entropy

MDPI

Article

A Scalable Bayesian Sampling Method Based on Stochastic Gradient Descent Isotropization

Giulio Franzese *, Dimitrios Milios, Maurizio Filippone and Pietro Michiardi

Data Science Department, Eurecom, 06410 Biot, France; dimitrios.milios@eurecom.fr (D.M.); maurizio.filippone@eurecom.fr (M.F.); pietro.michiardi@eurecom.fr (P.M.)
* Correspondence: giulio.franzese@eurecom.fr

Abstract: Stochastic gradient SG-based algorithms for Markov chain Monte Carlo sampling (SGMCMC) tackle large-scale Bayesian modeling problems by operating on mini-batches and injecting noise on SGsteps. The sampling properties of these algorithms are determined by user choices, such as the covariance of the injected noise and the learning rate, and by problem-specific factors, such as assumptions on the loss landscape and the covariance of SG noise. However, current SGMCMC algorithms applied to popular complex models such as Deep Nets cannot simultaneously satisfy the assumptions on loss landscapes and on the behavior of the covariance of the SG noise, while operating with the practical requirement of non-vanishing learning rates. In this work we propose a novel practical method, which makes the SG noise isotropic, using a fixed learning rate that we determine analytically. Extensive experimental validations indicate that our proposal is competitive with the state of the art on SGMCMC.

Keywords: Bayesian sampling; stochastic gradients; Monte Carlo integration

Citation: Franzese, G.; Milios, D.; Filippone, M.; Michiardi, P. A Scalable Bayesian Sampling Method Based on Stochastic Gradient Descent Isotropization. *Entropy* **2021**, *23*, 1426. https://doi.org/10.3390/e23111426

Academic Editor: Pierre Alquier

Received: 21 September 2021
Accepted: 26 October 2021
Published: 28 October 2021

1. Introduction

Stochastic gradient (SG) methods have been extensively studied as a means for MCMC-based Bayesian posterior sampling algorithms to scale to large data regimes. Variants of SG-MCMC algorithms have been studied through the lens of first [1–3] or second-order [4,5] Langevin Dynamics, which are mathematically convenient continuous-time processes that correspond to discrete-time gradient methods with and without momentum, respectively. The common traits underlying many methods from the literature can be summarized as follows: they address large data requirements using SG and mini-batching, they inject Gaussian noise throughout the algorithm execution, and they avoid the expensive Metropolis-Hasting accept/reject tests that use the whole data [1,2,4].

Despite mathematical elegance and some promising results restricted to simple models, current approaches fall short in dealing with the complexity of the loss landscape typical of popular modern machine learning models, e.g., neural networks [6,7], for which stochastic optimization poses some serious challenges [8,9].

In general, SG-MCMC algorithms inject random noise to SG descent algorithms: the covariance of such noise and the learning rate, or step-size in the stochastic differential equation simulation community, are tightly related to the assumptions on the loss landscape, which together with the SG noise, determine the sampling properties of these methods [5]. However, current SG-MCMC algorithms applied to popular complex models such as Deep Nets, cannot simultaneously satisfy the assumptions on posterior distribution geometry and on the behavior of the covariance of the SG noise, while operating with the practical requirement of non-vanishing learning rates. In this paper, in accordance with most of the Neural Network related literature, we refer to the posterior distribution geometry as loss landscape. Some recent work [10], instead, argue for fixed step sizes, but settle for variational approximations of quadratic losses. Although we are not the first to highlight these issues, including the lack of a unified notation [5], we believe that studying the

role of noise in SG-MCMC algorithms has not received enough attention, and a deeper understanding is truly desirable, as it can clarify how various methods compare. Most importantly, this endeavor can suggest novel and more practical algorithms relying on fewer parameters and less restrictive assumptions.

In this work we chose a mathematical notation that emphasizes the role of noise covariances and learning rate on the behavior of SG-MCMC algorithms (Section 2). As a result, the equivalence between learning rate annealing and extremely large injected noise covariance becomes apparent, and this allows us to propose a novel practical SG-MCMC algorithm (Section 3). We derive our proposal, by first analyzing the case where we inject the smallest complementary noise such that its combined effects with the SG noise result in an isotropic noise. Thanks to this isotropic property of the noise, it is possible to deal with intricate loss surfaces typical of deep models, and produce samples from the true posterior without learning rate annealing. This, however, comes at the expense of cubic complexity matrix operations. We address such issues through a practical variant of our scheme, which employs well-known approximations to the SG noise covariance (see, e.g., [11]). The result is an algorithm that produces approximate posterior samples with a fixed, theoretically derived, learning rate. Please note that in generic Bayesian deep learning setting, none of the existing implementations of SG-MCMC methods converge to the true posterior without learning rate annealing. In contrast, our method automatically determines an appropriate learning rate through a simple estimation procedure. Furthermore, our approach can be readily applied to pre-trained models: after a "warmup" phase to compute SG noise estimates, it can efficiently perform Bayesian posterior sampling.

We evaluate SG-MCMC algorithms (Section 4) through an extensive experimental campaign, where we compare our approach to several alternatives, including Monte Carlo Dropout (MCD) [12] and Stochastic Weighted Averaging Gaussians (SWAG, [9]), which have been successfully applied to the Bayesian deep learning setting. Our results indicate that our approach offers performance that are competitive to the state of the art, according to metrics that aim at assessing the predictive accuracy and uncertainty.

2. Preliminaries and Related Work

Consider a dataset of $m-$dimensional observations $\mathcal{D} = \{\boldsymbol{U}_i\}_{i=1}^N$. Given prior $p(\boldsymbol{\theta})$ for a d-dimensional set of parameters, and a likelihood model $p(\mathcal{D}|\boldsymbol{\theta})$, the posterior is obtained by means of Bayes theorem as follows:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\mathcal{D})} \tag{1}$$

where $p(\mathcal{D})$ is also known as the model evidence, defined as the integral $p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\theta})\, p(\boldsymbol{\theta}) d\boldsymbol{\theta}$. Except when the prior and the likelihood function are conjugate, Equation (1) is analytically intractable [13]. However, the joint likelihood term in the numerator is typically not hard to compute; this is a key element of many MCMC algorithms, since the normalization constant $p(\mathcal{D})$ does not affect the shape of the distribution in any way other than scaling. The posterior distribution is necessary to obtain predictive distributions for new test observations $\boldsymbol{U}_*$, as:

$$p(\boldsymbol{U}_*|\mathcal{D}) = \int p(\boldsymbol{U}_*|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} \tag{2}$$

We focus in particular on Monte Carlo methods to obtain an estimate of this predictive distribution, by averaging over N_{MC} samples obtained from the posterior over $\boldsymbol{\theta}$, i.e., $\boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta}|\mathcal{D})$

$$p(\boldsymbol{U}_*|\mathcal{D}) \approx \frac{1}{N_{\text{MC}}} \sum_{i=1}^{N_{\text{MC}}} p(\boldsymbol{U}_*|\boldsymbol{\theta}^{(i)}) \tag{3}$$

We develop our work by working with an unnormalized version of the logarithm of the posterior density, by expressing the negative logarithm of the joint distribution of the dataset $\mathcal{D}$ and parameters $\boldsymbol{\theta}$ as:

$$-f(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\boldsymbol{U}_i|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}). \tag{4}$$

For computational efficiency, we use a minibatch stochastic gradient $\boldsymbol{g}(\boldsymbol{\theta})$, which guarantees that the estimated gradient is an unbiased estimate of the true gradient $\nabla f(\boldsymbol{\theta})$, and we assume that the randomness due to the minibatch introduces a Gaussian noise:

$$\boldsymbol{g}(\boldsymbol{\theta}) \sim N(\nabla f(\boldsymbol{\theta}), 2\boldsymbol{B}(\boldsymbol{\theta})), \tag{5}$$

where the matrix $\boldsymbol{B}(\boldsymbol{\theta})$ denotes the SG noise covariance, which depends on the parametric model, the data distribution and the minibatch size.

A survey of algorithms to sample from the posterior using SG methods can be found in Ma et al. [5]. In Appendix A we report some well-known facts which are relevant for the derivations in our paper. As shown in the literature [10,14], there are structural similarities between SG-MCMC algorithms and stochastic optimization methods, and both can be used to draw samples from posterior distributions. Notice that the original goal of stochastic optimization is to find the minimum of a given cost function, and the stochasticity is introduced by sub-sampling the dataset to scale. SG-MCMC methods instead aim at sampling from a given distribution, i.e., collecting multiple values, and the stochasticity is necessary explore the whole landscape. In what follows, we use a unified notation to compare many existing algorithms in light of the role played by their noise components.

It is well-known [15–17] that stochastic gradient descent (SGD), with and without momentum, can be studied through the following stochastic differential equation (SDE), when the learning rate η is small enough (In this work we do not consider discretization errors. The reader can refer to classical SDE texts such as [18] to investigate the topic in greater depth.):

$$d\boldsymbol{z}_t = \boldsymbol{s}(\boldsymbol{z}_t)dt + \sqrt{2\eta \boldsymbol{D}(\boldsymbol{z}_t)}d\boldsymbol{W}_t. \tag{6}$$

where $\boldsymbol{s}$ is usually referred to as driving force and $\boldsymbol{D}$ as diffusion matrix We use a generic form of the SDE, with variable $\boldsymbol{z}$ instead of $\boldsymbol{\theta}$, which accommodates SGD variants, with and without momentum. By doing this, we will be able to easily cast the expression for the two cases in what follows (The operator $\nabla^\top$ applied to matrix $\boldsymbol{D}(\boldsymbol{z})$ produces a row vector whose elements are the divergences of the $\boldsymbol{D}(\boldsymbol{z})$ columns. Our notation is aligned with Chen et al. [4]).

Definition 1. *A distribution $\rho(\boldsymbol{z}) \propto \exp(-\phi(\boldsymbol{z}))$ is said to be a **stationary** distribution for the* SDE *of the form* (6)*, if and only if it satisfies the following Fokker-Planck equation* (FPE)*:*

$$0 = \mathrm{Tr}\Big\{\nabla\Big[-\boldsymbol{s}(\boldsymbol{z})^\top \rho(\boldsymbol{z}) + \nabla^\top(\boldsymbol{D}(\boldsymbol{z})\rho(\boldsymbol{z}))\Big]\Big\}. \tag{7}$$

Please note that in general, the stationary distribution does not converge to the desired posterior distribution, i.e., $\phi(\boldsymbol{z}) \neq f(\boldsymbol{z})$, as shown by Chaudhari and Soatto [8]. Additionally, given an initial condition for $\boldsymbol{z}_t$, its distribution is going to converge to $\rho(\boldsymbol{z})$ only for $t \to \infty$. In practice, we observe the SDE dynamics for a finite amount of time: then, we declare that the process is approximately in the stationary regime once the potential has reached low and stable values.

Next, we briefly overview known approaches to Bayesian posterior sampling, and interpret them as variants of an SGD process, using the FPE formalism.

2.1. Gradient Methods without Momentum

The generalized updated rule of SGD, described as a discrete-time stochastic process, writes as:

$$\delta\boldsymbol{\theta}_n = -\eta \boldsymbol{P}(\boldsymbol{\theta}_{n-1})(\boldsymbol{g}(\boldsymbol{\theta}_{n-1}) + \boldsymbol{w}_n), \tag{8}$$

where $\boldsymbol{P}(\boldsymbol{\theta}_{n-1})$ is a user-defined preconditioning matrix, and $\boldsymbol{w}_n$ is a noise term, distributed as $\boldsymbol{w}_n \sim N(\mathbf{0}, 2\boldsymbol{C}(\boldsymbol{\theta}_n))$, with a user-defined covariance matrix $\boldsymbol{C}(\boldsymbol{\theta}_n)$. Then, the corresponding continuous-time SDE is [15]:

$$d\boldsymbol{\theta}_t = -\boldsymbol{P}(\boldsymbol{\theta}_t)\nabla f(\boldsymbol{\theta}_t)dt + \sqrt{2\eta \boldsymbol{P}(\boldsymbol{\theta}_t)^2 \boldsymbol{\Sigma}(\boldsymbol{\theta}_t)}d\boldsymbol{W}_t. \tag{9}$$

In this paper we use the symbol n to indicate discrete time, while t for continuous time. We denote by $\boldsymbol{C}(\boldsymbol{\theta})$ the covariance of the *injected noise* and $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ the *composite noise* covariance. Please note that $\boldsymbol{\Sigma}(\boldsymbol{\theta}_t) = \boldsymbol{B}(\boldsymbol{\theta}_t) + \boldsymbol{C}(\boldsymbol{\theta}_t)$ combines the SG and the injected noise. Notice that our choice of notation is different from the standard one, in which the starting discrete-time process is in the form $\delta\boldsymbol{\theta}_n = -\eta \boldsymbol{P}(\boldsymbol{\theta}_{n-1})(\boldsymbol{g}(\boldsymbol{\theta}_{n-1})) + \boldsymbol{w}_n$. By directly grouping the injected noise with the stochastic gradient we can better appreciate the relationship between annealing the learning rate and extremely large injected noise. Moreover, as will be explained in Section 3, this allows derivation of a new sampling algorithm.

We define the stationary distribution of the SDE in Equation (9) as $\rho(\boldsymbol{\theta}) \propto \exp(-\phi(\boldsymbol{\theta}))$. Please note that when $\boldsymbol{C} = \mathbf{0}$, the potential $\phi(\boldsymbol{\theta})$ differs from the desired posterior $f(\boldsymbol{\theta})$ [8]. The following theorem, which is an adaptation of known results in light of our formalism, states the conditions for which the *noisy* SGD converges to the true posterior distribution (proof in Appendix A).

Theorem 1. *Consider dynamics of the form* (9) *and define the stationary distribution* $\rho(\boldsymbol{\theta}) \propto \exp(-\phi(\boldsymbol{\theta}))$. *If*

$$\nabla^\top \left(\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\right) = \mathbf{0}^\top \quad \textit{and} \quad \eta \boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}, \tag{10}$$

then $\phi(\boldsymbol{\theta}) = f(\boldsymbol{\theta})$.

Stochastic Gradient Langevin Dynamics (SGLD) [1] is a simple approach to satisfy Equation (10); it uses no preconditioning, $\boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{I}$, and sets the injected noise covariance to $\boldsymbol{C}(\boldsymbol{\theta}) = \eta^{-1}\boldsymbol{I}$. In the limit for $\eta \to 0$, it holds that $\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \boldsymbol{B}(\boldsymbol{\theta}) + \eta^{-1}\boldsymbol{I} \simeq \eta^{-1}\boldsymbol{I}$. Then, $\nabla^\top \left(\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\right) = \eta \nabla^\top \boldsymbol{I} = \mathbf{0}^\top$, and $\eta \boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}$. Although SGLD succeeds in (asymptotically) generating samples from the true posterior, its mixing rate is unnecessarily slow, due to the extremely small learning rate [2].

An extension to SGLD is Stochastic Gradient Fisher Scoring (SGFS) [2], which can be tuned to switch between sampling from an approximate posterior, using a non-vanishing learning rate, and the true posterior, by annealing the learning rate to zero. SGFS uses preconditioning, $\boldsymbol{P}(\boldsymbol{\theta}) \propto \boldsymbol{B}(\boldsymbol{\theta})^{-1}$. In practice, however, $\boldsymbol{B}(\boldsymbol{\theta})$ is ill conditioned for complex models such as deep neural networks. Then, many of its eigenvalues are almost zero [8], and computing $\boldsymbol{B}(\boldsymbol{\theta})^{-1}$ is problematic. An in-depth analysis of SGFS reveals that conditions (10) would be met with a non-vanishing learning rate only if, at convergence, $\nabla^\top(\boldsymbol{B}(\boldsymbol{\theta})^{-1}) = \mathbf{0}^\top$, which would be trivially true if $\boldsymbol{B}(\boldsymbol{\theta})$ was constant. However, recent work [6,7] suggest that this condition is difficult to justify for deep neural networks.

The Stochastic Gradient Riemannian Langevin Dynamics (SGRLD) algorithm [3] extends SGFS to the setting in which $\nabla^\top(\boldsymbol{B}(\boldsymbol{\theta})^{-1}) \neq \mathbf{0}^\top$. The process dynamic is adjusted by adding the term $\nabla^\top(\boldsymbol{B}(\boldsymbol{\theta})^{-1})$. However, the term $\nabla^\top(\boldsymbol{B}(\boldsymbol{\theta})^{-1})$ has not a clear estimation procedure, restricting SGRLD to cases where it can be computed analytically.

The work by [10] investigates constant-rate SGD (with no injected noise), and determines analytically the learning rate and preconditioning that minimize the Kullback–Leibler (KL) divergence between an approximation and the true posterior. Moreover, it shows

that the preconditioning used in SGFS is optimal, in the sense that it converges to the true posterior, when $\boldsymbol{B}(\boldsymbol{\theta})$ is constant and the true posterior has a quadratic form.

In summary, to claim convergence to the true posterior distribution, existing approaches require either vanishing learning rates or assumptions on the SG noise covariance that are difficult to verify in practice, especially when considering deep models. We instead propose a novel practical method that induces isotropic SG noise and thus satisfies Theorem 1. We determine analytically a fixed learning rate, and we require weaker assumptions on the loss shape.

2.2. Gradient Methods with Momentum

Momentum-corrected methods emerge as a natural extension to SGD approaches. The general set of update equations for (discrete-time) momentum-based algorithms is:

$$\begin{cases} \delta\boldsymbol{\theta}_n = \eta \boldsymbol{P}(\boldsymbol{\theta}_{n-1})\boldsymbol{M}^{-1}\boldsymbol{r}_{n-1} \\ \delta\boldsymbol{r}_n = -\eta \boldsymbol{A}(\boldsymbol{\theta}_{n-1})\boldsymbol{M}^{-1}\boldsymbol{r}_{n-1} - \eta \boldsymbol{P}(\boldsymbol{\theta}_{n-1})(\boldsymbol{g}(\boldsymbol{\theta}_{n-1}) + \boldsymbol{w}_n), \end{cases}$$

where $\boldsymbol{P}(\boldsymbol{\theta}_{n-1})$ is a preconditioning matrix, M is the mass matrix and $\boldsymbol{A}(\boldsymbol{\theta}_{n-1})$ is the friction matrix, as shown by [4,19]. As with the first order counterpart, the noise term is distributed as $\boldsymbol{w}_n \sim N(\boldsymbol{0}, 2\boldsymbol{C}(\boldsymbol{\theta}_n)))$. Then, the SDE to describe continuous-time system dynamics is:

$$\begin{cases} d\boldsymbol{\theta}_t = \boldsymbol{P}(\boldsymbol{\theta}_t)\boldsymbol{M}^{-1}\boldsymbol{r}_t dt \\ d\boldsymbol{r}_t = -(A(\boldsymbol{\theta}_t)\boldsymbol{M}^{-1}r_t + \boldsymbol{P}(\boldsymbol{\theta}_t)\nabla f(\boldsymbol{\theta}_t))dt + \sqrt{2\eta \boldsymbol{P}(\boldsymbol{\theta}_t)^2\boldsymbol{\Sigma}(\boldsymbol{\theta}_t)}d\boldsymbol{W}_t. \end{cases} \tag{11}$$

where $\boldsymbol{P}(\boldsymbol{\theta}_t)^2 = \boldsymbol{P}(\boldsymbol{\theta}_t)\boldsymbol{P}(\boldsymbol{\theta}_t)$ and we assume $\boldsymbol{P}(\boldsymbol{\theta}_t)$ to be symmetric. The theorem hereafter describes the conditions for which noisy SGD with momentum converges to the true posterior distribution (Appendix A).

Theorem 2. *Consider dynamics of the form* (11) *and define the stationary distribution for* $\boldsymbol{\theta}_t$ *as* $\rho(\boldsymbol{\theta}) \propto \exp(-\phi(\boldsymbol{\theta}))$. *If*

$$\nabla^\top \boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{0}^\top \quad \textit{and} \quad \boldsymbol{A}(\boldsymbol{\theta}) = \eta \boldsymbol{P}(\boldsymbol{\theta})^2\boldsymbol{\Sigma}(\boldsymbol{\theta}), \tag{12}$$

then $\phi(\boldsymbol{\theta}) = f(\boldsymbol{\theta})$.

In the naive case, where $\boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{I}, \boldsymbol{A}(\boldsymbol{\theta}) = \boldsymbol{0}, \boldsymbol{C}(\boldsymbol{\theta}) = \boldsymbol{0}$, Equation (12) are not satisfied and the stationary distribution does not correspond to the true posterior [4]. To generate samples from the true posterior it is sufficient to set $\boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{I}, \boldsymbol{A}(\boldsymbol{\theta}) = \eta \boldsymbol{B}(\boldsymbol{\theta}), \boldsymbol{C}(\boldsymbol{\theta}) = \boldsymbol{0}$ (as in Equation (9) in [4]).

Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) [4] suggests that estimating $\boldsymbol{B}(\boldsymbol{\theta})$ can be costly. Hence, the injected noise $\boldsymbol{C}(\boldsymbol{\theta})$ is chosen such that $\boldsymbol{C}(\boldsymbol{\theta}) = \eta^{-1}\boldsymbol{A}(\boldsymbol{\theta})$, where $\boldsymbol{A}(\boldsymbol{\theta})$ is user-defined. When $\eta \to 0$, the following approximation holds: $\boldsymbol{\Sigma}(\boldsymbol{\theta}) \simeq \boldsymbol{C}(\boldsymbol{\theta})$. It is then trivial to check that conditions (12) hold without the need for explicitly estimating $\boldsymbol{B}(\boldsymbol{\theta})$. A further practical reason to avoid setting $\boldsymbol{A}(\boldsymbol{\theta}) = \eta \boldsymbol{B}(\boldsymbol{\theta})$ is that the computational cost for the operation $\boldsymbol{A}(\boldsymbol{\theta}_{n-1})\boldsymbol{M}^{-1}\boldsymbol{r}_{n-1}$ has $\mathcal{O}(D^2)$ complexity, whereas if $\boldsymbol{C}(\boldsymbol{\theta})$ is diagonal, this is reduced to $\mathcal{O}(D)$. This, however, severely slows down the sampling process.

Stochastic Gradient Riemannian Hamiltonian Monte Carlo (SGRHMC) is an extension to SGHMC [5]), which considers a generic, space-varying preconditioning matrix $\boldsymbol{P}(\boldsymbol{\theta})$ derived from information geometric arguments [20]. SGRHMC suggests setting $\boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{G}(\boldsymbol{\theta})^{-\frac{1}{2}}$, where $\boldsymbol{G}(\boldsymbol{\theta})$ is the Fisher Information matrix. To meet the requirement $\nabla^\top \boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{0}^\top$, it includes a correction term, $-\nabla^\top \boldsymbol{P}(\boldsymbol{\theta})$. The injected noise is set to $\boldsymbol{C}(\boldsymbol{\theta}) = \eta^{-1}\boldsymbol{I} - \boldsymbol{B}(\boldsymbol{\theta})$, consequently $\boldsymbol{\Sigma} = \eta^{-1}\boldsymbol{I}$, and the friction matrix is set to $\boldsymbol{A}(\boldsymbol{\theta}) = \boldsymbol{P}(\boldsymbol{\theta})^2$. With all these choices, Theorem 2 is satisfied. Although appealing, the main drawbacks of this method are the need for an analytical expression of $\nabla^\top \boldsymbol{P}(\boldsymbol{\theta})$, and the assumption for $\boldsymbol{B}(\boldsymbol{\theta})$ to be known.

From a practical standpoint, momentum-based methods suffer from the requirement to tune many hyperparameters, including the learning rate, and the parameters that govern the simulation of a second-order Langevin dynamics.

The method we propose in this work can be applied to momentum-based algorithms; in this case, it could be viewed as an extension of the work in [11], albeit addressing the complex loss landscapes typical of deep neural networks. However, we leave this avenue of research for future work.

3. Sampling by Layer-Wise Isotropization

We present a simple and practical approach to inject noise to SGD iterates to perform Bayesian posterior sampling. Our goal is to sample from the true posterior distribution (or approximations thereof) using a *constant* learning rate, and to rely on more lenient assumptions about the shape of the loss landscape that characterize deep models, compared to previous works. In general, in modern machine learning applications, we deal with multi-layer neural networks [21]. We exploit the natural subdivision of the parameters of these architecture into different layers to propose a practical sampling scheme

Careful inspection of Theorem 1 reveals that the matrices $\boldsymbol{P}(\boldsymbol{\theta}), \boldsymbol{\Sigma}(\boldsymbol{\theta})$ are instrumental in determining the convergence properties of SG methods to the true posterior. Therefore, we consider the constructive approach of *designing* $\eta \boldsymbol{P}(\boldsymbol{\theta})$ to obtain a sampling scheme that meets our goals; we set $\eta \boldsymbol{P}(\boldsymbol{\theta})$ to be a constant, diagonal matrix which we constrain to be layer-wise uniform:

$$\eta \boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{\Lambda}^{-1} = \operatorname{diag}([\underbrace{\lambda^{(1)}, \ldots, \lambda^{(1)}}_{\text{layer } 1}, \ldots, \underbrace{\lambda^{(N_l)}, \ldots \lambda^{(N_l)}}_{\text{layer } N_l}])^{-1}. \tag{13}$$

By properly selecting the set of parameters $\{\lambda^i\}$ we can achieve the simultaneous result of non-vanishing learning rate and well-conditioned preconditioning matrix. This implies a layer-wise learning rate $\eta^{(p)} = \frac{1}{\lambda^{(p)}}$ for the p-th layer, without further preconditioning.

We can now prove (see Appendix B), as a corollary to Theorem 1, that our design choices can guarantee convergence to the true posterior distribution.

Corollary 1. *(Theorem 1) Consider dynamics of the form (9) and define the stationary distribution* $\rho(\boldsymbol{\theta}) \propto \exp(-\phi(\boldsymbol{\theta}))$. *If* $\eta \boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{\Lambda}^{-1}$ *as in (13),* $\boldsymbol{C}(\boldsymbol{\theta}) = \boldsymbol{\Lambda} - \boldsymbol{B}(\boldsymbol{\theta})$ *and* $\boldsymbol{C}(\boldsymbol{\theta}) \succ 0 \quad \forall \boldsymbol{\theta}$, *then* $\phi(\boldsymbol{\theta}) = f(\boldsymbol{\theta})$.

If aforementioned conditions are satisfied, it is in fact simple to show that the relevant matrices satisfy the conditions in Equation (10). The covariance matrix of the composite noise is said to be *isotropic* within the layers of (deep) models. In fact, $\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \boldsymbol{C}(\boldsymbol{\theta}) + \boldsymbol{B}(\boldsymbol{\theta}) = \operatorname{diag}\left(\left[\lambda^{(1)}, \ldots, \lambda^{(1)}, \ldots, \lambda^{(N_l)}, \ldots \lambda^{(N_l)}\right]\right)$. From a practical point of view, we choose $\boldsymbol{\Lambda}$ to be, among all valid matrices satisfying $\boldsymbol{\Lambda} - \boldsymbol{B}(\boldsymbol{\theta}) \succ 0$, the smallest (the one with the smallest λ's). Indeed, larger $\boldsymbol{\Lambda}$ induce a smaller learning rate, thus unnecessarily reducing sampling speed.

Now, let us consider an ideal case, in which we assume the SG noise covariance $\boldsymbol{B}(\boldsymbol{\theta})$ and $\boldsymbol{\Lambda}$ to be known in advance. The procedure described in Algorithm 1 illustrates a naive SG method that uses the *injected noise* covariance $\boldsymbol{C}(\boldsymbol{\theta})$ to sample from the true posterior.

Algorithm 1 Idealized posterior sampling

{Initialization: $\boldsymbol{\theta}_0$}

SAMPLE $(\boldsymbol{\theta}_0, \boldsymbol{B}(\boldsymbol{\theta}), \boldsymbol{\Lambda})$**:**

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$

loop

$\boldsymbol{g} = \nabla \tilde{f}(\boldsymbol{\theta})$

$\boldsymbol{n} \sim N(0, \boldsymbol{I})$

$\boldsymbol{C}(\boldsymbol{\theta})^{1/2} \leftarrow (\boldsymbol{\Sigma} - \boldsymbol{B}(\boldsymbol{\theta}))^{1/2}$

$\boldsymbol{g} \leftarrow \boldsymbol{\Sigma}^{-1}(\boldsymbol{g} + \sqrt{2}\boldsymbol{C}(\boldsymbol{\theta})^{1/2}\boldsymbol{n})$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \boldsymbol{g}$

end loop

This deceivingly simple procedure generate samples from the true posterior, with a non-vanishing learning rate, as shown earlier. However, it cannot be used in practice as $\boldsymbol{B}(\boldsymbol{\theta})$ and $\boldsymbol{\Lambda}$ are unknown. Furthermore, the algorithm requires computationally expensive operations, i.e., to compute $(\boldsymbol{\Sigma} - \boldsymbol{B}(\boldsymbol{\theta}))^{\frac{1}{2}}$, which requires $\mathcal{O}(d^3)$ operations, and $\boldsymbol{C}(\boldsymbol{\theta})^{\frac{1}{2}}$, which costs $\mathcal{O}(d^2)$ multiplications.

Next, we describe a practical variant of our approach, where we use approximations at the expense of generating samples from the true posterior distribution. We note that [10] suggest exploring a related preconditioning, but do not develop this path in their work. Moreover, the proposed method shares similarities with a scheme proposed in [22] although the analysis we perform here is different.

3.1. A Practical Method: Isotropic SGD

To render the idealized sampling method practical, it is necessary to consider some additional assumptions. As we explain at the end of this section, the assumptions that follow are less strict than other approaches in the literature.

Assumption 1. *The* SG *noise covariance* $\boldsymbol{B}(\boldsymbol{\theta})$ *can be approximated with a diagonal matrix, i.e.,* $\boldsymbol{B}(\boldsymbol{\theta}) = \mathrm{diag}(\boldsymbol{b}(\boldsymbol{\theta}))$.

Assumption 2. *The signal-to-noise ratio (SNR) of a gradient is small enough such that in the stationary regime, the second-order moment of the gradient is a good estimate of the true variance. Hence, combining with Assumption 1,* $\boldsymbol{b}(\boldsymbol{\theta}) \simeq \frac{\mathrm{E}[\boldsymbol{g}(\boldsymbol{\theta}) \odot \boldsymbol{g}(\boldsymbol{\theta})]}{2}$*, where* $\odot$ *indicates the element-wise product.*

Assumption 3. *The sum of the variances of noise components, layer by layer, can be assumed to constant in the stationary regime. Then,* $\beta^{(p)} = \sum_{j \in I_p} b_j(\boldsymbol{\theta})$*, where* I_p *is the set of indices of parameters belonging to* p_{th} *layer.*

The diagonal covariance assumption (i.e., Assumption 1) is common in other works, such as [2,11]. The small signal-to-noise ratio as stated in Assumption 2 is in line with recent studies, such as [11,23]. Assumption 3 is similar to those appeared in earlier work, such as [24]. Please note that Assumptions 2 and 3 must hold in the stationary regime when the process reaches the bottom valley of the loss landscape. The matrix $(b(\boldsymbol{\theta}))$ has been associated in the literature with the *empirical* Fisher information matrix [2,25]. As we

do not consider this matrix for preconditioning purposes, we do not further investigate this connection.

Given our assumptions, and our design choices, it is then possible to show (see Appendix B) that the optimal (i.e., the smallest possible) $\mathbf{\Lambda} = \left[\lambda^{(1)}, \ldots, \lambda^{(1)}, \ldots, \lambda^{(N_l)}, \ldots \lambda^{(N_l)}\right]$ satisfying Corollary 1 can be obtained as $\lambda^{(p)} = \beta^{(p)}$. Please note that we do not assume $\boldsymbol{B}(\boldsymbol{\theta})$ to be known, but use a simple procedure to estimate its components by computing: $\lambda^{(p)} = \sum_{j \in I_p} b_j(\boldsymbol{\theta}) = \frac{||\boldsymbol{g}^{(p)}(\boldsymbol{\theta})||^2}{2}$, where $\boldsymbol{g}^{(p)}(\boldsymbol{\theta})$ is the portion of stochastic gradient corresponding to the p-th layer. Then, the composite noise matrix $\mathbf{\Sigma} = \mathbf{\Lambda}$ is a layer-wise isotropic covariance matrix, which inspires the name of our proposed method as *Isotropic* SGD (I-SGD).

The practical implementation of I-SGD is shown in Algorithm 2. The advantage of I-SGD is that it can either be used to obtain posterior samples starting from a pre-trained model, or do so by training a model from scratch. In either case, the estimates of $\boldsymbol{B}(\boldsymbol{\theta})$ are used to compute $\mathbf{\Lambda}$, as discussed above. An important consideration is that once all $\lambda^{(i)}$ have been estimated, the learning rate, layer by layer, is determined *automatically*. In fact, for the p-th layer, the learning rate is: $\eta^{(p)} = {\lambda^{(p)}}^{-1}$. A simpler approach is to use a unique learning rate for all layers, where the equivalent λ is the sum of all $\lambda^{(p)}$.

Algorithm 2 I-SGD: practical posterior sampling

SAMPLE ($\boldsymbol{\theta}_0$):

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$

loop

$\boldsymbol{g} = \nabla \tilde{f}(\boldsymbol{\theta})$

for $p \leftarrow 1$ to N_l **do**

$\boldsymbol{n} \sim N(\mathbf{0}, \boldsymbol{I})$

$\boldsymbol{C}(\boldsymbol{\theta})^{1/2} \leftarrow \left(\lambda^{(p)} - (1/2)\left(\boldsymbol{g}^{(p)} \odot \boldsymbol{g}^{(p)}\right)\right)$

$\boldsymbol{g}^{(p)} \leftarrow 1/\lambda^{(p)}\left(\boldsymbol{g}^{(p)} + \sqrt{2}\boldsymbol{C}(\boldsymbol{\theta})^{1/2}\boldsymbol{n}\right)$

end for

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \boldsymbol{g}$

end loop

A Remark on Convergence

In summary, I-SGD is a practical method to perform approximate Bayesian posterior sampling, backed up by solid theoretical foundations. Our assumptions, which are at the origin of the approximate nature of I-SGD, are less strict than those used in the literature of SG-MCMC methods. More precisely, the theory behind I-SGD can explain convergence to the true posterior with a non-vanishing learning rate in the particular case when Assumption 1 holds and the estimation of $\boldsymbol{B}(\boldsymbol{\theta})$ is perfect. Even with perfect estimates, this is not the case for SGFS, which requires the correction term $\nabla^\top \boldsymbol{B}(\boldsymbol{\theta})^{-1} = 0$. Additionally, both SGRLD and SGRHMC are more demanding than I-SGD because they require computing $\nabla^\top \boldsymbol{B}(\boldsymbol{\theta})^{-1}$, for which an estimation procedure is elusive. Finally, the method by Springenberg et al. [11] needs a *constant*, diagonal $\boldsymbol{B}(\boldsymbol{\theta})$, a condition that does not necessarily hold for deep models.

3.2. Computational Cost

The computational cost of I-SGD is as follows. As with [4], we define the cost of computing a gradient minibatch as $C_g(N_b, d)$. Thanks to Assumptions 1 and 2, the computational cost for estimating the noise covariance scales as $\mathcal{O}(d)$ multiplications. The computational cost of generating random samples with the desired covariance scales as $\mathcal{O}(d)$ square roots and $\mathcal{O}(d)$ multiplications (without considering the cost of generating random numbers). The overall cost of our method is the sum of the above terms. Notice that the cost of estimating the noise covariance does not depend on the minibatch size N_b. We would like to stress that in many modern models, the real computational bottleneck is the backward propagation for the computation of the gradients. As all the SG-MCMC methods considered in this work require one gradient evaluation per step, the different methods have in practice the same complexity.

The space complexity of I-SGD is the same as SGHMC,SGFS and variants: it scales as $\mathcal{O}(N_{\text{sam}}d)$, where N_{sam} is the number of posterior samples.

4. Experiments

The empirical analysis of our method, and its comparison to alternative approaches from the literature, is organized as follows. First, we proceed with a validation of I-SGD using the standard UCI datasets [26] and a shallow neural network. Then we move to the case of deeper models: we begin with a simple CNN used on the MNIST [27] dataset, then move to the standard ResNet-18 [28] deep network using the Cifar-10 [29] dataset.

We compare I-SGD to other Bayesian sampling methods such as SGHMC [4], SGLD [2], and to alternative approaches to approximate Bayesian inference, including MCD [12], SWAG [9] and VSGD [10]. In general, our result indicates that: (1) I-SGD achieves similar or superior performance regarding competitors, when measuring uncertainty quantification, even with simple datasets and models; (2) I-SGD is simple to tune, when compared to alternatives; (3) I-SGD is competitive when used for deep Bayesian modeling, even when compared to standard methods used in the literature. In particular, the proposed method shares some of the strengths of VSGD, such as learning rates determined automatically and the simplicity of SGLD. Appendix B includes additional implementation details on I-SGD. Appendix C presents detailed configurations of all methods we compare, and additional experimental results.

4.1. A Disclaimer on Performance Characterization

It is important to stress a detail on the analysis of the experimental campaign. The discussion is usually focused on the goodness of the various methods for representing the true posterior distribution. Different methods can or cannot claim convergence to the true posterior according to certain assumptions and the nature of the hyperparameters. In the experimental validation of the results, however, we do not have access to the form of the true posterior as it is exactly the problem we are trying to solve. The practical solution adopted is to compare the different methods in terms of *proxy* metrics evaluated on the test sets, such as the accuracy and uncertainty metrics. Being better in terms of these performance metrics does not imply that the sampling method is better at approximating the posterior distribution, and outperforming competitors in terms of these metric do not provide sufficient information about the intrinsic quality of the sampling scheme.

4.2. Regression Tasks, with Simple Models

We consider several regression tasks defined on the UCI datasets. We use a simple neural network configuration with two fully connected layers and a ReLU activation function; the hidden layer includes 50 units. In this set of experiments, we use the following metrics: the root mean square error (RMSE) to judge the model predictive performance and the mean negative log-likelihood (MNLL) as a proxy for uncertainty quantification. We note that the task of tuning our competitors was far from trivial. We used our own version of SGHMC, based on [11], to ensure a proper understanding of the implementation internals,

and we proceeded with a tuning process to find appropriate values for the numerous hyperparameters. In this set of experiments, we omit results for SWAG, which we keep for more involved scenarios.

Tables 1 and 2 report a complete overview of our results, for a selection of UCI datasets. For each method and each dataset, we also included how many out of the 10 splits considered failed to converge, indicated as $F = \ldots$. As explained in Appendix C we implemented a temperature scaled version of VSGD. A clear picture emerges from this first set of experiments: while for the RMSE the performance is similar for different methods, for the MNLL averaging over multiple samples clearly improves the uncertainty quantification capabilities. SGHMC is in many cases better than alternatives, considering however the standard deviation of the results it is difficult to claim clear superiority of one method over the others.

Table 1. RMSE results for regression on UCI datasets.

Method	WINE	PROTEIN	NAVAL	KIN8NM	POWER	BOSTON
SGLD	0.759 ± 0.07	5.687 ± 0.05	0.007 ± 0.00 (F = 6.000)	0.171 ± 0.07 (F = 3.000)	11.753 ± 3.25	9.602 ± 2.06
I-SGD	0.635 ± 0.05	4.699 ± 0.03	0.001 ± 0.00	0.079 ± 0.00	4.320 ± 0.13	3.703 ± 1.19
Baseline	0.641 ± 0.05	4.733 ± 0.05	0.001 ± 0.00	0.080 ± 0.00	4.354 ± 0.12	3.705 ± 1.19
VSGD	0.635 ± 0.05	4.699 ± 0.03	0.001 ± 0.00	0.079 ± 0.00	4.325 ± 0.13	3.588 ± 1.06 (F = 1.000)
SGHMC	0.628 ± 0.04	4.712 ± 0.03	0.000 ± 0.00 (F = 2.000)	0.076 ± 0.00 (F = 1.000)	4.310 ± 0.14	3.659 ± 1.24
SGLD T	0.752 ± 0.07	5.673 ± 0.04	0.007 ± 0.00 (F = 6.000)	0.169 ± 0.07 (F = 3.000)	11.351 ± 3.02	9.417 ± 2.07
DROP	0.637 ± 0.04	4.968 ± 0.05	0.003 ± 0.00	0.139 ± 0.01	4.531 ± 0.16	3.803 ± 1.26
SGHMC T	0.628 ± 0.04	4.684 ± 0.03	0.000 ± 0.00 (F = 6.000)	0.076 ± 0.00	4.326 ± 0.13	3.692 ± 1.19

Table 2. MNLL results for regression on UCI datasets.

Method	WINE	PROTEIN	NAVAL	KIN8NM	POWER	BOSTON
SGLD	1.546 ± 0.25	5.604 ± 0.08	−1.751 ± 0.28 (F = 6.000)	5.140 ± 7.05 (F=3.000)	8.429 ± 3.14	30.386 ± 15.77
I-SGD	1.129 ± 0.15	4.371 ± 0.03	−2.466 ± 1.12	−0.460 ± 0.65	3.122 ± 0.07	9.799 ± 5.69
Baseline	1.182 ± 0.03	3.964 ± 0.04	0.920 ± 0.00	0.924 ± 0.00	3.071 ± 0.06	5.421 ± 2.73
VSGD	1.128 ± 0.15	4.371 ± 0.03	−2.466 ± 1.12	−0.480 ± 0.65	3.088 ± 0.06	8.413 ± 5.89 (F = 1.000)
SGHMC	1.041 ± 0.12	4.142 ± 0.02	−2.763 ± 1.33 (F = 2.000)	−0.798 ± 0.39 (F = 1.000)	2.924 ± 0.04	3.097 ± 0.83
SGLD T	1.526 ± 0.24	5.591 ± 0.07	−1.752 ± 0.28 (F = 6.000)	5.118 ± 7.06 (F = 3.000)	8.288 ± 3.04	33.212 ± 19.69
DROP	1.065 ± 0.12	4.218 ± 0.06	−2.322 ± 0.75	−0.086 ± 0.41	2.941 ± 0.04	3.989 ± 1.23
SGHMC T	1.104 ± 0.14	4.191 ± 0.02	−2.966 ± 1.89 (F = 6.000)	−0.756 ± 0.42	3.116 ± 0.07	9.826 ± 5.72

4.3. Classification Tasks, with Deeper Models

Next, we compare I-SGD against competitors on image classification tasks. First, we use the MNIST dataset, and a simple LeNet-5 CNN [30]. All methods are compared based on the test accuracy ACC,MNLL and the expected calibration error (ECE, [31]). Additionally, at test time, we carry out predictions on both MNIST and NOT-MNIST; the latter is a dataset equivalent to MNIST , but it represents letters rather than numbers. (http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html, accessed on 24 October 2021) This experimental setup is often used to check whether the entropy of the predictions on NOT-MNIST is higher than the entropy of the predictions on MNIST (the entropy of the output of an N_{cl} classes classifier, represented by the vector $\mathbf{p}$, is defined as $-\sum_{i=1}^{N_{cl}} p_i \log p_i$).

Table 3 indicates that all methods are essentially equivalent in terms of accuracy and MNLL. We consider, together with the classical in and out of distribution entropies the regions of convergence (ROCS) diagrams comparing detection of out of distribution

samples and false alarms when using as test statistic the entropy. Results, reported in Figure 1, clearly shows that: (1) collecting multiple samples improve the uncertainty quantification capabilities (2) I-SGD is competitive (but not the best scheme) and importantly outperform the closest approach to ours, i.e., VSGD. The experimental results show that I-SGD improves the quality of the BASELINE model with respect to all metrics. To test whether the improvements are due just to "*additional training*" or are intrinsically due to the Bayesian averaging properties, we do consider alternative deterministic baselines (details in Appendix C). For this set of experiments the best performing is BASELINE R. As can be appreciated by comparing Table 3 and Figure 1, while it is possible to increase the classical metrics, I-SGD (and other methods) still outperform by a large margin the baselines in terms of detection of out of distribution samples.

Table 3. Results for classification on MNIST dataset.

Method	ACC	MNLL	Mean H_0	ECE	Mean H_1	Failed
I-SGD	9916.3333 ± 2.8674	263.5311 ± 16.3600	0.0368 ± 0.0019	0.0491 ± 0.0003	0.4558 ± 0.0591	0.0000
SGHMC	9930.6667 ± 2.4944	268.2559 ± 6.8172	0.0593 ± 0.0018	0.0531 ± 0.0003	1.0369 ± 0.0346	0.0000
DROP	9912.6667 ± 6.0185	362.8973 ± 24.8881	0.0960 ± 0.0090	0.0541 ± 0.0011	0.5507 ± 0.0577	0.0000
BASELINE	9886.6667 ± 11.0252	352.6640 ± 20.8622	0.0353 ± 0.0058	0.0468 ± 0.0001	0.0019 ± 0.0003	0.0000
BASELINE r	9919.0000 ± 9.4163	242.7644 ± 17.0736	0.0303 ± 0.0001	0.0482 ± 0.0006	0.0021 ± 0.0002	0.0000
SWAG	9917.0000 ± 2.8284	308.8182 ± 20.0979	0.0675 ± 0.0108	0.0524 ± 0.0011	0.3953 ± 0.0442	0.0000
SGLD	9927.0000 ± 1.0000	279.7685 ± 16.6563	0.0556 ± 0.0034	0.0531 ± 0.0004	1.3032 ± 0.1942	1.0000
VSGD	9927.3333 ± 6.7987	225.3725 ± 16.3739	0.0274 ± 0.0008	0.0481 ± 0.0005	0.0414 ± 0.0070	0.0000
I-SGD T	9915.6667 ± 0.9428	255.9641 ± 12.8051	0.0289 ± 0.0014	0.0478 ± 0.0002	0.0284 ± 0.0122	0.0000
SGHMC T	9937.0000 ± 0.0000	231.5332 ± 0.0000	0.0434 ± 0.0000	0.0518 ± 0.0000	0.4623 ± 0.0000	2.0000

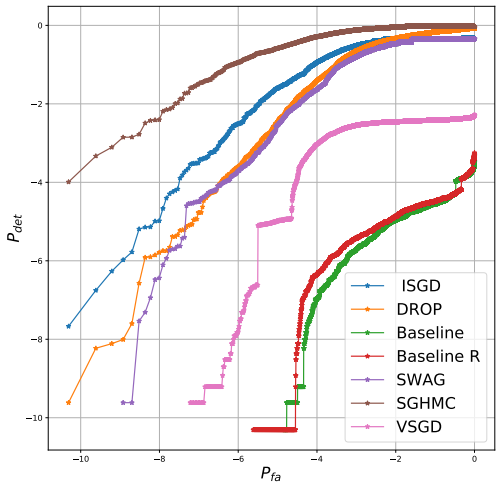

Figure 1. Detection/False alarm diagrams for different methods.

We now move on to a classical image classification problem with deep convolutional networks, whereby we use the CIFAR10 dataset, and the RESNET-18 network architecture. For this set of experiments, we compare I-SGD, SGHMC, SWAG, and VSGD using again test accuracy and MNLL, which we report in Table 4. As usual, we compare the results against the baseline of the individual network resulting from the pre-training phase. Results are obtained averaging over three independent seeds. Notice, as expanded in Appendix C that for SWAG we do consider two variants: the Bayesian correct one (SWAG) and a second variant that has better performance (SWAG wd). We stress again, as highlighted in Section 4.1

that not always goodness of approximation of the posterior and performance correlate positively. Additionally in this case, we found I-SGD to be competitive with other methods and superior to the baseline. Among the competitors, we found I-SGD to the easiest to tune, given the feature of a fixed learning rate informed by theoretical considerations; we believe that this is an important aspect to consider for a wide adoption of our proposal by practitioners.

Table 4. Results for classification on CIFAR10 10 dataset.

Method	ACC	MNLL	mean H_0	ECE
I-SGD	8591.3333 ± 17.4611	4393.3557 ± 107.0878	0.6107 ± 0.0337	0.0731 ± 0.0075
SGHMC	8634.6667 ± 5.1854	4357.8998 ± 11.2722	0.6300 ± 0.0023	0.0819 ± 0.0017
SWAG wd	8740.6667 ± 35.5653	3931.9900 ± 45.6605	0.4130 ± 0.0066	0.0275 ± 0.0015
SWAG	8061.0000 ± 11.4310	5903.2605 ± 62.8167	0.5308 ± 0.0135	0.0163 ± 0.0019
BASELINE	8273.3333 ± 26.7872	8050.4467 ± 109.9864	0.2250 ± 0.0005	0.0809 ± 0.0020
VSGD	8255.6667 ± 24.1155	8919.8062 ± 106.3571	0.1761 ± 0.0078	0.0905 ± 0.0020

5. Conclusions

SG methods allowed Bayesian posterior sampling algorithms, such as MCMC, to regain relevance in an age when datasets have reached extremely large sizes. However, despite mathematical elegance and promising results, current approaches from the literature are restricted to simple models. Indeed, the sampling properties of these algorithms are determined by simplifying assumptions on the loss landscape, which do not hold for the kind of complex models which are popular these days, such as deep models. Meanwhile, SG-MCMC algorithms require vanishing learning rates, which force practitioners to develop creative annealing schedules that are often model specific and difficult to justify.

We have attempted to target these weaknesses by suggesting a simpler algorithm that relies on fewer parameters and less strict assumptions compared to the literature on SG-MCMC. We used a unified mathematical notation to deepen our understanding of the role of the covariance of the noise of stochastic gradients and learning rate on the behavior of SG-MCMC algorithms. We then presented a practical variant of the SGD algorithm, which uses a constant learning rate, and an additional noise to perform Bayesian posterior sampling. Our proposal is derived from the ideal method, in which it is guaranteed that samples are generated from the true posterior. When the learning rate and noise terms are empirically estimated, with no user intervention, our method offers a very good approximation to the posterior, as demonstrated by the extensive experimental campaign.

We verified empirically the quality of our approach, and compared its performance to state-of-the-art SG-MCMC and alternative methods. Results, which span a variety of settings, indicated that our method is competitive to the alternatives from the state-of-the-art, while being much simpler to use.

Author Contributions: Formal analysis, G.F., D.M., M.F. and P.M.; Methodology, G.F., D.M., M.F. and P.M.; Software, G.F., D.M., M.F. and P.M.; Writing—original draft, G.F., D.M., M.F. and P.M.; Writing—review & editing, G.F., D.M., M.F. and P.M. All authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: MF gratefully acknowledges support from the AXA Research Fund and the Agence Nationale de la Recherche (grant ANR-18-CE46-0002 and ANR-19-P3IA-0002).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Background and Related Material

Appendix A.1. The Minibatch Gradient Approximation

Starting from the gradient of the logarithm of the posterior density:

$$-\nabla f(\boldsymbol{\theta}) = \sum_{i=1}^{N} \nabla \log p(\boldsymbol{U}_i|\boldsymbol{\theta}) + \nabla \log p(\boldsymbol{\theta}),$$

it is possible to define its *minibatch* version by computing the gradient on a random subset $\mathcal{I}_{N_b}$ with cardinality N_b of all the indices. The minibatch gradient $\boldsymbol{g}(\boldsymbol{\theta})$ is computed as

$$-\boldsymbol{g}(\boldsymbol{\theta}) = \frac{N}{N_b} \sum_{i=1}^{N_b} \nabla \log p(\boldsymbol{U}_i|\boldsymbol{\theta}) + \nabla \log p(\boldsymbol{\theta}),$$

By simple calculations it is possible to show that the estimation is unbiased ($E(\boldsymbol{g}(\boldsymbol{\theta})) = \nabla f(\boldsymbol{\theta})$). The estimation error covariance is defined to be $E\Big[(\boldsymbol{g}(\boldsymbol{\theta}) - \nabla f(\boldsymbol{\theta}))(\boldsymbol{g}(\boldsymbol{\theta}) - \nabla f(\boldsymbol{\theta}))^\top\Big] = 2\boldsymbol{B}(\boldsymbol{\theta})$.

If the minibatch size is large enough, invoking the central limit theorem, we can state that the minibatch gradient is normally distributed:

$$\boldsymbol{g}(\boldsymbol{\theta}) \sim N(\nabla f(\boldsymbol{\theta}), 2\boldsymbol{B}(\boldsymbol{\theta})).$$

Appendix A.2. Gradient Methods without Momentum

Appendix A.2.1. The SDE from Discrete Time

We start from the generalized updated rule of SGD:

$$\delta\boldsymbol{\theta}_n = -\eta \boldsymbol{P}(\boldsymbol{\theta}_{n-1})(\boldsymbol{g}(\boldsymbol{\theta}_{n-1}) + \boldsymbol{w}_n).$$

Since $\boldsymbol{g}(\boldsymbol{\theta}_{n-1}) \sim N(\nabla f(\boldsymbol{\theta}_{n-1}), 2\boldsymbol{B}(\boldsymbol{\theta}_{n-1}))$ we can rewrite the above equation as:

$$\delta\boldsymbol{\theta}_n = -\eta \boldsymbol{P}(\boldsymbol{\theta}_{n-1})(\nabla f(\boldsymbol{\theta}_{n-1}) + \boldsymbol{w}_n^{'}),$$

where $\boldsymbol{w}_n^{'} \sim N(0, 2\boldsymbol{\Sigma}(\boldsymbol{\theta}_{n-1}))$. If we separate deterministic and random component we can equivalently write:

$$\delta\boldsymbol{\theta}_n = -\eta \boldsymbol{P}(\boldsymbol{\theta}_{n-1})\nabla f(\boldsymbol{\theta}_{n-1}) + \eta \boldsymbol{P}(\boldsymbol{\theta}_{n-1})\boldsymbol{w}_n^{'} = -\eta \boldsymbol{P}(\boldsymbol{\theta}_{n-1})\nabla f(\boldsymbol{\theta}_{n-1}) + \sqrt{2\eta \boldsymbol{P}^2(\boldsymbol{\theta}_{n-1})\boldsymbol{\Sigma}(\boldsymbol{\theta}_{n-1})}\boldsymbol{v}_n$$

where $\boldsymbol{v}_n \sim N(0, \sqrt{\eta}\boldsymbol{I})$. When η is small enough ($\eta \to dt$) we can interpret the above equation as the discrete-time simulation of the following SDE [15]:

$$d\boldsymbol{\theta}_t = -\boldsymbol{P}(\boldsymbol{\theta}_t)\nabla f(\boldsymbol{\theta}_t)dt + \sqrt{2\eta \boldsymbol{P}(\boldsymbol{\theta}_t)^2\boldsymbol{\Sigma}(\boldsymbol{\theta}_t)}d\boldsymbol{W}_t,$$

where $d\boldsymbol{W}_t$ is a $d-$dimensional Brownian motion.

Appendix A.2.2. Proof of Theorem 1

The stationary distribution of the above SDE, $\rho(\boldsymbol{\theta}) \propto \exp(-\phi(\boldsymbol{\theta}))$, satisfies the following FPE

$$0 = \mathrm{Tr}\Big\{\nabla\Big[\nabla^\top(f(\boldsymbol{\theta}))\boldsymbol{P}(\boldsymbol{\theta})\rho(\boldsymbol{\theta}) + \eta\nabla^\top(\boldsymbol{P}(\boldsymbol{\theta})^2\boldsymbol{\Sigma}(\boldsymbol{\theta})\rho(\boldsymbol{\theta}))\Big]\Big\},$$

that we rewrite as

$$0 = \mathrm{Tr}\{\nabla[\nabla^\top(f(\boldsymbol{\theta}))\boldsymbol{P}(\boldsymbol{\theta})\rho(\boldsymbol{\theta}) - \eta\nabla^\top(\phi(\boldsymbol{\theta}))\boldsymbol{P}(\boldsymbol{\theta})^2\boldsymbol{\Sigma}(\boldsymbol{\theta})\rho(\boldsymbol{\theta}) + \eta\nabla^\top(\boldsymbol{P}(\boldsymbol{\theta})^2\boldsymbol{\Sigma}(\boldsymbol{\theta}))\rho(\boldsymbol{\theta})]\}.$$

The above equation is verified with $\nabla f(\boldsymbol{\theta}) = \nabla \phi(\boldsymbol{\theta})$ if

$$\begin{cases} \nabla^\top (\boldsymbol{P}(\boldsymbol{\theta})^2 \boldsymbol{\Sigma}(\boldsymbol{\theta})) = \mathbf{0} \\ \eta \boldsymbol{P}(\boldsymbol{\theta})^2 \boldsymbol{\Sigma}(\boldsymbol{\theta}) = \boldsymbol{P}(\boldsymbol{\theta}) \rightarrow \eta \boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1} \end{cases}$$

that proves Theorem 1.

Appendix A.3. Gradient Methods with Momentum

Appendix A.3.1. The SDE from Discrete Time

The general set of update equations for (discrete-time) momentum-based algorithms is:

$$\begin{cases} \delta \boldsymbol{\theta}_n = \eta \boldsymbol{P}(\boldsymbol{\theta}_{n-1}) \boldsymbol{M}^{-1} \boldsymbol{r}_{n-1} \\ \delta \boldsymbol{r}_n = -\eta \boldsymbol{A}(\boldsymbol{\theta}_{n-1}) \boldsymbol{M}^{-1} \boldsymbol{r}_{n-1} - \eta \boldsymbol{P}(\boldsymbol{\theta}_{n-1})(\boldsymbol{g}(\boldsymbol{\theta}_{n-1}) + \boldsymbol{w}_n). \end{cases}$$

Similarly to the case without momentum, we rewrite the second equation of the system as

$$\begin{aligned} &\delta \boldsymbol{r}_n = -\eta \boldsymbol{A}(\boldsymbol{\theta}_{n-1}) \boldsymbol{M}^{-1} \boldsymbol{r}_{n-1} - \eta \boldsymbol{P}(\boldsymbol{\theta}_{n-1})(\boldsymbol{g}(\boldsymbol{\theta}_{n-1}) + \boldsymbol{w}_n) = \\ &- \eta \boldsymbol{A}(\boldsymbol{\theta}_{n-1}) \boldsymbol{M}^{-1} \boldsymbol{r}_{n-1} - \eta \boldsymbol{P}(\boldsymbol{\theta}_{n-1}) \nabla f(\boldsymbol{\theta}_{n-1}) + \sqrt{2\eta \boldsymbol{P}^2(\boldsymbol{\theta}_{n-1}) \boldsymbol{\Sigma}(\boldsymbol{\theta}_{n-1})} \boldsymbol{v}_n \end{aligned}$$

where again $\boldsymbol{v}_n \sim N(0, \sqrt{\eta}\boldsymbol{I})$. If we define the supervariable $\boldsymbol{z} = [\boldsymbol{\theta}, \boldsymbol{r}]^\top$ we can rewrite the system as

$$\delta \boldsymbol{z}_n = -\eta \begin{bmatrix} \mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta}_{n-1}) \\ \boldsymbol{P}(\boldsymbol{\theta}_{n-1}) & \boldsymbol{A}(\boldsymbol{\theta}_{n-1}) \end{bmatrix} \boldsymbol{s}(\boldsymbol{z}_{n-1}) + \sqrt{2\eta \boldsymbol{D}(\boldsymbol{z}_{n-1})} \boldsymbol{\nu}_n$$

where $\boldsymbol{s}(\boldsymbol{z}) = \begin{bmatrix} \nabla f(\boldsymbol{\theta}) \\ \boldsymbol{M}^{-1} \boldsymbol{r} \end{bmatrix}$, $\boldsymbol{D}(\boldsymbol{z}) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{P}(\boldsymbol{\theta})^2 \boldsymbol{\Sigma}(\boldsymbol{\theta}) \end{bmatrix}$ and $\boldsymbol{\nu}_n \sim N(0, \sqrt{\eta}\boldsymbol{I})$.

As the learning rate goes to zero ($\eta \rightarrow dt$), similarly to the previous case, we can interpret the above difference equation as a discretization of the following FPE

$$d\boldsymbol{z}_t = -\begin{bmatrix} \mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta}_t) \\ \boldsymbol{P}(\boldsymbol{\theta}_t) & \boldsymbol{A}(\boldsymbol{\theta}_t) \end{bmatrix} \boldsymbol{s}(\boldsymbol{z}_t) + \sqrt{2\eta \boldsymbol{D}(\boldsymbol{z}_t)} d\boldsymbol{W}_t$$

Appendix A.3.2. Proof of Theorem 2

As before we assume that the stationary distribution has form $\rho(\boldsymbol{z}) \propto \exp(-\phi(\boldsymbol{z}))$. The corresponding FPE is

$$0 = \mathrm{Tr}\left(\nabla \left(\boldsymbol{s}(\boldsymbol{z})^\top \begin{bmatrix} \mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta}) \\ \boldsymbol{P}(\boldsymbol{\theta}) & \boldsymbol{A}(\boldsymbol{\theta}) \end{bmatrix} \rho(\boldsymbol{z}) + \eta \left(\nabla^\top (\boldsymbol{D}(\boldsymbol{z}) \rho(\boldsymbol{z})) \right) \right) \right).$$

Notice that since $\nabla^\top \boldsymbol{D}(\boldsymbol{z}) = 0$ we can rewrite

$$\begin{aligned} 0 &= \mathrm{Tr}\left(\nabla \left(\boldsymbol{s}(\boldsymbol{z})^\top \begin{bmatrix} \mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta}) \\ \boldsymbol{P}(\boldsymbol{\theta}) & \boldsymbol{A}(\boldsymbol{\theta}) \end{bmatrix} \rho(\boldsymbol{z}) + \eta \nabla^\top (\rho(\boldsymbol{z})) \boldsymbol{D}(\boldsymbol{z}) \right) \right) \\ &= \mathrm{Tr}\left(\nabla \left(\boldsymbol{s}(\boldsymbol{z})^\top \begin{bmatrix} \mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta}) \\ \boldsymbol{P}(\boldsymbol{\theta}) & \boldsymbol{A}(\boldsymbol{\theta}) \end{bmatrix} \rho(\boldsymbol{z}) - \eta \nabla^\top (\phi(\boldsymbol{z})) \boldsymbol{D}(\boldsymbol{z}) \rho(\boldsymbol{z}) \right) \right) \\ &= \mathrm{Tr}\left(\nabla \left(\boldsymbol{s}(\boldsymbol{z})^\top \begin{bmatrix} \mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta}) \\ \boldsymbol{P}(\boldsymbol{\theta}) & \boldsymbol{A}(\boldsymbol{\theta}) \end{bmatrix} \rho(\boldsymbol{z}) - \eta \nabla^\top (\phi(\boldsymbol{z})) \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{P}(\boldsymbol{\theta})^2 \boldsymbol{\Sigma}(\boldsymbol{\theta}) \end{bmatrix} \rho(\boldsymbol{z}) \right) \right) \end{aligned}$$

that is verified with $\nabla \phi(\boldsymbol{z}) = \boldsymbol{s}(\boldsymbol{z})$ if

$$\begin{cases} \nabla^\top \boldsymbol{P}(\boldsymbol{\theta}) = \mathbf{0} \\ \boldsymbol{A}(\boldsymbol{\theta}) = \eta \boldsymbol{P}(\boldsymbol{\theta})^2 \boldsymbol{\Sigma}(\boldsymbol{\theta}). \end{cases}$$

If $\nabla^\top \boldsymbol{P}(\boldsymbol{\theta}) = \mathbf{0}$ in fact

$$\mathrm{Tr}\left(\nabla\left(\nabla^\top(\phi(z))\rho(z)\begin{bmatrix}\mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta})\\ \boldsymbol{P}(\boldsymbol{\theta}) & \mathbf{0}\end{bmatrix}\right)\right) = \nabla^\top\left(\begin{bmatrix}\mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta})\\ \boldsymbol{P}(\boldsymbol{\theta}) & \mathbf{0}\end{bmatrix}\nabla(\phi(z))\rho(z)\right) =$$
$$\nabla^\top\left(\begin{bmatrix}\mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta})\\ \boldsymbol{P}(\boldsymbol{\theta}) & \mathbf{0}\end{bmatrix}\right)\nabla(\phi(z))\rho(z) + \mathrm{Tr}\left(\begin{bmatrix}\mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta})\\ \boldsymbol{P}(\boldsymbol{\theta}) & \mathbf{0}\end{bmatrix}\nabla\left(\nabla^\top(\phi(z))\rho(z)\right)\right) = 0,$$

since $\nabla^\top\begin{bmatrix}\mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta})\\ \boldsymbol{P}(\boldsymbol{\theta}) & \mathbf{0}\end{bmatrix} = \mathbf{0}$ and the second term is zero due to the fact that $\begin{bmatrix}\mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta})\\ \boldsymbol{P}(\boldsymbol{\theta}) & \mathbf{0}\end{bmatrix}$ is anti-symmetric while $\nabla\left(\nabla^\top(\phi(z))\rho(z)\right)$ is symmetric.

Thus, we can rewrite

$$\mathrm{Tr}\left(\nabla\left(\boldsymbol{s}(z)^\top\begin{bmatrix}\mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta})\\ \boldsymbol{P}(\boldsymbol{\theta}) & \boldsymbol{A}(\boldsymbol{\theta})\end{bmatrix}\rho(z) - \eta\nabla^\top(\phi(z))\begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{P}(\boldsymbol{\theta})^2\boldsymbol{\Sigma}(\boldsymbol{\theta})\end{bmatrix}\rho(z)\right)\right) =$$
$$\mathrm{Tr}\left(\nabla\left(\boldsymbol{s}(z)^\top\begin{bmatrix}\mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta})\\ \boldsymbol{P}(\boldsymbol{\theta}) & \boldsymbol{A}(\boldsymbol{\theta})\end{bmatrix}\rho(z) - \nabla^\top(\phi(z))\begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \eta\boldsymbol{P}(\boldsymbol{\theta})^2\boldsymbol{\Sigma}(\boldsymbol{\theta})\end{bmatrix}\rho(z)\right)\right) =$$
$$\mathrm{Tr}\left(\nabla\left(\boldsymbol{s}(z)^\top\begin{bmatrix}\mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta})\\ \boldsymbol{P}(\boldsymbol{\theta}) & \boldsymbol{A}(\boldsymbol{\theta})\end{bmatrix}\rho(z) - \nabla^\top(\phi(z))\begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{A}(\boldsymbol{\theta})\end{bmatrix}\rho(z)\right)\right) =$$
$$\mathrm{Tr}\left(\nabla\left(\left(\boldsymbol{s}(z)^\top - \nabla^\top(\phi(z))\right)\begin{bmatrix}\mathbf{0} & -\boldsymbol{P}(\boldsymbol{\theta})\\ \boldsymbol{P}(\boldsymbol{\theta}) & \boldsymbol{A}(\boldsymbol{\theta})\end{bmatrix}\rho(z)\right)\right) = 0$$

and then $\nabla\phi(z) = \boldsymbol{s}(z)$ proving Theorem 2.

Appendix B. I-SGD Method Proofs and Details

Appendix B.1. Proof of Corollary 1

The requirement $\boldsymbol{C}(\boldsymbol{\theta}) \succeq 0 \quad \forall\boldsymbol{\theta}$, ensures that the injected noise covariance is valid. The composite noise matrix is equal to $\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \boldsymbol{\Lambda}$. Since $\nabla^\top\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \nabla^\top\boldsymbol{\Lambda} = \mathbf{0}$ and $\eta\boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{\Lambda}^{-1}$ by construction, then Theorem 1 is satisfied.

Appendix B.2. Proof of Optimality of $\boldsymbol{\Lambda}$

Our design choice is to select $\lambda^{(p)} = \beta^{(p)}$. By the assumptions, the matrix $\boldsymbol{B}(\boldsymbol{\theta})$ is diagonal, and consequently $\boldsymbol{C}(\boldsymbol{\theta}) = \boldsymbol{\Lambda} - \boldsymbol{B}(\boldsymbol{\theta})$ is diagonal as well. The preconditioner $\boldsymbol{\Lambda}$ must be chosen to satisfy the positive semidefinite constraint, i.e., $\boldsymbol{C}(\boldsymbol{\theta})_{ii} \geq 0 \quad \forall i, \forall\boldsymbol{\theta}$. Equivalently, we must satisfy $\lambda^{(p)} - b_j(\boldsymbol{\theta}) \geq 0 \quad \forall j \in I_p, \forall p, \forall\boldsymbol{\theta}$, where I_p is the set of indices of parameters belonging to p_{th} layer. By assumption 3, i.e., $\beta^{(p)} = \sum_{k\in I_p} b_k(\boldsymbol{\theta})$, it is easy to show that $b_j(\boldsymbol{\theta}), j \in I_p$, is upper bounded as $b_j(\boldsymbol{\theta}) \leq \beta^{(p)}$. To satisfy the positive semidefinite requirement in all cases the minimum valid set of $\lambda^{(p)}$ is then determined as $\lambda^{(p)} = \beta^{(p)}$.

Appendix B.3. Algorithmic Details

In this section, we provide further details about the practical implementation of the proposed scheme. At any (discrete) time instant a minibatch version of the gradient is computed that is distributed, according to the hypotheses of the main paper, as $\boldsymbol{g}(\boldsymbol{\theta}) \sim N(\nabla f(\boldsymbol{\theta}), 2b(\boldsymbol{\theta}))$. Since we assumed that the second-order moment is a good approximation of the variance, we can estimate $b(\boldsymbol{\theta})$ as $\frac{1}{2}(\boldsymbol{g}(\boldsymbol{\theta}) \odot \boldsymbol{g}(\boldsymbol{\theta}))$. In practice, we found that the following running average estimation procedure to be the most robust

$$b(\boldsymbol{\theta}) \leftarrow \mu b(\boldsymbol{\theta}) + (1-\mu)\frac{1}{2}(\boldsymbol{g}(\boldsymbol{\theta}) \odot \boldsymbol{g}(\boldsymbol{\theta})) \tag{A1}$$

where $\mu \in (0, 1]$. In all experiments we considered $\mu = 0.5$

After a warmup period, the various $\lambda^{(p)}$, layer per layer, are estimated as $\lambda^{(p)} = \sum_{k\in I_p} b_k(\boldsymbol{\theta})$ and kept constant until the end. The estimation procedure continues during sampling phase,

as the quantity $\lambda^{(p)} - b(\boldsymbol{\theta})$ is necessary at every step. As the learning rate is derived as $\frac{2}{\lambda^{(p)}}$, we found that the usage of second-order moments instead of variances, and in certain cases temperature scaling, kept the simulated trajectories more stable.

Appendix C. Methodology

We hereafter present additional implementation details.

Appendix C.1. Regression Tasks, with Simple Models

For this set of experiments we considered , the BASELINE is obtained by running the ADAM optimizer for 20,000 steps with learning rate 0.01 and default parameters. At test time we use 100 samples to estimate the predictive posterior distribution, using Equation (3), for the *sampling* methods (I-SGD,SGLD,SGHMC,VSGD), with a keep-every value equal to 1000. The I-SGD and VSGD sampling methods are started from the BASELINE. For I-SGD we selected temperature 0.01, while for SGHMC and SGLD we do performed experiments for temperatures 1 and 0.01. We modified the implementation of VSGD as the original implementation produced unstable learning rates (as noticed also in [9]). A simple and effective solution we implement that we kept throughout the experimental campaigns is to divide the learning rate by the number of parameters (thus performing variational inference on a tempered version of the posterior). For SGLD the learning rate decay is the one suggested in [2], with initial and finial learning rate equal to 10^{-6} and 10^{-8} respectively. For MCD we collected 1000 samples with standard dropout rate of 0.5. All our experiments use 10-splits. The considered batch size is 64 for all methods.

Appendix C.2. Classification Task, CONVNET

For the LENET-5 on MNIST experiment, we do consider also the SWAG algorithm. At test time we use 30 samples for all methods. Baselines are again trained using ADAM optimizer for 20,000 steps with learning rate 0.01 and default parameters. For I-SGD and SGHMC we collected samples for the different temperatures of 1 and 0.01. SGLD has initial and final learning rates of 10^{-3} and 10^{-5}. For all the sampling methods we do collect 100 samples with a keep-every of 10,000 steps. SWAG results are obtained by collecting the statistics over 300 epochs using ADAM optimizer and decreasing the learning rate every epoch in accordance with the original paper schedule [9]. DROP results are obtained by training the networks with SGD, with learning rate 0.005 and momentum 0.5. The number of collected samples for this method is 1000. The batch size for all the methods is 128.

As explained in the main text, we performed an ablation study on the considered baselines. In Table A1 we do report the results for the additional variants obtained by early stopping (10,000 iterations instead of 20,000) BASELINE S, to ablate overfitting, and BASELINE L, by training for 30,000 iterations. Finally, we include the best performing BASELINE R, obtained starting from BASELINE, reducing the learning rate by a factor of 10 and training for 10,000 more iterations.

Table A1. Baselines comparison for classification on MNIST dataset.

Method	ACC	MNLL	Mean H_0	ECE	Mean H_1	Failed
BASELINE	9886.6667 ± 11.0252	352.6640 ± 20.8622	0.0353 ± 0.0058	0.0468 ± 0.0001	0.0019 ± 0.0003	0.0000
BASELINE l	9871.6667 ± 20.7579	389.7142 ± 79.0354	0.0378 ± 0.0051	0.0468 ± 0.0008	0.0025 ± 0.0006	0.0000
BASELINE s	9893.0000 ± 4.8990	339.8170 ± 7.9855	0.0392 ± 0.0042	0.0477 ± 0.0008	0.0024 ± 0.0001	0.0000
BASELINE r	9919.0000 ± 9.4163	242.7644 ± 17.0736	0.0303 ± 0.0001	0.0482 ± 0.0006	0.0021 ± 0.0002	0.0000

Appendix C.3. Classification Task, Deeper Models

We here report details for the RESNET-18 on CIFAR10 experiments. The BASELINE is obtained with ADAM optimizer with learning rate 0.01 decreased by a factor of 10 every 50 epochs for a total of 200 epochs and weight decay of 0.05. For this set of experiments no

temperature scaling was required. We could not find good hyperparameters for the SGLD scheme. Concerning I-SGD, SGHMC and VSGD the keep-every value is chosen as 10,000 and the number of collected samples is 30. For SWAG we used the default parameters described in [9]. Notice that for SWAG we performed the following ablation study: we trained the networks considering as loss function the joint log-likelihood and included or not the suggested weight decay of the original work [9]. From a purely Bayesian perspective no weight decay should be considered to be the information is implicit in the prior; however, we found that without the extra decay SWAG was not able to obtain competitive results. As underlined in Section 4.1, not necessarily a better posterior approximation translates into better empirical results.

Appendix C.4. Definition of the Metrics

For regression datasets, we consider RMSE and MNLL. Consider a single datapoint $\boldsymbol{U}_i = (\boldsymbol{x}_i, \boldsymbol{y}_i)$, with $\boldsymbol{x}_i$ the input of the model and $\boldsymbol{y}_i$ the true corresponding output. The output of the model, for a single sample of parameters $\boldsymbol{\theta}_j$, is $\hat{\boldsymbol{y}}_{\boldsymbol{\theta}_j}(\boldsymbol{x}_i)$. RMSE is defined as $\frac{1}{N}\sum_{i=1}^{N}||\boldsymbol{y}_i - \mu(\boldsymbol{x}_i)||^2$, where $\mu(\boldsymbol{x}_i)$ is the empirical mean $\frac{1}{N_{MC}}\sum_{j=1}^{N_{MC}}\hat{\boldsymbol{y}}_{\boldsymbol{\theta}_j}(\boldsymbol{x}_i)$. MNLL is defined instead as $(\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}\log(2\pi\sigma_i^2) + \frac{1}{2}\frac{||\boldsymbol{y}_i-\mu(\boldsymbol{x}_i)||^2}{\sigma_i^2}\right)$, where σ_i^2 is the empirical variance.

For classification datasets, we consider ACC,MNLL and entropy. Consider a single datapoint $\boldsymbol{U}_i = (\boldsymbol{x}_i, y_i)$, with $\boldsymbol{x}_i$ the input of the model and y_i the true corresponding label. The output of the model, for a single sample of parameters $\boldsymbol{\theta}_j$, is the N_{cl} vector $\mathbf{p}_{\boldsymbol{\theta}_j}(\boldsymbol{x}_i)$. The averaged probability vector for a single sample is $\mathbf{p}(\boldsymbol{x}_i) = \frac{1}{N_{MC}}\sum_{i=1}^{N_{MC}}\mathbf{p}_{\boldsymbol{\theta}_j}(\boldsymbol{x}_i)$.ACC is defined as $\frac{1}{N}\sum_{i=1}^{N}1(\arg\max\mathbf{p}(\boldsymbol{x}_i) = y_i)$. MNLL is computed as $\frac{1}{N}\sum_{i=1}^{N}\log\left(\mathbf{p}_{y_i}(\boldsymbol{x}_i)\right)$. Entropy, as stated in the main text, is instead computed according to $\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{k=1}^{N_{cl}}\mathbf{p}_k(\boldsymbol{x}_i)\log(\mathbf{p}_k(\boldsymbol{x}_i))\right)$.

References

1. Welling, M.; Teh, Y.W. Bayesian learning via stochastic gradient Langevin dynamics. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), Bellevue, WA, USA, 28 June–2 July 2011; pp. 681–688.
2. Ahn, S.; Korattikara, A.; Welling, M. Bayesian posterior sampling via stochastic gradient Fisher scoring. *arXiv* **2012**, arXiv:1206.6380.
3. Patterson, S.; Teh, Y.W. Stochastic Gradient Riemannian Langevin Dynamics on the Probability Simplex. In *Advances in Neural Information Processing Systems 26*; Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2013; pp. 3102–3110.
4. Chen, T.; Fox, E.; Guestrin, C. Stochastic gradient hamiltonian monte carlo. In Proceedings of the International Conference on Machine Learning, Bejing, China, 22–24 June 2014; pp. 1683–1691.
5. Ma, Y.A.; Chen, T.; Fox, E. A complete recipe for stochastic gradient MCMC. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 2917–2925.
6. Draxler, F.; Veschgini, K.; Salmhofer, M.; Hamprecht, F.A. Essentially no barriers in neural network energy landscape. *arXiv* **2018**, arXiv:1803.00885.
7. Garipov, T.; Izmailov, P.; Podoprikhin, D.; Vetrov, D.; Wilson, A.G. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018.
8. Chaudhari, P.; Soatto, S. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In Proceedings of the 2018 Information Theory and Applications Workshop (ITA), San Diego, CA, USA, 11–16 February 2018; pp. 1–10.
9. Maddox, W.J.; Izmailov, P.; Garipov, T.; Vetrov, D.P.; Wilson, A.G. A simple baseline for bayesian uncertainty in deep learning. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 13132–13143.
10. Mandt, S.; Hoffman, M.D.; Blei, D.M. Stochastic gradient descent as approximate bayesian inference. *J. Mach. Learn. Res.* **2017**, *18*, 4873–4907.
11. Springenberg, J.T.; Klein, A.; Falkner, S.; Hutter, F. Bayesian optimization with robust Bayesian neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 4134–4142.

12. Gal, Y.; Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the International Conference on Machine Learning, ICML, New York, NY, USA, 19–24 June 2016; pp. 1050–1059.
13. Bishop, C.M. *Pattern Recognition and Machine Learning*, 1st ed.; 2006. corr. 2nd printing 2011 ed.; Springer: Berlin/Heidelberg, Germany, 2006.
14. Chen, C.; Carlson, D.; Gan, Z.; Li, C.; Carin, L. Bridging the gap between stochastic gradient MCMC and stochastic optimization. In Proceedings of the Artificial Intelligence and Statistics, Cadiz, Spain, 9–11 May 2016; pp. 1051–1060.
15. Gardiner, C.W. *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*, 3rd ed.; Springer Series in Synergetics; Springer: Berlin/Heidelberg, Germany, 2004; Volume 13.
16. Kushner, H.; Yin, G. *Stochastic Approximation and Recursive Algorithms and Applications*; Stochastic Modelling and Applied Probability; Springer: New York, NY, USA, 2003.
17. Ljung, L.; Pflug, G.; Walk, H. *Stochastic Approximation and Optimization of Random Systems*; Birkhauser Verlag: Basel, Switzerland, 1992.
18. Kloeden, P.E.; Platen, E. *Numerical Solution of Stochastic Differential Equations*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013; Volume 23.
19. Neal, R.M. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*; CRC Press: Boca Raton, FL, USA, 2011; Volume 2, p. 2.
20. Girolami, M.; Calderhead, B. Riemann manifold langevin and hamiltonian monte carlo methods. *J. R. Stat. Soc. Ser. B (Stat. Methodol.)* **2011**, *73*, 123–214. [CrossRef]
21. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
22. Vollmer, S.J.; Zygalakis, K.C.; Teh, Y.W. Exploration of the (non-) asymptotic bias and variance of stochastic gradient Langevin dynamics. *J. Mach. Learn. Res.* **2016**, *17*, 5504–5548.
23. Saxe, A.M.; Bansal, Y.; Dapello, J.; Advani, M.; Kolchinsky, A.; Tracey, B.D.; Cox, D.D. On the information bottleneck theory of deep learning. *J. Stat. Mech. Theory Exp.* **2019**, *2019*, 124020. [CrossRef]
24. Zhu, Z.; Wu, J.; Yu, B.; Wu, L.; Ma, J. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from minima and regularization effects. *arXiv* **2018**, arXiv:1803.00195.
25. Scott, W.A. Maximum likelihood estimation using the empirical fisher information matrix. *J. Stat. Comput. Simul.* **2002**, *72*, 599–611. [CrossRef]
26. Dua, D.; Graff, C. UCI Machine Learning Repository. 2017. Available online: https://archive.ics.uci.edu/ml/index.php (accessed on 25 October 2021)
27. LeCun, Y.; Cortes, C. MNIST Handwritten Digit Database. 2010. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 25 October 2021)
28. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
29. Krizhevsky, A.; Nair, V.; Hinton, G. CIFAR-10 (Canadian Institute for Advanced Research). Available online: http://www.cs.toronto.edu/~kriz/cifar.html (accessed on 25 October 2021)
30. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
31. Guo, C.; Pleiss, G.; Sun, Y.; Weinberger, K.Q. On calibration of modern neural networks. *arXiv* **2017**, arXiv:1706.04599.