A Progressive Batching L-BFGS Method for Machine Learning

Raghu Bollapragada [1] Dheevatsa Mudigere [2] Jorge Nocedal [1] Hao-Jun Michael Shi [1] Ping Tak Peter Tang [3]

Abstract

The standard L-BFGS method relies on gradient approximations that are not dominated by noise, so that search directions are descent directions, the line search is reliable, and quasi-Newton updating yields useful quadratic models of the objective function. All of this appears to call for a full batch approach, but since small batch sizes give rise to faster algorithms with better generalization properties, L-BFGS is currently not considered an algorithm of choice for large-scale machine learning applications. One need not, however, choose between the two extremes represented by the full batch or highly stochastic regimes, and may instead follow a progressive batching approach in which the sample size increases during the course of the optimization. In this paper, we present a new version of the L-BFGS algorithm that combines three basic components — progressive batching, a stochastic line search, and stable quasi-Newton updating — and that performs well on training logistic regression and deep neural networks. We provide supporting convergence theory for the method.

1. Introduction

The L-BFGS method (Liu & Nocedal, 1989) has traditionally been regarded as a batch method in the machine learning community. This is because quasi-Newton algorithms need gradients of high quality in order to construct useful quadratic models and perform reliable line searches. These algorithmic ingredients can be implemented, it seems, only by using very large batch sizes, resulting in a costly iteration that makes the overall algorithm slow compared with stochastic gradient methods (Robbins & Monro, 1951).

Even before the resurgence of neural networks, many researchers observed that a well-tuned implementation of the stochastic gradient (SG) method was far more effective on large-scale logistic regression applications than the batch L-BFGS method, even when taking into account the advantages of parallelism offered by the use of large batches. The preeminence of the SG method (and its variants) became more pronounced with the advent of deep neural networks, and some researchers have speculated that SG is endowed with certain regularization properties that are essential in the minimization of such complex nonconvex functions (Hardt et al., 2015; Keskar et al., 2016).

In this paper, we postulate that the most efficient algorithms for machine learning may not reside entirely in the highly stochastic or full batch regimes, but should employ a progressive batching approach in which the sample size is initially small, and is increased as the iteration progresses. This view is consistent with recent numerical experiments on training various deep neural networks (Smith et al., 2017; Goyal et al., 2017), where the SG method, with increasing sample sizes, yields similar test loss and accuracy as the standard (fixed mini-batch) SG method, while offering significantly greater opportunities for parallelism.

Progressive batching algorithms have received much attention recently from a theoretical perspective. It has been shown that they enjoy complexity bounds that rival those of the SG method (Byrd et al., 2012), and that they can achieve a fast rate of convergence (Friedlander & Schmidt, 2012). The main appeal of these methods is that they inherit the efficient initial behavior of the SG method, offer greater opportunities to exploit parallelism, and allow for the incorporation of second-order information. The latter can be done efficiently via quasi-Newton updating.

An integral part of quasi-Newton methods is the line search, which ensures that a convex quadratic model can be constructed at every iteration. One challenge that immediately arises is how to perform this line search when the objective function is stochastic. This is an issue that has not received sufficient attention in the literature, where stochastic line searches have been largely dismissed as inappropriate. In this paper, we take a step towards the development

[1] Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA [2] Intel Corporation, Bangalore, India [3] Intel Corporation, Santa Clara, CA, USA. Correspondence to: Raghu Bollapragada <raghu.bollapragada@u.northwestern.edu>, Dheevatsa Mudigere <dheevatsa.mudigere@intel.com>, Jorge Nocedal <j-nocedal@northwestern.edu>, Hao-Jun Michael Shi <hjmshi@u.northwestern.edu>, Ping Tak Peter Tang <peter.tang@intel.com>.

of stochastic line searches for machine learning by studying a key component, namely the initial estimate in the one-dimensional search. Our approach, which is based on statistical considerations, is designed for an Armijo-style backtracking line search.

1.1. Literature Review

Progressive batching (sometimes referred to as dynamic sampling) has been well studied in the optimization literature, both for stochastic gradient and subsampled Newton-type methods (Byrd et al., 2012; Friedlander & Schmidt, 2012; Cartis & Scheinberg, 2015; Pasupathy et al., 2015; Roosta-Khorasani & Mahoney, 2016a;b; Bollapragada et al., 2016; 2017; De et al., 2017). Friedlander and Schmidt (2012) introduced theoretical conditions under which a progressive batching SG method converges linearly for finite sum problems, and experimented with a quasi-Newton adaptation of their algorithm. Byrd et al. (2012) proposed a progressive batching strategy, based on a *norm test*, that determines when to increase the sample size; they established linear convergence and computational complexity bounds in the case when the batch size grows geometrically. More recently, Bollapragada et al. (2017) introduced a batch control mechanism based on an *inner product* test that improves upon the norm test mentioned above.

There has been a renewed interest in understanding the generalization properties of small-batch and large-batch methods for training neural networks; see (Keskar et al., 2016; Dinh et al., 2017; Goyal et al., 2017; Hoffer et al., 2017). Keskar et al. (2016) empirically observed that large-batch methods converge to solutions with inferior generalization properties; however, Goyal et al. (2017) showed that large-batch methods can match the performance of small-batch methods when a warm-up strategy is used in conjunction with scaling the step length by the same factor as the batch size. Hoffer et al. (2017) and You et al. (2017) also explored larger batch sizes and steplengths to reduce the number of updates necessary to train the network. All of these studies naturally led to an interest in progressive batching techniques. Smith et al. (2017) showed empirically that increasing the sample size and decaying the steplength are quantitatively equivalent for the SG method; hence, steplength schedules could be directly converted to batch size schedules. This approach was parallelized by Devarakonda et al. (2017). De et al. (2017) presented numerical results with a progressive batching method that employs the norm test. Balles et al. (2016) proposed an adaptive dynamic sample size scheme and couples the sample size with the steplength.

Stochastic second-order methods have been explored within the context of convex and non-convex optimization; see (Schraudolph et al., 2007; Sohl-Dickstein et al., 2014; Mokhtari & Ribeiro, 2015; Berahas et al., 2016; Byrd et al., 2016; Keskar & Berahas, 2016; Curtis, 2016; Berahas & Takáč, 2017; Zhou et al., 2017). Schraudolph et al. (2007) ensured stability of quasi-Newton updating by computing gradients using the same batch at the beginning and end of the iteration. Since this can potentially double the cost of the iteration, Berahas et al. (2016) proposed to achieve gradient consistency by computing gradients based on the overlap between consecutive batches; this approach was further tested by Berahas and Takac (2017). An interesting approach introduced by Martens and Grosse (2015; 2016) approximates the Fisher information matrix to scale the gradient; a distributed implementation of their K-FAC approach is described in (Ba et al., 2016). Another approach approximately computes the inverse Hessian by using the Neumann power series representation of matrices (Krishnan et al., 2017).

1.2. Contributions

This paper builds upon three algorithmic components that have recently received attention in the literature — progressive batching, stable quasi-Newton updating, and adaptive steplength selection. It advances their design and puts them together in a novel algorithm with attractive theoretical and computational properties.

The cornerstone of our progressive batching strategy is the mechanism proposed by Bollapragada et al. (2017) in the context of first-order methods. We extend their *inner product* control test to second-order algorithms, something that is delicate and leads to a significant modification of the original procedure. Another main contribution of the paper is the design of an Armijo-style backtracking line search where the initial steplength is chosen based on statistical information gathered during the course of the iteration. We show that this steplength procedure is effective on a wide range of applications, as it leads to well scaled steps and allows for the BFGS update to be performed most of the time, even for nonconvex problems. We also test two techniques for ensuring the stability of quasi-Newton updating, and observe that the overlapping procedure described by Berahas et al. (2016) is more efficient than a straightforward adaptation of classical quasi-Newton methods (Schraudolph et al., 2007).

We report numerical tests on large-scale logistic regression and deep neural network training tasks that indicate that our method is robust and efficient, and has good generalization properties. An additional advantage is that the method requires almost no parameter tuning, which is possible due to the incorporation of second-order information. All of this suggests that our approach has the potential to become one of the leading optimization methods for training deep neural networks. In order to achieve this, the algorithm must be optimized for parallel execution, something that was only briefly explored in this study.

2. A Progressive Batching Quasi-Newton Method

The problem of interest is

$$\min_{x \in \mathbb{R}^d} F(x) = \int f(x; z, y) dP(z, y), \tag{1}$$

where f is the composition of a prediction function (parametrized by x) and a loss function, and (z, y) are random input-output pairs with probability distribution $P(z, y)$. The associated empirical risk problem consists of minimizing

$$R(x) = \frac{1}{N} \sum_{i=1}^{N} f(x; z^i, y^i) \triangleq \frac{1}{N} \sum_{i=1}^{N} F_i(x),$$

where we define $F_i(x) = f(x; z^i, y^i)$. A stochastic quasi-Newton method is given by

$$x_{k+1} = x_k - \alpha_k H_k g_k^{S_k}, \tag{2}$$

where the batch (or subsampled) gradient is given by

$$g_k^{S_k} = \nabla F_{S_k}(x_k) \triangleq \frac{1}{|S_k|} \sum_{i \in S_k} \nabla F_i(x_k), \tag{3}$$

the set $S_k \subset \{1, 2, \cdots\}$ indexes data points (y^i, z^i) sampled from the distribution $P(z, y)$, and H_k is a positive definite quasi-Newton matrix. We now discuss each of the components of the new method.

2.1. Sample Size Selection

The proposed algorithm has the form (29)-(30). Initially, it utilizes a small batch size $|S_k|$, and increases it gradually in order to attain a fast local rate of convergence and permit the use of second-order information. A challenging question is to determine when, and by how much, to increase the batch size $|S_k|$ over the course of the optimization procedure based on observed gradients — as opposed to using prescribed rules that depend on the iteration number k.

We propose to build upon the strategy introduced by Bollapragada et al. (2017) in the context of first-order methods. Their *inner product test* determines a sample size such that the search direction is a descent direction with high probability. A straightforward extension of this strategy to the quasi-Newton setting is not appropriate since requiring only that a stochastic quasi-Newton search direction be a descent direction with high probability would underutilize the curvature information contained in the search direction.

We would like, instead, for the search direction $d_k = -H_k g_k^{S_k}$ to make an acute angle with the true quasi-Newton search direction $-H_k \nabla F(x_k)$, with high probability. Although this does not imply that d_k is a descent direction for F, this will normally be the case for any reasonable quasi-Newton matrix.

To derive the new inner product quasi-Newton (IPQN) test, we first observe that the stochastic quasi-Newton search direction makes an acute angle with the true quasi-Newton direction in expectation, i.e.,

$$\mathbb{E}_k \left[(H_k \nabla F(x_k))^T (H_k g_k^{S_k}) \right] = \|H_k \nabla F(x_k)\|^2, \tag{4}$$

where $\mathbb{E}_k$ denotes the conditional expectation at x_k. We must, however, control the variance of this quantity to achieve our stated objective. Specifically, we select the sample size $|S_k|$ such that the following condition is satisfied:

$$\mathbb{E}_k \left[\left((H_k \nabla F(x_k))^T (H_k g_k^{S_k}) - \|H_k \nabla F(x_k)\|^2 \right)^2 \right] \leq \theta^2 \|H_k \nabla F(x_k)\|^4, \tag{5}$$

for some $\theta > 0$. The left hand side of (5) is difficult to compute but can be bounded by the true variance of individual search directions, i.e.,

$$\frac{\mathbb{E}_k \left[\left((H_k \nabla F(x_k))^T (H_k g_k^i) - \|H_k \nabla F(x_k)\|^2 \right)^2 \right]}{|S_k|} \leq \theta^2 \|H_k \nabla F(x_k)\|^4, \tag{6}$$

where $g_k^i = \nabla F_i(x_k)$. This test involves the true expected gradient and variance, but we can approximate these quantities with sample gradient and variance estimates, respectively, yielding the practical inner product quasi-Newton test:

$$\frac{\mathrm{Var}_{i \in S_k^v} \left((g_k^i)^T H_k^2 g_k^{S_k} \right)}{|S_k|} \leq \theta^2 \left\| H_k g_k^{S_k} \right\|^4, \tag{7}$$

where $S_k^v \subseteq S_k$ is a subset of the current sample (batch), and the variance term is defined as

$$\frac{\sum_{i \in S_k^v} \left((g_k^i)^T H_k^2 g_k^{S_k} - \left\| H_k g_k^{S_k} \right\|^2 \right)^2}{|S_k^v| - 1}. \tag{8}$$

The variance (8) may be computed using just one additional Hessian vector product of H_k with $H_k g_k^{S_k}$. Whenever condition (7) is not satisfied, we increase the sample size $|S_k|$. In order to estimate the increase that would lead to a satisfaction of (7), we reason as follows. If we assume that new sample $|\bar{S}_k|$ is such that

$$\left\| H_k g_k^{S_k} \right\| \approxeq \left\| H_k g_k^{\bar{S}_k} \right\|,$$

and similarly for the variance estimate, then a simple computation shows that a lower bound on the new sample size is

$$|\bar{S}_k| \geq \frac{\mathrm{Var}_{i \in S_k^v}\left((g_k^i)^T H_k^2 g_k^{S_k}\right)}{\theta^2 \left\|H_k g_k^{S_k}\right\|^4} \triangleq b_k. \tag{9}$$

In our implementation of the algorithm, we set the new sample size as $|S_{k+1}| = \lceil b_k \rceil$. When the sample approximation of $F(x_k)$ is not accurate, which can occur when $|S_k|$ is small, the progressive batching mechanism just described may not be reliable. In this case we employ the moving window technique described in Section 4.2 of Bollapragada et al. (2017), to produce a sample estimate of $\nabla F(x_k)$.

2.2. The Line Search

In deterministic optimization, line searches are employed to ensure that the step is not too short and to guarantee sufficient decrease in the objective function. Line searches are particularly important in quasi-Newton methods since they ensure robustness and efficiency of the iteration with little additional cost.

In contrast, stochastic line searches are poorly understood and rarely employed in practice because they must make decisions based on sample function values

$$F_{S_k}(x) = \frac{1}{|S_k|} \sum_{i \in S_k} F_i(x), \tag{10}$$

which are noisy approximations to the true objective F. One of the key questions in the design of a stochastic line search is how to ensure, with high probability, that there is a decrease in the true function when one can only observe stochastic approximations $F_{S_k}(x)$. We address this question by proposing a formula for the step size α_k that controls possible increases in the true function. Specifically, the first trial steplength in the stochastic backtracking line search is computed so that the predicted decrease in the expected function value is sufficiently large, as we now explain.

Using Lipschitz continuity of $\nabla F(x)$ and taking conditional expectation, we can show the following inequality

$$\mathbb{E}_k\left[F_{k+1}\right] \leq F_k - \alpha_k \nabla F(x_k)^T H_k^{1/2} W_k H_k^{1/2} \nabla F(x_k) \tag{11}$$

where

$$W_k = \left(I - \frac{L\alpha_k}{2}\left(1 + \frac{\mathrm{Var}\{H_k g_k^i\}}{|S_k| \|H_k \nabla F(x_k)\|^2}\right) H_k\right),$$

$\mathrm{Var}\{H_k g_k^i\} = \mathbb{E}_k\left[\|H_k g_k^i - H_k \nabla F(x_k)\|^2\right]$, $F_k = F(x_k)$, and L is the Lipschitz constant. The proof of (A.1) is given in the supplement.

The only difference in (A.1) between the deterministic and stochastic quasi-Newton methods is the additional variance term in the matrix W_k. To obtain decrease in the function value in the deterministic case, the matrix $\left(I - \frac{L\alpha_k}{2} H_k\right)$ must be positive definite, whereas in the stochastic case the matrix W_k must be positive definite to yield a decrease in F *in expectation*. In the deterministic case, for a reasonably good quasi-Newton matrix H_k, one expects that $\alpha_k = 1$ will result in a decrease in the function, and therefore the initial trial steplength parameter should be chosen to be 1. In the stochastic case, the initial trial value

$$\hat{\alpha}_k = \left(1 + \frac{\mathrm{Var}\{H_k g_k^i\}}{|S_k| \|H_k \nabla F(x_k)\|^2}\right)^{-1} \tag{12}$$

will result in decrease in the expected function value. However, since formula (12) involves the expensive computation of the individual matrix-vector products $H_k g_k^i$, we approximate the variance-bias ratio as follows:

$$\bar{\alpha}_k = \left(1 + \frac{\mathrm{Var}\{g_k^i\}}{|S_k| \|\nabla F(x_k)\|^2}\right)^{-1}, \tag{13}$$

where $\mathrm{Var}\{g_k^i\} = \mathbb{E}_k\left[\|g_k^i - \nabla F(x_k)\|^2\right]$. In our practical implementation, we estimate the population variance and gradient with the sample variance and gradient, respectively, yielding the initial steplength

$$\alpha_k = \left(1 + \frac{\mathrm{Var}_{i \in S_k^v}\{g_k^i\}}{|S_k| \left\|g_k^{S_k}\right\|^2}\right)^{-1}, \tag{14}$$

where

$$\mathrm{Var}_{i \in S_k^v}\{g_k^i\} = \frac{1}{|S_k^v| - 1} \sum_{i \in S_k^v} \left\|g_k^i - g_k^{S_k}\right\|^2 \tag{15}$$

and $S_k^v \subseteq S_k$. With this initial value of α_k in hand, our algorithm performs a backtracking line search that aims to satisfy the Armijo condition

$$\begin{aligned} &F_{S_k}(x_k - \alpha_k H_k g_k^{S_k}) \\ &\qquad \leq F_{S_k}(x_k) - c_1 \alpha_k (g_k^{S_k})^T H_k g_k^{S_k}, \end{aligned} \tag{16}$$

where $c_1 > 0$.

2.3. Stable Quasi-Newton Updates

In the BFGS and L-BFGS methods, the inverse Hessian approximation is updated using the formula

$$\begin{aligned} H_{k+1} &= V_k^T H_k V_k + \rho_k s_k s_k^T \\ \rho_k &= (y_k^T s_k)^{-1} \\ V_k &= I - \rho_k y_k s_k^T \end{aligned} \tag{17}$$

where $s_k = x_{k+1} - x_k$ and y_k is the difference in the gradients at x_{k+1} and x_k. When the batch changes from one iteration to the next ($S_{k+1} \neq S_k$), it is not obvious how y_k should be defined. It has been observed that when y_k is computed using different samples, the updating process may be unstable, and hence it seems natural to use the same sample at the beginning and at the end of the iteration (Schraudolph et al., 2007), and define

$$y_k = g_{k+1}^{S_k} - g_k^{S_k}. \tag{18}$$

However, this requires that the gradient be evaluated twice for every batch S_k at x_k and x_{k+1}. To avoid this additional cost, Berahas et al. (2016) propose to use the overlap between consecutive samples in the gradient differencing. If we denote this overlap as $O_k = S_k \cap S_{k+1}$, then one defines

$$y_k = g_{k+1}^{O_k} - g_k^{O_k}. \tag{19}$$

This requires no extra computation since the two gradients in this expression are subsets of the gradients corresponding to the samples S_k and S_{k+1}. The overlap should not be too small to avoid differencing noise, but this is easily achieved in practice. We test both formulas for y_k in our implementation of the method; see Section 4.

2.4. The Complete Algorithm

The pseudocode of the progressive batching L-BFGS method is given in Algorithm 1. Observe that the limited memory Hessian approximation H_k in Line 8 is independent of the choice of the sample S_k. Specifically, H_k is defined by a collection of curvature pairs $\{(s_j, y_j)\}$, where the most recent pair is based on the sample S_{k-1}; see Line 14. For the batch size control test (7), we choose $\theta = 0.9$ in the logistic regression experiments, and θ is a tunable parameter chosen in the interval $[0.9, 3]$ in the neural network experiments. The constant c_1 in (16) is set to $c_1 = 10^{-4}$. For L-BFGS, we set the memory as $m = 10$. We skip the quasi-Newton update if the following curvature condition is not satisfied:

$$y_k^T s_k > \epsilon \|s_k\|^2, \quad \text{with } \epsilon = 10^{-2}. \tag{20}$$

The initial Hessian matrix H_0^k in the L-BFGS recursion at each iteration is chosen as $\gamma_k I$ where $\gamma_k = y_k^T s_k / y_k^T y_k$.

3. Convergence Analysis

We now present convergence results for the proposed algorithm, both for strongly convex and nonconvex objective functions. Our emphasis is in analyzing the effect of progressive sampling, and therefore, we follow common practice and assume that the steplength in the algorithm is fixed ($\alpha_k = \alpha$), and that the inverse L-BFGS matrix H_k has bounded eigenvalues, i.e.,

$$\Lambda_1 I \preceq H_k \preceq \Lambda_2 I. \tag{21}$$

Algorithm 1 Progressive Batching L-BFGS Method

```
Input: Initial iterate $x_0$, initial sample size $|S_0|$;
Initialization: Set $k \leftarrow 0$
Repeat until convergence:
 1: Sample $S_k \subseteq \{1, \cdots, N\}$ with sample size $|S_k|$
 2: if condition (7) is not satisfied then
 3:     Compute $b_k$ using (9), and set $\hat{b}_k \leftarrow \lceil b_k \rceil - |S_k|$
 4:     Sample $S^+ \subseteq \{1, \cdots, N\} \setminus S_k$ with $|S^+| = \hat{b}_k$
 5:     Set $S_k \leftarrow S_k \cup S^+$
 6: end if
 7: Compute $g_k^{S_k}$
 8: Compute $p_k = -H_k g_k^{S_k}$ using L-BFGS Two-Loop
    Recursion in (Nocedal & Wright, 1999)
 9: Compute $\alpha_k$ using (14)
10: while the Armijo condition (16) not satisfied do
11:     Set $\alpha_k = \alpha_k/2$
12: end while
13: Compute $x_{k+1} = x_k + \alpha_k p_k$
14: Compute $y_k$ using (18) or (19)
15: Compute $s_k = x_{k+1} - x_k$
16: if $y_k^T s_k > \epsilon \|s_k\|^2$ then
17:     if number of stored $(y_j, s_j)$ exceeds $m$ then
18:         Discard oldest curvature pair $(y_j, s_j)$
19:     end if
20:     Store new curvature pair $(y_k, s_k)$
21: end if
22: Set $k \leftarrow k + 1$
23: Set $|S_k| = |S_{k-1}|$
```

This assumption can be justified both in the convex and nonconvex cases under certain conditions; see (Berahas et al., 2016). We assume that the sample size is controlled by the exact inner product quasi-Newton test (31). This test is designed for efficiency, and in rare situations could allow for the generation of arbitrarily long search directions. To prevent this from happening, we introduce an additional control on the sample size $|S_k|$, by extending (to the quasi-Newton setting) the orthogonality test introduced in (Bollapragada et al., 2017). This additional requirement states that the current sample size $|S_k|$ is acceptable only if

$$\frac{\mathbb{E}_k\left[\left\| H_k g_k^i - \frac{\left(H_k g_k^{S_k}\right)^T (H_k \nabla F(x_k))}{\|H_k \nabla F(x_k)\|^2} H_k \nabla F(x_k) \right\|^2\right]}{|S_k|} \leq \nu^2 \|H_k \nabla F(x_k)\|^2, \tag{22}$$

for some given $\nu > 0$.

We now establish linear convergence when the objective is strongly convex.

Theorem 3.1. *Suppose that F is twice continuously differentiable and that there exist constants $0 < \mu \leq L$ such that*

$$\mu I \preceq \nabla^2 F(x) \preceq LI, \quad \forall x \in \mathbb{R}^d. \tag{23}$$

Let $\{x_k\}$ be generated by iteration (29), for any x_0, where $|S_k|$ is chosen by the (exact variance) inner product quasi-Newton test (31). Suppose that the orthogonality condition (32) holds at every iteration, and that the matrices H_k satisfy (B.2). Then, if

$$\alpha_k = \alpha \leq \frac{1}{(1+\theta^2+\nu^2)L\Lambda_2}, \tag{24}$$

we have that

$$\mathbb{E}[F(x_k) - F(x^*)] \leq \rho^k (F(x_0) - F(x^*)), \tag{25}$$

where x^* denotes the minimizer of F, $\rho = 1 - \mu\Lambda_1\alpha$, and $\mathbb{E}$ denotes the total expectation.

The proof of this result is given in the supplement. We now consider the case when F is nonconvex and bounded below.

Theorem 3.2. *Suppose that F is twice continuously differentiable and bounded below, and that there exists a constant $L > 0$ such that*

$$\nabla^2 F(x) \preceq LI, \quad \forall x \in \mathbb{R}^d. \tag{26}$$

Let $\{x_k\}$ be generated by iteration (29), for any x_0, where $|S_k|$ is chosen so that (31) and (32) are satisfied, and suppose that (B.2) holds. Then, if α_k satisfies (36), we have

$$\lim_{k\to\infty} \mathbb{E}[\|\nabla F(x_k)\|^2] \to 0. \tag{27}$$

Moreover, for any positive integer T we have that

$$\min_{0\leq k\leq T-1} \mathbb{E}[\|\nabla F(x_k)\|^2] \leq \frac{2}{\alpha T \Lambda_1}(F(x_0) - F_{min}),$$

where F_{min} is a lower bound on F in $\mathbb{R}^d$.

The proof is given in the supplement. This result shows that the sequence of gradients $\{\|\nabla F(x_k)\|\}$ converges to zero in expectation, and establishes a global sublinear rate of convergence of the smallest gradients generated after every T steps.

4. Numerical Results

In this section, we present numerical results for the proposed algorithm, which we refer to as PBQN for the **P**rogressive **B**atching **Q**uasi-**N**ewton algorithm.

4.1. Experiments on Logistic Regression Problems

We first test our algorithm on binary classification problems where the objective function is given by the logistic loss with ℓ_2 regularization:

$$R(x) = \frac{1}{N}\sum_{i=1}^{N} \log(1 + \exp(-z^i x^T y^i)) + \frac{\lambda}{2}\|x\|^2, \tag{28}$$

with $\lambda = 1/N$. We consider the 8 datasets listed in the supplement. An approximation R^* of the optimal function value is computed for each problem by running the full batch L-BFGS method until $\|\nabla R(x_k)\|_\infty \leq 10^{-8}$. Training error is defined as $R(x_k) - R^*$, where $R(x_k)$ is evaluated over the training set; test loss is evaluated over the test set without the ℓ_2 regularization term.

We tested two options for computing the curvature vector y_k in the PBQN method: the multi-batch (MB) approach (19) with 25% sample overlap, and the full overlap (FO) approach (18). We set $\theta = 0.9$ in (7), chose $|S_0| = 512$, and set all other parameters to the default values given in Section 2. Thus, none of the parameters in our PBQN method were tuned for each individual dataset. We compared our algorithm against two other methods: (i) Stochastic gradient (SG) with a batch size of 1; (ii) SVRG (Johnson & Zhang, 2013) with the inner loop length set to N. The steplength for SG and SVRG is constant and tuned for each problem ($\alpha_k \equiv \alpha = 2^j$, for $j \in \{-10, -9, ..., 9, 10\}$) so as to give best performance.

In Figures 9 and 2 we present results for two datasets, `spam` and `covertype`; the rest of the results are given in the supplement. The horizontal axis measures the number of full gradient evaluations, or equivalently, the number of times that N component gradients ∇F_i were evaluated. The left-most figure reports the long term trend over 100 gradient evaluations, while the rest of the figures zoom into the first 10 gradient evaluations to show the initial behavior of the methods. The vertical axis measures training error, test loss, and test accuracy, respectively, from left to right.

The proposed algorithm competes well for these two datasets in terms of training error, test loss and test accuracy, and decreases these measures more evenly than the SG and SVRG. Our numerical experience indicates that formula (14) is quite effective at estimating the steplength parameter, as it is accepted by the backtracking line search for most iterations. As a result, the line search computes very few additional function values.

It is interesting to note that SVRG is not as efficient in the initial epochs compared to PBQN or SG, when measured either in terms of test loss and test accuracy. The training error for SVRG decreases rapidly in later epochs but this rapid improvement is not observed in the test loss and accuracy. Neither the PBQN nor SVRG significantly outperforms the other across all datasets tested in terms of training error, as observed in the supplement.

Our results indicate that defining the curvature vector using the MB approach is preferable to using the FB approach. The number of iterations required by the PBQN method is significantly smaller compared to the SG method, suggesting the potential efficiency gains of a parallel implementa-

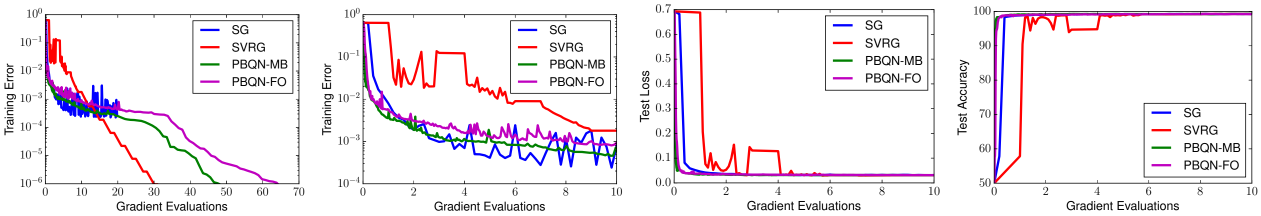

Figure 1. **spam dataset:** Performance of the progressive batching L-BFGS method (PBQN), with multi-batch (25% overlap) and full-overlap approaches, and the SG and SVRG methods.

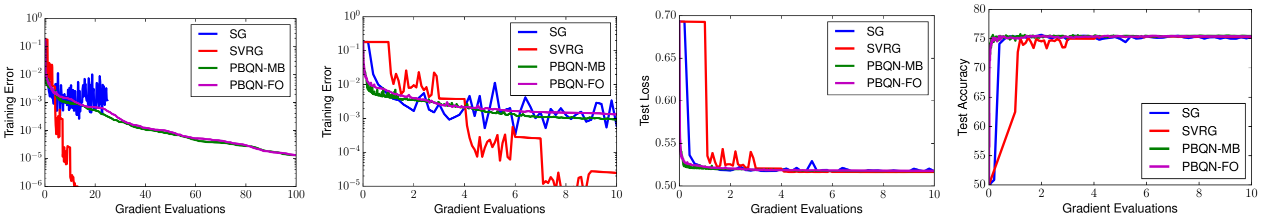

Figure 2. **covertype dataset:** Performance of the progressive batching L-BFGS methods, with multi-batch (25% overlap) and full-overlap approaches, and the SG and SVRG methods.

tion of our algorithm.

4.2. Results on Neural Networks

We have performed a preliminary investigation into the performance of the PBQN algorithm for training neural networks. As is well-known, the resulting optimization problems are quite difficult due to the existence of local minimizers, some of which generalize poorly. Thus our first requirement when applying the PBQN method was to obtain as good generalization as SG, something we have achieved.

Our investigation into how to obtain fast performance is, however, still underway for reasons discussed below. Nevertheless, our results are worth reporting because they show that our line search procedure is performing as expected, and that the overall number of iterations required by the PBQN method is small enough so that a parallel implementation could yield state-of-the-art results, based on the theoretical performance model detailed in the supplement.

We compared our algorithm, as described in Section 2, against SG and Adam (Kingma & Ba, 2014). It has taken many years to design regularizations techniques and heuristics that greatly improve the performance of the SG method for deep learning (Srivastava et al., 2014; Ioffe & Szegedy, 2015). These include batch normalization and dropout, which (in their current form) are not conducive to the PBQN approach due to the need for gradient consistency when evaluating the curvature pairs in L-BFGS. Therefore, we do not implement batch normalization and dropout in any of the methods tested, and leave the study of their extension to the PBQN setting for future work.

We consider three network architectures: (i) a small convolutional neural network on CIFAR-10 ($\mathcal{C}$) (Krizhevsky, 2009), (ii) an AlexNet-like convolutional network on MNIST and CIFAR-10 ($\mathcal{A}_1, \mathcal{A}_2$, respectively) (LeCun et al., 1998; Krizhevsky et al., 2012), and (iii) a residual network (ResNet18) on CIFAR-10 ($\mathcal{R}$) (He et al., 2016). The network architecture details and additional plots are given in the supplement. All of these networks were implemented in PyTorch (Paszke et al., 2017). The results for the CIFAR-10 AlexNet and CIFAR-10 ResNet18 are given in Figures 15 and 16, respectively. We report results both against the total number of iterations and the total number of gradient evaluations. Table 1 shows the best test accuracies attained by each of the four methods over the various networks.

In all our experiments, we initialize the batch size as $|S_0| = 512$ in the PBQN method, and fix the batch size to $|S_k| = 128$ for SG and Adam. The parameter θ given in (7), which controls the batch size increase in the PBQN method, was tuned lightly by chosing among the 3 values: 0.9, 2, 3. SG and Adam are tuned using a development-based decay (dev-decay) scheme, which track the best validation loss at each epoch and reduces the steplength by a constant factor δ if the validation loss does not improve after e epochs.

We observe from our results that the PBQN method achieves a similar test accuracy as SG and Adam, but requires more gradient evaluations. Improvements in performance can be obtained by ensuring that the PBQN method exerts a finer control on the sample size in the small batch regime — something that requires further investigation. Nevertheless, the small number of iterations required by the PBQN method, together with the fact that it employs larger batch sizes than

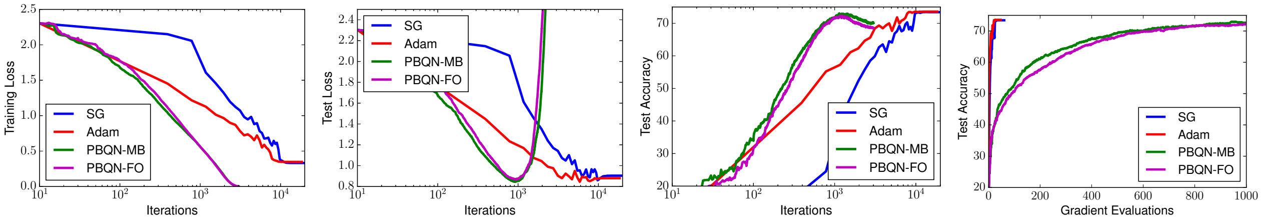

Figure 3. **CIFAR-10 AlexNet** $(\mathcal{A}_2)$**:** Performance of the progressive batching L-BFGS methods, with multi-batch (25% overlap) and full-overlap approaches, and the SG and Adam methods. The best results for L-BFGS are achieved with $\theta = 0.9$.

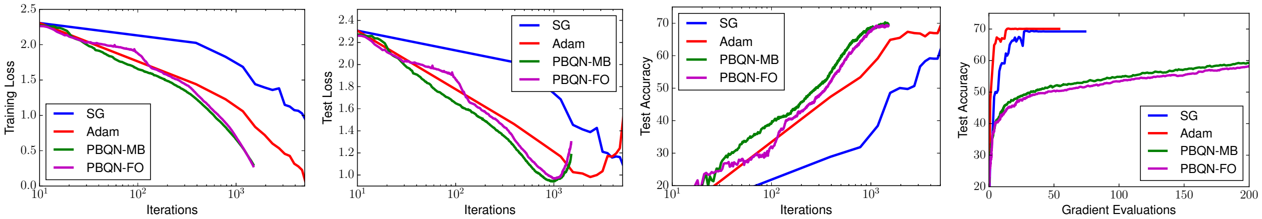

Figure 4. **CIFAR-10 ResNet18** $(\mathcal{R})$**:** Performance of the progressive batching L-BFGS methods, with multi-batch (25% overlap) and full-overlap approaches, and the SG and Adam methods. The best results for L-BFGS are achieved with $\theta = 2$.

Table 1. Best test accuracy performance of SG, Adam, multi-batch L-BFGS, and full overlap L-BFGS on various networks over 5 different runs and initializations.

Network	SG	Adam	MB	FO
$\mathcal{C}$	66.24	67.03	67.37	62.46
$\mathcal{A}_1$	99.25	99.34	99.16	99.05
$\mathcal{A}_2$	73.46	73.59	73.02	72.74
$\mathcal{R}$	69.5	70.16	70.28	69.44

SG during much of the run, suggests that a distributed version similar to a data-parallel distributed implementation of the SG method (Chen et al., 2016; Das et al., 2016) would lead to a highly competitive method.

Similar to the logistic regression case, we observe that the steplength computed via (14) is almost always accepted by the Armijo condition, and typically lies within $(0.1, 1)$. Once the algorithm has trained for a significant number of iterations using full-batch, the algorithm begins to overfit on the training set, resulting in worsened test loss and accuracy, as observed in the graphs.

5. Final Remarks

Several types of quasi-Newton methods have been proposed in the literature to address the challenges arising in machine learning. Some of these method operate in the purely stochastic setting (which makes quasi-Newton updating difficult) or in the purely batch regime (which leads to generalization problems). We believe that progressive batching is the right context for designing an L-BFGS method that has good generalization properties, does not expose any free parameters, and has fast convergence. The advantages of our approach are clearly seen in logistic regression experiments. To make the new method competitive with SG and Adam for deep learning, we need to improve several of its components. This includes the design of a more robust progressive batching mechanism, the redesign of batch normalization and dropout heuristics to improve the generalization performance of our method for training larger networks, and most importantly, the design of a parallelized implementation that takes advantage of the higher granularity of each iteration. We believe that the potential of the proposed approach as an alternative to SG for deep learning is worthy of further investigation.

Acknowledgements

We thank Albert Berahas for his insightful comments regarding multi-batch L-BFGS and probabilistic line searches, as well as for his useful feedback on earlier versions of the manuscript. We also thank the anonymous reviewers for their useful feedback. Bollapragada is supported by DOE award DE-FG02-87ER25047. Nocedal is supported by NSF award DMS-1620070. Shi is supported by Intel grant SP0036122.

References

Ba, J., Grosse, R., and Martens, J. Distributed second-order optimization using kronecker-factored approximations.

2016.

Balles, L., Romero, J., and Hennig, P. Coupling adaptive batch sizes with learning rates. *arXiv preprint arXiv:1612.05086*, 2016.

Berahas, A. S. and Takáč, M. A robust multi-batch l-bfgs method for machine learning. *arXiv preprint arXiv:1707.08552*, 2017.

Berahas, A. S., Nocedal, J., and Takác, M. A multi-batch l-bfgs method for machine learning. In *Advances in Neural Information Processing Systems*, pp. 1055–1063, 2016.

Bertsekas, D. P., Nedić, A., and Ozdaglar, A. E. *Convex analysis and optimization*. Athena Scientific Belmont, 2003.

Bollapragada, R., Byrd, R., and Nocedal, J. Exact and inexact subsampled Newton methods for optimization. *arXiv preprint arXiv:1609.08502*, 2016.

Bollapragada, R., Byrd, R., and Nocedal, J. Adaptive sampling strategies for stochastic optimization. *arXiv preprint arXiv:1710.11258*, 2017.

Byrd, R. H., Chin, G. M., Nocedal, J., and Wu, Y. Sample size selection in optimization methods for machine learning. *Mathematical Programming*, 134(1):127–155, 2012.

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

Carbonetto, P. *New probabilistic inference algorithms that harness the strengths of variational and Monte Carlo methods*. PhD thesis, University of British Columbia, 2009.

Cartis, C. and Scheinberg, K. Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. *Mathematical Programming*, pp. 1–39, 2015.

Chang, C. and Lin, C. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Chen, J., Monga, R., Bengio, S., and Jozefowicz, R. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

Cormack, G. and Lynam, T. Spam corpus creation for TREC. In *Proc. 2nd Conference on Email and Anti-Spam*, 2005. http://plg.uwaterloo.ca/gvcormac/treccorpus.

Curtis, F. A self-correcting variable-metric algorithm for stochastic optimization. In *International Conference on Machine Learning*, pp. 632–641, 2016.

Das, D., Avancha, S., Mudigere, D., Vaidynathan, K., Sridharan, S., Kalamkar, D., Kaul, B., and Dubey, P. Distributed deep learning using synchronous stochastic gradient descent. *arXiv preprint arXiv:1602.06709*, 2016.

De, S., Yadav, A., Jacobs, D., and Goldstein, T. Automated inference with adaptive batches. In *Artificial Intelligence and Statistics*, pp. 1504–1513, 2017.

Devarakonda, A., Naumov, M., and Garland, M. Adabatch: Adaptive batch sizes for training deep neural networks. *arXiv preprint arXiv:1712.02029*, 2017.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. *arXiv preprint arXiv:1703.04933*, 2017.

Friedlander, M. P. and Schmidt, M. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Grosse, R. and Martens, J. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pp. 573–582, 2016.

Guyon, I., Aliferis, C. F., Cooper, G. F., Elisseeff, A., Pellet, J., Spirtes, P., and Statnikov, A. R. Design and analysis of the causation and prediction challenge. In *WCCI Causation and Prediction Challenge*, pp. 1–33, 2008.

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

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

Hoffer, E., Hubara, I., and Soudry, D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *arXiv preprint arXiv:1705.08741*, 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456, 2015.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, pp. 315–323, 2013.

Keskar, N. S. and Berahas, A. S. adaqn: An adaptive quasi-newton algorithm for training rnns. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 1–16. Springer, 2016.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krishnan, S., Xiao, Y., and Saurous, R. A. Neumann optimizer: A practical optimization algorithm for deep neural networks. *arXiv preprint arXiv:1712.03298*, 2017.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

Kurth, T., Zhang, J., Satish, N., Racah, E., Mitliagkas, I., Patwary, M. M. A., Malas, T., Sundaram, N., Bhimji, W., Smorkalov, M., et al. Deep learning at 15pf: Supervised and semi-supervised classification for scientific data. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 7. ACM, 2017.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Liu, D. C. and Nocedal, J. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pp. 2408–2417, 2015.

Mokhtari, A. and Ribeiro, A. Global convergence of online limited memory bfgs. *Journal of Machine Learning Research*, 16(1):3151–3181, 2015.

Nocedal, J. and Wright, S. *Numerical Optimization*. Springer New York, 2 edition, 1999.

Pasupathy, R., Glynn, P., Ghosh, S., and Hashemi, F. S. On sampling rates in stochastic recursions. 2015. Under Review.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Roosta-Khorasani, F. and Mahoney, M. W. Sub-sampled Newton methods II: Local convergence rates. *arXiv preprint arXiv:1601.04738*, 2016a.

Roosta-Khorasani, F. and Mahoney, M. W. Sub-sampled Newton methods I: Globally convergent algorithms. *arXiv preprint arXiv:1601.04737*, 2016b.

Schraudolph, N. N., Yu, J., and Günter, S. A stochastic quasi-newton method for online convex optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 436–443, 2007.

Smith, S. L., Kindermans, P., and Le, Q. V. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

Sohl-Dickstein, J., Poole, B., and Ganguli, S. Fast large-scale optimization by unifying stochastic gradient and quasi-Newton methods. In *International Conference on Machine Learning*, pp. 604–612, 2014.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

You, Y., Gitman, I., and Ginsburg, B. Scaling sgd batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888*, 2017.

Zhou, C., Gao, W., and Goldfarb, D. Stochastic adaptive quasi-Newton methods for minimizing expected values. In *International Conference on Machine Learning*, pp. 4150–4159, 2017.

A. Initial Step Length Derivation

To establish our results, recall that the stochastic quasi-Newton method is defined as

$$x_{k+1} = x_k - \alpha_k H_k g_k^{S_k}, \tag{29}$$

where the batch (or subsampled) gradient is given by

$$g_k^{S_k} = \nabla F_{S_k}(x_k) = \frac{1}{|S_k|} \sum_{i \in S_k} \nabla F_i(x_k), \tag{30}$$

and the set $S_k \subset \{1, 2, \cdots\}$ indexes data points (y^i, z^i). The algorithm selects the Hessian approximation H_k through quasi-Newton updating prior to selecting the new sample S_k to define the search direction p_k. We will use $\mathbb{E}_k$ to denote the conditional expectation at x_k and use $\mathbb{E}$ to denote the total expectation.

The primary theoretical mechanism for determining batch sizes is the exact variance inner product quasi-Newton (IPQN) test, which is defined as

$$\frac{\mathbb{E}_k\left[\left((H_k \nabla F(x_k))^T (H_k g_k^i) - \|H_k \nabla F(x_k)\|^2\right)^2\right]}{|S_k|} \leq \theta^2 \|H_k \nabla F(x_k)\|^4. \tag{31}$$

We establish the inequality used to determine the initial steplength α_k for the stochastic line search.

Lemma A.1. *Assume that F is continuously differentiable with Lipschitz continuous gradient with Lipschitz constant L. Then*

$$\mathbb{E}_k\left[F(x_{k+1})\right] \leq F(x_k) - \alpha_k \nabla F(x_k)^T H_k^{1/2} W_k H_k^{1/2} \nabla F(x_k),$$

where

$$W_k = \left(I - \frac{L\alpha_k}{2}\left(1 + \frac{\mathrm{Var}\{H_k g_k^i\}}{|S_k| \|H_k \nabla F(x_k)\|^2}\right) H_k\right),$$

and $\mathrm{Var}\{H_k g_k^i\} = \mathbb{E}_k\left[\|H_k g_k^i - H_k \nabla F(x_k)\|^2\right]$.

Proof. By Lipschitz continuity of the gradient, we have that

$$\begin{aligned}
\mathbb{E}_k\left[F(x_{k+1})\right] &\leq F(x_k) - \alpha_k \nabla F(x_k)^T H_k \mathbb{E}_k\left[g_k^{S_k}\right] + \frac{L\alpha_k^2}{2} \mathbb{E}_k\left[\|H_k g_k^{S_k}\|^2\right] \\
&= F(x_k) - \alpha_k \nabla F(x_k)^T H_k \nabla F(x_k) + \frac{L\alpha_k^2}{2}\left(\|H_k \nabla F(x_k)\|^2 + \mathbb{E}_k\left[\|H_k g_k^{S_k} - H_k \nabla F(x_k)\|^2\right]\right) \\
&\leq F(x_k) - \alpha_k \nabla F(x_k)^T H_k \nabla F(x_k) + \frac{L\alpha_k^2}{2}\left(\|H_k \nabla F(x_k)\|^2 + \frac{\mathrm{Var}\{H_k g_k^i\}}{|S_k| \|H_k \nabla F(x_k)\|^2} \|H_k \nabla F(x_k)\|^2\right) \\
&= F(x_k) - \alpha_k \nabla F(x_k)^T H_k^{1/2}\left(I - \frac{L\alpha_k}{2}\left(1 + \frac{\mathrm{Var}\{H_k g_k^i\}}{|S_k| \|H_k \nabla F(x_k)\|^2}\right) H_k\right) H_k^{1/2} \nabla F(x_k) \\
&= F(x_k) - \alpha_k \nabla F(x_k)^T H_k^{1/2} W_k H_k^{1/2} \nabla F(x_k).
\end{aligned}$$

□

B. Convergence Analysis

For the rest of our analysis, we make the following two assumptions.

Assumptions B.1. *The orthogonality condition is satisfied for all k, i.e.,*

$$\frac{\mathbb{E}_k\left[\left\|H_k g_k^i - \frac{(H_k g_k^i)^T (H_k \nabla F(x_k))}{\|H_k \nabla F(x_k)\|^2} H_k \nabla F(x_k)\right\|^2\right]}{|S_k|} \leq \nu^2 \|H_k \nabla F(x_k)\|^2, \tag{32}$$

for some large $\nu > 0$.

Assumptions B.2. *The eigenvalues of H_k are contained in an interval in $\mathbb{R}^+$, i.e., for all k there exist constants $\Lambda_2 \geq \Lambda_1 > 0$ such that*

$$\Lambda_1 I \preceq H_k \preceq \Lambda_2 I. \tag{33}$$

Condition (32) ensures that the stochastic quasi-Newton direction is bounded away from orthogonality to $-H_k \nabla F(x_k)$, with high probability, and prevents the variance in the individual quasi-Newton directions to be too large relative to the variance in the individual quasi-Newton directions along $-H_k \nabla F(x_k)$. Assumption B.2 holds, for example, when F is convex and a regularization parameter is included so that any subsampled Hessian $\nabla^2 F_S(x)$ is positive definite. It can also be shown to hold in the non-convex case by applying cautious BFGS updating; e.g. by updating H_k only when $y_k^T s_k \geq \epsilon \|s_k\|_2^2$ where $\epsilon > 0$ is a predetermined constant (Berahas et al., 2016).

We begin by establishing a technical descent lemma.

Lemma B.3. *Suppose that F is twice continuously differentiable and that there exists a constant $L > 0$ such that*

$$\nabla^2 F(x) \preceq LI, \quad \forall x \in \mathbb{R}^d. \tag{34}$$

Let $\{x_k\}$ be generated by iteration (29) for any x_0, where $|S_k|$ is chosen by the (exact variance) inner product quasi-Newton test (31) for given constant $\theta > 0$ and suppose that assumptions (B.1) and (B.2) hold. Then, for any k,

$$\mathbb{E}_k \left[\|H_k g_k^{S_k}\|^2 \right] \leq (1 + \theta^2 + \nu^2) \|H_k \nabla F(x_k)\|^2. \tag{35}$$

Moreover, if α_k satisfies

$$\alpha_k = \alpha \leq \frac{1}{(1 + \theta^2 + \nu^2) L \Lambda_2}, \tag{36}$$

we have that

$$\mathbb{E}_k[F(x_{k+1})] \leq F(x_k) - \frac{\alpha}{2} \|H_k^{1/2} \nabla F(x_k)\|^2. \tag{37}$$

Proof. By Assumption (B.1), the orthogonality condition, we have that

$$\mathbb{E}_k \left[\left\| H_k g_k^{S_k} - \frac{(H_k g_k^{S_k})^T (H_k \nabla F(x_k))}{\|H_k \nabla F(x_k)\|^2} H_k \nabla F(x_k) \right\|^2 \right] \leq \frac{\mathbb{E}_k \left[\left\| H_k g_k^i - \frac{(H_k g_k^i)^T (H_k \nabla F(x_k))}{\|H_k \nabla F(x_k)\|^2} H_k \nabla F(x_k) \right\|^2 \right]}{|S_k|} \tag{38}$$
$$\leq \nu^2 \|H_k \nabla F(x_k)\|^2.$$

Now, expanding the left hand side of inequality (38), we get

$$\begin{aligned}
&\mathbb{E}_k \left[\left\| H_k g_k^{S_k} - \frac{(H_k g_k^{S_k})^T (H_k \nabla F(x_k))}{\|H_k \nabla F(x_k)\|^2} H_k \nabla F(x_k) \right\|^2 \right] \\
&= \mathbb{E}_k \left[\|H_k g_k^{S_k}\|^2 \right] - \frac{2\mathbb{E}_k \left[\left((H_k g_k^{S_k})^T (H_k \nabla F(x_k)) \right)^2 \right]}{\|H_k \nabla F(x_k)\|^2} + \frac{\mathbb{E}_k \left[\left((H_k g_k^{S_k})^T (H_k \nabla F(x_k)) \right)^2 \right]}{\|H_k \nabla F(x_k)\|^2} \\
&= \mathbb{E}_k \left[\|H_k g_k^{S_k}\|^2 \right] - \frac{\mathbb{E}_k \left[\left((H_k g_k^{S_k})^T (H_k \nabla F(x_k)) \right)^2 \right]}{\|H_k \nabla F(x_k)\|^2} \\
&\leq \nu^2 \|H_k \nabla F(x_k)\|^2.
\end{aligned}$$

Therefore, rearranging gives the inequality

$$\mathbb{E}_k \left[\|H_k g_k^{S_k}\|^2 \right] \leq \frac{\mathbb{E}_k \left[\left((H_k g_k^{S_k})^T (H_k \nabla F(x_k)) \right)^2 \right]}{\|H_k \nabla F(x_k)\|^2} + \nu^2 \|H_k \nabla F(x_k)\|^2. \tag{39}$$

To bound the first term on the right side of this inequality, we use the inner product quasi-Newton test; in particular, $|S_k|$ satisfies

$$\begin{aligned}\mathbb{E}_k\left[\left((H_k\nabla F(x_k))^T(H_k g_k^{S_k})) - \|H_k\nabla F(x_k)\|^2\right)^2\right] &\leq \frac{\mathbb{E}_k\left[\left((H_k\nabla F(x_k))^T(H_k g_k^{i}) - \|H_k\nabla F(x_k)\|^2\right)^2\right]}{|S_k|} \\ &\leq \theta^2\|H_k\nabla F(x_k)\|^4, \end{aligned} \tag{40}$$

where the second inequality holds by the IPQN test. Since

$$\mathbb{E}_k\left[\left((H_k\nabla F(x_k))^T(H_k g_k^{S_k}) - \|H_k\nabla F(x_k)\|^2\right)^2\right] = \mathbb{E}_k\left[\left((H_k\nabla F(x_k))^T(H_k g_k^{S_k})\right)^2\right] - \|H_k\nabla F(x_k)\|^4, \tag{41}$$

we have

$$\begin{aligned}\mathbb{E}_k\left[\left((H_k g_k^{S_k})^T(H_k\nabla F(x_k))\right)^2\right] &\leq \|H_k\nabla F(x_k)\|^4 + \theta^2\|H_k\nabla F(x_k)\|^4 \\ &= (1+\theta^2)\|H_k\nabla F(x_k)\|^4, \end{aligned} \tag{42}$$

by (40) and (41). Substituting (42) into (39), we get the following bound on the length of the search direction:

$$\mathbb{E}_k\left[\|H_k g_k^{S_k}\|^2\right] \leq (1+\theta^2+\nu^2)\|H_k\nabla F(x_k)\|^2,$$

which proves (35). Using this inequality, Assumption B.2, and bounds on the Hessian and steplength (34) and (36), we have

$$\begin{aligned}\mathbb{E}_k[F(x_{k+1})] &\leq F(x_k) - \mathbb{E}_k\left[\alpha(H_k g_k^{S_k})^T\nabla F(x_k)\right] + \mathbb{E}_k\left[\frac{L\alpha^2}{2}\|H_k g_k^{S_k}\|^2\right] \\ &= F(x_k) - \alpha\nabla F(x_k)^T H_k\nabla F(x_k) + \frac{L\alpha^2}{2}\mathbb{E}_k[\|H_k g_k^{S_k}\|^2] \\ &\leq F(x_k) - \alpha\nabla F(x_k)^T H_k\nabla F(x_k) + \frac{L\alpha^2}{2}(1+\theta^2+\nu^2)\|H_k\nabla F(x_k)\|^2 \\ &= F(x_k) - \alpha(H_k^{1/2}\nabla F(x_k))^T\left(I - \frac{L\alpha(1+\theta^2+\nu^2)}{2}H_k\right)H_k^{1/2}\nabla F(x_k) \\ &\leq F(x_k) - \alpha\left(1 - \frac{L\Lambda_2\alpha(1+\theta^2+\nu^2)}{2}\right)\|H_k^{1/2}\nabla F(x_k)\|^2 \\ &\leq F(x_k) - \frac{\alpha}{2}\|H_k^{1/2}\nabla F(x_k)\|^2. \end{aligned}$$

□

We now show that the stochastic quasi-Newton iteration (29) with a fixed steplength α is linearly convergent when F is strongly convex. In the following discussion, x^* denotes the minimizer of F.

Theorem B.4. *Suppose that F is twice continuously differentiable and that there exist constants $0 < \mu \leq L$ such that*

$$\mu I \preceq \nabla^2 F(x) \preceq LI, \quad \forall x \in \mathbb{R}^d. \tag{43}$$

Let $\{x_k\}$ be generated by iteration (29), for any x_0, where $|S_k|$ is chosen by the (exact variance) inner product quasi-Newton test (31) and suppose that the assumptions (B.1) and (B.2) hold. Then, if α_k satisfies (36) we have that

$$\mathbb{E}[F(x_k) - F(x^*)] \leq \rho^k(F(x_0) - F(x^*)), \tag{44}$$

where x^* denotes the minimizer of F, and $\rho = 1 - \mu\Lambda_1\alpha$.

Proof. It is well-known (Bertsekas et al., 2003) that for strongly convex functions,

$$\|\nabla F(x_k)\|^2 \geq 2\mu[F(x_k) - F(x^*)].$$

Substituting this into (37) and subtracting $F(x^*)$ from both sides and using Assumption B.2, we obtain

$$\begin{aligned}\mathbb{E}_k[F(x_{k+1}) - F(x^*)] &\leq F(x_k) - F(x^*) - \frac{\alpha}{2}\|H_k^{1/2}\nabla F(x_k)\|^2 \\ &\leq F(x_k) - F(x^*) - \frac{\alpha}{2}\Lambda_1\|\nabla F(x_k)\|^2 \\ &\leq (1 - \mu\Lambda_1\alpha)(F(x_k) - F(x^*)).\end{aligned}$$

The theorem follows from taking total expectation.

□

We now consider the case when F is nonconvex and bounded below.

Theorem B.5. *Suppose that F is twice continuously differentiable and bounded below, and that there exists a constant $L > 0$ such that*

$$\nabla^2 F(x) \preceq LI, \quad \forall x \in \mathbb{R}^d. \tag{45}$$

Let $\{x_k\}$ be generated by iteration (29), for any x_0, where $|S_k|$ is chosen by the (exact variance) inner product quasi-Newton test (31) and suppose that the assumptions (B.1) and (B.2) hold. Then, if α_k satisfies (36), we have

$$\lim_{k\to\infty} \mathbb{E}[\|\nabla F(x_k)\|^2] \to 0. \tag{46}$$

Moreover, for any positive integer T we have that

$$\min_{0\leq k\leq T-1} \mathbb{E}[\|\nabla F(x_k)\|^2] \leq \frac{2}{\alpha T\Lambda_1}(F(x_0) - F_{min}),$$

where F_{min} is a lower bound on F in $\mathbb{R}^d$.

Proof. From Lemma B.3 and by taking total expectation, we have

$$\mathbb{E}[F(x_{k+1})] \leq \mathbb{E}[F(x_k)] - \frac{\alpha}{2}\mathbb{E}[\|H_k^{1/2}\nabla F(x_k)\|^2],$$

and hence

$$\mathbb{E}[\|H_k^{1/2}\nabla F(x_k)\|^2] \leq \frac{2}{\alpha}\mathbb{E}[F(x_k) - F(x_{k+1})].$$

Summing both sides of this inequality from $k = 0$ to $T - 1$, and since F is bounded below by F_{min}, we get

$$\sum_{k=0}^{T-1} \mathbb{E}[\|H_k^{1/2}\nabla F(x_k)\|^2] \leq \frac{2}{\alpha}\mathbb{E}[F(x_0) - F(x_{\mathrm{T}})] \leq \frac{2}{\alpha}[F(x_0) - F_{min}].$$

Using the bound on the eigenvalues of H_k and taking limits, we obtain

$$\Lambda_1 \lim_{T\to\infty}\sum_{k=0}^{T-1}\mathbb{E}[\|\nabla F(x_k)\|^2] \leq \lim_{T\to\infty}\sum_{k=0}^{T-1}\mathbb{E}[\|H_k^{1/2}\nabla F(x_k)\|^2] < \infty,$$

which implies (46). We can also conclude that

$$\min_{0\leq k\leq T-1}\mathbb{E}[\|\nabla F(x_k)\|^2] \leq \frac{1}{T}\sum_{k=0}^{T}\mathbb{E}[\|\nabla F(x_k)\|^2] \leq \frac{2}{\alpha T\Lambda_1}(F(x_0) - F_{min}).$$

□

C. Additional Numerical Experiments

C.1. Datasets

Table 2 summarizes the datasets used for the experiments. Some of these datasets divide the data into training and testing sets; for the rest, we randomly divide the data so that the training set constitutes 90% of the total.

Table 2. Characteristics of all datasets used in the experiments.

Dataset	# Data Points (train; test)	# Features	# Classes	Source
gisette	(6,000; 1,000)	5,000	2	(Chang & Lin, 2011)
mushrooms	(7,311; 813)	112	2	(Chang & Lin, 2011)
sido	(11,410; 1,268)	4,932	2	(Guyon et al., 2008)
ijcnn	(35,000; 91701)	22	2	(Chang & Lin, 2011)
spam	(82,970; 9,219)	823,470	2	(Cormack & Lynam, 2005; Carbonetto, 2009)
alpha	(450,000; 50,000)	500	2	synthetic
covertype	(522,910; 58,102)	54	2	(Chang & Lin, 2011)
url	(2,156,517; 239,613)	3,231,961	2	(Chang & Lin, 2011)
MNIST	(60,000; 10,000)	28×28	10	(LeCun et al., 1998)
CIFAR-10	(50,000; 10,000)	32×32	10	(Krizhevsky, 2009)

The alpha dataset is a synthetic dataset that is available at `ftp://largescale.ml.tu-berlin.de`.

C.2. Logistic Regression Experiments

We report the numerical results on binary classification logistic regression problems on the 8 datasets given in Table 2. We plot the performance measured in terms of training error, test loss and test accuracy against gradient evaluations. We also report the behavior of the batch sizes and steplengths for both variants of the PBQN method.

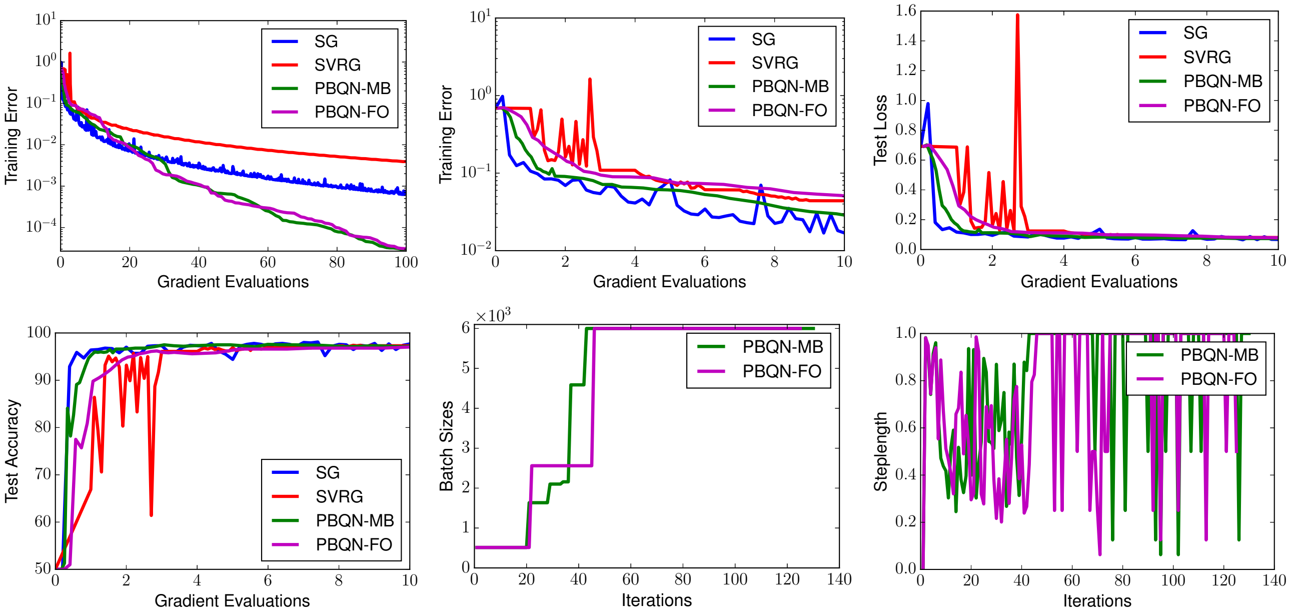

Figure 5. **gisette dataset:** Performance of the progressive batching L-BFGS methods, with multi-batch (MB) (25% overlap) and full-overlap (FO) approaches, and the SG and SVRG methods.

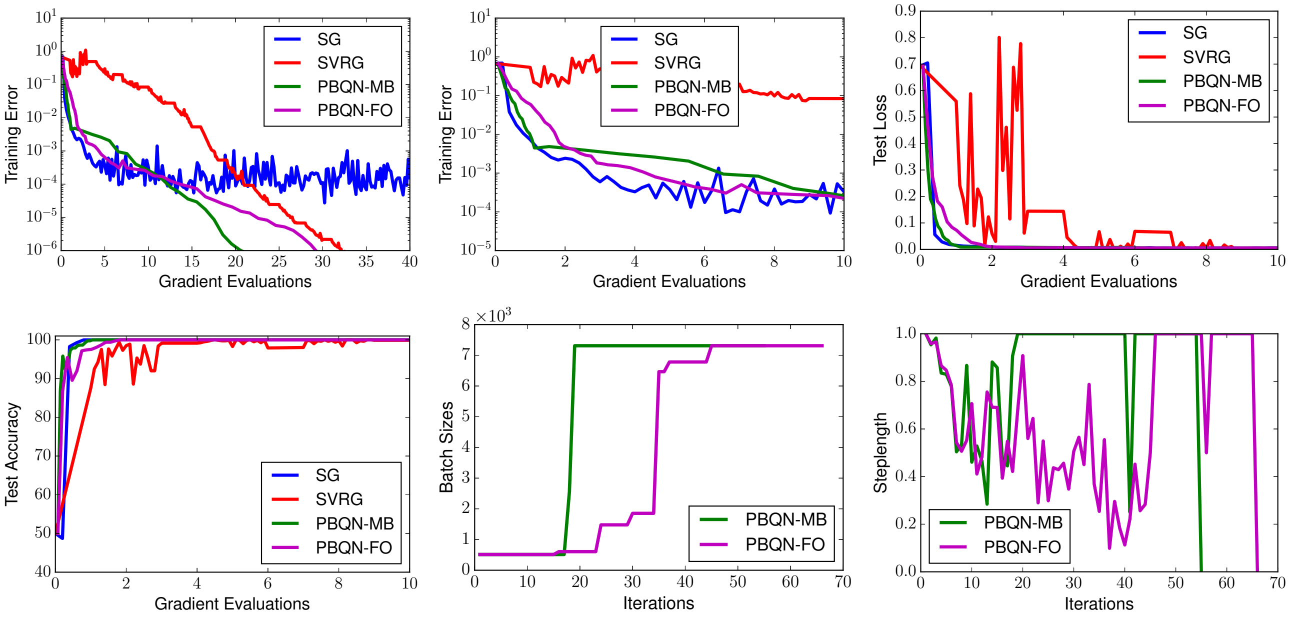

Figure 6. **mushrooms dataset:** Performance of the progressive batching L-BFGS methods, with multi-batch (MB) (25% overlap) and full-overlap (FO) approaches, and the SG and SVRG methods.

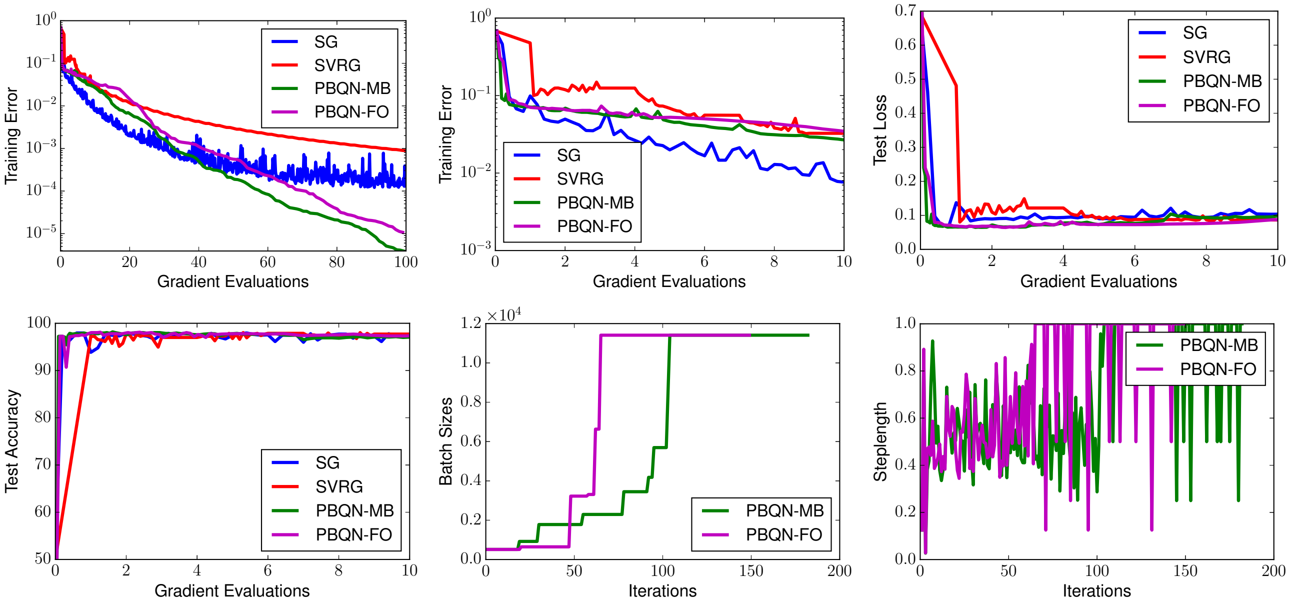

Figure 7. **sido dataset:** Performance of the progressive batching L-BFGS methods, with multi-batch (MB) (25% overlap) and full-overlap (FO) approaches, and the SG and SVRG methods.

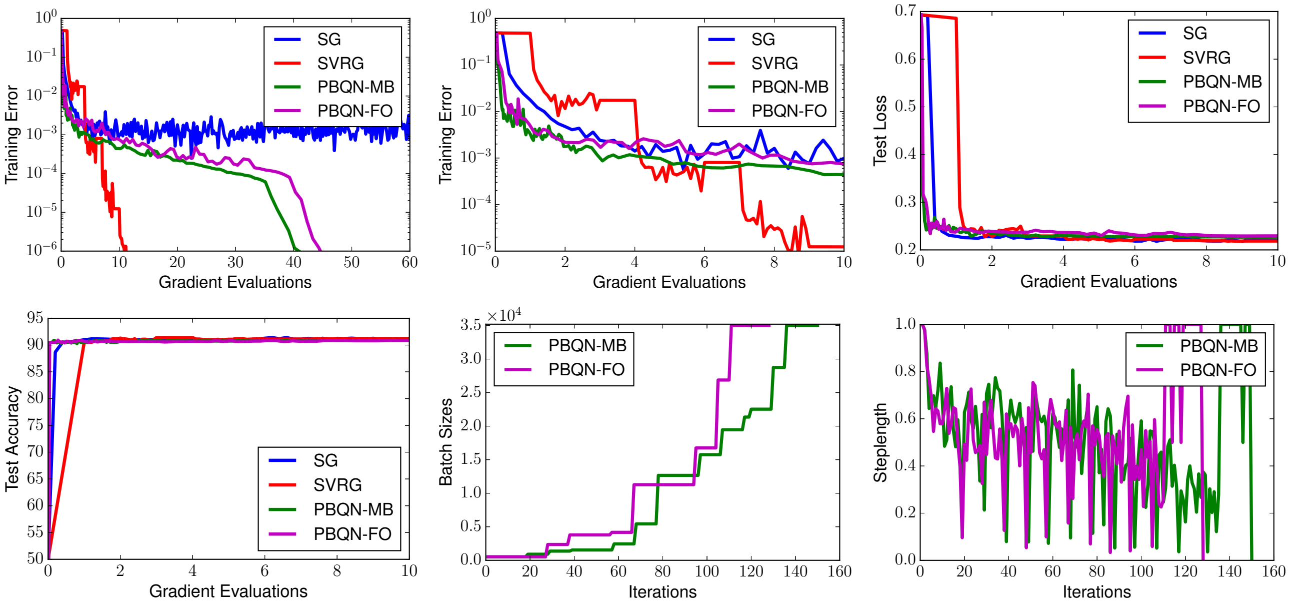

Figure 8. **ijcnn dataset:** Performance of the progressive batching L-BFGS methods, with multi-batch (MB) (25% overlap) and full-overlap(FO) approaches, and the SG and SVRG methods.

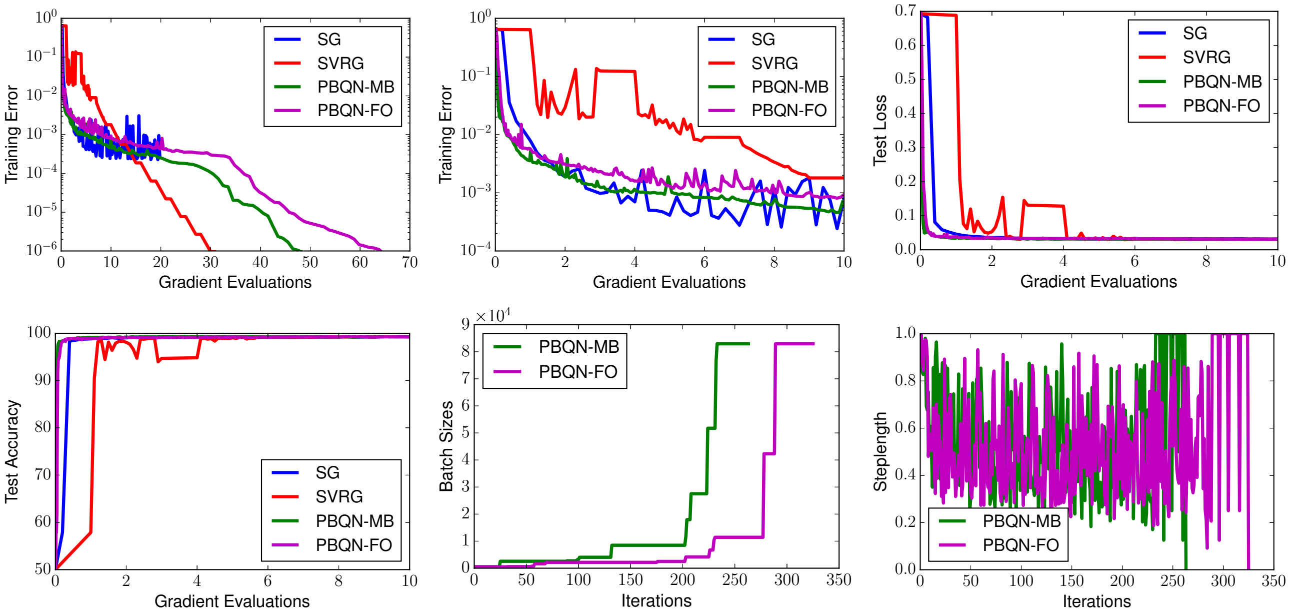

Figure 9. **spam dataset:** Performance of the progressive batching L-BFGS methods, with multi-batch (MB) (25% overlap) and full-overlap (FO) approaches, and the SG and SVRG methods.

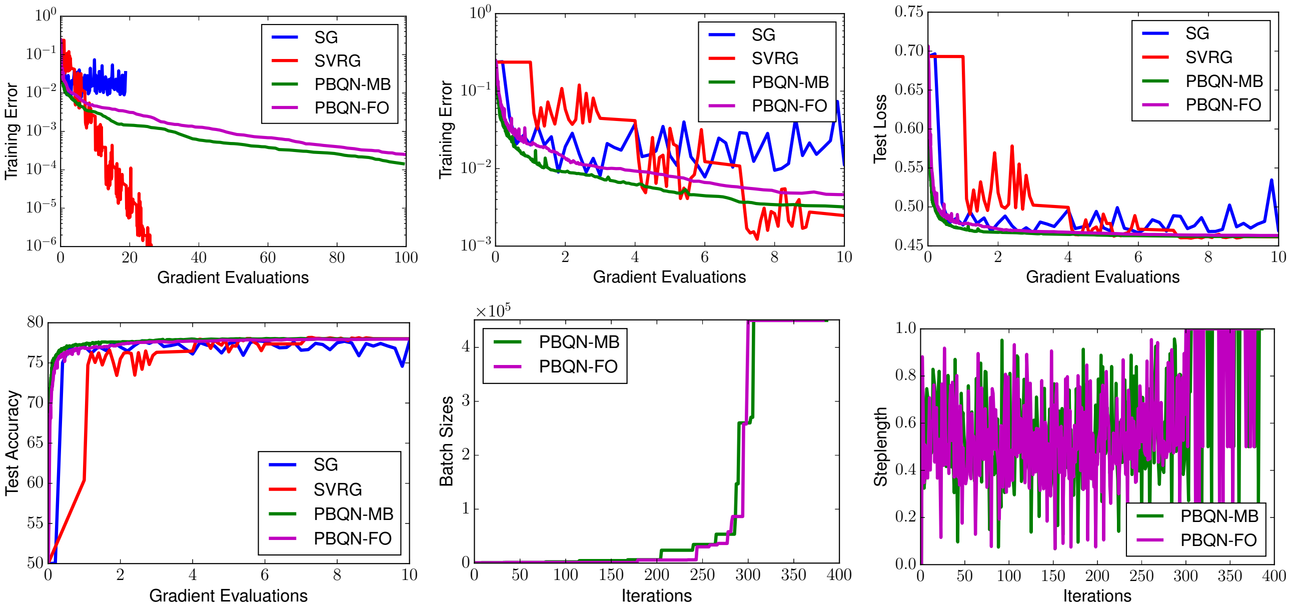

Figure 10. **alpha dataset:** Performance of the progressive batching L-BFGS methods, with multi-batch (MB) (25% overlap) and full-overlap (FO) approaches, and the SG and SVRG methods.

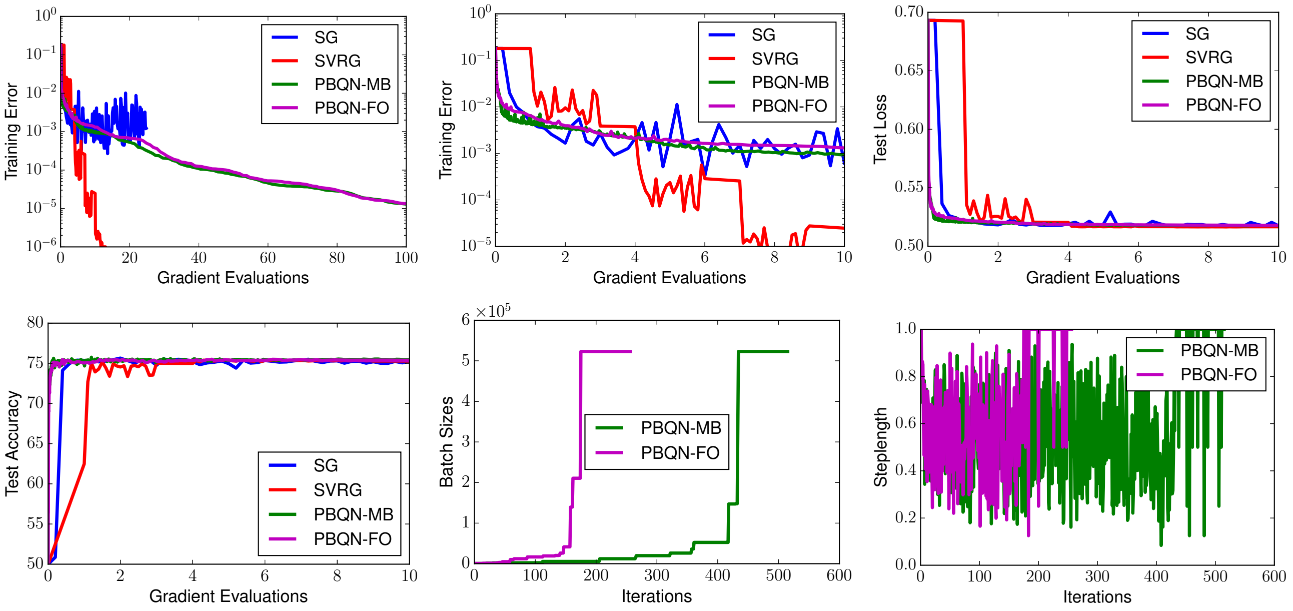

Figure 11. **covertype dataset:** Performance of the progressive batching L-BFGS methods, with multi-batch (MB) (25% overlap) and full-overlap (FO) approaches, and the SG and SVRG methods.

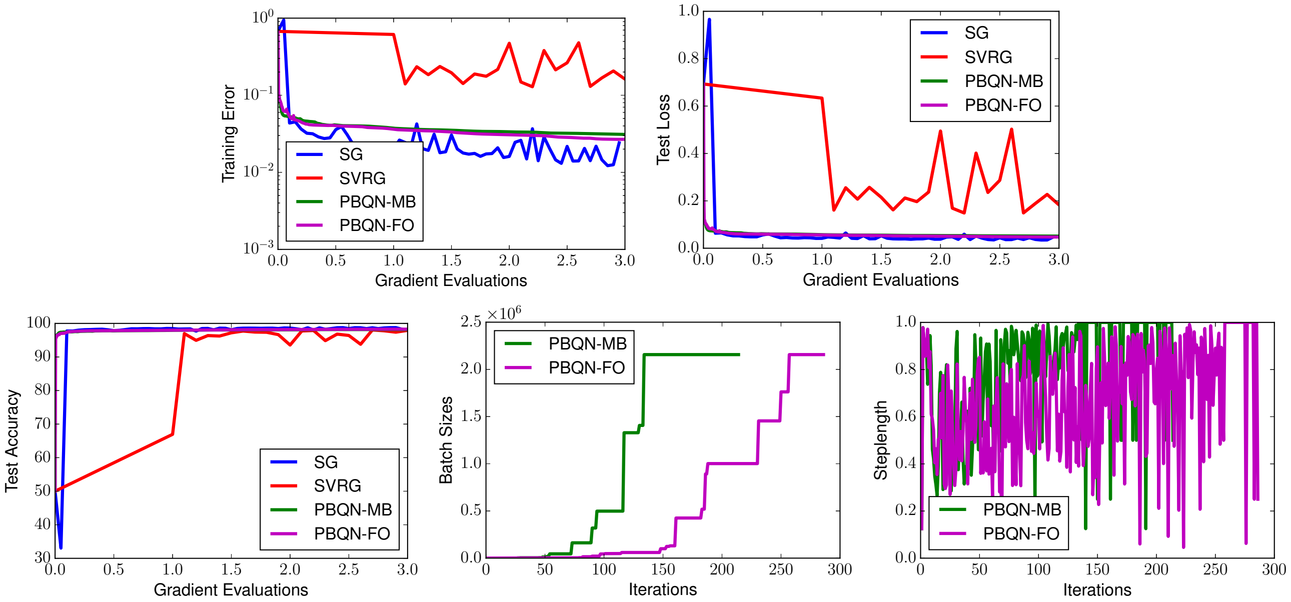

Figure 12. **url dataset:** Performance of the progressive batching L-BFGS methods, with multi-batch (MB) (25% overlap) and full-overlap (FO) approaches, and the SG and SVRG methods. Note that we only ran the SG and SVRG algorithms for 3 gradient evaluations since the equivalent number of iterations already reached of order of magnitude 10^7.

C.3. Neural Network Experiments

We describe each neural network architecture below. We plot the training loss, test loss and test accuracy against the total number of iterations and gradient evaluations. We also report the behavior of the batch sizes and steplengths for both variants of the PBQN method.

C.3.1. CIFAR-10 Convolutional Network ($\mathcal{C}$) Architecture

The small convolutional neural network (ConvNet) is a 2-layer convolutional network with two alternating stages of 5×5 kernels and 2×2 max pooling followed by a fully connected layer with 1000 ReLU units. The first convolutional layer yields 6 output channels and the second convolutional layer yields 16 output channels.

C.3.2. CIFAR-10 and MNIST AlexNet-like Network ($\mathcal{A}_1, \mathcal{A}_2$) Architecture

The larger convolutional network (AlexNet) is an adaptation of the AlexNet architecture (Krizhevsky et al., 2012) for CIFAR-10 and MNIST. The CIFAR-10 version consists of three convolutional layers with max pooling followed by two fully-connected layers. The first convolutional layer uses a 5×5 kernel with a stride of 2 and 64 output channels. The second and third convolutional layers use a 3×3 kernel with a stride of 1 and 64 output channels. Following each convolutional layer is a set of ReLU activations and 3×3 max poolings with strides of 2. This is all followed by two fully-connected layers with 384 and 192 neurons with ReLU activations, respectively. The MNIST version of this network modifies this by only using a 2×2 max pooling layer after the last convolutional layer.

C.3.3. CIFAR-10 Residual Network ($\mathcal{R}$) Architecture

The residual network (ResNet18) is a slight modification of the ImageNet ResNet18 architecture for CIFAR-10 (He et al., 2016). It follows the same architecture as ResNet18 for ImageNet but removes the global average pooling layer before the 1000 neuron fully-connected layer. ReLU activations and max poolings are included appropriately.

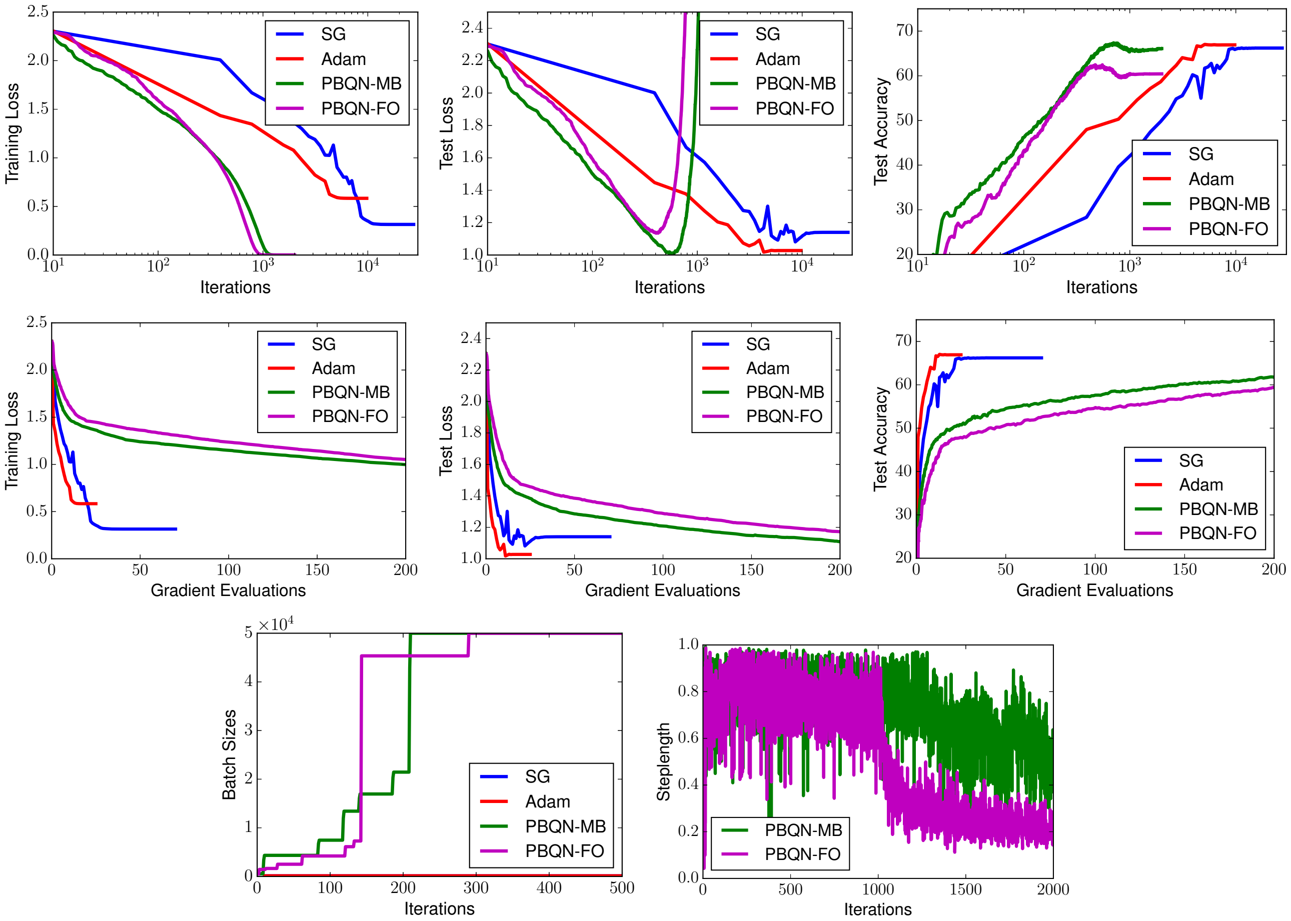

Figure 13. **CIFAR-10 ConvNet** $(\mathcal{C})$**:** Performance of the progressive batching L-BFGS methods, with multi-batch (MB) (25% overlap) and full-overlap (FO) approaches, and the SG and Adam methods. The best results for L-BFGS are achieved with $\theta = 0.9$.

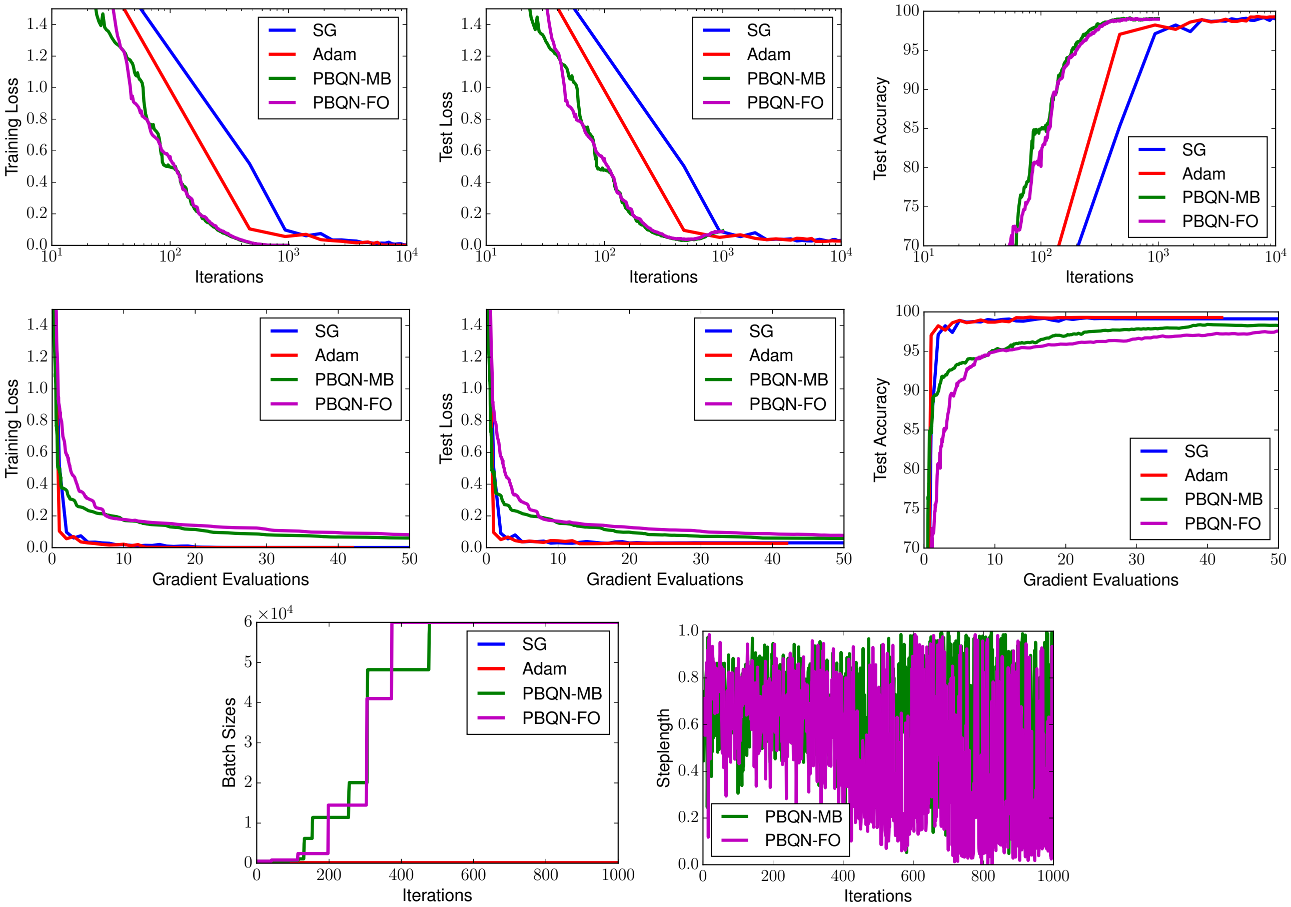

Figure 14. **MNIST AlexNet** $(\mathcal{A}_1)$**:** Performance of the progressive batching L-BFGS methods, with multi-batch (MB) (25% overlap) and full-overlap (FO) approaches, and the SG and Adam methods. The best results for L-BFGS are achieved with $\theta = 2$.

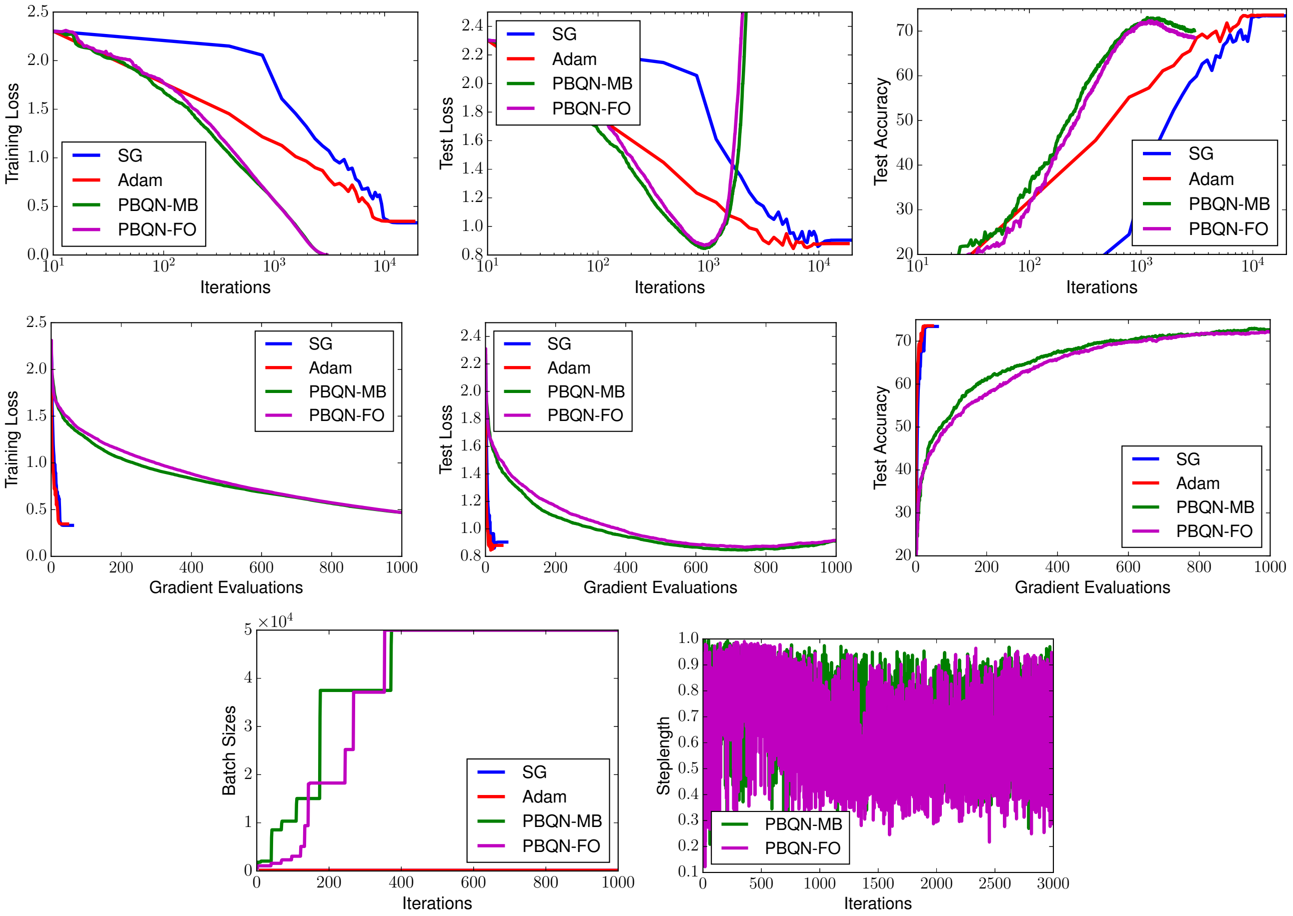

Figure 15. **CIFAR-10 AlexNet** $(\mathcal{A}_2)$**:** Performance of the progressive batching L-BFGS methods, with multi-batch (MB) (25% overlap) and full-overlap (FO) approaches, and the SG and Adam methods. The best results for L-BFGS are achieved with $\theta = 0.9$.

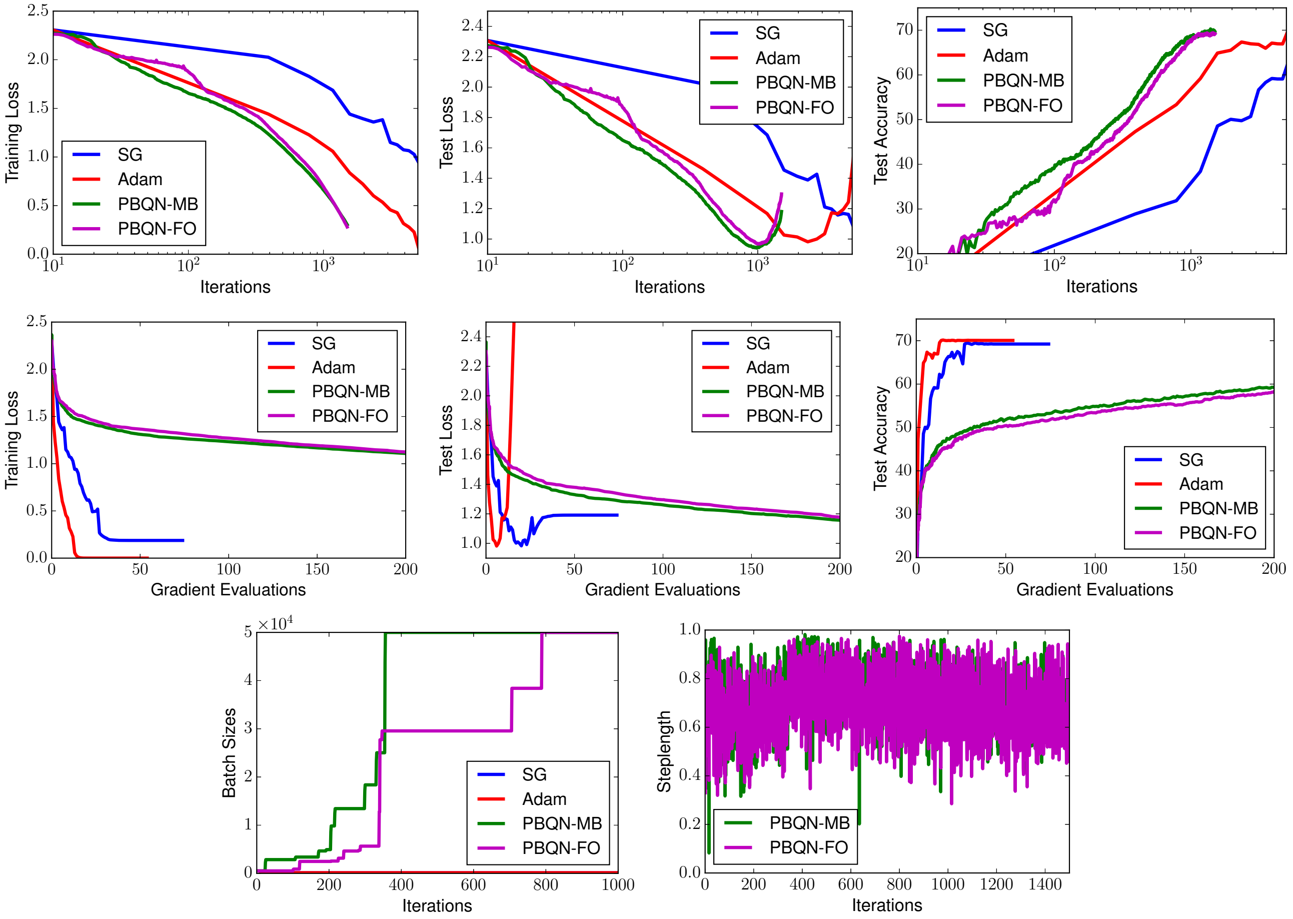

Figure 16. **CIFAR-10 ResNet18** ($\mathcal{R}$)**:** Performance of the progressive batching L-BFGS methods, with multi-batch (MB) (25% overlap) and full-overlap (FO) approaches, and the SG and Adam methods. The best results for L-BFGS are achieved with $\theta = 2$.

D. Performance Model

The use of increasing batch sizes in the PBQN algorithm yields a larger effective batch size than the SG method, allowing PBQN to scale to a larger number of nodes than currently permissible even with large-batch training (Goyal et al., 2017). With improved scalability and richer gradient information, we expect reduction in training time. To demonstrate the potential to reduce training time of a parallelized implementation of PBQN, we extend the idealized performance model from (Keskar et al., 2016) to the PBQN algorithm. For PBQN to be competitive, it must achieve the following: (i) the quality of its solution should match or improve SG's solution (as shown in Table 1 of the main paper); (ii) it should utilize a larger effective batch size; and (iii) it should converge to the solution in a lower number of iterations. We provide an initial analysis for this by establishing the analytic requirements for improved training time; we leave discussion on implementation details, memory requirements, and large-scale experiments for future work.

Let the effective batch size for PBQN and conventional SG batch size be denoted as $\widehat{B_L}$ and B_S, respectively. From Algorithm 1, we observe that the PBQN iteration involves extra computation in addition to the gradient computation as in SG. The additional steps are as follows: the L-BFGS two-loop recursion, which includes several operations over the stored curvature pairs and network parameters (Algorithm 1:6); the stochastic line search for identifying the steplength (Algorithm 1:7-16); and curvature pair updating (Algorithm 1:18-21). However, most of these supplemental operations are performed on the weights of the network, which is orders of magnitude lower than computing the gradient. The two-loop recursion performs $O(10)$ operations over the network parameters and curvature pairs. The cost for variance estimation is negligible since we may use a fixed number of samples throughout the run for its computation which can be parallelized while avoiding becoming a serial bottleneck.

The only exception is the stochastic line search, which requires additional forward propagations over the model for different sets of network parameters. However, this happens only when the step-length is not accepted, which happens infrequently in practice. We make the pessimistic assumption of an addition forward propagation every iteration, amounting to an additional $\frac{1}{3}$ the cost of the gradient computation (forward propagation, back propagation with respect to activations and weights). Hence, the ratio of cost-per-iteration for PBQN C_L to SG's cost-per-iteration C_S is $\frac{4}{3}$. Let I_S and I_L be the number of iterations that it takes SG and PBQN, respectively, to reach similar test accuracy. The target number of nodes to be used for training is N, such that $N < \widehat{B_L}$. For N nodes, the parallel efficiency of SG is assumed to be $P_e(N)$ and we assume that for the target node count, there is no drop in parallel efficiency for PBQN due to the large effective batch size.

For a lower training time with the PBQN method, the following relation should hold:

$$I_L C_L \frac{\widehat{B_L}}{N} < I_S C_S \frac{B_S}{N P_e(N)}. \tag{47}$$

In terms of iterations, we can rewrite this as

$$\frac{I_L}{I_S} < \frac{C_S}{C_L} \frac{B_S}{\widehat{B_L}} \frac{1}{P_e(N)}. \tag{48}$$

Assuming target node count $N = B_S < \hat{B}_L$, the scaling efficiency of SG drops significantly due to the reduced work per single node, giving a parallel efficiency of $P_e(N) = 0.2$; see (Kurth et al., 2017; You et al., 2017). If we additionally assume that effective batch size for PBQN is 4× larger, with SG large batch ≈ 8K and PBQN ≈ 32K as observed in our experiments (from Section 4), this gives $\widehat{B_L}/B_S = 4$. PBQN must converge with about the same number of iterations as SG in order to achieve lower training time. From Section 4, the results show that PBQN converges in significantly fewer iterations than SG, hence establishing the potential for lower training times. We refer the reader to (Das et al., 2016) for a more detailed model and commentary on the effect of batch size on performance.