# Completely Randomized Trust-Region Algorithms Incorporating Barzilai–Borwein Step Sizes

**Ahmed Ibrahim Hussein**

**Emil: a07826148119@gmail.com**

**Abstract**

This study introduces novel stochastic gradient algorithms that integrate Barzilai–Borwein adaptive step sizes within a trust-region-inspired framework for solving finite-sum optimization problems. The proposed approach, referred to as **TRishBB**, extends the foundation of the Trust-Region-ish (TRish) framework developed by Curtis, Scheinberg, and Shi (2019) in the *INFORMS Journal on Optimization*. The objective of TRishBB is to improve the computational efficiency and optimization performance of the original TRish scheme while avoiding the high cost associated with its second-order variant. Three distinct algorithms under the TRishBB family are introduced, and their convergence behavior is analyzed for both convex and nonconvex objectives using biased or unbiased stochastic gradients. The theoretical results are obtained without assuming diminishing step lengths or full gradient evaluations. Experimental evaluations on machine learning benchmarks confirm that employing Barzilai–Borwein step sizes within stochastic trust-region schemes enhances both convergence speed and testing accuracy relative to the standard TRish method.

**Keywords:** stochastic trust-region algorithms; finite-sum optimization; Barzilai–Borwein step length; stochastic gradients; machine learning.

# 1 .Introduction

We study the optimization problem defined as

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

where $f$ is a finite-sum objective composed of $N$ continuously differentiable component functions $f_i: \mathbb{R}^n \to \mathbb{R}$, $i \in \mathcal{N} = \{1, \dots, N\}$. This general formulation encompasses a wide range of practical problems such as classification in Machine Learning (ML) and Deep Learning (DL), data-fitting tasks, and sample average approximations of expected-value objectives.

Among first-order optimization algorithms, Gradient Descent (GD) is a fundamental approach that iteratively updates the variable sequence as

$$x_{k+1} = x_k - \mu_k \nabla f(x_k), \tag{2}$$

where $\mu_k > 0$ denotes the steplength. The convergence rate and computational efficiency of GD and its variants depend strongly on how $\mu_k$ is selected [1, 2, 3]. Barzilai and Borwein [?] introduced a strategy to choose $\mu_k$ that approximates second-order information without explicitly computing or inverting the Hessian matrix. In essence, the Barzilai–Borwein (BB) technique can be interpreted as a quasi-Newton approach where the Hessian is approximated by $\mu_k^{-1} I$, and $\mu_k$ is chosen to minimize the least-squares residual of a secant-like condition involving consecutive iterates and gradients.

However, computing the full gradient $\nabla f(x_k)$ at each iteration becomes computationally burdensome in ML and DL contexts, where $f$ represents a loss over massive datasets (large $N$). The Stochastic Gradient (SG) method [4, 5] and its mini-batch version [6] mitigate this cost by approximating $\nabla f(x_k)$ through a random subset of samples:

$$\nabla f_{\mathcal{N}_k}(x_k) = \frac{1}{|\mathcal{N}_k|} \sum_{i \in \mathcal{N}_k} \nabla f_i(x_k),$$

where $\mathcal{N}_k \subseteq \mathcal{N}$ and $|\mathcal{N}_k|$ denotes the mini-batch size. These approaches, however, require careful tuning of problem-specific step sizes to ensure stability and convergence.

To alleviate this sensitivity, the Trust-Region-ish (TRish) framework was proposed in [7], where the step length is automatically adapted according to a trust-region-like rule. The algorithm selects normalized steps whenever the gradient norm falls within a predefined interval whose limits are adjusted empirically. Unlike traditional trust-region methods, TRish does not require evaluating the objective function or comparing predicted and actual reductions. Subsequent works [8, 9] extended TRish for first-order and quadratic models, showing improved performance compared to standard stochastic gradient approaches, and more recently it has been adapted to equality-constrained settings [10].

The current work aims to merge the TRish framework with stochastic Barzilai–Borwein steplengths, allowing exploitation of approximate second-order information at low computational cost. In this setting, the BB steplength selection is embedded directly within the TRish mechanism. We present a stochastic formulation of BB step computation and establish convergence guarantees without assuming diminishing step sizes or unbiased gradient estimates. Furthermore, we introduce three algorithmic variants under this framework and provide

empirical evidence demonstrating that these methods improve the performance of the baseline TRish algorithm [11].

The structure of this paper is as follows: Section 2 describes the proposed stochastic algorithms; Section 3 develops their theoretical analysis; Section 4 provides implementation details and experimental evaluations; and Section 5 concludes with key findings and future perspectives.

**Notation.**

The Euclidean norm is denoted by $\|\cdot\|$, while $|\cdot|$ refers to the absolute value of a scalar, the cardinality of a set, or the number of columns in a matrix, depending on context. The floor operator $\lfloor\cdot\rfloor$ returns the greatest integer less than or equal to its argument, and $\mathrm{mod}(p,t)$ denotes the remainder after dividing $p$ by $t$. The symbols $\mathbb{E}_k[\cdot]$ and $\mathbb{P}_k[\cdot]$ represent conditional expectation and probability given that the iterate $x_k$ has been reached. The unconditional expectation over all randomness is denoted by $\mathbb{E}[\cdot]$.

## 1.1 Related Work and Motivation

Incorporating Barzilai–Borwein (BB) step sizes into stochastic optimization requires defining stochastic analogues of the classic BB formulas and establishing convergence guarantees (in expectation or with high probability) for the resulting schemes. A number of contributions in this direction exist [12, 13, 14, 15, 16], many of which stem from the broader family of stochastic quasi-Newton methods [17, 18, 19, 20, 21, 22, 23].

In [24], BB step sizes were embedded within both Stochastic Variance Reduced Gradient (SVRG) [25] and standard stochastic gradient (SG) methods using nested (inner/outer) loops. Because SVRG computes a full gradient at each outer iteration, the BB parameter at the outer level can be formed with the standard (deterministic) formula, and convergence is proved for strongly convex problems. For the SG variant in [26], single-sample gradients $\nabla f_i$ drive inner iterations, while a moving average of $N$ stochastic gradients at distinct points is used to build a BB-type parameter at outer iterations; however, a complete theoretical analysis of this "SGD-BB" construction was not provided. The work of [27] employed deterministic BB step sizes [28]—relying on exact function and gradient information—inside SVRG and established convergence under strong convexity. In [29], stochastic BB parameters were computed via moving averages of mini-batch gradients evaluated at different iterates and were accepted only

when they fell into a shrinking interval. Although diminishing step sizes ensure convergence by classical SG theory, this mechanism can eventually reject BB updates, since the BB parameter is not inherently vanishing. A related approach in [30] also enforced diminishing step sizes while forming stochastic BB parameters, with the distinctive design choice of keeping the mini-batch fixed for several iterations to probe the approximate Hessian spectrum. Finally, [31] examined BB step sizes together with projected stochastic gradients for constrained problems and for objectives given as expectations.

Across these studies, common constraints include reliance on diminishing step sizes, growth in mini-batch size $|\mathcal{N}_k|$, occasional use of full gradients, and the need for objective evaluations. Moreover, most theoretical results target (strongly) convex objectives. Our framework sidesteps these issues by leveraging the TRish paradigm [32, 33], which employs a simple stochastic quadratic model inspired by BB ideas and adapts the step via a trust-region-style rule without evaluating $f$. The resulting algorithms are lightweight, and we provide theory accommodating possibly nonconvex objectives and either biased or unbiased stochastic gradients.

[H]

TRishBB

[1] Choose an initial point $x_0 \in \mathbb{R}^n$ and an initial scalar $\mu_0 > 0$. Select parameters $(\alpha, \gamma_1, \gamma_2)$ with $\alpha > 0$, $0 < \gamma_2 \leq \gamma_1 < \infty$, and a lower bound $\mu_{\min} > 0$. $k = 0,1,2,\ldots$ Set $H_k = \mu_k^{-1} I$. Compute a stochastic gradient $g_k$. Solve the trust-region subproblem

$$\min_{p \in \mathbb{R}^n} \quad m_k(p) = g_k^{\top} p + \frac{1}{2} p^{\top} H_k p \qquad \text{s.t.} \qquad \| p \| \leq \Delta_k, \tag{3}$$

where the radius $\Delta_k$ is defined by

$$\Delta_k = \begin{cases} \alpha\gamma_1 \| g_k \|, & \text{if } \| g_k \| \in [0, \frac{1}{\gamma_1}), \\ \alpha, & \text{if } \| g_k \| \in [\frac{1}{\gamma_1}, \frac{1}{\gamma_2}], \\ \alpha\gamma_2 \| g_k \|, & \text{if } \| g_k \| \in (\frac{1}{\gamma_2}, \infty). \end{cases} \tag{4}$$

Update the iterate $x_{k+1} = x_k + p_k$, where $p_k$ solves (3). Form the next BB parameter $\mu_{k+1} \geq \mu_{\min}$.

## 2 .New Stochastic Trust-Region-ish Algorithms

This section presents a class of stochastic trust-region-ish algorithms that employ

Barzilai–Borwein (BB) inspired step sizes. We first recall the classical deterministic BB method [34, 35, 36, 37], then describe the main principles of the proposed **TRishBB** framework, and finally introduce three specific algorithmic variants.

At iteration $k$, the deterministic BB method minimizes the quadratic model

$$m_k(p) = \nabla f(x_k)^\top p + \frac{1}{2} p^\top H_k p,$$

where $H_k = \mu_k^{-1} I$ for some scalar $\mu_k > 0$. The resulting update has the same structure as the gradient descent iteration (3). Once $x_{k+1}$ is obtained, the BB step size $\mu_{k+1}^{\mathrm{BB}}$ is computed by solving

$$\mu_{k+1}^{\mathrm{BB}} = \operatorname*{argmin}_{\mu} \| \mu^{-1} s_k - y_k \|^2 = \frac{s_k^\top s_k}{s_k^\top y_k}, \tag{5}$$

where $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ form the so-called *correction pair*. For strictly convex $f$, this ratio is positive.

If $f$ is twice continuously differentiable, $\mu_{k+1}^{\mathrm{BB}}$ can be expressed using the mean-value theorem [38]:

$$\mu_{k+1}^{\mathrm{BB}} = \frac{s_k^\top s_k}{s_k^\top \left( \int_0^1 \nabla^2 f(x_k + t s_k) \, dt \right) s_k} = \frac{1}{q\left( \int_0^1 \nabla^2 f(x_k + t s_k) \, dt, s_k \right)}, \tag{6}$$

where $q(A, v)$ denotes the Rayleigh quotient of matrix $A$ and vector $v$. Hence,

$$\mu_{k+1}^{\mathrm{BB}} \in \left[ \frac{1}{\lambda_{\max}\left( \int_0^1 \nabla^2 f(x_k + t s_k) \, dt \right)}, \; \frac{1}{\lambda_{\min}\left( \int_0^1 \nabla^2 f(x_k + t s_k) \, dt \right)} \right]$$

if $f$ is strictly convex. In general, however, $\mu_{k+1}^{\mathrm{BB}}$ may be negative.

In practice, the parameter $\mu_{k+1}$ is bounded to avoid extreme values:

$$\mu_{k+1} = \max\{\mu_{\min}, \min(|\mu_{k+1}^{\mathrm{BB}}|, \mu_{\max})\},$$

where $0 < \mu_{\min} < \mu_{\max}$. In convex problems, $|\cdot|$ can be omitted. We refer to [39] for a detailed review of heuristic rules for BB updates and step acceptance strategies.

## 2.1 Trust-Region-ish Formulation

We now focus on the **TRishBB** algorithm, a stochastic extension of the second-order TRish method [40], specialized to the case where $H_k = \mu_k^{-1} I$ for all $k$. The main idea is to integrate BB-inspired step sizes $\mu_k$ to enrich the first-order TRish variant [?], where $H_k = 0$. Because stochastic gradients are used, the correction pair $(s_k, y_k)$ is not deterministic, making $\mu_{k+1}^{\mathrm{BB}}$ a random variable. The definition of $(s_k, y_k)$—and hence of $\mu_{k+1}$—is a crucial design component

that will be discussed in Sections 2.1–2.3.

At each iteration, the stochastic gradient $g_k$ is computed. Given a trust-region radius $\Delta_k$ and a scalar $\mu_k > \mu_{\min}$, the quadratic subproblem

$$\min_{p\in\mathbb{R}^n} \quad m_k(p) = g_k^\top p + \frac{1}{2}p^\top H_k p \qquad \text{s.t.} \qquad \| p \| \leq \Delta_k$$

is solved with $H_k = \mu_k^{-1} I$. This has a simple closed-form solution:

$$p_k = \begin{cases} -\mu_k g_k, & \text{if } \| \mu_k g_k \| < \Delta_k, \\ -\frac{\Delta_k}{\|g_k\|} g_k, & \text{otherwise.} \end{cases} \tag{7}$$

When $p_k = -\mu_k g_k$, it is the unconstrained minimizer of the quadratic model; otherwise, it lies on the trust-region boundary (the constrained Cauchy point). Thus, $p_k$ always follows $-g_k$, making the subproblem inexpensive to solve. In contrast, full second-order trust-region methods would require costly linear algebra procedures such as conjugate-gradient iterations [41].

The radius $\Delta_k$ is determined by the rule

$$\Delta_k = \begin{cases} \alpha\gamma_1 \| g_k \|, & \| g_k \| \in [0, 1/\gamma_1), \\ \alpha, & \| g_k \| \in [1/\gamma_1, 1/\gamma_2], \\ \alpha\gamma_2 \| g_k \|, & \| g_k \| \in (1/\gamma_2, \infty), \end{cases} \tag{8}$$

where $\alpha$, $\gamma_1$, and $\gamma_2$ are positive constants. This rule—central to TRish—prevents excessive normalization of gradients that could lead to ascent directions and instability [?]. In this work, we assume fixed parameters, though variable sequences $\{\alpha_k\}$, $\{\gamma_{1,k}\}$, and $\{\gamma_{2,k}\}$ can be used.

Combining (8) and (7), the constrained trust-region step becomes

$$p_k = \begin{cases} -\alpha\gamma_1 g_k, & \| g_k \| \in [0, 1/\gamma_1), \\ -\alpha \frac{g_k}{\|g_k\|}, & \| g_k \| \in [1/\gamma_1, 1/\gamma_2], \\ -\alpha\gamma_2 g_k, & \| g_k \| \in (1/\gamma_2, \infty), \end{cases} \tag{9}$$

which mirrors the first-order TRish formulation in [?]. The unconstrained step $p_k = -\mu_k g_k$ occurs whenever

$$\mu_k < \begin{cases} \alpha\gamma_1, & \| g_k \| \in [0, 1/\gamma_1), \\ \frac{\alpha}{\|g_k\|}, & \| g_k \| \in [1/\gamma_1, 1/\gamma_2], \\ \alpha\gamma_2, & \| g_k \| \in (1/\gamma_2, \infty). \end{cases} \tag{10}$$

[H]

TRishBB v1 (Iteration $k$)

[1] Given $x_k \in \mathbb{R}^n$, parameters $(\alpha, \gamma_1, \gamma_2)$, $\mu_k > 0$, bounds $0 < \mu_{\min} < \mu_{\max}$, and an integer $m \geq 1$. Choose $\mathcal{N}_k \subseteq \mathcal{N}$, compute $g_k = \nabla f_{\mathcal{N}_k}(x_k)$, and set $H_k = \mu_k^{-1} I$. Obtain $p_k$ from (7) and update $x_{k+1} = x_k + p_k$. $\mathrm{mod}(k, m) = 0$ Set $s_k = p_k$, $y_k = \nabla f_{\mathcal{N}_k}(x_{k+1}) - g_k$. Update

$$\mu_{k+1} = \max\left\{\mu_{\min}, \min\left(\frac{s_k^{\mathrm{T}} s_k}{s_k^{\mathrm{T}} y_k}, \mu_{\max}\right)\right\}.$$

$\mu_{k+1} = \mu_k$.

Notably, after computing $p_k$, the new iterate $x_{k+1}$ is accepted directly—without performing any predicted-versus-actual reduction test as in classical trust-region schemes [22]. Therefore, the TRishBB framework operates without evaluating the objective function $f$.

Each step $p_k$ lies in the direction of $-g_k$, similar to first-order methods, but the unconstrained variant incorporates approximate second-order curvature through $\mu_k$. In the following subsections, we present three alternative ways to construct stochastic BB step sizes, each defining a distinct version of the TRishBB algorithm.

## 2.2 TRishBB v1

The **TRishBB v1** algorithm represents a direct stochastic adaptation of the classical Barzilai–Borwein (BB) approach, combined with insights from stochastic quasi-Newton strategies. After computing the trust-region step $p_k$ according to (7), the following BB-type step length $\mu_{k+1}$ is derived from a correction pair $(s_k, y_k)$, as indicated in line 5 of Algorithm 2.1. The displacement vector $s_k$ is identified with $p_k$, i.e., $s_k = x_{k+1} - x_k$, while the vector $y_k$ captures the change between two mini-batch gradients computed over the same sample subset:

$$y_k = \nabla f_{\mathcal{N}_k}(x_{k+1}) - \nabla f_{\mathcal{N}_k}(x_k).$$

Using the same sample indices $\mathcal{N}_k$ for both gradients—recommended in [?, ?, ?]—ensures that relation (6) remains valid with $\int_0^1 \nabla^2 f_{\mathcal{N}_k}(x_k + t s_k)\ dt$. Therefore, as long as $\mu_{k+1}^{\mathrm{BB}} > 0$ and $f$ is twice continuously differentiable, the BB step length implicitly reflects the spectral behavior of the stochastic Hessian. Following common practice [27], the parameter $\mu_{k+1}$ is updated every $m$ iterations, where $m \geq 1$ is a fixed integer.

[H]

TRishBB v2 (Iteration $k$)

[1] Given $x_k \in \mathbb{R}^n$, parameters $(\alpha, \gamma_1, \gamma_2)$, $\mu_k > 0$, $\bar{\mu} > 0$, bounds $0 < \mu_{\min} < \mu_{\max}$, an

integer $m \geq 1$, $\beta = \frac{m-1}{m}$, $\eta \in (0,1)$, and auxiliary variables $\bar{x}_{\text{old}}, \bar{g}_{\text{old}}, \bar{g}$ (initialized as $\bar{x}_{\text{old}} = x_0$, $\bar{g}_{\text{old}} = 0$, $\bar{g} = 0$, and $\bar{\mu} = \mu_0$ for $k = 0$). Choose $\mathcal{N}_k \subseteq \mathcal{N}$, compute $g_k = \nabla f_{\mathcal{N}_k}(x_k)$, and set $H_k = \mu_k^{-1} I$. Compute $p_k$ from (7) and update $x_{k+1} = x_k + p_k$. Update the moving-average gradient $\bar{g} = \beta \bar{g} + (1-\beta) g_k$. $k > 0$ and $\text{mod}(k, m) = 0$ $\bar{x} = x_{k+1}$. $s_k = \bar{x} - \bar{x}_{\text{old}}$, $y_k = \bar{g} - \bar{g}_{\text{old}}$. $\tilde{\mu}_k = \frac{1}{m} \frac{s_k^{\mathrm{T}} s_k}{s_k^{\mathrm{T}} y_k}$, $\bar{\mu} = \eta \bar{\mu} + (1-\eta) \tilde{\mu}_k$. $\mu_{k+1} = \max\{\mu_{\min}, \min\{\bar{\mu}, \mu_{\max}\}\}$. $\bar{x}_{\text{old}} = \bar{x}$, $\bar{g}_{\text{old}} = \bar{g}$, $\bar{g} = 0$. $\mu_{k+1} = \mu_k$.

Compared with the original TRish algorithm, the additional computational overhead consists solely of evaluating $\nabla f_{\mathcal{N}_k}(x_{k+1})$ and two inner products to form $\mu_{k+1}$.

## 2.3 TRishBB v2

Unlike TRishBB v1, where $(s_k, y_k)$ relies on consecutive iterates and mini-batch gradients, **TRishBB v2** aggregates gradient information over a fixed number $m$ of iterations. The displacement $s_k$ now represents the difference between two non-consecutive iterates separated by $m$ steps, while $y_k$ captures the change in averaged stochastic gradients accumulated during the same $m$ iterations. Specifically, the moving-average gradient $\bar{g}$ is updated via

$$\bar{g} = \beta \bar{g} + (1-\beta) g_k, \qquad \beta = \frac{m-1}{m},$$

as suggested by [24], which avoids manual hyperparameter tuning.

This scheme extends the stochastic gradient descent with BB step size (SGD-BB) of [33], where $|\mathcal{N}_k| = 1$ and $m = N$. In contrast, TRishBB v2 allows mini-batches of arbitrary size $|\mathcal{N}_k| > 1$. If we define $N_b = \lfloor N / |\mathcal{N}_k| \rfloor$ as the approximate number of disjoint mini-batches and set $m = N_b$, then computing $\bar{g}$ over $m$ iterations incurs roughly the same cost as one full gradient evaluation.

Every $m$ iterations, the new step size $\mu_{k+1}$ is updated using the convex combination

$$\bar{\mu} \leftarrow \eta \bar{\mu} + (1-\eta) \tilde{\mu}_k,$$

where $\tilde{\mu}_k$ is the instantaneous stochastic BB estimate and $\eta \in (0,1)$ controls the level of smoothing. Larger values of $\eta$ reduce the influence of $\tilde{\mu}_k$, mitigating the effect of gradient noise. The additional computational effort is limited to two dot products, while memory usage involves three vectors of size $n$.

## 2.4 TRishBB v3

The third variant, **TRishBB v3**, follows the rationale of [?], designed to alleviate instability from small displacements ($\| s_k \|$ close to zero) and noisy gradient differences. In this approach, given $s_k = \bar{x} - \bar{x}_{\text{old}}$, the curvature vector $y_k$ is defined as

$$y_k = \nabla^2 f_{\mathcal{N}_k^f}(\bar{x}) s_k,$$

where $\nabla^2 f_{\mathcal{N}_k^f}(\bar{x})$ denotes a subsampled Hessian approximation based on a larger batch $\mathcal{N}_k^f$. For common objectives such as least-squares or cross-entropy, the Hessian can be approximated by the empirical Fisher Information Matrix (eFIM) [?, §11]:

$$\frac{1}{|\mathcal{N}_k^f|} \sum_{i \in \mathcal{N}_k^f} \nabla f_i(\bar{x}) \nabla f_i(\bar{x})^\top.$$

To reduce cost, [15, 17] proposed using the accumulated Fisher Information Matrix (aFIM). In TRishBB v3, $s_k$ and $y_k$ are updated every $m$ iterations using this aFIM and a first-in–first-out (FIFO) memory mechanism that maintains recent gradients.

Let $F_k \in \mathbb{R}^{n \times q}$ be a limited-memory matrix containing up to $m_F$ most recent stochastic gradients. The curvature vector is computed as

$$y_k = \frac{1}{|F_k|} F_k (F_k^\top s_k),$$

where $|F_k|$ denotes the number of stored columns. Consequently, since $F_k F_k^\top$ is a sum of $|F_k|$ rank-one matrices, the expected value of $\mu_k$ satisfies

$$\tilde{\mu}_k = \frac{|F_k|}{m} \left( \sum_{t=k-|F_k|+1}^{k} q(\nabla f_{\mathcal{N}_t}(x_t) \nabla f_{\mathcal{N}_t}(x_t)^\top, s_k) \right)^{-1}.$$

[H]

TRishBB v3 (Iteration $k$)

[1] Given $x_k \in \mathbb{R}^n$, $(\alpha, \gamma_1, \gamma_2)$, $\mu_k > 0$, $\bar{\mu} > 0$, $0 < \mu_{\min} < \mu_{\max}$, integers $m \geq 1$ and $m_F \in \mathbb{N}$, matrix $F_k \in \mathbb{R}^{n \times q}$ with $q \leq m_F$, vectors $x_{\text{avg}}$, $\bar{x}_{\text{old}}$ (initialize $F_0 = [\ ]$, $x_{\text{avg}} = 0$, $\bar{x}_{\text{old}} = 0$, $\bar{\mu} = \mu_0$ for $k = 0$). Accumulate $x_{\text{avg}} = x_{\text{avg}} + x_k$. Choose $\mathcal{N}_k$, compute $g_k = \nabla f_{\mathcal{N}_k}(x_k)$, and set $H_k = \mu_k^{-1} I$. Compute $p_k$ from (7) and update $x_{k+1} = x_k + p_k$. $k \leq m_F$ Store $g_k$ in the $k$-th column of $F_k$. Discard the oldest column and insert $g_k$ as the newest. $\text{mod}(k, m) = 0$ $\bar{x} = x_{\text{avg}}/m$, reset $x_{\text{avg}} = 0$. $k > 0$ $s_k = \bar{x} - \bar{x}_{\text{old}}$, $y_k = \frac{1}{|F_k|} F_k (F_k^\top s_k)$. $\tilde{\mu}_k = \frac{1}{m} \frac{s_k^\top s_k}{s_k^\top y_k}$, $\bar{\mu} = \eta \bar{\mu} + (1 - \eta) \tilde{\mu}_k$. $\mu_{k+1} = \max\{\mu_{\min}, \min\{\bar{\mu}, \mu_{\max}\}\}$. $\bar{x}_{\text{old}} = \bar{x}$. $\mu_{k+1} =$

$\mu_k$.

This algorithm incurs an additional cost of approximately $m_F + 2$ scalar products per update and requires storage for two $n$-dimensional vectors and an $n \times m_F$ matrix. Like the previous variants, it updates $\mu_k$ every $m$ iterations using accumulated curvature information.

# 3. Convergence Analysis

In this section, we establish convergence guarantees for the TRishBB family under both *biased* and *unbiased* stochastic gradient oracles. We cover two regimes: (i) objectives satisfying the Polyak–Łojasiewicz (PL) condition, and (ii) general nonconvex objectives. We begin with the standing smoothness and PL assumptions.

[Smoothness] The function $f: \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable, bounded below by $f^\star$, and has $L$-Lipschitz gradient, i.e.,

$$f(x) \le f(y) + \nabla f(y)^\top (x - y) + \frac{L}{2} \| x - y \|^2 \qquad \text{forall} x, y \in \mathbb{R}^n. \tag{10}$$

[Polyak–Łojasiewicz] There exists $c > 0$ such that, for all $x \in \mathbb{R}^n$,

$$2c(f(x) - f^\star) \le \| \nabla f(x) \|^2. \tag{11}$$

Throughout the analysis, we partition iterations into three mutually exclusive cases according to the norm of the stochastic gradient $g_k$:

$\text{Case1}: \| g_k \| \in [0, 1/\gamma_1)$, $\text{Case2}: \| g_k \| \in [1/\gamma_1, 1/\gamma_2]$, $\text{Case3}: \| g_k \| \in (1/\gamma_2, \infty)$.

Let $C_{i,k}$ denote the event that Case $i \in \{1,2,3\}$ occurs at iteration $k$.

## 3.1 Analysis with Biased Stochastic Gradients

We first handle biased gradient estimators satisfying the following moments conditions.

[Biased oracle] There exist constants $\omega > 0$ and $M_1, M_2 > 0$ such that

$$\nabla f(x_k)^\top \mathbb{E}_k[g_k] \ge \omega \| \nabla f(x_k) \|^2, \tag{12}$$

$$\mathbb{E}_k[\| g_k \|^2] \le M_1 + M_2 \| \nabla f(x_k) \|^2. \tag{13}$$

Moreover, letting $\mathcal{E}_k$ denote the event $\{\nabla f(x_k)^\top g_k \ge 0\}$, it follows that

$$\mathbb{P}_k[\mathcal{E}_k] \;\; \mathbb{E}_k[\nabla f(x_k)^\top g_k | \mathcal{E}_k] \le h_1 + h_2 \| \nabla f(x_k) \|^2, \qquad h_1 = \frac{1}{2}\sqrt{M_1}, \; h_2 = h_1 + \sqrt{M_2}. \tag{14}$$

We also constrain the minimal BB parameter used by TRishBB.

[Lower bound on $\mu$] For all $k \geq 0$, $\mu_{\min} \leq \gamma_2 \alpha$.

Our derivations adapt the arguments in [36]. For clarity, define the mutually exclusive events for the *unconstrained* and *constrained* solutions of the trust-region subproblem (7) by

$$\mathcal{S}_{\mathrm{U}} = \{\| \mu_k g_k \| < \Delta_k\}, \qquad \mathcal{S}_{\mathrm{C}} = \{\| \mu_k g_k \| \geq \Delta_k\},$$

and the intersections $C_{i,k}^{\mathrm{U}} = C_{i,k} \cap \mathcal{S}_{\mathrm{U}}$, $C_{i,k}^{\mathrm{C}} = C_{i,k} \cap \mathcal{S}_{\mathrm{C}}$. We also use the sign events

$$\mathcal{E}_k = \{\nabla f(x_k)^{\top} g_k \geq 0\}, \qquad \bar{\mathcal{E}}_k = \{\nabla f(x_k)^{\top} g_k < 0\}.$$

Under Assumptions 3 and 3.1, the TRishBB iterates satisfy

$$\mathbb{E}_k[f(x_{k+1})] - f(x_k) \leq (\gamma_1 \alpha - \mu_{\min})\ \mathbb{P}_k[\mathcal{E}_k]\ \mathbb{E}_k[\nabla f(x_k)^{\top} g_k | \mathcal{E}_k] + \frac{L}{2}\gamma_1^2 \alpha^2\ \mathbb{E}_k[\| g_k \|^2] - \gamma_1 \alpha\ \mathbb{E}_k[\nabla f(x_k)^{\top} g_k]. \tag{15}$$

*Proof.* See Appendix B.1.

Suppose Assumptions 3, 3.1, and 3.1 hold. Then

$$\mathbb{E}_k[f(x_{k+1})] - f(x_k) \leq (\gamma_1 \alpha - \mu_{\min}) h_1 + \frac{L}{2}\gamma_1^2 \alpha^2 M_1 - \gamma_1 \alpha \left(\omega - \frac{L}{2}\gamma_1 M_2 \alpha\right) \| \nabla f(x_k) \|^2 + (\gamma_1 \alpha - \mu_{\min}) h_2 \| \nabla f(x_k) \|^2. \tag{16}$$

*Proof.* Combine Lemma 3.1 with (??) and Assumption 3.1.

Define

$$\theta_1 = \frac{1}{2}\omega\gamma_1 - h_2(\gamma_1 - \rho\gamma_2), \qquad \theta_2 = h_1(\gamma_1 \alpha - \mu_{\min}) + \frac{1}{2}\gamma_1^2 L M_1 \alpha^2, \qquad \rho = \frac{\mu_{\min}}{\alpha \gamma_2} \in (0,1]. \tag{17}$$

If $h_2 - \frac{1}{2}\omega \leq 0$, then $\theta_1 > 0$ automatically; otherwise $\theta_1 > 0$ is ensured by

$$\frac{\gamma_1}{\gamma_2} < \frac{\rho h_2}{h_2 - \frac{1}{2}\omega}. \tag{18}$$

Note that $\theta_2 > 0$ by Assumption 3.1.

**PL case.**

Under Assumptions 3–3.1, assume (23) when $h_2 - \frac{1}{2}\omega > 0$ and choose

$$\alpha \leq \min\left\{\frac{\omega}{\gamma_1 L M_2}, \frac{1}{2c\theta_1}\right\}. \tag{19}$$

Then the expected suboptimality of TRishBB satisfies

$$\mathbb{E}[f(x_{k+1})] - f^\star \xrightarrow[k\to\infty]{} \frac{\theta_2}{2c\alpha\theta_1}. \tag{20}$$

*Proof.* From (18) and (19),

$$\mathbb{E}_k[f(x_{k+1})] - f(x_k) \le -\alpha\theta_1 \parallel \nabla f(x_k) \parallel^2 + \theta_2.$$

Using the PL inequality (20) yields

$$\mathbb{E}_k[f(x_{k+1})] - f(x_k) \le -2c\alpha\theta_1(f(x_k) - f^\star) + \theta_2. \tag{21}$$

Rearranging, taking total expectations, and applying the standard contraction argument gives (21); see also

$$\mathbb{E}[f(x_{k+1})] - f^\star \le \frac{\theta_2}{2c\alpha\theta_1} + (1 - 2c\alpha\theta_1)\left(\mathbb{E}[f(x_k)] - f^\star - \frac{\theta_2}{2c\alpha\theta_1}\right). \tag{22}$$

**General nonconvex case.**

Under Assumptions 3, 3.1, and 3.1, assume (??) when $h_2 - \frac{1}{2}\omega > 0$ and

$$\alpha \le \frac{\omega}{\gamma_1 L M_2}. \tag{23}$$

Then

$$\frac{1}{K}\sum_{k=0}^{K} \mathbb{E}[\parallel \nabla f(x_k) \parallel^2] \xrightarrow[K\to\infty]{} \frac{\theta_2}{\alpha\theta_1}. \tag{24}$$

*Proof.* From (23) and (24),

$$\mathbb{E}_k[f(x_{k+1})] - f(x_k) \le -\alpha\theta_1 \parallel \nabla f(x_k) \parallel^2 + \theta_2.$$

Taking total expectation and summing from $k = 0$ to $K$ gives

$$\sum_{k=0}^{K} \mathbb{E}[\parallel \nabla f(x_k) \parallel^2] \le \frac{f(x_0) - \mathbb{E}[f(x_{K+1})]}{\alpha\theta_1} + \frac{(K+1)\theta_2}{\alpha\theta_1} \le \frac{f(x_0) - f^\star}{\alpha\theta_1} + \frac{(K+1)\theta_2}{\alpha\theta_1}, \tag{25}$$

and division by $K + 1$ yields (25).

## 3.2 Analysis with Unbiased Stochastic Gradients

We now turn to unbiased gradient estimators with bounded variance.

[Unbiased oracle] For all $k$, $\mathbb{E}_k[g_k] = \nabla f(x_k)$ and there exists $M_g > 0$ such that

$$\mathbb{E}_k[\parallel \nabla f(x_k) - g_k \parallel^2] \le M_g. \tag{26}$$

Since

$$\mathbb{E}_k[\parallel \nabla f(x_k) - g_k \parallel^2] = \parallel \nabla f(x_k) \parallel^2 - 2\ \nabla f(x_k)^\top \mathbb{E}_k[g_k] + \mathbb{E}_k[\parallel g_k \parallel^2]$$

$$= -\| \nabla f(x_k) \|^2 + \mathbb{E}_k[\| g_k \|^2], \tag{27}$$

Assumption 3.1 holds with $\omega = 1$, $M_1 = M_g$, and $M_2 = 1$. Hence, the results of Section 3.1 apply. We now obtain bounds that avoid the extra restriction (??).

Suppose Assumption 3 holds and parameters satisfy

$$0 < \alpha \leq \frac{\gamma_2}{8\gamma_1^2 L}, \qquad \mu_{\min} \geq \frac{4\gamma_1 \alpha}{5}. \tag{28}$$

Then TRishBB obeys

$$\mathbb{E}_k[f(x_{k+1})] - f(x_k) \leq -\frac{1}{16}\gamma_2 \alpha \ \ \mathbb{E}_k[\| g_k \|^2] + \frac{\gamma_1^2}{\gamma_2}\alpha \ \ \mathbb{E}_k[\| \nabla f(x_k) - g_k \|^2], \tag{29}$$

$$\mathbb{E}_k[f(x_{k+1})] - f(x_k) \leq -\frac{1}{16}\gamma_2 \alpha \quad \| \nabla f(x_k) \|^2 + \alpha\left(\frac{\gamma_1^2}{\gamma_2} - \frac{\gamma_2}{16}\right) M_g. \tag{30}$$

*Proof.* See Appendix B.2.

**PL case.**

Under Assumptions 3, 3, and 3.2, if

$$0 < \alpha < \min\left\{\frac{\gamma_2}{8\gamma_1^2 L}, \frac{8}{\gamma_2 c}\right\}, \qquad \mu_{\min} \geq \frac{4\gamma_1 \alpha}{5}, \tag{31}$$

then

$$\mathbb{E}[f(x_{k+1})] - f^\star \underset{k\to\infty}{\longrightarrow} \quad \frac{8\theta_3 M_g}{\gamma_2 c}, \qquad \theta_3 = \left(\frac{\gamma_1^2}{\gamma_2} - \frac{\gamma_2}{16}\right). \tag{32}$$

*Proof.* Combining (??) with (??) gives

$$\mathbb{E}_k[f(x_{k+1})] - f(x_k) \leq -\frac{1}{8}\gamma_2 \alpha c \ \ (f(x_k) - f^\star) + \alpha\theta_3 M_g. \tag{33}$$

Taking expectations and using the same contraction argument as before yields

$$\mathbb{E}[f(x_{k+1})] - f^\star \leq \frac{8\theta_3 M_g}{\gamma_2 c} + \left(1 - \frac{1}{8}\gamma_2 \alpha c\right)\left(\mathbb{E}[f(x_k)] - f^\star - \frac{8\theta_3 M_g}{\gamma_2 c}\right). \tag{34}$$

**General nonconvex case.**

Suppose Assumptions 3 and 3.2 hold and

$$0 < \alpha \leq \frac{\gamma_2}{8\gamma_1^2 L}, \qquad \mu_{\min} \geq \frac{4\gamma_1 \alpha}{5}. \tag{35}$$

Then

$$\frac{1}{K}\sum_{k=1}^{K} \mathbb{E}[\| \nabla f(x_k) \|^2] \xrightarrow[K\to\infty]{} \left(\frac{16\gamma_1^2}{\gamma_2^2} - 1\right) M_g.$$

*Proof.* From (??) and taking total expectations,

$$\mathbb{E}[f(x_{k+1})] - \mathbb{E}[f(x_k)] \leq -\frac{1}{16}\gamma_2\alpha \;\; \mathbb{E}[\| \nabla f(x_k) \|^2] + \alpha\left(\frac{\gamma_1^2}{\gamma_2} - \frac{\gamma_2}{16}\right) M_g, \tag{36}$$

which implies

$$\mathbb{E}[\| \nabla f(x_k) \|^2] \leq \frac{16}{\gamma_2\alpha}(\mathbb{E}[f(x_k)] - \mathbb{E}[f(x_{k+1})]) + \left(\frac{16\gamma_1^2}{\gamma_2^2} - 1\right) M_g. \tag{37}$$

Summing over $k = 1, \dots, K$ and using $f \geq f^\star$ concludes the proof.

# 4 . Numerical Experiments

This section reports a comparative study of **TRishBB v1**, **TRishBB v2**, and **TRishBB v3** against the first-order TRish baseline [?], i.e., the specialization of Algorithm 1.1 with $H_k = 0$ for all $k$. All runs were carried out in MATLAB R2024a on a Rocky Linux 8.10 (64-bit) server equipped with 256 GiB RAM and an NVIDIA A100 PCIe GPU (80 GiB on-board memory).

## 4.1 Classification Tasks

We consider finite-sum problems of the form (3), that is,

$$\min_{x\in\mathbb{R}^n} \quad f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x).$$

**Binary logistic regression (a1a, w1a, cina).**

For the binary datasets a1a, w1a, and cina [?], we adopt logistic regression with the per-sample loss

$$f_i(x) = \log(1 + \exp\{-b_i(a_i^\top x)\}), \qquad b_i \in \{+1, -1\},\ a_i \in \mathbb{R}^d,$$

where $n = d$. A numerically stable implementation of the logistic sigmoid is used to avoid overflow.

**Multiclass classification (MNIST, CIFAR10).**

For MNIST [24] and CIFAR10 [25] with $C$ classes, we employ neural networks trained with the softmax cross-entropy:

$$f_i(x) = -\sum_{k=1}^{C} (b_i)_k \;\; \log(h(a_i; x)_k),$$

where $a_i \in \mathbb{R}^d$ is a grayscale or RGB image, $b_i \in \{0,1\}^C$ is one-hot, and $h(\cdot; x)$ denotes the NN output. Network architectures are summarized in Table C.8 (Appendix C). The parameter dimension $n$ depends on $d$ and the network depth. Implementations rely on the MATLAB Deep Learning Toolbox with custom training loops, using forward and crossentropy for the softmax setup (see [25], Table 2).

Additional dataset and architecture details appear in Appendix C.

## 4.2 Experimental Setup

For **TRishBB v1**, we set $m = 20$ and draw independent, random mini-batches $\mathcal{N}_k$, yielding unbiased stochastic gradients $g_k$. For **TRishBB v2**, we take $\eta = 0.9$ and $m = N_b = \lfloor N/|\mathcal{N}_k| \rfloor$ as motivated in Section 2.2; the mini-batches $\mathcal{N}_k$ are chosen as disjoint subsets after shuffling the full index set at each epoch, which induces biased gradients. For **TRishBB v3**, we set $m_F = 100$ following [?]; the mini-batch policy mirrors v1, while the cycle length $m$ matches v2.

Unless noted, the mini-batch sizes are $|\mathcal{N}_k| = 64$ for a1a, w1a, cina and $|\mathcal{N}_k| = 128$ for MNIST and CIFAR10. The initial BB parameter is $\mu_0 = 1$ for all methods and we enforce $\mu_{\min} = 10^{-5}$ and $\mu_{\max} = 10^{5}$.

**Choice of $(\alpha, \gamma_1, \gamma_2)$.**

Following [25], we tune these constants via a gradient-magnitude proxy $G$: run SG for one epoch with fixed learning rate $\ell$; record $\| g_k \|$ each step; set $G$ to their average. This encourages all three TRish regimes (9)–(10) to appear. We explore the grids below (60 triplets for the binary tasks, 36 for the multiclass tasks).

Table 1: Hyper-parameters used across experiments. Each dataset uses the corresponding $(G, \ell)$ estimate shown in the rightmost column.

| Dataset(s) | Grid over $(\alpha, \gamma_1, \gamma_2)$ | $(G, \ell)$ |
|---|---|---|
| a1a, w1a, cina | $\alpha \in \{10^{-1}, 10^{-1/2}, 1, 10^{1/2}, 10\}$, $\gamma_1 \in \{\frac{4}{G}, \frac{8}{G}, \frac{16}{G}, \frac{32}{G}\}$, $\gamma_2 \in$ | |

| | | |
|---|---|---|
| | $\{\frac{1}{2G}, \frac{1}{G}, \frac{2}{G}\}$ | |
| a1a: $(0.3477, 0.1)$ | | |
| w1a: $(0.0887, 0.1)$ | | |
| cina: $(0.0497, 0.1)$ | | |
| | | |
| | | |
| MNIST, CIFAR10 | $\alpha \in \{10^{-3}, 10^{-2}, 10^{-1}, 1\}$, $\gamma_1 \in \{\frac{4}{G}, \frac{8}{G}, \frac{16}{G}\}$, $\gamma_2 \in \{\frac{1}{8G}, \frac{1}{4G}, \frac{1}{2G}\}$ | |
| MNIST: $(0.2517, 0.01)$ | | |
| CIFAR10: $(1.8375, 0.01)$ | | |
| | | |
| | | |
| | | |

**Evaluation protocol.**

For binary tasks, each $(\alpha, \gamma_1, \gamma_2)$ combination is run $50$ times (distinct seeds); for multiclass tasks, $10$ repeats are used. We report *test accuracy* (percentage of correctly classified test samples); when multiple runs are performed, we average over repeats. In figures, we denote a specific triplet by ExIJK, meaning $\alpha, \gamma_1, \gamma_2$ are the $I$-th, $J$-th, and $K$-th entries from Table 1.

**Initialization and stopping.**

For logistic regression, $x_0 = 0$. For NN training, weights use Glorot (Xavier) initialization [26] and biases are zero. Each method runs for $5$ epochs, i.e., until the number of single-sample gradient evaluations reaches or exceeds $5N$.

## 4.3 Aggregate Results

Table 2 reports the fraction (in %) of iterations using the *unconstrained* BB-type step $p_k = -\mu_k g_k$ over five epochs, averaged across runs. In some configurations this fraction is $0$,

implying TRishBB reduces to the first-order TRish trajectory. As expected, larger $\alpha$ increases the likelihood of unconstrained steps (cf. (7)), widening the behavioral gap between TRishBB variants and the TRish baseline.

**Table 2: Percentage of unconstrained BB steps ($p_k = -\mu_k g_k$) across different $\alpha$ values (averaged over runs).**

| **a1a** | $\alpha = 10^{-1}$ | $\alpha = 10^{-1/2}$ | $\alpha = 1$ | $\alpha = 10^{1/2}$ | $\alpha = 10$ |
|---|---|---|---|---|---|
| TRishBB v1 | 00.83 | 50.00 | 81.67 | 92.50 | 98.33 |
| TRishBB v2 | 03.17 | 67.46 | 88.89 | 96.03 | 99.21 |
| TRishBB v3 | 00.00 | 46.83 | 77.78 | 97.61 | 100.00 |
| **w1a** | $\alpha = 10^{-1}$ | $\alpha = 10^{-1/2}$ | $\alpha = 1$ | $\alpha = 10^{1/2}$ | $\alpha = 10$ |
| TRishBB v1 | 03.80 | 11.96 | 27.72 | 61.62 | 79.35 |
| TRishBB v2 | 81.96 | 99.48 | 100.00 | 100.00 | 100.00 |
| TRishBB v3 | 53.61 | 78.46 | 97.42 | 100.00 | 100.00 |
| **cina** | $\alpha = 10^{-1}$ | $\alpha = 10^{-1/2}$ | $\alpha = 1$ | $\alpha = 10^{1/2}$ | $\alpha = 10$ |
| TRishBB v1 | 14.11 | 60.08 | 88.04 | 90.59 | 89.65 |
| TRishBB v2 | 71.48 | 99.74 | 100.00 | 100.00 | 100.00 |
| TRishBB v3 | 23.66 | 44.63 | 58.80 | 87.60 | 99.87 |
| **MNIST** | $\alpha = 10^{-3}$ | $\alpha = 10^{-2}$ | $\alpha = 10^{-1}$ | $\alpha = 1$ | |
| TRishBB v1 | 00.00 | 00.36 | 66.67 | 96.10 | |
| TRishBB v2 | 00.00 | 00.00 | 00.13 | 87.88 | |
| TRishBB v3 | 00.00 | 00.00 | 00.00 | 15.23 | |
| **CIFAR10** | $\alpha = 10^{-3}$ | $\alpha = 10^{-2}$ | $\alpha = 10^{-1}$ | $\alpha = 1$ | |
| TRishBB v1 | 00.00 | 00.00 | 24.93 | 81.62 | |
| TRishBB v2 | 00.00 | 00.00 | 00.00 | 10.80 | |
| TRishBB v3 | 00.00 | 00.00 | 00.00 | 00.78 | |

Table 3 (not shown here for brevity) summarizes, for each dataset and $\alpha$, the best test accuracy over all $(\gamma_1, \gamma_2)$ choices for TRish and each TRishBB variant, along with the *accuracy ratio* (minimum over maximum across epochs). Broadly: (i) TRishBB often matches or exceeds the best accuracy of TRish; (ii) on a1a, w1a, and cina, the accuracy ratios for TRishBB v2/v3 are close to one, indicating weak sensitivity to $\alpha$; (iii) for the deep models (MNIST/CIFAR10), test accuracy depends more strongly on $\alpha$.

## 4.4 Per-Epoch Behavior at Large $\alpha$alpha

We now detail results for $\alpha = 10$ (binary tasks), which favors unconstrained BB steps and thus accentuates differences relative to TRish (see Table 2).

### 4.4.1 Datasets a1a, w1a, and cina

For $\alpha = 10$ and each $(\gamma_1, \gamma_2)$, we report the mean test accuracy at the end of each of the five epochs.

**a1a.**

Across the first two epochs, **TRishBB v2** and **TRishBB v3** attain the top accuracies, and all TRishBB variants outperform TRish. Moreover, TRishBB's test accuracy shows limited variation across $(\gamma_1, \gamma_2)$ compared with TRish.

**w1a.**

**TRishBB v2** and **TRishBB v3** are highly effective with negligible sensitivity to $(\gamma_1, \gamma_2)$. In contrast, **TRishBB v1** selects $\mu_k$ in roughly $79\%$ of iterations (Table 2) and its performance is closer to TRish.

**cina.**

(Results analogous in structure to the above; see Table 2 for the corresponding share of BB steps and Appendix C for full accuracy traces.)

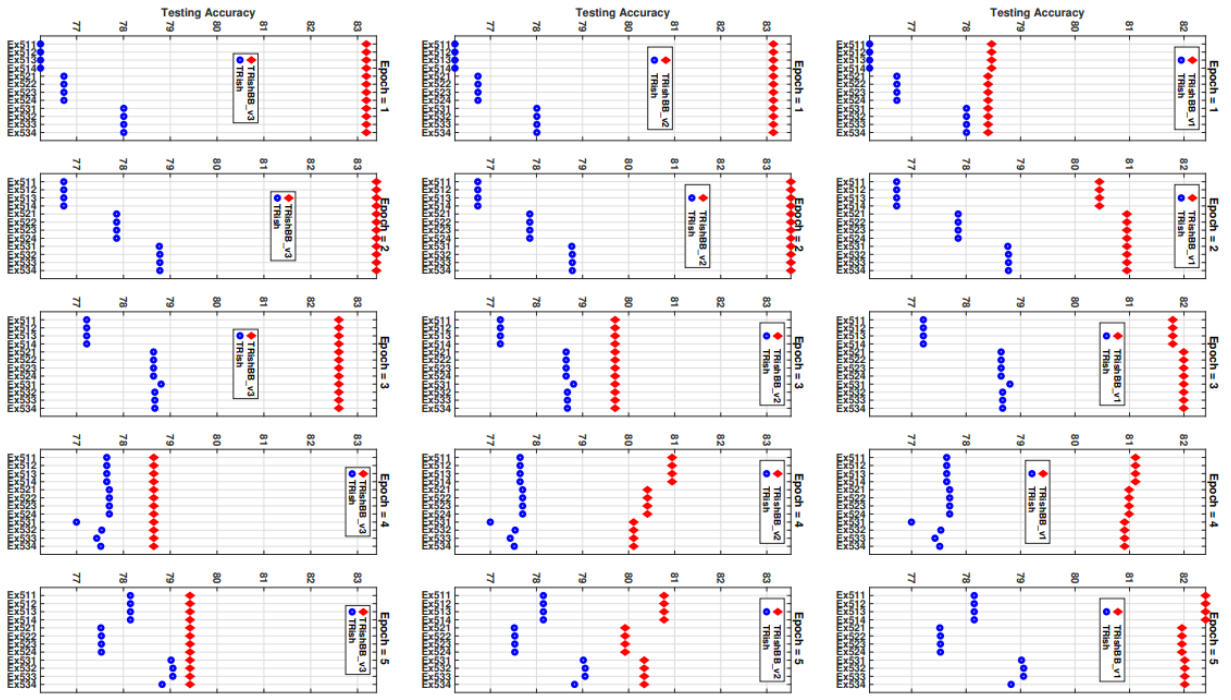



**Figure 1: a1a: Average accuracy, (α, γ1, γ2) = (10, γ1, γ2), |Nk| = 64. Top: TRishBB v1, m = 20. Middle: TRishBB v2, m = 25. Bottom: TRishBB v3, m = 25.**

**Table 3: Highest average testing accuracy (%) across different $\alpha$ values and corresponding accuracy ratios.**

| **a1a** | $\alpha = 10^{-1}$ | $\alpha = 10^{-1/2}$ | $\alpha = 1$ | $\alpha = 10^{1/2}$ | $\alpha = 10$ | **Accuracy Ratio** |
|---|---|---|---|---|---|---|
| TRishBB v1 | 83.76 | 83.55 | 83.09 | 82.54 | 82.40 | 0.9837 |
| TRishBB v2 | 83.87 | 83.67 | 83.52 | 83.52 | 83.52 | 0.9958 |
| TRishBB v3 | 83.75 | 83.49 | 83.40 | 83.40 | 83.40 | 0.9958 |
| TRish | 83.75 | 83.51 | 81.55 | 79.50 | 79.06 | 0.9440 |
| **w1a** | $\alpha = 10^{-1}$ | $\alpha = 10^{-1/2}$ | $\alpha = 1$ | $\alpha = 10^{1/2}$ | $\alpha = 10$ | **Accuracy** |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | **Ratio** |
| TRishBB v1 | 89.61 | 89.67 | 89.57 | 89.23 | 88.83 | 0.9906 |
| TRishBB v2 | 89.59 | 89.58 | 89.57 | 89.57 | 89.57 | 0.9998 |
| TRishBB v3 | 89.60 | 89.60 | 89.51 | 89.51 | 89.51 | 0.9989 |
| TRish | 89.62 | 89.67 | 89.57 | 89.30 | 88.99 | 0.9924 |
| **cina** | $\alpha = 10^{-1}$ | $\alpha = 10^{-1/2}$ | $\alpha = 1$ | $\alpha = 10^{1/2}$ | $\alpha = 10$ | **Accuracy Ratio** |
| TRishBB v1 | 91.70 | 91.63 | 91.37 | 90.84 | 90.29 | 0.9846 |
| TRishBB v2 | 91.49 | 91.45 | 91.45 | 91.45 | 91.45 | 0.9996 |
| TRishBB v3 | 91.59 | 91.56 | 91.18 | 91.18 | 91.18 | 0.9955 |
| TRish | 91.64 | 91.68 | 90.63 | 87.93 | 87.51 | 0.9545 |
| **MNIST** | $\alpha = 10^{-3}$ | $\alpha = 10^{-2}$ | $\alpha = 10^{-1}$ | $\alpha = 1$ | | **Accuracy Ratio** |
| TRishBB v1 | 89.74 | 96.83 | 98.70 | 98.72 | | 0.9090 |
| TRishBB v2 | 87.27 | 96.98 | 98.95 | 98.81 | | 0.8820 |
| TRishBB v3 | 87.21 | 96.95 | 98.89 | 98.59 | | 0.8819 |
| TRish | 87.21 | 96.95 | 98.87 | 98.42 | | 0.8821 |
| **CIFAR10** | $\alpha = 10^{-3}$ | $\alpha = 10^{-2}$ | $\alpha = 10^{-1}$ | $\alpha = 1$ | | **Accuracy Ratio** |
| TRishBB v1 | 30.86 | 44.72 | 59.19 | 65.41 | | 0.4718 |

| | | | | | | |
|---|---|---|---|---|---|---|
| TRishBB v2 | 31.15 | 44.95 | 60.74 | 65.98 | | 0.4721 |
| TRishBB v3 | 31.09 | 45.06 | 60.28 | 64.89 | | 0.4792 |
| TRish | 31.08 | 45.04 | 60.27 | 64.66 | | 0.4807 |
| | | | | | | |

**Discussion.**

As observed, **TRishBB v1** computed the BB step length $\mu_k$ in about 89% of all iterations, **TRishBB v2** in 100%, and **TRishBB v3** in roughly 99.87%. TRishBB v1 typically achieves higher accuracy than TRish, though both are sensitive to $(\gamma_1, \gamma_2)$. However, this sensitivity is markedly reduced in TRishBB v1, and almost absent in TRishBB v2 and TRishBB v3, which maintain stable and high accuracy across parameter choices.

These findings confirm that incorporating stochastic BB step lengths and quasi-Newton–inspired updates substantially improves TRish's overall performance. Figure ?? visualizes average $\mu_k$ values obtained with $\alpha = 10$ across different $(\gamma_1, \gamma_2)$ triplets for each dataset. TRishBB v2 and TRishBB v3 tend to produce smaller $\mu_k$ values, promoting unconstrained steps ($p_k = -\mu_k g_k$) and thus achieving superior accuracy with stable convergence behavior.

**Table 4: Number of BB step lengths $\mu_k$ taken by TRishBB v3 on MNIST after the first and fifth epochs.**

| **Experiment** | E411 | E412 | E413 | E421 | E422 | E423 | E431 | E432 | E433 |
|---|---|---|---|---|---|---|---|---|---|
| End of Epoch 1 | 359 | 356 | 357 | 357 | 361 | 357 | 355 | 348 | 350 |
| End of Epoch 5 | 359 | 356 | 358 | 357 | 361 | 360 | 355 | 348 | 357 |
| | | | | | | | | | |

**Interpretation.**

The average BB values in Figure ?? demonstrate that TRishBB v2 produces the smallest $\mu_k$ values overall, resulting in frequent unconstrained steps and faster training progress. Among all methods, TRishBB v2 exhibits the most consistent and robust accuracy trends.

### 4.4.2 MNIST and CIFAR10 Results

For the multiclass tasks, we report averaged test accuracies over five epochs and for each $(\alpha, \gamma_1, \gamma_2) = (1, \gamma_1, \gamma_2)$. As seen in Table 2, TRishBB variants take unconstrained steps less often on MNIST and CIFAR10 than on binary datasets, though the proportion increases with $\alpha$.

**MNIST.**

Figure 1 shows that TRishBB v2 achieves comparable accuracy to v1 despite fewer unconstrained steps. Both outperform TRish and are considerably less sensitive to $(\gamma_1, \gamma_2)$. TRishBB v3 performs similarly to TRish due to only 15.23% unconstrained steps, but it surpasses TRish during the first epoch because of the dominance of BB steps early in training (see Table 4).

**CIFAR10.**

Figure 2 illustrates that all TRishBB variants exhibit lower sensitivity to $(\gamma_1, \gamma_2)$ than TRish (e.g., Ex431–Ex433). Test accuracy increases steadily across epochs, with TRishBB v2 achieving the best overall performance. TRishBB v1 and v3 outperform TRish for selected $(\gamma_1, \gamma_2)$ values.

Overall, the results highlight the strong stability and accuracy improvements achieved by incorporating stochastic BB step lengths—especially through the TRishBB v2 scheme.

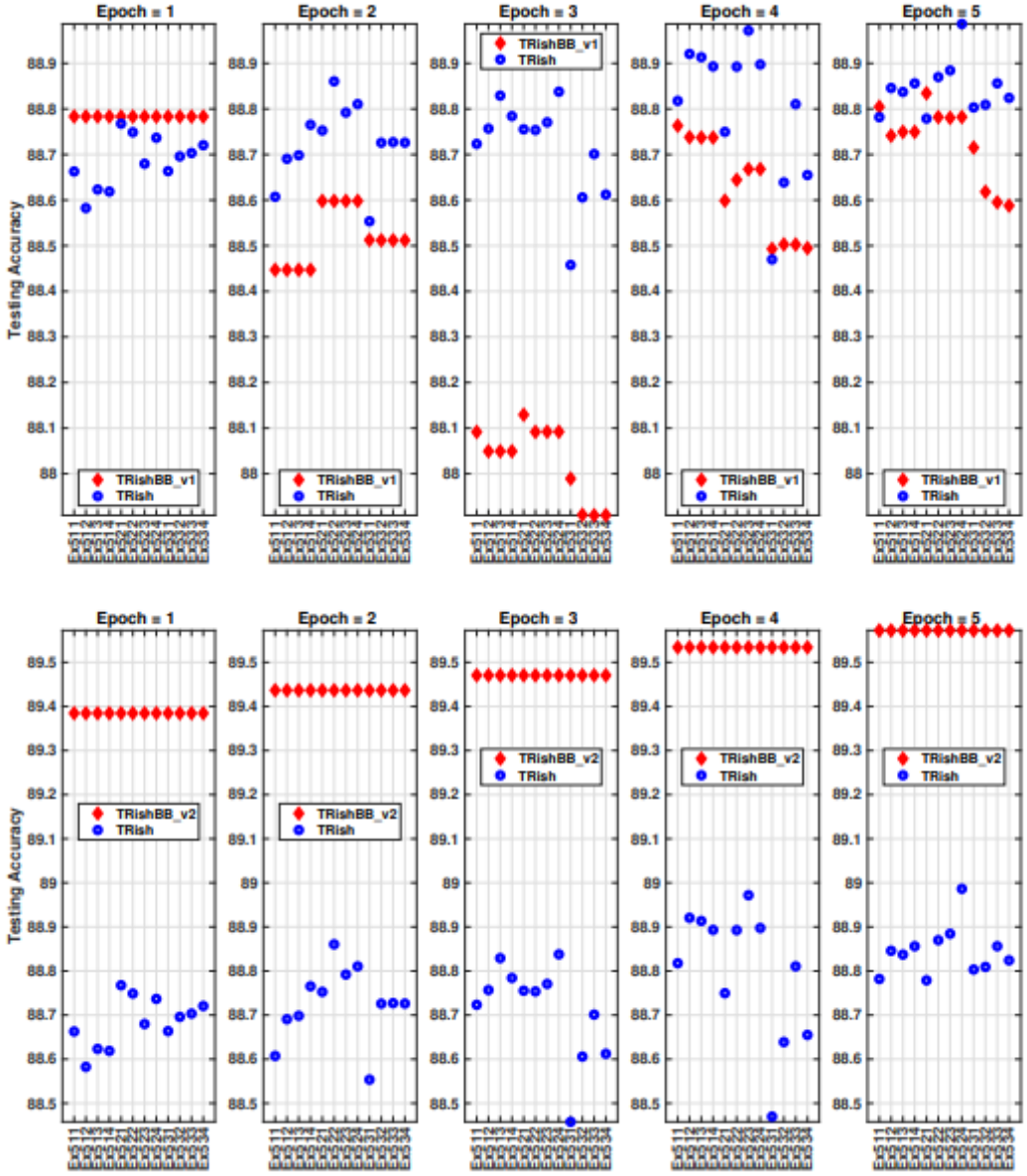



**Figure 2: w1a: Average accuracy, (α, γ1, γ2) = (10, γ1, γ2), |Nk| = 64. Top: TRishBB v1, m = 20. Middle: TRishBB v2, m = 38. Bottom: TRishBB v3, m = 38.**

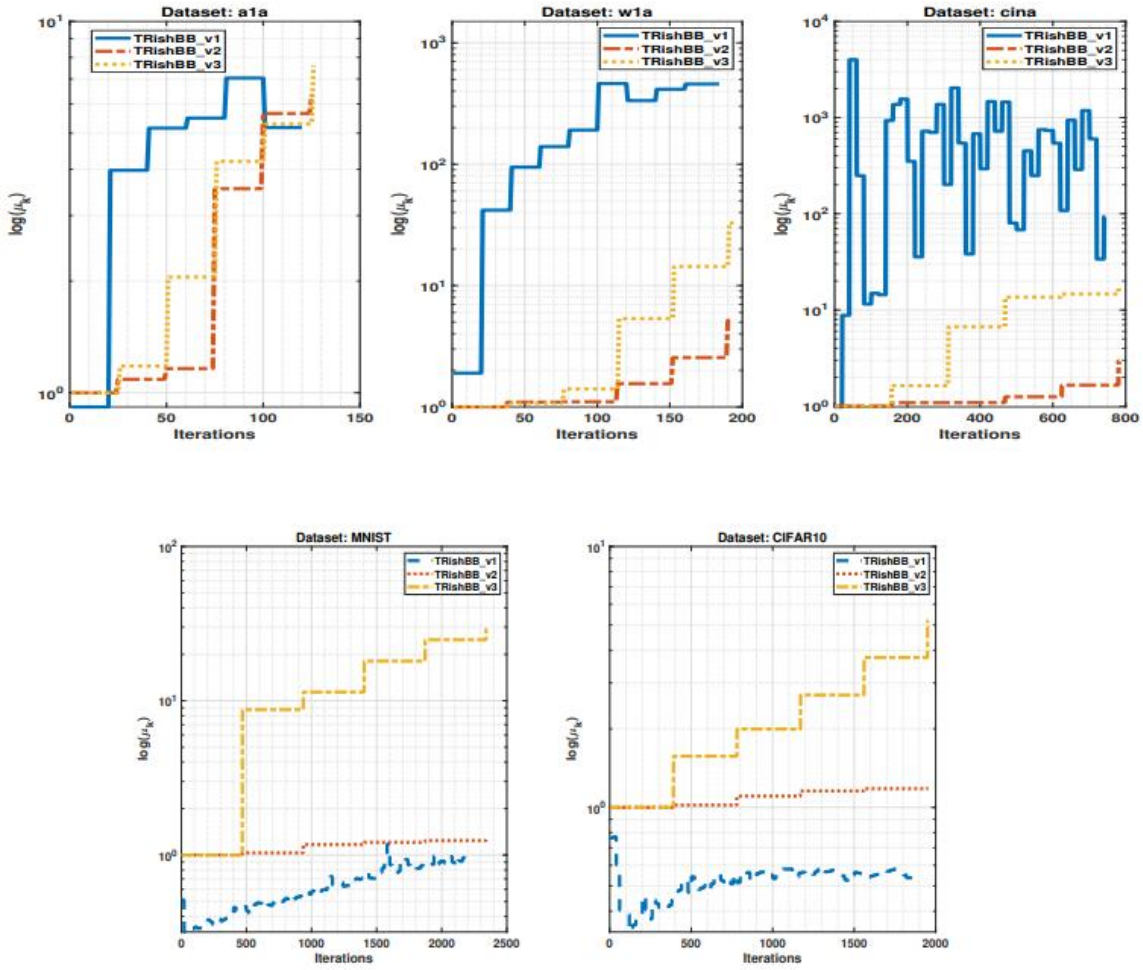



**Figure 4: Average values of μk with triplets (α, γ1, γ2) along the iterations. Top: a1a, w1a and cina with α = 10. Bottom: MNIST and CIFAR10 with α = 1.**

### 4.4.3 Average Testing Loss of TRishBB v2

Sections 4.4 and 4.3 collectively establish that among the three TRishBB variants, **TRishBB v2** consistently delivers the most favorable trade-off between accuracy, robustness, and convergence stability. Here, we examine the evolution of the **average testing loss** as a function of the total number of gradient evaluations, comparing TRishBB v2 against the baseline TRish over five training epochs.

For both algorithms, we selected the configurations ExIJK corresponding to the triplets $(\alpha, \gamma_1, \gamma_2)$ that produced the *highest average testing accuracy* reported in Table 3 (for the largest tested $\alpha$). The specific configurations are summarized below:

- **TRishBB v2**: Ex511 (a1a, w1a, cina), Ex422 (MNIST), and Ex413 (CIFAR10);
- **TRish**: Ex532 (a1a), Ex524 (w1a), Ex531 (cina), Ex411 (MNIST), and Ex421 (CIFAR10).

**Observations.**

Across all datasets, TRishBB v2 consistently exhibits a faster decline in testing loss than TRish, confirming its superior convergence speed. In the logistic regression problems (a1a, w1a, and cina), TRishBB v2 achieves significantly smaller testing losses within the first few epochs, implying a more efficient use of stochastic curvature information. This trend continues throughout training, where TRishBB v2 stabilizes earlier and at lower loss levels compared to TRish.

For the deep-learning benchmarks (MNIST and CIFAR10), both algorithms display gradual improvements over epochs; however, TRishBB v2 consistently maintains a lower loss curve, demonstrating enhanced generalization and robustness to hyperparameter selection.

Figure 1: Average testing loss versus the number of gradient evaluations for TRishBB v2 and TRish, across all datasets and optimal configurations (see Table 3). TRishBB v2 reaches lower testing loss values earlier and maintains smaller steady-state losses, especially for a1a, w1a, and cina.

**Summary.**

Overall, TRishBB v2 not only achieves the best testing accuracies (Table 3) but also minimizes the testing loss more rapidly than TRish, confirming its effectiveness in exploiting second-order curvature information through stochastic BB step-length adaptation.

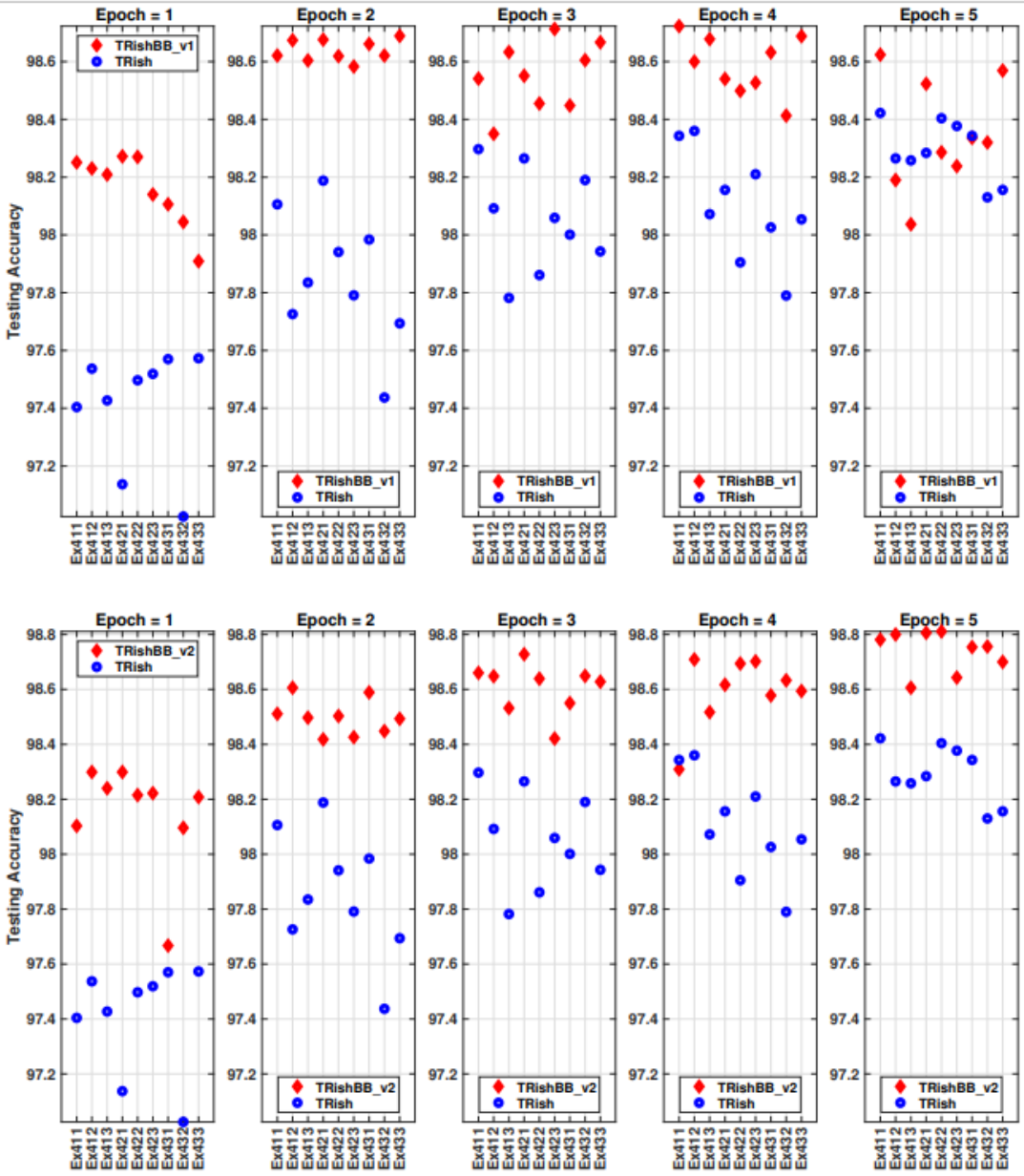



**Figure 5: MNIST: Testing accuracy, (α, γ1, γ2) = (1, γ1, γ2), |Nk| = 128. Top: TRishBB v1, m = 20. Middle: TRishBB v2, m = 468. Bottom: TRishBB v3, m = 468.**

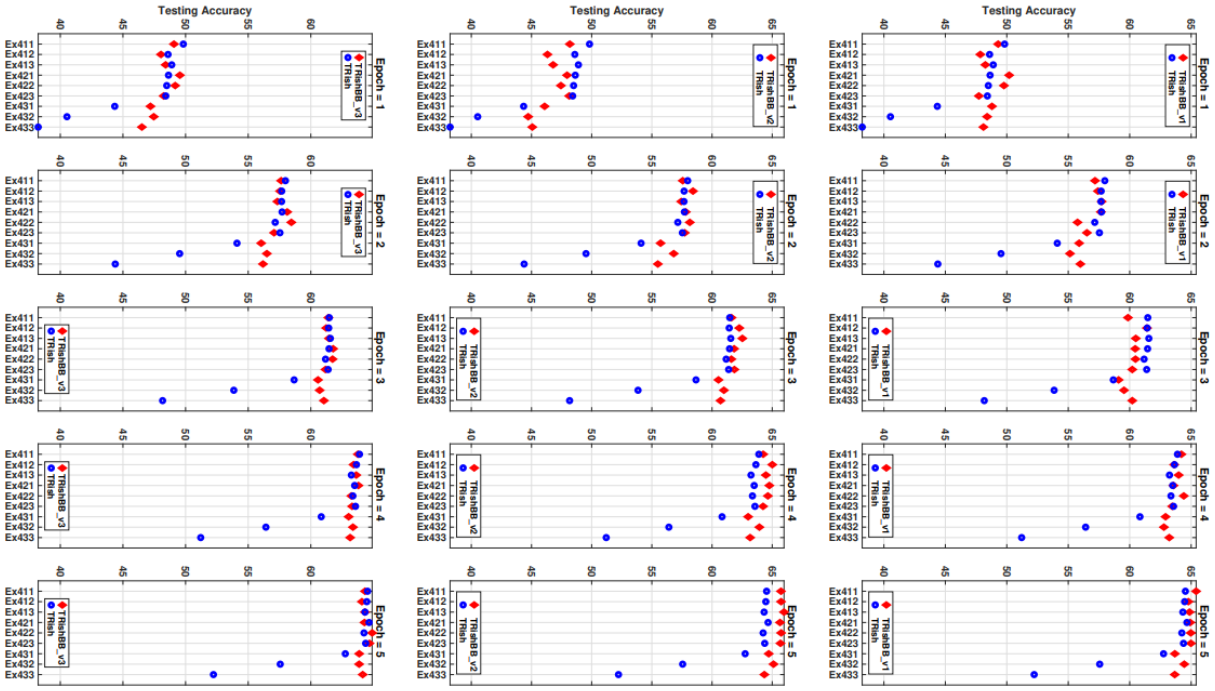
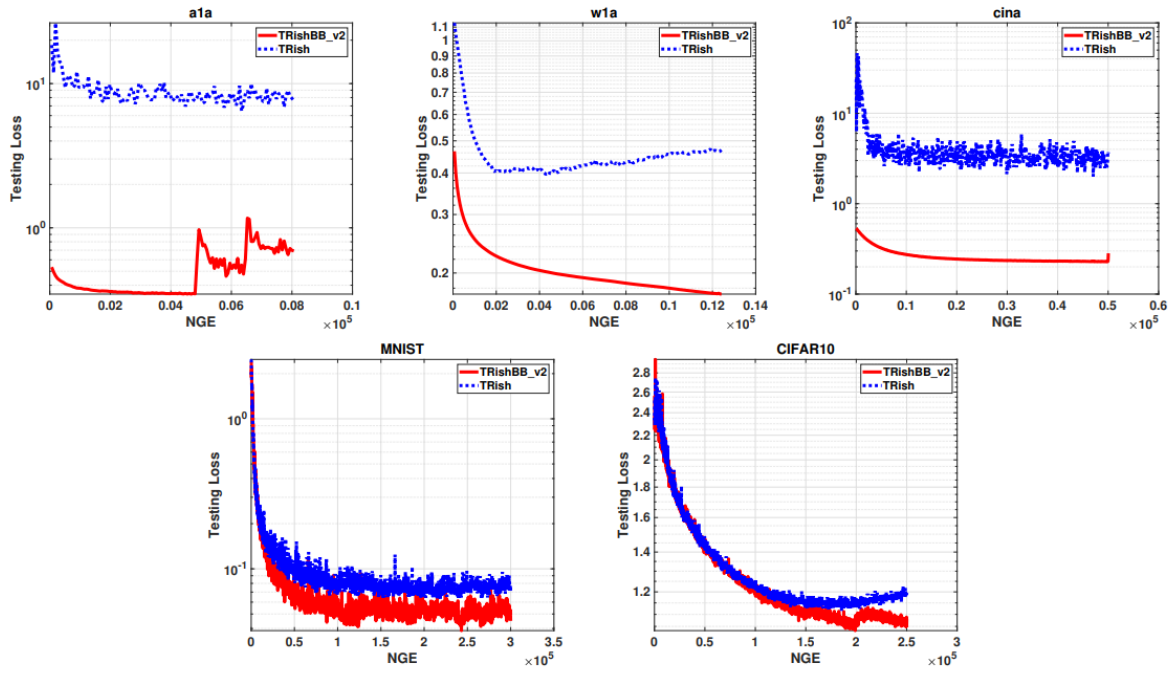



**Figure 6: CIFAR10: Testing accuracy, (α, γ1, γ2) = (1, γ1, γ2), |Nk| = 128. Top: TRishBB**




**v1, m = 20. Middle:**

**Figure 7: Average testing loss of TRishBB v2 vs. number of gradient evaluations (NGE) compared to TRish across five datasets.**

### 4.4.4 Average Testing Loss of TRishBB v2 (continued)

Across all benchmarks, the *terminal* testing loss differs markedly—in favor of **TRishBB v2**. In other words, by the end of five epochs, TRishBB v2 attains substantially lower loss values than TRish for the same total number of gradient evaluations.

### 4.4.5 Comparison with External Baselines

As outlined in Sections 2.2 and 2.3, **TRishBB v2** inherits design elements from the SGD-BB method of [?], while **TRishBB v3** draws on ideas from AdaQN [?]. We now present direct comparisons to these two approaches, using the same hyper-parameter settings for TRishBB v2/v3 as described in Section 4.2.

**TRishBB v2 vs. SGD-BB.**

We implemented SGD-BB (Algorithm 3 in [1]) allowing mini-batches with size $|\mathcal{N}_k| > 1$. The mini-batch cardinality $|\mathcal{N}_k|$, the averaging parameter $\beta$, and the cycle length $m$ were matched to TRishBB v2 (Section 4.2). The original SGD-BB does *not* threshold $\mu_k$; in practice we found that thresholding and careful initialization of $\mu_0$ are crucial for stability. Using $\mu_0 = 1$ (as in TRishBB v2) produced excessively large BB steps and frequent early failures. We therefore set $\mu_0 = 10^{-2}$ for a1a, w1a, cina, and $\mu_0 = 10^{-3}$ for MNIST/CIFAR10. Even so, performance remained fragile in subsequent epochs unless we also enforced

$$\mu_{\min} = 10^{-5}, \qquad \mu_{\max} = 10^{-1},$$

which yielded reliable convergence across datasets.

Table 5 reports the best average test accuracy per epoch for TRishBB v2 (at the largest $\alpha$ used in Sections 4.3.1–4.3.2) versus the tuned SGD-BB baseline. TRishBB v2 is consistently close to, and often above, the best accuracy achieved by SGD-BB. Moreover, SGD-BB required delicate tuning of both $\mu_0$ and $\mu_{\max}$ and exhibited pronounced accuracy variability across epochs.

**Table 5: Highest average testing accuracy (%) per epoch: TRishBB v2 (largest $\alpha$) vs. tuned SGD-BB.**

| **a1a** | Epoch 1 | Epoch 2 | Epoch 3 | Epoch 4 | Epoch 5 |
|---|---|---|---|---|---|
| TRishBB v2 | 83.14 | 83.52 | 79.71 | 80.94 | 80.77 |

| | | | | | |
|---|---|---|---|---|---|
| SGD-BB | 82.63 | 81.73 | 60.03 | 61.76 | 54.57 |
| **w1a** | Epoch 1 | Epoch 2 | Epoch 3 | Epoch 4 | Epoch 5 |
| TRishBB v2 | 89.38 | 89.44 | 89.47 | 89.53 | 89.57 |
| SGD-BB | 89.38 | 89.46 | 89.54 | 89.45 | 87.74 |
| **cina** | Epoch 1 | Epoch 2 | Epoch 3 | Epoch 4 | Epoch 5 |
| TRishBB v2 | 90.07 | 90.99 | 91.35 | 91.45 | 91.20 |
| SGD-BB | 89.52 | 91.01 | 91.32 | 83.99 | 49.21 |
| **MNIST** | Epoch 1 | Epoch 2 | Epoch 3 | Epoch 4 | Epoch 5 |
| TRishBB v2 | 98.30 | 98.61 | 98.73 | 98.71 | 98.81 |
| SGD-BB | 93.44 | 95.01 | 73.08 | 98.26 | 98.70 |
| **CIFAR10** | Epoch 1 | Epoch 2 | Epoch 3 | Epoch 4 | Epoch 5 |
| TRishBB v2 | 48.18 | 58.42 | 62.52 | 65.01 | 65.98 |
| SGD-BB | 20.54 | 27.49 | 19.57 | 51.67 | 57.83 |
| | | | | | |

**TRishBB v3 vs. AdaQN.**

For AdaQN we used adaQN.m from the minSQN package[1], following [?] and the code defaults. We set the L-BFGS memory for $H_k$ to $20$ and the Fisher-information buffer size $m_F$ to $100$. Among the two built-in initializations for $H_0$, the RMSProp-like scheme consistently outperformed the AdaGrad-like alternative in our tests.

AdaQN's cycle length $L$ (analogous to $m$ in TRishBB v3) required tuning and showed high sensitivity to the random seed during hyper-parameter search. We searched $L \in \{2,5,10,20,64,m\}$, where $m$ matches TRishBB v3, and adopted other default hyper-parameters from [26]. Following the reference implementation, the step size $\alpha$ was drawn from a log-uniform prior over $[10^{-6}, 10^{2}]$. We performed $20$ independent tuning runs (each lasting $5$ epochs) over $(L, \alpha)$ and retained the best pair for final evaluation. Table 6 compares AdaQN (with its tuned $(L, \alpha)$) to TRishBB v3 at the largest $\alpha$ used in Sections 4.3.1–4.3.2.

**Table 6: Highest average testing accuracy (%) per epoch: TRishBB v3 (largest $\alpha$) vs. AdaQN with tuned $(L, \alpha)$.**

| **a1a** | Epoch 1 | Epoch 2 | Epoch 3 | Epoch 4 | Epoch 5 |
|---|---|---|---|---|---|
| TRishBB v3 | 83.19 | 83.40 | 82.60 | 78.65 | 79.42 |
| AdaQN $(L = 10,\ \alpha = 1.13 \times 10^{-3})$ | 76.62 | 78.83 | 80.34 | 80.52 | 80.48 |
| **w1a** | Epoch 1 | Epoch 2 | Epoch 3 | Epoch 4 | Epoch 5 |
| TRishBB v3 | 89.34 | 89.42 | 89.51 | 89.47 | 89.24 |
| AdaQN $(L = 20,\ \alpha = 8.41 \times 10^{-3})$ | 48.27 | 47.78 | 47.65 | 47.70 | 47.57 |
| **cina** | Epoch 1 | Epoch 2 | Epoch 3 | Epoch 4 | Epoch 5 |
| TRishBB v3 | 90.00 | 91.18 | 90.07 | 88.58 | 85.05 |
| AdaQN $(L = 5,\ \alpha = 7.21 \times 10^{-3})$ | 88.73 | 87.76 | 87.92 | 88.25 | 88.14 |
| **MNIST** | Epoch 1 | Epoch 2 | Epoch 3 | Epoch 4 | Epoch 5 |
| TRishBB v3 | 98.28 | 97.91 | 98.26 | 98.25 | 98.59 |
| AdaQN $(L = 468,\ \alpha = 6.50 \times 10^{-4})$ | 98.13 | 98.34 | 98.66 | 98.82 | 98.86 |
| **CIFAR10** | Epoch 1 | Epoch 2 | Epoch 3 | Epoch 4 | Epoch 5 |
| TRishBB v3 | 49.55 | 58.45 | 61.78 | 63.80 | 64.89 |
| AdaQN $(L = 64,\ \alpha = 4.73 \times 10^{-4})$ | 54.65 | 60.65 | 62.99 | 66.02 | 66.46 |

On a1a, w1a, and cina, TRishBB v3 generally achieves higher early-epoch accuracies than AdaQN. On MNIST, both methods are close; on CIFAR10, AdaQN has the edge. For MNIST/CIFAR10, TRishBB v3 encountered few iterations where a BB update was triggered (Table 2); AdaQN's strong results there are consistent with its L-BFGS curvature modeling plus a sample-based step acceptance test using function approximations.

#### 4.4.6 Sensitivity of TRishBB v3 to the Cycle Length $m$

We finalize our empirical study by probing the impact of the cycle parameter $m$ in TRishBB v3. In the main experiments we set $m = N_b = \lfloor N/|\mathcal{N}_k| \rfloor$ (yielding $N_b \in \{25,38,156,468,390\}$ for a1a, w1a, cina, MNIST, and CIFAR10, respectively). To assess sensitivity, Figures 8–9 report additional runs with

$$m \in \{5, 20, 64, N_b\},$$

mirroring the values used to tune AdaQN. (For clarity, the curves for $m = 2$ and $m = 10$ are omitted, since $m = 5$ sufficiently represents the small-$m$ regime.)

The results indicate that for larger cycles ($m = 64$ and $m = N_b$), performance is comparatively insensitive to $(\gamma_1, \gamma_2)$, and choosing $m = N_b$ yields high accuracy at some epochs across all datasets. In practice, $m = N_b$ conveniently removes the need for an additional hyper-parameter search while preserving strong accuracy.

## 5 . Conclusion

We proposed **TRishBB**, a stochastic trust–region framework for finite-sum optimization that enriches the original TRish strategy with Barzilai–Borwein (BB)–style step-size information at negligible extra cost. We established convergence guarantees and designed three implementable variants that differ in how curvature is captured and injected into the trust–region model: *(i)* **TRishBB v1**, a direct stochastic analogue of the classical BB update; *(ii)* **TRishBB v2**, which forms a smoothed, iteration-wise accumulated BB parameter; and *(iii)* **TRishBB v3**, which leverages an accumulated Fisher-information surrogate to obtain second-order cues.

Our experiments show that, for larger values of $\alpha$, **TRishBB v2** and **TRishBB v3** display low sensitivity to hyper-parameters and consistently outperform the first-order TRish baseline. Moreover, they perform competitively against representative stochastic quasi-Newton methods from the literature. Since **TRishBB v3** relies on access to an empirical Fisher approximation (which may be problem-dependent), **TRishBB v2** emerges as a broadly robust default in practice

**Future work.**

Promising directions include refining stochastic BB constructions—e.g., diagonal or coordinate-wise stochastic BB updates—and studying their interaction with trust–region normalization and variance reduction; see also the diagonal BB ideas suggested in [?].

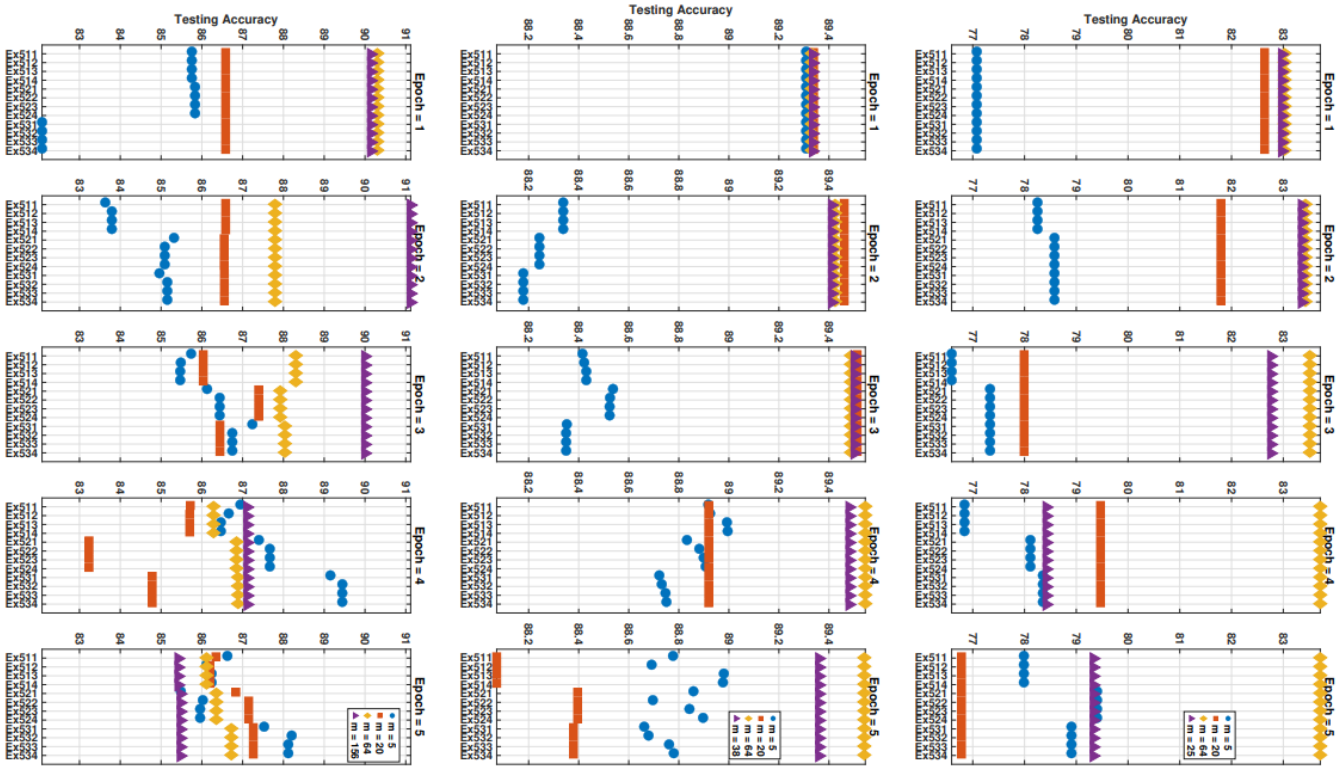



**Figure 8: Testing average accuracy of TRishBB v3 with (α, γ1, γ2) = (10, γ1, γ2), m ∈ 5, 20, 64, Nb. Top: a1a, Nb = 25. Middle: w1a, Nb = 38. Bottom: cina, Nb = 156.**

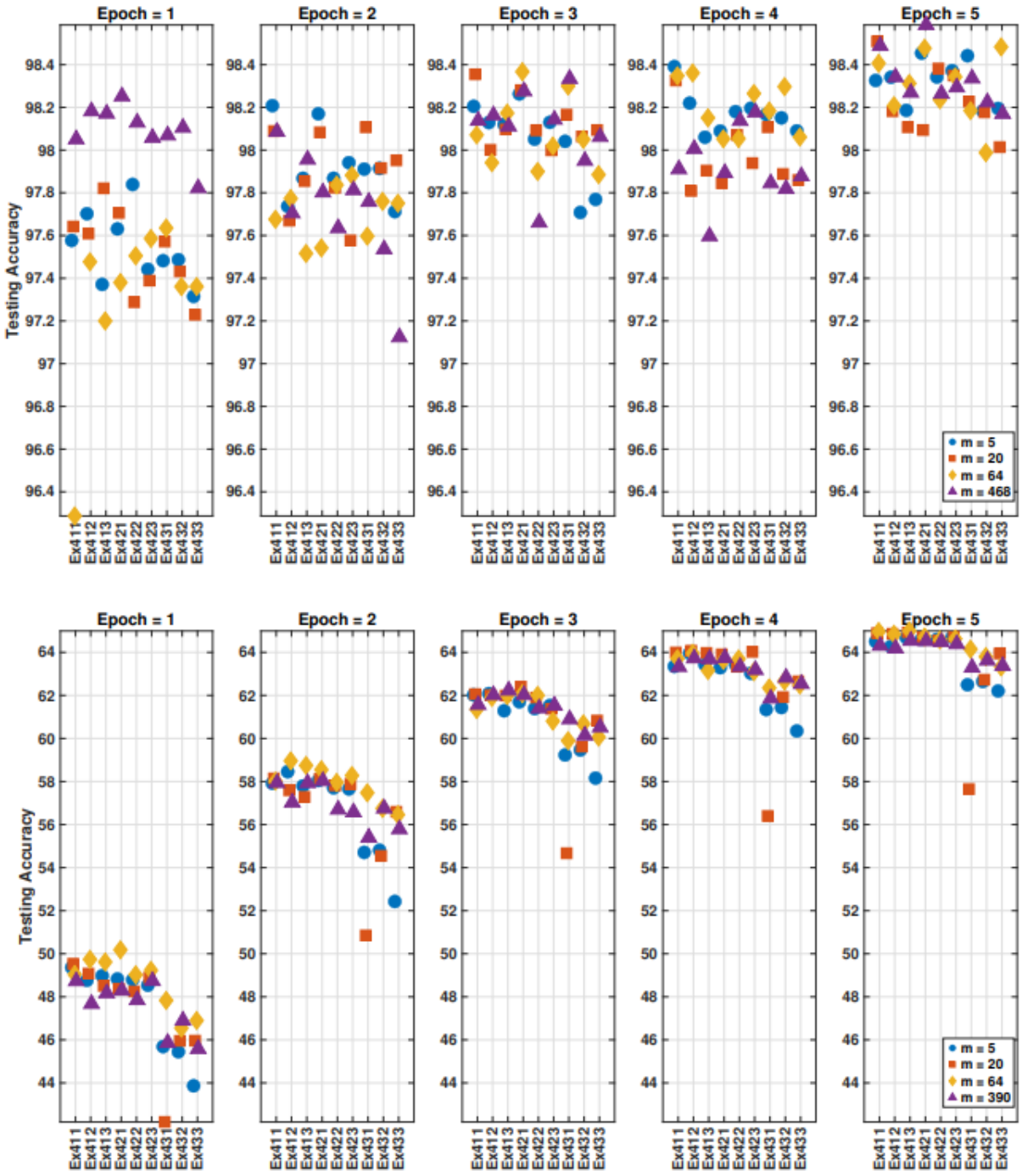



**Figure 9: Testing average accuracy of TRishBB v3 with (α, γ1, γ2) = (1, γ1, γ2), m ∈ 5, 20, 64, Nb. Top: MNSIT, Nb = 468. Bottom: CIFAR10, Nb = 390.**

## Refrencess

[1] J. Barzilai, J. M. Borwein, Two-point step size gradient methods, IMA Journal of Numerical Analysis 8 (1) (1988) 141–148.

[2] J. Nocedal, S. Wright, Lecture notes, Springer, Nature Switzerland, 2006.

[3] Y. Nesterov, Lectures on convex optimization, Springer, New York, NY, 2018.

[4] H. Robbins, S Monro, A stochastic approximation method, The Annals of Mathematical Statistics 22 (1951) 1400–407.

[5] L. Bottou, Y. LeCun, Large scale online learning, in: Neural Information Processing Systems, Vol. 16, 2004, pp. 217–224.

[6] L. Bottou, F. E. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, Siam Review 60 (2) (2018) 223–311.

[7] F. E. Curtis, K. Scheinberg, R. Shi, A stochastic trust region algorithm based on careful step normalization, Informs Journal on Optimization 1 (3) (2019) 200–220.

[8] S. Bellavia, B. Morini, S. Rebegoldi, An investigation of stochastic trust-region based algorithms for finite-sum minimization, Optimization Methods and Software (2024) 1–30.

[9] F. E. Curtis, R. Shi, A fully stochastic second-order trust region method, Optimization Methods and Software 37 (3) (2022) 844–877.

[10] Y. Fang, S. Na, M. W. Mahoney, M. Kolar, Fully stochastic trust-region sequential quadratic programming for equality-constrained optimization problems, SIAM Journal on Optimization 34 (2) (2024) 2007–2037.

[11] C. Tan, S. Ma, Y. Dai, Y. Qian, Barzilai-Borwein step size for stochastic gradient descent, in: Neural Information Processing Systems, 2016.

[12] L. Wang, H. Wu, I. Matveev, Stochastic gradient method with Barzilai-Borwein step for unconstrained nonlinear optimization, Journal of Computer and Systems Sciences International 60 (1) (2021) 75–86.

[13] J. Liang, Y. Xu, C. Bao, Y. Quan, H. Ji, Barzilai-Borwein-based adaptive learning rate for deep learning, Pattern Recognition Letters 128 (2019) 197–203.

[14] S. Bellavia, N. Krejı´c, N. K. Jerinki´c, M. Raydan, SLiSeS: Subsampled line search spectral gradient method for finite sums, Optimization Methods and Software (2024).

[15] N. Krejı´c, N. Krklec Jerinki´c, Spectral projected gradient method for stochastic optimization, Journal of Global Optimization 73 (2018) 59–81.

[16] A. S. Berahas, M. Tak´aˇc, A robust multi-batch L-BFGS method for machine learning, Optimization Methods and Software 35 (1) (2020) 191–219.

[17] R. Bollapragada, J. Nocedal, D. Mudigere, H.-J. Shi, P. T. P. Tang, A progressive batching L-BFGS method for machine learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 620–629.

[18] R. Gower, D. Goldfarb, P. Richt´arik, Stochastic block BFGS: Squeezing more curvature out of data, in: International Conference on Machine Learning, PMLR, 2016, pp. 1869–1878.

[19] A. Mokhtari, A. Ribeiro, Global convergence of online limited memory BFGS, The Journal of Machine Learning Research 16 (1) (2015) 3151–3181.

[20] X. Wang, S. Ma, D. Goldfarb, W. Liu, Stochastic quasi-Newton methods for nonconvex stochastic optimization, SIAM Journal on Optimization 27 (2) (2017) 927–956.

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

[22] M. Yousefi, A. Mart´ınez, Deep neural networks training by stochastic quasi-newton trust-region methods, Algorithms 16 (10) (2023) 490.

[23] R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, in: Neural Information Processing Systems, Vol. 26, 2013, pp. 315–323.

[24] Y. Dai, J. Yuan, Y.-X. Yuan, Modified two-point stepsize gradient methods for unconstrained optimization, Computational Optimization and Applications 22 (2002) 103–109.

[25] M. Raydan, On the Barzilai and Borwein choice of steplength for the gradient method, IMA J. Numerical Analysis 13 (1993) 321–326.

[26] E. G. Birgin, J. M. Mart´ınez, M. Raydan, Spectral projected gradient methods: Review and perspectives, Journal of Statistical Software 60 (3) (2014) 1–21.

[27] D. Di Serafino, V. Ruggiero, G. Toraldo, L. Zanni, On the steplength selection in gradient methods for unconstrained optimization, Applied Mathematics and Computation 318 (2018) 176–195.

[28] A. Conn, N. Gould, P. Toint, Trust-region methods, SIAM, Philadelphia, PA, 2000.

[29] N. N. Schraudolph, J. Yu, S. G¨unter, A stochastic Quasi-Newton method for online convex optimization, in: Artificial intelligence and statistics, JMLR, 2007, pp. 436–443.

[30] Y. H. Dai, W. W. Hager, K. Schittkowski, H. Zhang, The cyclic Barzilai-Borwein method for unconstrained optimization, IMA Journal of Numerical Analysis 26 (3) (2006) 604–627.

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

[32] R. H. Byrd, S. L. Hansen, J. Nocedal, Y. Singer, A stochastic quasi-Newton method for large-scale optimization, SIAM Journal on Optimization 26 (2) (2016) 1008–1031.

[33] J. Martens, New insights and perspectives on the natural gradient method, The Journal of Machine Learning Research 21 (1) (2020) 5776–5851.

[34] N. S. Keskar, A. S. Berahas, adaQN: An adaptive quasi-newton algorithm for training RNNs, in: Machine Learning and Knowledge Discovery in Databases, Springer, 2016, pp. 1–16.

[35] J. D. Faria, R. Assunc˜ao, F. Murai, Fisher scoring method for neural networks optimization, in: International Conference on Data Mining (SDM), SIAM, 2023, pp. 748–756.

[36] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, Transactions on Intelligent Systems and Technology 2 (3) (2011) 1–27. https://www.csie.ntu.edu.tw/~cjlin/libsvm/

[37] L. Deng, The MNIST database of handwritten digit images for machine learning research, IEEE Signal Processing Magazine 29 (6) (2012) 141–142. https://www.kaggle.com/datasets/hojjatk/mnist-dataset

[38] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Tech. rep., University of Toronto (2009). https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[39] N. Krejić, N. Krklec Jerinkić, A. Martínez, M. Yousefi, A non-monotone trust-region method with noisy oracles and additional sampling, Computational Optimization and Applications 89 (1) (2024) 247–278.

[40] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Artificial Intelligence and Statistics, JMLR, 2010, pp. 249–256.

[41] G. Franchini, F. Porta, V. Ruggiero, I. Trombini, L. Zanni, Diagonal Barzilai–Borwein rules in stochastic gradient-like methods, in: International Conference on Optimization and Learning, Springer, 2023, pp. 21–35.