
The best hyper-parameters were those that yielded the highest testing accuracy within a fixed budget of gradient evaluations. In all experiments, the optimal performance with Adam was consistently achieved using $\alpha = 10^{-3}$ and $bs = 128$. These settings were employed with ADAM in Figs. 9,10 and 11 (first rows) for comparison purposes against ASNTR with $C_2 = 10^8$ demonstrated in Figs. 1 to 6. As these figures show, the tuned ADAM could produce the highest accuracy and lowest loss



(a) MNIST ($N = 6 \times 10^4$)



(b) `CIFAR10` ($N = 5 \times 10^4$)



(c) `DIGITS` ($N = 5 \times 10^3$)

Fig. 8: Batch size progress with initial `rng(42)`.

with fewer gradient evaluations in comparison with ASNTR. The common observation of rapid initial improvement achieved by ADAM, followed by a drastic slowdown, is well understood in practice for some first-order methods (see, e.g., [13]). First-order methods such as ADAM are computationally less expensive per iteration compared to second-order methods such as ASNTR, as they only involve the computation of one gradient compared to two gradients. Moreover, as already mentioned the initial sample size for ASNTR was set as $N_0 = d + 1$ where $d = 784$ for both `MNIST` and `DIGITS`

and $d = 3072$ for `CIFAR10`. By initially employing such large batch sizes against an obtained optimal batch size for ADAM, i.e., $N_k = 128$, there are only a small number of updated iterates (w_k) during training with both second-order methods within the fixed budget of gradient evaluations. Therefore, allowing ASNTR to train networks within a larger number of gradient evaluations helps it to eventually achieve a higher level of accuracy while this cannot help ADAM. Note that we have performed this analysis between the tuned ADAM ($N_k = 128$, and $\alpha_k = 10^{-3}$) and ASNTR, where tuning the hyper-parameters of ADAM incurs significant time costs. When ADAM is used with suboptimal hyper-parameters, its sensitivity becomes evident, as illustrated in Figs. 9,10 and 11 (second rows) where batch size $N_k = d + 1$ and different learning rates are considered. As these figures show, for a challenging problem such as `CIFAR10` classification, ADAM may not achieve a lower loss value than ASNTR, even when employing an optimal learning rate ($\alpha = 10^{-2}$). Neither for the other challenging problem `DIGITS` ADAM could produce higher testing Accuracy than ASNTR. All the results shown in the three figures were obtained using the same seed for the random number generator (`rng(42)`). Moreover, due to awkward oscillations in Loss and Accuracy obtained by ADAM in `CIFAR10` and `DIGITS` classification problems, we have imposed a filter using `movmean` MATLAB built-in function (moving mean with a window of length c) in Figs. 10 and 11; in our experiments $c = 30$.

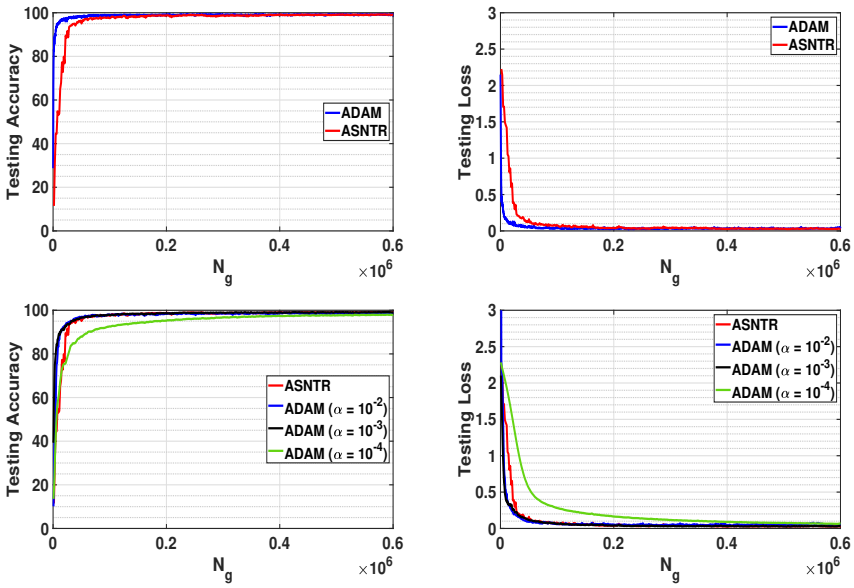


Fig. 9: Comparison of ASNTR vs. tuned ADAM (first row), and vs. untuned ADAM (second row) on `MNIST` for training `LeNet-like` with `rng(42)`.

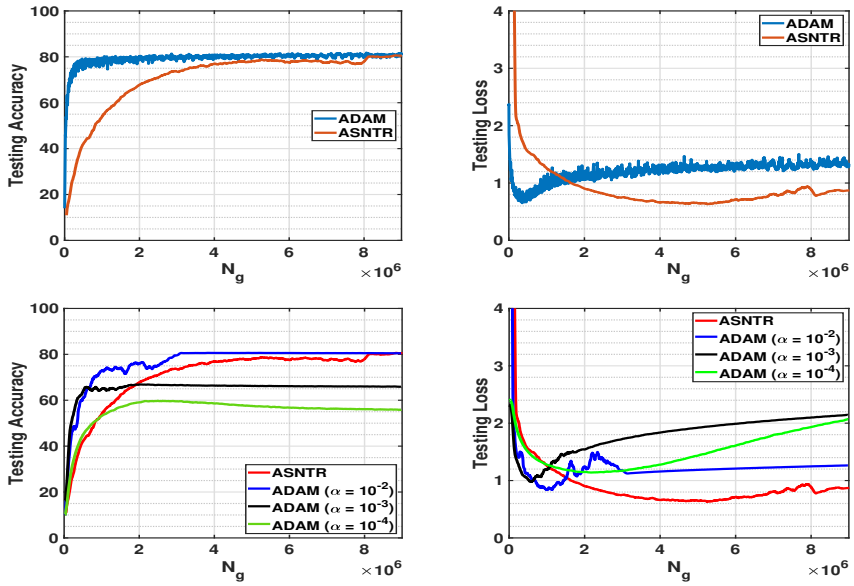


Fig. 10: Comparison of ASNTR vs. tuned ADAM (first row), and vs. untuned ADAM (second row) on `CIFAR10` for training `ResNet-20` with `rng(42)`.

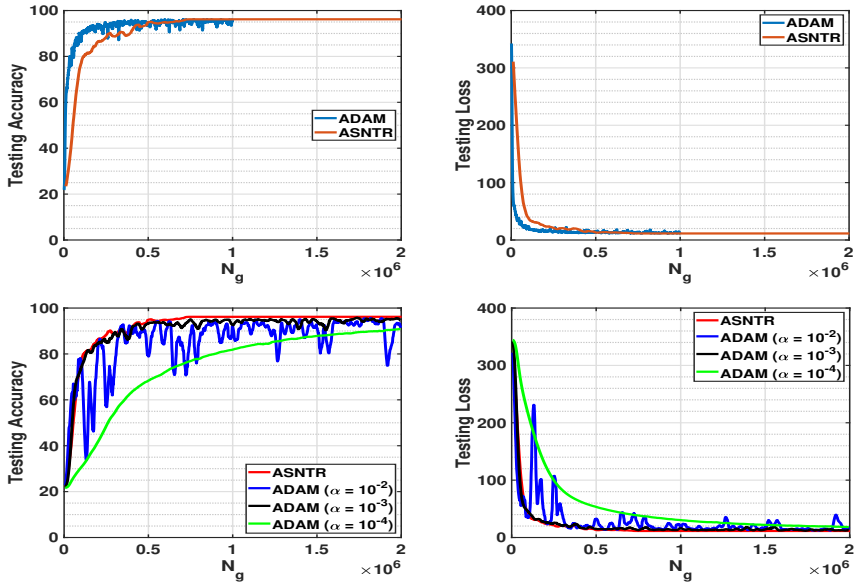


Fig. 11: Comparison of ASNTR vs. tuned ADAM (first row), and vs. untuned ADAM (second row) on `DIGITS` for training `CNN-Rn` with `rng(42)`.

5 Conclusion

In this work, we have presented ASNTR, a second-order non-monotone trust-region method that employs an adaptive subsampling strategy. We have incorporated additional sampling into the TR framework to control the noise and overcome issues in the convergence analysis coming from biased estimators. Depending on the estimated progress of the algorithm, this can yield different scenarios ranging from mini-batch to full sample functions. We provide convergence analysis for all possible scenarios and show that the proposed method achieves almost sure convergence under standard assumptions for the TR framework. The experiments in deep neural network training for both image classification and regression show the efficiency of the proposed method. In comparison to the state-of-the-art second-order method STORM, ASNTR achieves higher testing accuracy with a fixed budget of gradient evaluations. However, our experiments show that the popular first-order method, tuned ADAM using its optimal hyper-parameters, can produce higher accuracy than ASNTR with fixed and fewer gradient computations if one does not count the effort needed for tuning the parameters. As expected, ASNTR is more robust and performs better than ADAM with suboptimal parameters. Future work on ASNTR could include more specified sample size updates and Hessian approximation strategies.

Acknowledgments. We are grateful to the two anonymous referees whose constructive comments helped us to improve the paper.

The work of NK and NKJ was supported by the Science Fund of the Republic of Serbia, Grant no. 7359, Project LASCADO. AM and MY gratefully acknowledge the support of the INdAM-GNCS Project CUP_E53C22001930001. The work of AM was carried out within the PNRR research activities of the consortium iNEST (Interconnected North-East Innovation Ecosystem) funded by the European Union Next-GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, ECS_00000043). This manuscript reflects only the Authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

Competing interests declarations

The authors have no relevant financial or non-financial interests to disclose.

Data availability statements

The datasets utilized in this research, `DIGITS`, `MNIST`, and `CIFAR-10`, are publicly accessible and commonly employed benchmarks in the field of Machine Learning and Deep Learning, see https://www.mathworks.com/help/deeplearning/ug/data-sets-for-deep-learning.html, https://www.kaggle.com/datasets/hojjatk/mnist-dataset and [49, 50].

References

[1] Nocedal, J., Wright, S.: Numerical Optimization. Springer, New York, NY (2006). https://doi.org/10.1007/978-0-387-40065-5

[2] Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust-region Methods. SIAM, Philadelphia, PA (2000). https://doi.org/10.1137/1.9780898719857

[3] Ahookhosh, M., Amini, K., Peyghami, M.R.: A non-monotone trust-region line search method for large-scale unconstrained optimization. Applied Mathematical Modelling **36**(1), 478–487 (2012) https://doi.org/10.1016/j.apm.2011.07.021

[4] Di Serafino, D., Krejić, N., Krklec Jerinkić, N., Viola, M.: LSOS: line-search second-order stochastic optimization methods for nonconvex finite sums. Mathematics of Computation **92**(341), 1273–1299 (2023) https://doi.org/10.1090/mcom/3802

[5] Robbins, H., Monro, S.: A stochastic approximation method. The Annals of Mathematical Statistics **22**, 400–407 (1951) https://doi.org/10.1214/aoms/1177729586

[6] Bottou, L., LeCun, Y.: Large scale online learning. In: Advances in Neural Information Processing Systems, vol. 16, pp. 217–224 (2004). available at: https://proceedings.neurips.cc/paper_files/paper/2003

[7] 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). Available at: https://proceedings.neurips.cc/paper_files/paper/2014

[8] Johnson, R., Zhang, T.: Accelerating stochastic gradient descent using predictive variance reduction. In: Advances in Neural Information Processing Systems, vol. 26, pp. 315–323 (2013). Available at: https://proceedings.neurips.cc/paper_files/paper/2013

[9] Schmidt, M., Le Roux, N., Bach, F.: Minimizing finite sums with the stochastic average gradient. Mathematical Programming **162**(1-2), 83–112 (2017) https://doi.org/10.1007/s10107-016-1030-6

[10] Nguyen, L.M., Liu, J., Scheinberg, K., Takáč, M.: SARAH: A novel method for machine learning problems using stochastic recursive gradient. In: International Conference on Machine Learning, pp. 2613–2621 (2017). PMLR. Available at: https://proceedings.mlr.press/v70/

[11] Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research **12**(7) (2011). Available at: https://www.jmlr.org/papers/v12/

[12] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings (2015). Available at: http://arxiv.org/abs/1412.6980

[13] Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. Siam Review **60**(2), 223–311 (2018) https://doi.org/10.1137/16M1080173

[14] 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 (2019). https://doi.org/10.1137/1.9781611975673.79 . SIAM

[15] Martens, J.: Deep learning via Hessian-free optimization. In: Proceedings of the 27th International Conference on Machine Learning, pp. 735–742 (2010). Available at: https://www.icml2010.org/abstracts.html

[16] Martens, J., Sutskever, I.: Training deep and recurrent networks with Hessian-free optimization. In: Neural Networks: Tricks of the Trade, pp. 479–535. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35289-8_27

[17] Bollapragada, R., Byrd, R.H., Nocedal, J.: Exact and inexact subsampled Newton methods for optimization. IMA Journal of Numerical Analysis **39**(2), 545–578 (2019) https://doi.org/10.1093/imanum/dry009

[18] 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 (2020). https://doi.org/10.1137/1.9781611976236.23 . SIAM

[19] Martens, J., Grosse, R.: Optimizing neural networks with Kronecker-factored approximate curvature. In: International Conference on Machine Learning, pp. 2408–2417 (2015). PMLR. Available at: https://proceedings.mlr.press/v37/

[20] Goldfarb, D., Ren, Y., Bahamou, A.: Practical quasi-newton methods for training deep neural networks. In: Advances in Neural Information Processing Systems, vol. 33, pp. 2386–2396 (2020). Available at: https://proceedings.neurips.cc/paper_files/paper/2020

[21] Mokhtari, A., Ribeiro, A.: Global convergence of online limited memory BFGS. The Journal of Machine Learning Research **16**(1), 3151–3181 (2015). Available at: https://www.jmlr.org/papers/v16/

[22] 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). PMLR. Available at: https://proceedings.mlr.press/v48/

[23] Wang, X., Ma, S., Goldfarb, D., Liu, W.: Stochastic quasi-Newton methods for nonconvex stochastic optimization. SIAM Journal on Optimization **27**(2), 927–956 (2017) https://doi.org/10.1137/15M1053141

[24] Berahas, A.S., Takáč, M.: A robust multi-batch L-BFGS method for machine learning. Optimization Methods and Software **35**(1), 191–219 (2020) https://doi.org/10.1080/10556788.2019.1658107

[25] Bollapragada, R., Nocedal, J., Mudigere, D., Shi, H.-J., Tang, P.T.P.: A progressive batching L-BFGS method for machine learning. In: International Conference on Machine Learning, pp. 620–629 (2018). PMLR. Available at: https://proceedings.mlr.press/v80/

[26] Jahani, M., Nazari, M., Rusakov, S., Berahas, A.S., Takáč, M.: Scaling up quasi-newton algorithms: communication efficient distributed SR1. In: *et al.*(ed.) Machine Learning, Optimization, and Data Science. LOD 2020. Lecture Notes in Computer Science, vol. 12565, pp. 41–54. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64583-0_5

[27] Berahas, A.S., Jahani, M., Richtárik, P., Takáč, M.: Quasi-newton methods for machine learning: forget the past, just sample. Optimization Methods and Software **37**(5), 1668–1704 (2022) https://doi.org/10.1080/10556788.2021.1977806

[28] 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 (2018). https://doi.org/10.1109/ICMLA.2018.00081 . IEEE

[29] Yousefi, M., Martínez Calomardo, Á.: A stochastic modified limited memory BFGS for training deep neural networks. In: Intelligent Computing: Proceedings of the 2022 Computing Conference, Volume 2, pp. 9–28 (2022). https://doi.org/10.1007/978-3-031-10464-0_2 . Springer

[30] Erway, J.B., Griffin, J., Marcia, R.F., Omheni, R.: Trust-region algorithms for training responses: machine learning methods using indefinite Hessian approximations. Optimization Methods and Software **35**(3), 460–487 (2020) https://doi.org/10.1080/10556788.2019.1624747

[31] Grippo, L., Lampariello, F., Lucidi, S.: A non-monotone line search technique for Newton's method. SIAM Journal on Numerical Analysis **23**(4), 707–716 (1986) https://doi.org/10.1137/0723046

[32] Deng, N., Xiao, Y., Zhou, F.: Nonmonotonic trust-region algorithm. Journal of optimization theory and applications **76**(2), 259–285 (1993) https://doi.org/10.1007/BF00939608

[33] Cui, Z., Wu, B., Qu, S.: Combining non-monotone conic trust-region and line

search techniques for unconstrained optimization. Journal of computational and applied mathematics **235**(8), 2432–2441 (2011) https://doi.org/10.1016/j.cam.2010.10.044

[34] Krejić, N., Krklec Jerinkić, N.: Non-monotone line search methods with variable sample size. Numerical Algorithms **68**(4), 711–739 (2015) https://doi.org/10.1007/s11075-014-9869-1

[35] Yousefi, M., Martínez Calomardo, Á.: A stochastic nonmonotone trust-region training algorithm for image classification. In: 2022 16th International Conference on Signal-Image Technology and Internet-Based Systems (SITIS), pp. 522–529 (2022). https://doi.org/10.1109/SITIS57111.2022.00084 . IEEE

[36] Sun, S., Nocedal, J.: A trust-region method for noisy unconstrained optimization. Mathematical Programming, 1–28 (2023) https://doi.org/10.1007/s10107-023-01941-9

[37] Cao, L., Berahas, A.S., Scheinberg, K.: First- and second-order high probability complexity bounds for trust-region methods with noisy oracles. Mathematical Programming (2023) https://doi.org/10.1007/s10107-023-01999-5

[38] Iusem, A.N., Jofré, A., Oliveira, R.I., Thompson, P.: Variance-based extra gradient methods with line search for stochastic variational inequalities. SIAM Journal on Optimization **29**(1), 175–206 (2019) https://doi.org/10.1137/17M1144799

[39] Krejić, N., Lužanin, Z., Ovcin, Z., Stojkovska, I.: Descent direction method with line search for unconstrained optimization in noisy environment. Optimization Methods and Software **30**(6), 1164–1184 (2015) https://doi.org/10.1080/10556788.2015.1025403

[40] Blanchet, J., Cartis, C., Menickelly, M., Scheinberg, K.: Convergence rate analysis of a stochastic trust-region method via supermartingales. INFORMS journal on optimization **1**(2), 92–119 (2019) https://doi.org/10.1287/ijoo.2019.0016

[41] Chen, R., Menickelly, M., Scheinberg, K.: Stochastic optimization using a trust-region method and random models. Mathematical Programming **169**(2), 447–487 (2018) https://doi.org/10.1007/s10107-017-1141-8

[42] Bellavia, S., Krejić, N., Morini, B., Rebegoldi, S.: A stochastic first-order trust-region method with inexact restoration for finite-sum minimization. Computational Optimization and Applications **84**(1), 53–84 (2023) https://doi.org/10.1007/s10589-022-00430-7

[43] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, pp. 249–256 (2010). JMLR Workshop and Conference Proceedings. Available at: https://proceedings.mlr.press/v9/

[44] Brust, J., Erway, J.B., Marcia, R.F.: On solving L-SR1 trust-region subproblems. Computational Optimization and Applications **66**(2), 245–266 (2017) https://doi.org/10.1007/s10589-016-9868-3

[45] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge, MA (2016). Available at: http://www.deeplearningbook.org

[46] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90

[47] Yousefi, M., Martínez, Á.: Deep neural networks training by stochastic quasi-newton trust-region methods. Algorithms **16**(10), 490 (2023) https://doi.org/10.3390/a16100490

[48] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998) https://doi.org/10.1109/5.726791

[49] LeCun, Y.: The MNIST database of handwritten digits. Available at: https://www.kaggle.com/datasets/hojjatk/mnist-dataset (1998)

[50] Krizhevsky, A.: Learning multiple layers of features from tiny images. Available at: https://api.semanticscholar.org/CorpusID:18268744 (2009)

Table 2: Architectures of the networks.

Regression	
`CNN-Rn`	$(Conv(3\times 3@8, 1, \text{same})/BN/ReLU/AvgPool(2\times 2, 2, 0))$ $(Conv(3\times 3@16, 1, \text{same})/BN/ReLU/AvgPool(2\times 2, 2, 0))$ $(Conv(3\times 3@32, 1, \text{same})/BN/ReLU)$ $(Conv(3\times 3@32, 1, \text{same})/BN/ReLU/DropOut(0.2))$ $FC(1)$
Classification	
`ResNet-20`	$(Conv(3\times 3@16, 1, 1)/BN/ReLU)$ B_1 { $(Conv(3\times 3@16, 1, 1)/BN/ReLU)$ $(Conv(3\times 3@16, 1, 1)/BN) + \text{addition}(1)/ReLU$ B_2 { $(Conv(3\times 3@16, 1, 1)/BN/ReLU)$ $(Conv(3\times 3@16, 1, 1)/BN) + \text{addition}(1)/ReLU$ B_3 { $(Conv(3\times 3@16, 1, 1)/BN/ReLU)$ $(Conv(3\times 3@16, 1, 1)/BN) + \text{addition}(1)/ReLU$ B_1 { $(Conv(3\times 3@32, 2, 1)/BN/ReLU)$ $(Conv(3\times 3@32, 1, 1)/BN)$ $(Conv(1\times 1@32, 2, 0)/BN) + \text{addition}(2)/ReLU$ B_2 { $(Conv(3\times 3@32, 1, 1)/BN/ReLU)$ $(Conv(3\times 3@32, 1, 1)/BN) + \text{addition}(1)/ReLU$ B_3 { $(Conv(3\times 3@32, 1, 1)/BN/ReLU)$ $(Conv(3\times 3@32, 1, 1)/BN) + \text{addition}(1)/ReLU$ B_1 { $(Conv(3\times 3@64, 2, 1)/BN/ReLU)$ $(Conv(3\times 3@64, 1, 1)/BN)$ $(Conv(1\times 1@64, 2, 0)/BN) + \text{addition}(2)/ReLU$ B_2 { $(Conv(3\times 3@64, 1, 1)/BN/ReLU)$ $(Conv(3\times 3@64, 1, 1)/BN) + \text{addition}(1)/ReLU$ B_3 { $(Conv(3\times 3@64, 1, 1)/BN/ReLU)$ $(Conv(3\times 3@64, 1, 1)/BN) + \text{addition}(1)/gAvgPool/ReLU)$ $FC(C/Softmax)$
Classification	
`LeNet-like`	$(Conv(5\times 5@20, 1, 0)/ReLU/MaxPool(2\times 2, 2, 0))$ $(Conv(5\times 5@50, 1, 0)/ReLU/MaxPool(2\times 2, 2, 0))$ $FC(500/ReLU)$ $FC(C/Softmax)$

TABLE'S NOTES: See [45, 46] for more details about the different layers in a deep neural network. The compound $(Conv(5\times 5@32,\ 1, 2)/BN/ReLu/MaxPool(2\times 2,\ 1, 0)))$ indicates a simple convolutional network (ConvNet) including a convolutional layer $(Conv)$ using 32 filters of size 5×5, stride 1, padding 2, followed by a batch normalization layer (BN), a nonlinear activation function $(ReLu)$ and, finally, a 2-D max-pooling layer with a channel of size 2×2, stride 1 and padding 0. The syntax $FC(C/Softmax)$ denotes a layer of C fully connected neurons followed by the $softmax$ layer. Moreover, $(AvgPool)$, $(gAvg.Pool)$, and $(DropOut)$ refer to the 2D average-pooling, global average-pooling, and drop-out layers, respectively. The syntax $addition(1)/ReLu$ indicates the existence of an *identity shortcut* with functionality such that the output of a given block, say B_1 (or B_2 or B_3), is directly fed to the *addition* layer and then to the ReLu layer while $addition(2)/ReLu$ in a block shows the existence of a *projection shortcut* with functionality such that the output from the two first ConvNets is added to the output of the third ConvNet and then the output is passed through the $ReLu$ layer. An open-source implementation of the `ResNet-20` and `LeNet-like` networks described above as components in Matlab programs of algorithms presented in [47] is available on https://github.com/MATHinDL/sL_QN_TR/.)