Communication-Efficient Stochastic Gradient MCMC for Neural Networks

Chunyuan Li,[1] **Changyou Chen,**[2] **Yunchen Pu,**[3] **Ricardo Henao,**[4] **Lawrence Carin**[4]
[1]Microsoft Research, Redmond [2]University at Buffalo, SUNY [3]Facebook [4]Duke University

Abstract

Learning probability distributions on the weights of neural networks has recently proven beneficial in many applications. Bayesian methods such as Stochastic Gradient Markov Chain Monte Carlo (SG-MCMC) offer an elegant framework to reason about model uncertainty in neural networks. However, these advantages usually come with a high computational cost. We propose accelerating SG-MCMC under the master-worker framework: workers *asynchronously* and in parallel share responsibility for gradient computations, while the master collects the final samples. To reduce communication overhead, two protocols (downpour and elastic) are developed to allow periodic interaction between the master and workers. We provide a theoretical analysis on the finite-time estimation consistency of posterior expectations, and establish connections to sample thinning. Our experiments on various neural networks demonstrate that the proposed algorithms can greatly reduce training time while achieving comparable (or better) test accuracy/log-likelihood levels, relative to traditional SG-MCMC. When applied to reinforcement learning, it naturally provides exploration for asynchronous policy optimization, with encouraging performance improvement.

Introduction

Deep neural networks (DNNs) have become widely used in machine learning, often achieving state-of-the-art performance across a variety of applications. Recently, there has been considerable interest in developing principled yet scalable Bayesian learning methods (Blundell et al. 2015; Hernández-Lobato and Adams 2015; Korattikara et al. 2015; Kingma, Salimans, and Welling 2015; Gal and Ghahramani 2016; Bui et al. 2016; Liu and Wang 2016), to obtain good estimates of uncertainty for the DNN weights. The learned uncertainty is then transferred to predictions during testing, to help alleviate overfitting, and/or to quantify confidence about predictive estimates. Markov chain Monte Carlo (MCMC) is perhaps the most well-known family of (sample-based) uncertainty-estimation methods for DNNs. Hamiltonian Monte Carlo (HMC) (Neal 1995) is a popular member of this family. More recently, mini-batch-based methods, such as stochastic gradient MCMC (SG-MCMC) (Welling and Teh 2011; Chen, Fox, and Guestrin 2014; Ding et al. 2014; Li et al. 2016a; Liu, Zhu, and Song 2016; Durmus et al. 2016; Fu and Zhang 2017; Cong et al. 2017; Li et al. 2016b; Gan et al. 2017) have been developed to deal with the inherent scalability problems of MCMC methods when applied to large data sets.

The standard formulation of SG-MCMC is sequential, *i.e.*, it alternates between two operations: gradient evaluation and parameter updates. To improve the speed of SG-MCMC algorithms, we consider estimating weight uncertainty through parallelizing across multiple compute cores (processing nodes or processors). A natural approach to parallel SG-MCMC consists of allowing each compute core to access the full data set, run separate SG-MCMC chains (without communication), and then combine their results as independent samples. Though a larger number of samples can be obtained, each chain has in principle roughly the same mixing speed w.r.t. serial SG-MCMC. However, it is likely to be better at exploring parameter space, because each chain can be initialized to a different starting point in such a space.

An alternative approach to implementing parallel SG-MCMC allows communication between compute cores: some cores (workers) are tasked with improving the mixing (via stochastic gradients) of sample paths (Markov chains) maintained by a different core (master), which regularly sends globally aggregated parameter summaries back to the workers. The cross-talk between workers through the master allows the parameter space to be efficiently represented with a smaller number of samples, collected by the master core. Unfortunately, communication overhead prevents instantaneous communication between master and workers at every step of the learning procedure. One compromise strategy consists of allowing the master to interact with workers periodically. This raises a different concern: to the best of our knowledge, there are no effective protocols for SG-MCMC to coordinate the exchange of information between cores, under communication constraints.

In this paper, we propose leveraging parallelization to accelerate SG-MCMC under a master-worker framework. Multiple workers run individual SG-MCMC chains to explore the parameter space at the same time; they periodically communicate with the master to share information about model parameters. In the distributed *optimization* literature, numerous approaches have been proposed for coordination of work among concurrent processes, including downpour stochastic gradient descent (SGD) (Dean and others 2012) and elastic SGD (Zhang, Choromanska, and LeCun 2015). We argue that these ideas can be properly adjusted to improve SG-MCMC on two fronts: (i) For training, the mixing speed of samples kept on the master is improved, as more gradient evaluations per time interval can be performed. (ii) The periodic communication protocols help reduce communication overhead, while maintaining high-quality testing performance, using a small number of *effective* samples.

Our contributions are summarized as follows. First, we develop two periodic communication protocols to accelerate SG-MCMC. In support of these ideas, we provide finite-time estimation error bounds, and show their connection to sample thinning, to illustrate its role in aggregating knowledge from multiple chains into a more compact sample "ensemble". Second, we apply the new algorithms to parallelized estimation of uncertainty for DNN weights. Experimental results demonstrate that it provides significant acceleration without performance penalties, including test accuracy and prediction uncertainty, compared to traditional SG-MCMC. Third, we develop the first asynchronous SG-MCMC to learn the policy weight uncertainty in reinforcement learning. It naturally favours exploration and stabilize training, with improved performance.

Weight Uncertainty with SG-MCMC

Bayesian View of Neural Networks

Consider i.i.d. data $\mathcal{D} = \{\mathbf{D}_1, \cdots, \mathbf{D}_N\}$, where $\mathbf{D}_n \triangleq (\mathbf{X}_n, \mathbf{Y}_n)$ with input $\mathbf{X}_n$ and output $\mathbf{Y}_n$. The explicit forms of $\mathbf{X}_n$ and $\mathbf{Y}_n$ can be specified for different models and applications. Our goal is to learn model parameters $\boldsymbol{\theta}$ to best characterize the relationship from $\mathbf{X}_n$ to $\mathbf{Y}_n$, via the data likelihood $p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\mathbf{D}_n|\boldsymbol{\theta})$. In Bayesian statistics, one sets a prior on $\boldsymbol{\theta}$ via distribution $p(\boldsymbol{\theta})$. The posterior $p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})$ reflects the belief concerning the model parameter distribution after observing data $\mathcal{D}$. Different DNNs imply different parametric forms of the likelihood $p(\mathcal{D}|\boldsymbol{\theta})$, thus providing a Bayesian treatment for the feedforward, convolutional and recurrent neural networks.

During testing, given an input $\tilde{\mathbf{X}}$ (with missing output $\tilde{\mathbf{Y}}$), the uncertainty learned in training is transferred to prediction, yielding the posterior predictive distribution: $\bar{\phi} \triangleq p(\tilde{\mathbf{Y}}|\tilde{\mathbf{X}}, \mathcal{D}) = \int_{\boldsymbol{\theta}} p(\tilde{\mathbf{Y}}|\tilde{\mathbf{X}}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \mathrm{d}\boldsymbol{\theta}$. Typically $\bar{\phi}$ is not available in closed form. Online versions of variational Bayes (Blundell et al. 2015) and expectation propagation (Hernández-Lobato and Adams 2015) have been developed to approximate $\bar{\phi}$. Alternatively, we employ SG-MCMC to provide sample-based approximations to the posterior expectation.

Single-Worker SG-MCMC

The negative log-posterior is

$$U(\boldsymbol{\theta}) \triangleq -\log p(\boldsymbol{\theta}) - \sum_{n=1}^{N} \log p(\mathbf{D}_n|\boldsymbol{\theta}) \,, \qquad (1)$$

In (Ma, Chen, and Fox 2015) a framework is proposed to generate approximate samples from a family of continuous-time diffusions, whose stationary distribution coincides with the posterior distribution of interest. To generate samples from these diffusions, numerical methods are adopted to discretize the continuous-time system with step-size ϵ_t for step t. As a result, the t-th sample is typically generated using the update rule (Ma, Chen, and Fox 2015):

$$\begin{aligned} \boldsymbol{z}_{t+1} = \boldsymbol{z}_t - \epsilon_t \left[\left(W(\boldsymbol{z}_t) + Q(\boldsymbol{z}_t) \right) \nabla_{\boldsymbol{\theta}} H(\boldsymbol{z}_t) + \Gamma(\boldsymbol{z}_t) \right] + \boldsymbol{\xi}_t, \\ \boldsymbol{\xi}_t \sim \mathcal{N}(0, 2\epsilon_t W(\boldsymbol{z}_t)) \,, \qquad (2) \end{aligned}$$

where $\boldsymbol{z}$ is the system state containing the model parameters $\boldsymbol{\theta}$, $\nabla_{\boldsymbol{\theta}} H$ is often related to the gradient $\boldsymbol{f} = \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$ and $\{W(\boldsymbol{z}_t), Q(\boldsymbol{z}_t), \Gamma(\boldsymbol{z}_t)\}$ are functions of $\boldsymbol{z}_t$ to be specified for different algorithms (defined below). The computational efficiency of these sampling methods largely relies on the cost of computing $\boldsymbol{f}$. When N is large, SG-MCMC methods (Welling and Teh 2011; Chen, Fox, and Guestrin 2014; Ding et al. 2014; Li et al. 2016a) extend the sampling method in (2), by proposing to evaluate the gradient on a *mini-batch* of data $\mathcal{D}_M = \{\mathbf{D}_{i_1}, \cdots, \mathbf{D}_{i_M}\}$, where $\{i_1, \cdots, i_M\}$ are random subsets of the set $\{1, 2, \cdots, N\}$, such that $M \ll N$. Therefore, $\boldsymbol{f}_t$ is approximated with *stochastic gradient*:

$$\begin{aligned} &\tilde{\boldsymbol{f}}_t = \nabla \tilde{U}(\boldsymbol{\theta}_t), \text{ where} \\ &\tilde{U}(\boldsymbol{\theta}_t) \triangleq -\log p(\boldsymbol{\theta}_t) - \tfrac{N}{M} \textstyle\sum_{m=1}^{M} \log p(\mathbf{D}_{i_m}|\boldsymbol{\theta}_t) \,. \qquad (3) \end{aligned}$$

In (2), the choice of $W(\cdot)$, $Q(\cdot)$ and $\Gamma(\cdot)$ defines various SG-MCMC algorithms; see (Ma, Chen, and Fox 2015) for details, and below we describe two algorithms considered in this paper.

Stochastic Gradient Langevin Dynamics (SGLD) corresponds to $\boldsymbol{z} = \boldsymbol{\theta}$, $H(\boldsymbol{\theta}) = U(\boldsymbol{\theta})$, $W(\boldsymbol{\theta}) = \mathbf{I}$, $Q(\boldsymbol{\theta}) = \mathbf{0}$, and $\Gamma(\boldsymbol{\theta}) = \mathbf{0}$ (Welling and Teh 2011).

Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) extends HMC to use mini-batches for updates; $\boldsymbol{z} = (\boldsymbol{\theta}, \boldsymbol{q})$, $H(\boldsymbol{\theta}, \boldsymbol{q}) = U(\boldsymbol{\theta}) + \frac{1}{2}\boldsymbol{q}^T\boldsymbol{q}$, $W(\boldsymbol{\theta}, \boldsymbol{q}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & B\mathbf{I} \end{bmatrix}$, $Q(\boldsymbol{\theta}, \boldsymbol{q}) = \begin{bmatrix} \mathbf{0} & -\mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}$, and $\Gamma(\boldsymbol{\theta}, \boldsymbol{q}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$, where $\boldsymbol{q}$ is the momentum variable, and B is a constant (Chen, Fox, and Guestrin 2014).

After obtaining L parameter samples $\{\boldsymbol{\theta}_l\}_{l=1}^{L}$, a Monte Carlo (MC) approximation is assembled to estimate the expectation: $\bar{\phi} \approx \hat{\phi} \triangleq \frac{1}{L}\sum_{l=1}^{L} p(\tilde{\mathbf{Y}}|\tilde{\mathbf{X}}, \boldsymbol{\theta}_l)$. The key characteristic of SG-MCMC is that the cost of gradient evaluation is reduced from $\mathcal{O}(N)$ to $\mathcal{O}(M)$, thus allowing for many more parameter updates per unit time. In practice, this leads to much shorter burn-in times and faster mixing speeds (Ahn, Shahbaba, and Welling 2014). In the same spirit, SG-MCMC can be further accelerated if, via parallelization, more gradient evaluations per unit time can be completed.

Speed up with Communication Constraints

From Single Worker to Multiple Workers

We separate the two operations in SG-MCMC into two workflows: (i) parallel gradient evaluation on multiple workers, and (ii) parameter updates on the master. From this perspective, traditional SG-MCMC (serial) can be seen as evaluating gradients on a single worker, while the master waits until the worker has completed its task, as shown in Fig. 1(a).

We first introduce the intuition for accelerating SG-MCMC in the *synchronous* setting, shown in Fig. 1(b). In a star-shaped compute architecture, the master maintains the summary model parameters $\boldsymbol{\theta}$ (**center parameters**), P workers keep their own copy of parameters, $\boldsymbol{\theta}^{(p)}$ (**intermediate parameters**), which are updated by running SG-MCMC with their own gradients $\tilde{\boldsymbol{f}}^{(p)}$. All computing cores share the same global clock, and the center parameters $\boldsymbol{\theta}_t$ at the t-th step are updated by *aggregating* over $\{\boldsymbol{\theta}_t^{(p)}\}_{p=1}^P$. Ideally (with no communication overhead and identical worker capacity), the mini-batch is split on the P workers, reducing computation of gradient evaluation to $\mathcal{O}(M/P)$. To make the center parameters a proper sample from the posterior, higher noise is injected to each sample on each worker, *i.e.*, $\boldsymbol{\xi}_t$ in (2) is drawn instead from $\mathcal{N}(0, 2\epsilon_t PW(\boldsymbol{z}_t))$. We consider two aggregation schemes, summarized in Table 1:

- In Scheme A, the master averages the intermediate parameters from all the workers at the current step, t, then updated center parameters are sent back to each worker.
- In Scheme B, the master averages the parameters obtained from the workers and smoothes over time. This is inspired by the Adam algorithm (Kingma and Ba 2015), where a weighted average of historical gradients reduces the variance in gradient estimation. The center parameters are smoothed with parameter α (see Table 1), using historical parameters stored in the master, where $\alpha \in (0, 1]$ is the decay weight controlling the amount of smoothing (we use 0.9 as default value).

There are two practical issues with the synchronous setting: (i) instantaneous communication after every gradient evaluation results in prohibitive communication overhead, and (ii) synchronous updates can be inefficient, when workers have different processing capabilities or when certain workers are offline. We extend the two schemes in Table 1 to the *asynchronous* setting via periodic communication.

Periodic Communication Protocols

In the asynchronous setting, each worker maintains its own clock $t^{(p)}$, which starts from 0 and is incremented by 1 after each evaluation of $\tilde{\boldsymbol{f}}^{(p)}$. Multiple workers asynchronously update center parameters, leading to potentially higher mixing speeds compared to traditional SG-MCMC. For illustration, Fig. 1(c) shows two workers with different processing abilities. The gradient on worker p_1 is evaluated on the more recent $\boldsymbol{\theta}_2$ instead of $\boldsymbol{\theta}_0$, while the gradient on worker p_2 is evaluated on $\boldsymbol{\theta}_3$ instead of $\boldsymbol{\theta}_2$. To reduce the communication overhead, the master performs an update only when the local worker has finished π steps of its gradient evaluations, where π denotes the *communication period*. Figure 1(d) shows an example with $\pi = 2$. Below we develop two periodic communication protocols, based on (Dean and others 2012; Zhang, Choromanska, and LeCun 2015).

π-downpour protocol Scheme A in Table 1 is extended to accumulate the update during the π steps in $\boldsymbol{\nu}$ (Dean and others 2012), which denotes the space that the worker has explored since its last communication. Besides $\boldsymbol{\theta}^{(p)}$, the p-th worker also maintains $\boldsymbol{\nu}^{(p)}$. When the next communication happens, the master absorbs $\boldsymbol{\nu}^{(p)}$ and sends new center parameters back to the p-th worker to replace (update) $\boldsymbol{\theta}^{(p)}$.

(π, α)-elastic protocol The weighted average in Scheme B results in a difference between historical center parameter and current parameters sent from a worker, as the former are smoothed over previous steps. The master waits until the p-th worker has sent the requested $\boldsymbol{\theta}^{(p)}$, then computes the *elastic difference* $\alpha(\boldsymbol{\theta}^{(p)} - \boldsymbol{\theta})$ (Zhang, Choromanska, and LeCun 2015). Next, this difference is sent back to the worker who then updates $\boldsymbol{\theta}^{(p)}$.

Any SG-MCMC method can in principle incorporate the above protocols for speed up. Within this framework, we have implemented two novel algorithms as examples, to illustrate these procedures.

Downpour SGLD Algorithm 1 employs an asynchronous parallel procedure to combine SGLD (lines 5-9), with the downpour protocol (lines 12-17), termed *downpour SGLD*. Recent preconditioning techniques (Li et al. 2016a; Simsekli et al. 2016) leverage the local geometry of the parameter space to approximate the Fisher information matrix, and have equipped SGLD with adaptive step-sizes for DNNs. Similarly, we parallelize the preconditioned SGLD (Li et al. 2016a) to develop *downpour pSGLD*, shown in the Section A of Supplementary Material (SM).

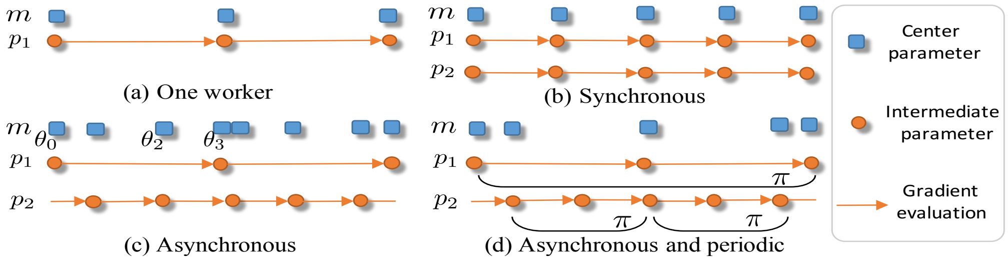
Elastic SGHMC Momentum has been shown capable of accelerating the learning trajectory along directions of low-curvature in the parameter space of DNNs, leading to faster convergence speeds (Sutskever et al. 2013). In Algorithm 2, we incorporate SGHMC (lines 5-10) into the elastic protocol (lines 12-17), termed *elastic SGHMC*.

Analysis and Practice

Finite-Time Estimation Errors

Samples obtained from MCMC methods are often used to estimate the posterior expectation $\bar{\phi} = \int \phi(\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}$ of a test function $\phi(\boldsymbol{\theta})$. In the context of DNNs, $\phi(\boldsymbol{\theta}) = p(\tilde{\mathbf{Y}}|\tilde{\mathbf{X}}, \boldsymbol{\theta})$, and $\bar{\phi}$ is the predictive distribution. For SG-MCMC, the model averaging approximation is implemented as $\hat{\phi} = \frac{1}{S_L} \sum_{l=1}^L \epsilon_l \phi(\boldsymbol{\theta}_l)$, where $S_L = \sum_{l,p} \epsilon_l^{(p)}$ and $\epsilon_l^{(p)} = \sum_{t=t^{(p)}-\pi}^{t^{(p)}} \epsilon_t$ is the accumulated step-size obtained from the p-th worker for l-th sample. For each worker, $S_L^{(p)} = \sum_l \epsilon_l^{(p)}$. The quality of the MCMC approximation to the true posterior expectation can be characterized by the bias and mean squared error (MSE), defined as: $|\mathbb{E}\hat{\phi}_L - \bar{\phi}|$ and $\mathbb{E}(\hat{\phi}_L - \bar{\phi})^2$, respectively. Furthermore, following (Chen et al. 2016), we also study the estimation variance defined as: $\mathbb{E}(\hat{\phi}_L - \mathbb{E}\hat{\phi}_L)^2$. We extend the work of (Chen, Ding, and Carin 2015; Teh, Thiéry, and Vollmer 2016; Vollmer, Zygalakis, and Teh 2015) to derive the bounds to account for the proposed communication protocols; See Section B of SM for all proofs.

The bias, variance and MSE of standard SG-MCMC ($P =$



(a) One worker

(b) Synchronous

(c) Asynchronous

(d) Asynchronous and periodic

Figure 1: Illustration of different SG-MCMC implementations. Each row represents a master, m, or a worker, p. An intermediate sample is collected at each arrow point, whose length represents the time required to evaluate the gradient. (a) Standard SG-MCMC corresponds to one worker. (b) Multiple workers accelerate gradient evaluation and update parameter synchronously. (c) Multiple workers update parameter asynchronously. (d) Asynchronous communication with period $\pi = 2$.

Table 1: Two synchronous communication schemes.

	Master	The p-th Worker
A	$\boldsymbol{\theta}_t = \frac{1}{P}\sum_{p=1}^{P}\boldsymbol{\theta}_t^{(p)}$	$\boldsymbol{\theta}_t^{(p)} = \boldsymbol{\theta}_t$
B	$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \frac{\alpha}{P}\sum_{p=1}^{P}(\boldsymbol{\theta}_t^{(p)} - \boldsymbol{\theta}_{t-1})$	$\boldsymbol{\theta}_t^{(p)} = \boldsymbol{\theta}_t^{(p)} + \alpha(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_t^{(p)})$

Algorithm 1 Downpour SGLD.

```
**Input**: $\epsilon_t, \pi \in \mathbb{N}$
**Output**: $\{\boldsymbol{\theta}_l\}_{l=1}^L$
**Initialize**: $t^{(p)} = 0, l = 0, \boldsymbol{\nu}^{(p)} = 0$
    $\tilde{\boldsymbol{\theta}} \sim \mathcal{N}(0, \mathbf{I}), \boldsymbol{\theta}^{(p)} = \tilde{\boldsymbol{\theta}}$
**while** maximum time not reached **do**
    % Estimate gradient from $\mathcal{D}_{\mathtt{M}}$
    $\tilde{\boldsymbol{f}}_t^{(p)} = \nabla\tilde{U}(\boldsymbol{\theta}_t^{(p)})$
    % Parameter update with SGLD
    $\boldsymbol{\xi}_t^{(p)} \sim \mathcal{N}(0, \mathbf{I})$
    $\boldsymbol{\theta}_{t+1}^{(p)} \leftarrow \boldsymbol{\theta}_t^{(p)} - \epsilon_t\tilde{\boldsymbol{f}}_t^{(p)} + \sqrt{2\epsilon_t}\boldsymbol{\xi}_t^{(p)}$
    $\boldsymbol{\nu}_{t+1}^{(p)} \leftarrow \boldsymbol{\nu}_t^{(p)} - \epsilon_t\tilde{\boldsymbol{f}}_t^{(p)} + \sqrt{2\epsilon_t}\boldsymbol{\xi}_t^{(p)}$
    $t^{(p)} \leftarrow t^{(p)} + 1$
    % $\pi-$Downpour Communication
    **if** $t^{(p)}$ divide $\pi$ **then**
        $\boldsymbol{\theta}_{l+1} \leftarrow \boldsymbol{\theta}_l + \boldsymbol{\nu}_{t+1}^{(p)}$
        $\boldsymbol{\theta}_{t+1}^{(p)} \leftarrow \boldsymbol{\theta}_{l+1}$    $\boldsymbol{\nu}_{t+1}^{(p)} \leftarrow 0$
        $l = l + 1$
    **end if**
**end while**
```

Algorithm 2 Elastic SGHMC.

```
**Input**: $\epsilon_t, \alpha, B, \pi \in \mathbb{N}$
**Output**: $\{\boldsymbol{\theta}_l\}_{l=1}^L$
**Initialize**: $t^{(p)} = 0, l = 0, \boldsymbol{q}^{(p)} = 0$
    $\tilde{\boldsymbol{\theta}} \sim \mathcal{N}(0, \mathbf{I}), \boldsymbol{\theta}^{(p)} = \tilde{\boldsymbol{\theta}}$
**while** maximum time not reached **do**
    % Estimate gradient from $\mathcal{D}_{\mathtt{M}}$
    $\tilde{\boldsymbol{f}}_t^{(p)} = \nabla\tilde{U}(\boldsymbol{\theta}_t^{(p)})$
    % Parameter update with SGHMC
    $\boldsymbol{\xi}_t^{(p)} \sim \mathcal{N}(0, \mathbf{I})$
    $\boldsymbol{q}_{t+1}^{(p)} \leftarrow \boldsymbol{q}_t^{(p)} - B\boldsymbol{q}_t^{(p)} - \epsilon_t\tilde{\boldsymbol{f}}_t^{(p)} + \sqrt{2B\epsilon_t}\boldsymbol{\xi}_t^{(p)}$
    $\boldsymbol{\theta}_{t+1}^{(p)} \leftarrow \boldsymbol{\theta}_t^{(p)} + \boldsymbol{q}_{t+1}^{(p)}$
    $t^{(p)} \leftarrow t^{(p)} + 1$
    % $(\pi, \alpha)-$Elastic Communication
    **if** $t^{(p)}$ divide $\pi$ **then**
        $\boldsymbol{\theta}_{l+1} \leftarrow \boldsymbol{\theta}_l + \alpha\big(\boldsymbol{\theta}_{t+1}^{(p)} - \boldsymbol{\theta}_l\big)$
        $\boldsymbol{\theta}_{t+1}^{(p)} \leftarrow \boldsymbol{\theta}_{t+1}^{(p)} + \alpha\big(\boldsymbol{\theta}_l - \boldsymbol{\theta}_{t+1}^{(p)}\big)$
        $l = l + 1$
    **end if**
**end while**
```

1 and $\pi = 1$) is bounded by (Chen, Ding, and Carin 2015):

$$\text{Bias:} \quad |\mathbb{E}\hat{\phi}_L - \bar{\phi}| \leq \mathcal{B}_0 = B_0\mathcal{E}_0, \tag{4}$$

$$\text{Variance: } \mathbb{E}(\hat{\phi}_L - \mathbb{E}\hat{\phi}_L)^2 \leq \mathcal{V}_0 = V_0\mathcal{E}_2\ . \tag{5}$$

$$\text{MSE:} \quad \mathbb{E}(\hat{\phi}_L - \bar{\phi})^2 \leq \mathcal{M}_0 = M_0(\mathcal{E}_1 + \mathcal{E}_2), \text{with} \tag{6}$$

$$\mathcal{E}_0 = \frac{1}{S_L} + \sum_l \frac{\epsilon_l^2}{S_L},\ \mathcal{E}_1 = \sum_l \frac{\epsilon_l^2}{S_L^2}\mathbb{E}\,\|\Delta V_l\|^2,\ \mathcal{E}_2 = \frac{1}{S_L} + \frac{(\sum_l \epsilon_l^2)^2}{S_L^2},$$

where $\Delta V_l = \tilde{\boldsymbol{f}}_l - \boldsymbol{f}_l$, and B_0, V_0 and M_0 are constant values independent of $\{\epsilon_l\}$ and L. The MSE bound includes two independent terms: $\mathcal{E}_1$ is the approximate error using stochastic gradients, and $\mathcal{E}_2$ is discretization error from the numerical integrator. When $S_L \to \infty$ and $\frac{\sum_l \epsilon_l^2}{S_L} \to 0$ with decreasing step-sizes, both terms diminish, leading the bias and MSE to asymptotically converging to zero.

Analysis of Downpour Protocal To obtain the bias bound for the downpour protocol, note that in the original proof (Chen, Ding, and Carin 2015) for standard SG-MCMC, there are extra terms $\|\mathbb{E}\Delta V_l\| = 0$, which are dropped in the bias bound. When considering the downpour protocol, however, these terms do not equal to zero, and thus cannot be dropped. Similarly, for the MSE bound, note that compared with standard SG-MCMC, there is additional error associated with the gradient approximation, *i.e.*, $\boldsymbol{\theta}_{l+1}$ is obtained with a stochastic gradient $\tilde{\boldsymbol{f}}_{l-\pi_l}$ evaluated on "old" parameters $\boldsymbol{\theta}_{l-\pi_l}$ for some integer $\pi_l \leq \pi(P-1)$, instead of $\boldsymbol{\theta}_l$. As a result, the term ΔV_l in (6) is replaced with $\Delta\tilde{V}_l \triangleq \tilde{\boldsymbol{f}}_{l-\pi_l} - \boldsymbol{f}_l = (\tilde{\boldsymbol{f}}_{l-\pi_l} - \tilde{\boldsymbol{f}}_l) + (\tilde{\boldsymbol{f}}_l - \boldsymbol{f}_l) = (\tilde{\boldsymbol{f}}_{l-\pi_l} - \tilde{\boldsymbol{f}}_l) + \Delta V_l$. Note that $\mathbb{E}\|\tilde{\boldsymbol{f}}_{l-\pi_l} - \tilde{\boldsymbol{f}}_l\|$ can be bounded under the Lipchitz assumption (Lian et al. 2015; Abdulle, Vilmart, and Zygalakis 2015). We summarize the bias, variance and MSE bounds for the downpour protocol, based on single-worker SG-MCMC:

$$\text{Bias: } \mathcal{B}_1 = B_1(\mathcal{E}_0 + \sqrt{\mathcal{E}_3}) \tag{7}$$

$$\text{Variance: } \mathcal{V}_1 = V_1\mathcal{E}_2 \tag{8}$$

$$\text{MSE: } \mathcal{M}_1 = M_1(\mathcal{E}_1 + \mathcal{E}_2 + \mathcal{E}_3), \text{ with} \tag{9}$$

$$\mathcal{E}_3 = \pi^2(P-1)^2\frac{(\sum_l \epsilon_l^2)^2}{S_L^2},$$

where B_1, V_1 and M_1 are constant values. The extra error term $\mathcal{E}_3$ is due to the approximation error introduced by using multiple workers and communication delays. Larger π and P lead to larger errors in $\mathcal{E}_3$. However, when P workers are employed, we can also get a P-times smaller $\frac{1}{S_L}$ in $\mathcal{E}_0$ for the bias and $\mathcal{E}_2$ for the MSE per wallclock time interval, possibly leading to lower bias and MSE bounds. If the same step-size conditions hold in this case, the bias and MSE will still asymptotically approach zero. Furthermore, we notice that the estimation variance is independent of P and π, leading to a linear speedup with respect to P.

Analysis of Elastic Protocal Concerning the MSE bound for the elastic protocol, we note that $\alpha = 1$ corresponds to the case for which the master essentially plays the role of only exchanging intermediate parameters among workers every π steps, an algorithm proposed in (Ahn, Shahbaba, and Welling 2014) when data are shared across workers. It is equal to running P independent chains, and collecting samples every π steps instead of all samples, bounded as:

$$\text{Bias: } \mathcal{B}_2 = B_2\mathcal{E}_0 \tag{10}$$

$$\text{Variance: } \mathcal{V}_2 = V_2\mathcal{E}_2' \tag{11}$$

$$\text{MSE: } \mathcal{M}_2 = M_2(\mathcal{E}_1' + \mathcal{E}_2'), \text{ with} \tag{12}$$

$$\mathcal{E}_1' = \frac{\sum_p (S_L^{(p)})^2}{S_L^2}\mathcal{E}_1^{(p)},\ \mathcal{E}_2' = \frac{1}{S_L} + \frac{\sum_{i,j}(\sum_l((\epsilon_l^{(i)})^2 + (\epsilon_l^{(j)})^2))^2}{S_L^2},$$

where B_2, V_2 and M_2 are constant values., and $\mathcal{E}_1^{(p)}$ is $\mathcal{E}_1$ for a single worker.

Role of the Communication Period

We show that periodic protocols relate to thinning samples on each worker; the thinned samples are then maintained on the master. We have the following observations for one worker case ($P = 1$).

- The samples obtained from a π-downpour SG-MCMC algorithm are equivalent to the samples obtained from a standard SG-MCMC algorithm with thinning interval π.
- When $\alpha = 1$, the samples obtained from a (π, α)-elastic SG-MCMC algorithm are equivalent to the samples obtained from a standard SG-MCMC algorithm with thinning interval π.

This can be shown by plugging in the special cases into the procedures in Algorithm 1 and 2, as the samples within the period π are not sent to the master. By "thinning", the total number of samples on the master are reduced, while reducing the communication overhead between the master and workers. Moreover, these thinned samples have a lower autocorrelation time and maintain a similar effective sample size (Li et al. 2016a). When $P > 1$, more samples are obtained per unit time, while containing information about the parameter space explored by multiple workers.

Related Work

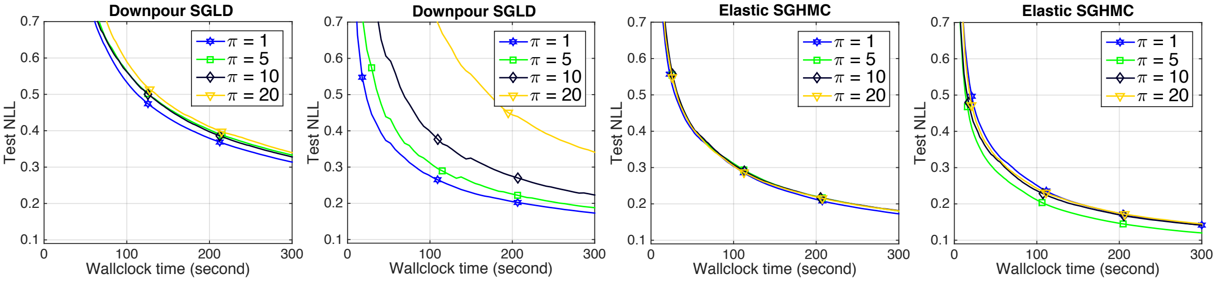
Note that one can parallelize any of the three components of SG-MCMC to accelerate learning: *data*, *model parameters* and *gradients*. Data parallelism of SG-MCMC was first implemented in (Ahn, Shahbaba, and Welling 2014), where each worker iteratively is run on local pool of data for an amount of time, followed by synchronizing with other workers. In the recent embarrassingly parallel SGLD (Yang, Chen, and Zhu 2016), a master is introduced to aggregate the sub-posteriors on all workers into a global one. Significant speedup has been shown on latent Dirichlet allocation with the data-parallelism scheme. Simultaneous parallelism of data and parameters have also been developed in (Ahn et al. 2015; Şimşekli et al. 2015). They leverage the conditional independence in matrix factorization to group data and parameters, where each chain is run on one group. We emphasize that our framework for parallel gradient evaluation under communication constraints is distinct from the above related works; in fact, it can be incorporated into their works to improve performance. While (Chen et al. 2016) developed the theory for the staleness of stochastic gradients in SG-MCMC recently, we focus on studying more efficient algorithms to reduce the communication cost. Recently, (Şimşekli et al. 2018) reformulate the original optimization problem within the sampling framework for distributed training. They further introduced additional hyperparameter "inverse temperature" to decouple the gradient term and noise term. We can borrow similar concept to balance sampling and optimization.

In reinforcement learning, our asynchronus SG-MCMC policy learning method also has interesting connections to existing algorithms. Compared with Asynchronous Advantage Actor-Critic (Mnih et al. 2016), we sample the weights rather than optimize them. It corresponds to choosing a different current policy, which naturally favours exploration. Recent work (Fortunato et al. 2018; Plappert et al. 2018) showed that adding noise to weights/units of DNNs can empirically lead to performance gain for many reinforcement learning tasks. However, their theoretical justification is lacked, and our work fill the gap.

Experimental Results

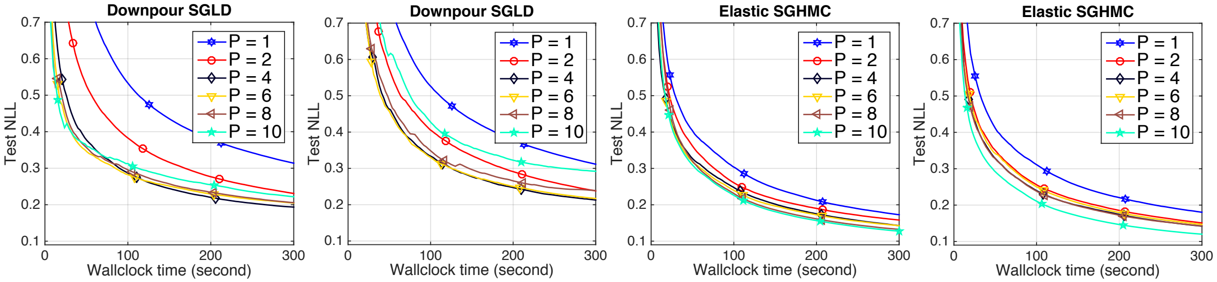
Feedforward Neural Networks

We first study FNN on the standard MNIST dataset, consisting of 28×28 images (thus of 784-dimensional input vectors) from 10 different classes (0 to 9), with 60000 training and 10000 test samples. Following (Blundell et al. 2015; Li et al. 2016a), we use rectified linear units (ReLUs) (Glorot, Bordes, and Bengio 2011) as the activation function, and



(a) $P = 1$ (b) $P = 4$ (c) $P = 1$ (d) $P = 10$

Figure 2: The effects of π in downpour and elastic protocols.



(a) $\pi = 1$ (b) $\pi = 5$ (c) $\pi = 1$ (d) $\pi = 5$

Figure 3: The effects of P in downpour and elastic protocols.

a two-layer model, 784-X-X-10, is employed, where X is the number of hidden units for each layer. Sizes (X-X) 400-400, 800-800 and 1200-1200 are considered.

Effects of P and π We first investigate the role of the number of workers and communication period in accelerating SG-MCMC, with the 400-400 network, shown in Fig. 2 and Fig. 3, respectively, where test negative log-likelihood (NLL) *vs.* wallclock time are shown. Only the first 300 seconds are displayed for clarity. We first employ 1, 4 or 10 workers, and vary the communication period as $\pi = \{1, 5, 10, 20\}$. As seen in Fig. 2(a)(c), single-worker SG-MCMC can maintain good test NLL, verifying our observation about the equivalence of the communication protocols to thinning. When more workers ($P = 4$) are used, downpour fails to tolerate delays (Fig. 2(b)), because directly incorporating the update from “long-runs” of many workers may lead to conflicts; while the elastic protocol is robust (Fig. 2(d)). We then study the impact of the number of workers (Fig. 3), by varying $P = \{1, 2, 4, 6, 8, 10\}$, in the setup of instantaneous messaging ($\pi = 1$) and periodic messaging ($\pi = 5$). While more workers generally provide higher speedup, there is a limit when too many workers are involved: downpour SGLD would slow down learning, especially when communication delay exist; Elastic SGHMC would yield less speedup gain per worker, but will be robust to delays, thus potentially allowing for a larger number of workers. The communication issues can be more significant in larger systems. However, we observed little improvement when $P > 10$ in our hardware setup. This could be the effect of scheduling overhead.

In Fig. 4(a), we compare our methods with their distributed optimization counterparts using the same communication protocols: downpour RMSprop and elastic AMSGD (Zhang, Choromanska, and LeCun 2015). SG-MCMC methods converges slower than their optimization counterparts, this is because the injecting noise encourage SG-MCMC’s to explore the parameter space during learning. However, the optimization methods tend to overfit as evidenced by the NLL. Our approach alleviates overfitting by model averaging, where more stable learning curves are obtained. Figure 4(b) shows the NLL for FNN (various network sizes) using elastic SGHMC, which exhibits consistent speedups.

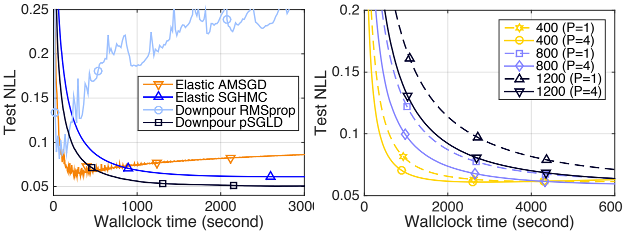
Table 2 shows the wallclock time needed to reach a stable NLL, and best test classification errors of different algorithms. SG-MCMC algorithms with downpour and elastic protocols ($P > 1$) can achieve the same levels of errors as their simpler versions ($P = 1$), but with significantly less time. The proposed downpour pSGLD also outperform other techniques developed to prevent overfitting (dropout) (Srivastava et al. 2014), or capture weight uncertainty (BPB, Gaussian and scale mixtures) (Blundell et al. 2015), and representative stochastic optimization methods: SGD, RMSprop (Tieleman and Hinton 2012) and RMSspectral (Carlson et al. 2015).

Convolutional Neural Networks

The CNN is tested on SVHN, which is a large dataset consisting of color images of size 32 × 32. The task is to recognize center digits in natural scene images. A standard 2 layers CNN is used. Figure 5(a) shows learning curves for test NLL using downpour pSGLD and elastic SGHMC. Multiple workers show consistent speedup in both NLL. To further study the effectiveness of our method in leveraging the benefits of the Bayesian approach, we use 20 model samples from downpour

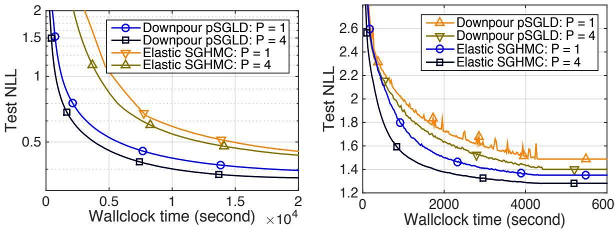
Table 2: Results of FNN on MNIST. For each parallel algorithm, the first row shows classification error, while the second row shows wallclock time in seconds. Results marked with [⋄] and [⋆] are from (Blundell et al. 2015) and (Li et al. 2016a) , respectively.

Method	Test Error (Wallclock Time)		
	400-400	800-800	1200-1200
Downpour pSGLD	1.40%	1.31%	1.25%
$(P = 1)$	3188	5719	7264
Downpour pSGLD	1.34%	1.32%	1.30%
$(P = 4, \pi = 3)$	**2664**	**4994**	**6385**
Elastic SGHMC	1.76%	1.77%	1.80%
$(P = 1)$	3393	5918	6776
Elastic SGHMC	1.79%	1.71%	1.77%
$(P = 4, \pi = 10)$	**2256**	**4315**	**5552**
BPB, Gaussian⋄	1.82%	1.99%	2.04%
BPB, Scale mixture⋄	1.32%	1.34%	1.32%
SGD, dropout⋄	1.51%	1.33%	1.36%
RMSprop⋆	1.59%	1.43%	1.39%
RMSspectral⋆	1.65%	1.56%	1.46%
SGD⋆	1.72%	1.47%	1.47%



(a) NLL (b) Network Size X

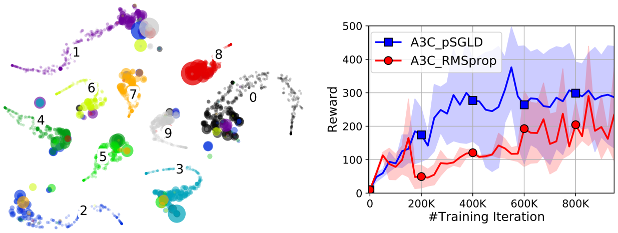
Figure 4: Comparison of parallel methods on FNN.



(a) NLL on CNN (b) NLL on 2-layer GRU

Figure 5: Learning curves on CNN and RNN.

pSGLD algorithm to estimate predictive means and standard deviations. In Fig. 6 (a) we embed the predictive means on the testing data to a 2D-space with t-SNE (Van der M. and Hinton 2008), with each point as a data instance. Color indicates true label of the point, whose size indicates the standard deviation on the predicted label. Interestingly, large points (high uncertainty) often lie in the wrong manifold, or near the boundary of different classes. One can leverage the uncertainty information to improve decision-making by manual judgement, when uncertainty is high and in cases where distributed optimization (Dean and others 2012; Zhang, Choromanska, and LeCun 2015) or distillation methods (Korattikara et al. 2015) cannot be directly applied.



(a) (b)

Figure 6: (a) t-SNE of SVHN; (b) A3C on cartpole..

Recurrent Neural Networks

We test the RNN with the task of character-level language modeling on the *War and Peace* (WP) novel dataset (Karpathy, Johnson, and Fei-Fei 2016). The training/testing sets contain 26000/3200 characters, and the vocabulary size is 87. We consider a 1 or 2-hidden-layer RNN of dimension 128, with Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) or Gated Recurrent Unit (GRU) (Cho and et al. 2014).

Figure 5(c) shows the learning curves on a 2-layer GRU model using downpour pSGLD and elastic SGHMC. The final test NLL in various scenarios are in SM. Interestingly, we observe a lower NLL when using 4 workers compared to 1 worker, *i.e.*, standard SG-MCMC. This is perhaps due to the existence of many local optima in large deep models, multiple workers allow more exploration of the space, and thus lead to improved performance.

Asynchronus Advantage Actor-Critic (A3C)

A3C (Mnih et al. 2016) uses asynchronous gradient descent for policy optimization of DNN agents. There is no explicit exploratory action selection scheme, and the chosen action is always from the current policy. Therefore, we apply SG-MCMC for direct exploration in the policy space.

Specifically, we implemented downpour pSGLD algorithm for A3C. For fair comparison, RMSprop optimizer is considered as a competitor, entropy regularisation is off, and $P = 5$ workers are used for both methods. Each algorithm runs 5 times on the `Cartpole-v1` environment, and we plot the average reward in Figure 6 (b). A3C with pSGLD converges faster and achieves significant higher reward than the RMSprop alternative.

Conclusion

We have developed two periodic communication protocols to coordinate the information exchange in asynchronous parallel SG-MCMC. While the downpour protocol is theoretically supported by our finite-time convergence theory, the elastic protocol has shown empirically to be very communication efficient. Experiments on various DNNs demonstrate that both protocols can significantly accelerate conventional SG-MCMC, to achieve the same or even better levels of test performance. When applied to A3C, it naturally endows the exploration ability, and shows higher improvement.

References

Abdulle, A.; Vilmart, G.; and Zygalakis, K. 2015. Long time accuracy of Lie-Trotter splitting methods for Langevin dynamics. *SIAM Journal on Numerical Analysis*.

Ahn, S.; Korattikara, A.; Liu, N.; Rajan, S.; and Welling, M. 2015. Large-scale distributed Bayesian matrix factorization using stochastic gradient MCMC. In *ACM SIGKDD*.

Ahn, S.; Shahbaba, B.; and Welling, M. 2014. Distributed stochastic gradient MCMC. In *ICML*.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural networks. In *ICML*.

Bui, T. D.; Hernández-Lobato, D.; Li, Y.; Hernández-Lobato, J.; and Turner, R. E. 2016. Deep Gaussian processes for regression using approximate expectation propagation. In *ICML*.

Carlson, D.; Collins, E.; Hsieh, Y. P.; Carin, L.; and Cevher, V. 2015. Preconditioned spectral descent for deep learning. In *NIPS*.

Chen, C.; Ding, N.; Li, C.; Zhang, Y.; and Carin, L. 2016. Stochastic gradient MCMC with stale gradients. In *NIPS*.

Chen, C.; Ding, N.; and Carin, L. 2015. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *NIPS*.

Chen, T.; Fox, E. B.; and Guestrin, C. 2014. Stochastic gradient Hamiltonian Monte Carlo. In *ICML*.

Cho, K., and et al. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Cong, Y.; Chen, B.; Liu, H.; and Zhou, M. 2017. Deep latent dirichlet allocation with topic-layer-adaptive stochastic gradient riemannian mcmc. *ICML*.

Dean, J., et al. 2012. Large scale distributed deep networks. In *NIPS*.

Ding, N.; Fang, Y.; Babbush, R.; Chen, C.; Skeel, R. D.; and Neven, H. 2014. Bayesian sampling using stochastic gradient thermostats. In *NIPS*.

Durmus, A.; Simsekli, U.; Moulines, E.; Badeau, R.; and Richard, G. 2016. Stochastic gradient richardson-romberg markov chain monte carlo. In *NIPS*.

Fortunato, M.; Azar, M. G.; Piot, B.; Menick, J.; Osband, I.; Graves, A.; Mnih, V.; Munos, R.; Hassabis, D.; Pietquin, O.; et al. 2018. Noisy networks for exploration. *ICLR*.

Fu, T., and Zhang, Z. 2017. Cpsg-mcmc: Clustering-based preprocessing method for stochastic gradient mcmc. In *Artificial Intelligence and Statistics*.

Gal, Y., and Ghahramani, Z. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *ICML*.

Gan, Z.; Li, C.; Chen, C.; Pu, Y.; Su, Q.; and Carin, L. 2017. Scalable bayesian learning of recurrent neural networks for language modeling. *ACL*.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *AISTATS*.

Hernández-Lobato, J. M., and Adams, R. P. 2015. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *ICML*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. In *Neural computation*.

Karpathy, A.; Johnson, J.; and Fei-Fei, L. 2016. Visualizing and understanding recurrent networks. *ICLR, workshop*.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. *ICLR*.

Kingma, D. P.; Salimans, T.; and Welling, M. 2015. Variational Dropout and the local reparameterization trick. *NIPS*.

Korattikara, A.; Rathod, V.; Murphy, K.; and Welling, M. 2015. Bayesian dark knowledge. In *NIPS*.

Li, C.; Chen, C.; Carlson, D.; and Carin, L. 2016a. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *AAAI*.

Li, C.; Stevens, A.; Chen, C.; Pu, Y.; Gan, Z.; and Carin, L. 2016b. Learning weight uncertainty with stochastic gradient mcmc for shape classification. In *CVPR*.

Lian, X.; Huang, Y.; Li, Y.; and Lix, J. 2015. Asynchronous parallel stochastic gradient for nonconvex optimization. In *NIPS*.

Liu, Q., and Wang, D. 2016. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *NIPS*.

Liu, C.; Zhu, J.; and Song, Y. 2016. Stochastic gradient geodesic mcmc methods. In *NIPS*.

Ma, Y. A.; Chen, T.; and Fox, E. B. 2015. A complete recipe for stochastic gradient MCMC. In *NIPS*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *ICML*.

Neal, R. M. 1995. *Bayesian learning for neural networks*. PhD thesis, University of Toronto.

Plappert, M.; Houthooft, R.; Dhariwal, P.; Sidor, S.; Chen, R. Y.; Chen, X.; Asfour, T.; Abbeel, P.; and Andrychowicz, M. 2018. Parameter space noise for exploration. *ICLR*.

Şimşekli, U.; Koptagel, H.; Güldaş, H.; Cemgil, T.; Öztoprak, F.; and Birbil, Ş. İ. 2015. Parallel stochastic gradient Markov Chain Monte Carlo for matrix factorisation models. *arXiv:1506.01418*.

Simsekli, U.; Badeau, R.; Richard, G.; and Cemgil, T. 2016. Stochastic Quasi-Newton Langevin Monte Carlo. In *ICML*.

Şimşekli, U.; Yıldız, Ç.; Nguyen, T. H.; Richard, G.; and Cemgil, A. T. 2018. Asynchronous stochastic quasi-newton mcmc for non-convex optimization. *ICML*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*.

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. E. 2013. On the importance of initialization and momentum in deep learning. In *ICML*.

Teh, Y. W.; Thiéry, A. H.; and Vollmer, S. J. 2016. Consistency and fluctuations for stochastic gradient Langevin dynamics. *JMLR*.

Tieleman, T., and Hinton, G. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural Networks for Machine Learning*.

Van der M., L., and Hinton, G. E. 2008. Visualizing data using t-SNE. *JMLR*.

Vollmer, S. J.; Zygalakis, K. C.; and Teh, Y. W. 2015. (Non-)asymptotic properties of stochastic gradient Langevin dynamics. Technical report.

Welling, M., and Teh, Y. W. 2011. Bayesian learning via stochastic gradient Langevin dynamics. In *ICML*.

Yang, Y.; Chen, J.; and Zhu, J. 2016. Distributing the stochastic gradient sampler for large-scale LDA. In *SIGKDD*.

Zhang, S.; Choromanska, A.; and LeCun, Y. 2015. Deep learning with elastic averaging SGD. In *NIPS*.