On the Convergence of Locally Adaptive and Scalable Diffusion-Based Sampling Methods for Deep Bayesian Neural Network Posteriors

Tim Rensmeyer
Helmut-Schmidt University
Hamburg

Oliver Niggemann
Helmut-Schmidt University
Hamburg

Abstract

Achieving robust uncertainty quantification for deep neural networks represents an important requirement in many real-world applications of deep learning such as medical imaging where it is necessary to assess the reliability of a neural network's prediction. Bayesian neural networks are a promising approach for modeling uncertainties in deep neural networks. Unfortunately, generating samples from the posterior distribution of neural networks is a major challenge. One significant advance in that direction would be the incorporation of adaptive step sizes, similar to modern neural network optimizers, into Monte Carlo Markov chain sampling algorithms without significantly increasing computational demand. Over the past years, several papers have introduced sampling algorithms with corresponding theorems stating that they achieve this property. In this paper, we demonstrate that these methods can have a substantial bias in the distribution they sample, even in the limit of vanishing step sizes and at full batch size. Furthermore, for most of the algorithms, we show that convergence to the correct distribution can be restored with a simple fix at the cost of increasing computational demand.

1 INTRODUCTION

Markov Chain methods have established themselves as valuable tools in statistics. One of their main attractions is their ability to accurately sample high-dimensional probability distributions [Luengo et al., 2020]. Due to this property, they can enable the estimation of expectation values with respect to such high-dimensional distributions. This is of particular interest in the field of Bayesian Neural Networks (BNNs), where all the weights and biases of a neural network are treated as the free parameters of a stochastic model over which Bayesian inference is performed [Goan and Fookes, 2020].

Since the beginning of the deep learning revolution, neural networks have achieved impressive results in many domains such as vision [Krizhevsky et al., 2012, He et al., 2015, Redmon et al., 2015, Ronneberger et al., 2015], language [Vaswani et al., 2017, Devlin et al., 2019, Brown et al., 2020] and Markovian decision processes [Mnih et al., 2015, Silver et al., 2018, Lillicrap et al., 2015]. However, despite their successes the adoption of deep learning models in safety-critical domains has been slow, in large part due to a lack of interpretability and limited robustness to outliers [Huang et al., 2020]. Another desirable property in safety-critical applications is a reliable uncertainty quantification for neural network predictions to assess which neural network predictions are reliable and which are likely to be wrong [Abdar et al., 2021].

Monte Carlo Markov Chain (MCMC) based Bayesian Neural Networks (BNNs) are a promising approach to improve robustness to outliers while also achieving high-quality uncertainty quantification [Carbone et al., 2020, Goan and Fookes, 2020, Yao et al., 2019]. Such methods work by simulating a stochastic process over the neural network parameters, which is known to converge in distribution to the true posterior for appropriate learning rate schedules. Unfortunately, sampling from the Bayesian posterior over neural network parameters via Markov chains represents a challenging task.

Two key components of modern (classical) neural network optimizers that enable the scalability of neural networks to deep architectures and large datasets are

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

the usage of adaptive step sizes for each weight and mini-batching [Ruder, 2017]. Adaptive step sizes are often necessary to compensate for the variability of how much each weight at each step of the training algorithm influences the neural network predictions and without them, training times for deeper neural network architectures are often unpractically large. Furthermore, the use of mini-batching is necessary, because it is also highly inefficient to evaluate the neural network on each sample in the dataset at each iteration of the training algorithm if the dataset is very large.

Motivated in large part by the challenges associated with applying MCMC-based approaches to BNNs, a general framework for using adaptive step sizes in Markov chain methods was developed by Ma et al. [2015]. Furthermore, different algorithms were developed, that could utilize utilize mini-batching [Welling and Teh, 2011, Chen et al., 2014]. However, combining both approaches is challenging.

Adaptive step-size methods work by introducing a Riemannian metric over the space of neural network parameters, that uses geometric information of the posterior density to increase step-sizes of weights that currently only have a weak influence on the predictions of the model on the dataset. The methods that use mini-batching work because it is possible to get an unbiased estimate of the drift of the Markov chain by sampling only a mini-batch, because the drift is proportional to the gradient of the log posterior w.r.t. the neural network parameters. However, it is generally not possible to get an unbiased estimate for the Riemannian metric this way because it will be (approximately) **inversely** proportional to the gradient of the log posterior (see Section 2 for examples).

Nonetheless, several algorithms have been introduced that attempt to combine adaptive step sizes with mini-batching. These methods keep a record of gradient estimates from previous time steps which can improve the estimation of the metric at the current time step, especially for small step sizes.

In addition, by proceeding this way the algorithms gained another beneficial property. When adaptive step sizes are introduced into MCMC methods via a Riemannian metric, another term has to be added to the drift of the Markov chain. The evaluation of this term will typically require the calculation of the second-order derivatives of the neural network predictions w. r. t. the neural network parameters. This approximately doubles the size of the computational graph and consequently also approximately doubles the computational time and memory requirements of each optimization step.

But for those algorithms that estimate the metric based on a record of previous time steps, it was shown that this term is very small. Consequently, it can be neglected without changing the stationary distributions that the algorithms converge to by much.

However, this property appears to directly contradict some principles from the ergodic theory of Stochastic Differential Equations (SDEs).

Given the apparent contradiction, the main research question of this paper is, whether those algorithms actually converge to the desired distribution.

The main contributions of this paper are the following:

- We demonstrate both theoretically as well as empirically that all of the algorithms discussed in this paper converge to distributions that deviate significantly from the true posterior.
- We show that the reason why in most of the algorithms it appears that the second-order correction term can be neglected is that due to a mathematical mistake, this term is significantly downscaled by a fixed factor.
- We show that for those algorithms, convergence to the right distributions can be restored by simply upscaling the second-order terms by the appropriate amount.

It is important to note here, that **the convergence issues are not a consequence of neglecting the second-order term**. They already exist in the full algorithms when this term is included. In fact, as far as we can tell, the conclusion that the stationary distributions of those algorithms do not change by much when the term is neglected is correct. But this is only because in the full algorithms the second-order term is already so much smaller than what it should be, that the full algorithms have an almost equally wrong stationary distribution as the approximate version of the algorithms where the term was dropped completely.

2 BACKGROUND: BAYESIAN PREDICTIVE MODELING

Throughout this paper, we make the conventions that bold case symbols represent vector-valued quantities and $\boldsymbol{\theta}$ is a vector containing all the free parameters of a predictive model e.g. the weights and biases of a neural network.

The main difference between frequentist and Bayesian models is that for Bayesian models the free parameters are modeled probabilistically. The starting point for Bayesian predictive models is a parametric probability distribution $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$ that models the relationship between an input variable $\boldsymbol{x}$ for example the history of a stock price and an output variable e.g. the future development of that price. Furthermore, Bayesian predictive models require a prior probability density $p(\boldsymbol{\theta})$, that expresses preexisting knowledge of which parameters are more likely to result in a good model of the relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$. This preexisting knowledge is then refined via a training dataset $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_M, \boldsymbol{y}_M)\}$ into the posterior density by using Bayes rule:

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

If the samples from the dataset are independent, $p(\mathcal{D}|\boldsymbol{\theta})$ can be simplified to $\Pi_{i=1}^{M} p(\boldsymbol{y}_i|\boldsymbol{x}_i, \boldsymbol{\theta})p(\boldsymbol{x}_i)$. A prediction on a new data point $\boldsymbol{x}^*, \boldsymbol{y}^*$ can then be made via

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) = \int p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta} \tag{2}$$

$$= \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta})]. \tag{3}$$

While this expectation w. r. t. to posterior density is almost always intractable for high-dimensional models such as neural networks, it can be evaluated through the law of large numbers as

$$\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta})] \approx \frac{1}{N}\sum_{k=1}^{N} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}_k) \tag{4}$$

where $\boldsymbol{\theta}_k, k \in \{1, ..., N\}$ are independent samples from the posterior distribution. These samples can be generated by simulating a stochastic process on the parameter space of the model, which is known to converge in distribution to the posterior. A short overview of relevant processes is given in the following subsections. A more in-depth discussion of BNNs can be found in the work by Goan and Fookes [2020].

2.1 Stochastic Gradient Langevin Dynamics

One of the most popular algorithms for sampling the posterior distribution of neural networks is Stochastic Gradient Langevin Dynamics (SDLD) [Welling and Teh, 2011]. This algorithm is based on a time-discretized simulation of the Stochastic Differential Equation (SDE)

$$d\boldsymbol{\theta}_t = \frac{1}{2}\nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t)dt + d\boldsymbol{B}_t, \tag{5}$$

where $u(\boldsymbol{\theta}_t) = \log(p(\boldsymbol{\theta}_t|\mathcal{D}))$ and $d\boldsymbol{B}_t$ are the increments of a Brownian motion. These diffusions provably converge to the distribution $p(\boldsymbol{\theta}|\mathcal{D})$. Unfortunately, objective functions such as $u(\boldsymbol{\theta}_t)$ will oftentimes have pathological curvature for deeper neural network architectures characterized by vanishing gradients for many of the neural network parameters. This can slow down the convergence of both classical gradient descent-based optimization and MCMC sampling methods for deep neural networks to the point where no convergence can be achieved in a practical amount of time. Modern neural network optimizers get around this problem by employing an individual adaptive step size for each parameter which is roughly inversely proportional to the current magnitude of its derivative [Ruder, 2017]. A major breakthrough for deep neural network-focused MCMC methods was the introduction of Stochastic Gradient Riemannian Langevin Dynamics (SGRLD) [Patterson and Teh, 2013] which can use adaptive step sizes similar to modern neural network optimizers by introducing a Riemannian metric $G(\boldsymbol{\theta})$ over the space of neural network parameters. The resulting stochastic differential equation is

$$d\boldsymbol{\theta}_t = \frac{1}{2}G(\boldsymbol{\theta}_t)\nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t)dt + \frac{1}{2}\boldsymbol{\Gamma}(\boldsymbol{\theta}_t)dt + G(\boldsymbol{\theta}_t)^{\frac{1}{2}}d\boldsymbol{B}_t, \tag{6}$$

which still converges to the posterior but allows for adaptive step sizes via the metric $G(\boldsymbol{\theta})$. One example metric is the diagonal metric

$$G(\boldsymbol{\theta}) = diag\left(\frac{1}{\lambda + \sqrt{\nabla_{\boldsymbol{\theta}} u(\boldsymbol{\theta}) \odot \nabla_{\boldsymbol{\theta}} u(\boldsymbol{\theta})}}\right) \tag{7}$$

where λ is a stability constant, $\odot$ is the element-wise product and the square root and division are applied element-wise. The drawback of this new equation lies in the introduction of a new term

$$\boldsymbol{\Gamma}_i(\boldsymbol{\theta}) := \sum_j \frac{\partial G_{i,j}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_j}. \tag{8}$$

Because the metric will typically contain the gradient of the log posterior since it is supposed to increase step sizes when those gradients get small, the evaluation of $\boldsymbol{\Gamma}$ usually requires the computation of second-order derivatives of the log posterior with respect to the parameters. This effectively doubles the size of the computational graph and hence also the required computation time and GPU memory of each time step. Later a very general framework for incorporating adaptive step sizes in sampling methods was discovered [Ma et al., 2015]. Unfortunately, it also required the computationally intensive calculation of additional terms each time step. Nonetheless, several algorithms where such terms are not required or the neglection causes only a small change in the distribution the algorithm converges to have recently been proposed.

2.2 Proposed Alternatives

In this subsection, we will give a short overview of the algorithms, which incorporate adaptive step sizes without a costly correction term and for which the authors claim, that the process will converge to almost the correct distribution. We discuss here only the full-batch variants of the proposed algorithms. The mini-batch versions can be found in the original works. By far the most popular one is the Preconditioned Stochastic Gradient Langevin Dynamics (PSGLD) algorithm [Li et al., 2016] which at the time of writing has over 350 citations. The algorithm is given by the updates

$$\boldsymbol{V}_t = \alpha \boldsymbol{V}_{t-\epsilon} + (1-\alpha)\nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t) \odot \nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta_t}) \tag{9}$$

$$G_t = diag\left(\frac{1}{\lambda + \sqrt{\boldsymbol{V}_t}}\right) \tag{10}$$

$$\boldsymbol{\theta}_{t+\epsilon} = \boldsymbol{\theta}_t + \frac{\epsilon}{2}\left(G_t \nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t) + \boldsymbol{\Gamma}(\boldsymbol{\theta}_t)\right) + G_t^{\frac{1}{2}} \boldsymbol{\mathcal{N}}_t(0, \epsilon\mathbb{I}). \tag{11}$$

Here, ϵ is the step size, α is a hyperparameter typically slightly smaller than one, λ is a stability constant close to zero, $\mathbb{I}$ is the identity matrix and $\boldsymbol{\mathcal{N}}_t(0, \epsilon\mathbb{I}), t \in \mathbb{N}_0$ are independent Gaussian random variables $\sim \boldsymbol{N}(0, \epsilon\mathbb{I})$. Again the square root and division are applied element-wise.

Two other similar algorithms were recently introduced called SGRLD in Monge metric and SGRLD in Shampoo metric [Yu et al., 2023]. SGRLD in Monge metric is similar to PSGLD but changes the definition of $\boldsymbol{V}_t$ and G_t to

$$\boldsymbol{V}_t = \alpha \boldsymbol{V}_{t-\epsilon} + (1-\alpha)\nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t), \tag{12}$$

$$G_t = \mathbb{I} - \frac{\beta^2}{1 + \beta^2 ||\boldsymbol{V}_t||^2} \boldsymbol{V}_t \boldsymbol{V}_t^T, \tag{13}$$

where β^2 is a hyperparameter. The construction of the Shampoo metric is quite involved but luckily in the case of one-dimensional distributions, the resulting algorithm coincides with PSGLD where the stability constant λ is set to zero. Since this is the only case we will analyze in the next section we do not introduce the general algorithm here. The interested reader can find the general definition in the work by Yu et al. [2023].

In all three algorithms, the authors claim that dropping $\boldsymbol{\Gamma}(\boldsymbol{\theta}_t)$ will not change the distribution that the process converges to significantly.

Finally, there is the Adam SGLD algorithm [Sehwan Kim and Liang, 2022] which is defined via

$$\boldsymbol{m}_t = \beta \boldsymbol{m}_{t-\epsilon} + (1-\beta)\nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t), \tag{14}$$

$$\boldsymbol{\theta}_{t+\epsilon} = \boldsymbol{\theta}_t + \frac{\epsilon}{2}(\nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t) + a G_t \boldsymbol{m}_t) + \boldsymbol{\mathcal{N}}_t(0, \epsilon\mathbb{I}) \tag{15}$$

where G_t coincides with the definition of the PSGLD algorithm, β is another hyperparameter that is typically slightly smaller than one, and $a > 0$ is a hyperparameter controlling the additional drift term. Again the authors claim, that this is a suitable algorithm for sampling $p(\boldsymbol{\theta}|\mathcal{D})$.

3 CONVERGENCE ANALYSIS OF SAMPLING ALGORITHMS WITH ADAPTIVE STEP SIZES

The main working principle of all SDE-based sampling approaches is that they converge to a stationary density $\pi(\boldsymbol{\theta})$ which should be equal to the target density one tries to sample, e.g. the posterior over the neural network parameters in this case. This density is invariant under the SDE [Ma et al., 2015], e.g. if at $t = 0$ the variable $\boldsymbol{\theta}_t$ is distributed according to π, then it will have this distribution for all $t > 0$. In fact, it is not difficult to see that any distribution a process can converge to has to be a stationary distribution of the process. For a given SDE, π can in principle be determined via solving the associated Fokker-Planck partial differential equation Øksendal [1987]. A fundamental property of the SDEs under consideration here is an ergodic property, that enables the computation of expectations of a function f w. r. t. the stationary density as a time average of the function [Borodin and Salminen, 2002]

$$\mathbb{E}_\pi\left[f(\boldsymbol{\theta})\right] = \lim_{T\to\infty} \frac{1}{T}\int_{t=0}^{T} f(\boldsymbol{\theta}_t)dt. \tag{16}$$

When convergence speed and the effects of time discretization were investigated in the papers for the proposed algorithms, it was done using this property by analyzing the discrepancy of $\mathbb{E}_\pi\left[f(\boldsymbol{\theta})\right]$ and the time discretized analog of $\frac{1}{T}\int_{t=0}^{T} f(\boldsymbol{\theta}_t)dt$ in the large T limit.

Unfortunately, this ergodic property also seems to indicate that incorporating adaptive step sizes into the sampling procedure without a correction term should at best be very difficult and at worst simply not possible. This can be seen by applying the ergodic property to the indicator function

$$\mathbb{1}_U(\boldsymbol{\theta}) = \left\{ \begin{array}{ll} 1, & \text{for } \boldsymbol{\theta} \in U \\ 0, & \text{otherwise} \end{array} \right\}. \tag{17}$$

This yields

$$\mathbb{P}_\pi(U) = \mathbb{E}_\pi\left[\mathbb{1}_U(\boldsymbol{\theta})\right] = \lim_{t\to\infty} \frac{1}{T}\int_{t=0}^{T} \mathbb{1}_U(\boldsymbol{\theta}_t)dt \tag{18}$$

- when sampling from the stationary density the probability of sampling a parameter vector $\boldsymbol{\theta}$ from some region U of parameter space is simply given by the fraction of time the process $\boldsymbol{\theta}_t$ spends in that region. If the step size is selectively doubled in some region U of parameter space the process will roughly spend only half as much time in that region as before and hence the stationary density changes.

3.1 The neccessity of the $\mathbf{\Gamma}$ Term in Riemannian Langevin Dynamics

The majority of the proposed sampling methods we discuss here are based on Riemannian Langevin dynamics. In all of those examples, it is claimed that the pathological $\mathbf{\Gamma}$ term can be neglected since the overall term is supposedly very small. But based on the previous analysis this seems strange. In fact, it will be shown here that it is unlikely that the $\mathbf{\Gamma}$ term can ever be neglected for any sensible adaptive metric $G(\boldsymbol{\theta})$ without significantly changing the stationary density. To see this one simply has to look at the one-dimensional case. In the case of a one dimensional SDE $d\theta_t = \mu(\theta_t)dt + \sigma(\theta_t)dB_t$ the stationary density $\pi(\theta)$ is known to be

$$\pi(\theta) = \frac{Z}{\sigma(\theta)^2} \exp\left(2\int^{\theta} \frac{\mu(x)}{\sigma(x)^2}dx\right), \tag{19}$$

where Z is a normalization constant [Borodin and Salminen, 2002].

Applying this formula to one-dimensional Riemannian Langevin Dynamics under neglection of Γ:

$$d\theta_t = \frac{1}{2}G(\theta_t)\frac{d}{d\theta_t}u(\theta_t)dt + G(\theta_t)^{\frac{1}{2}}dB_t, \tag{20}$$

results in (see Appendix A.1 for a derivation)

$$\pi(\theta) = \frac{Z}{G(\theta)}p(\theta|\mathcal{D}). \tag{21}$$

Unless the metric G has almost the same value for all θ where $p(\theta|\mathcal{D})$ is reasonably large, the stationary density π will significantly deviate from $p(\theta|\mathcal{D})$. However, the main function of the metric is to increase step sizes when the parameter derivatives get small. Since the derivatives vanish at local maxima of the log-likelihood it therefore seems unlikely that this approach can ever work without significantly changing the stationary distribution.

The mistake in the mentioned algorithms lies in the following:

All the sampling methods based on Riemannian Langevin dynamics where the $\mathbf{\Gamma}$ term was claimed to be negligible are not actually time discretizations of Riemannian Langevin dynamics. The main problem is, that in the limit of vanishing step sizes, the $\mathbf{\Gamma}$ term is much smaller than what would be dictated by Riemannian Langevin dynamics.

All the algorithms involve a valid metric $G(\boldsymbol{\theta}_t) = F(\boldsymbol{h}(\boldsymbol{\theta}_t))$ for which $\mathbf{\Gamma}(\boldsymbol{\theta}_t)$ is not negligible. However, instead of using the term $\boldsymbol{h}(\boldsymbol{\theta}_t)$ itself, which carries the entire dependence on $\boldsymbol{\theta}_t$ it is replaced by its exponentially moving average $\boldsymbol{V}_t = \alpha\boldsymbol{V}_{t-\epsilon} + (1-\alpha)\boldsymbol{h}(\boldsymbol{\theta}_t)$. For example in PSGLD the function $\boldsymbol{h}(\boldsymbol{\theta})$ is given by $\nabla_{\boldsymbol{\theta}}u(\boldsymbol{\theta}) \odot \nabla_{\boldsymbol{\theta}}u(\boldsymbol{\theta})$. From now on we will denote with $G^{\alpha,t}$ and $\mathbf{\Gamma}^{\alpha,t}$ the corresponding quantities where $\boldsymbol{h}(\boldsymbol{\theta}_t)$ has been replaced by its exponential moving average.

Now the only explicit dependence of the metric $G^{\alpha,t}$ on the current parameter vector $\boldsymbol{\theta}_t$ comes from the $(1-\alpha)\boldsymbol{h}(\boldsymbol{\theta}_t)$ term in the exponential moving average, where $(1-\alpha)$ will be close to zero for typical values of α. Consequently, by introducing this exponential moving average almost all explicit dependence of $G^{\alpha,t}$ on the current parameter vector $\boldsymbol{\theta}_t$ is lost, even though implicitly $G^{\alpha,t}$ still depends strongly $\boldsymbol{\theta}_t$ due to the high correlation of $\boldsymbol{h}(\boldsymbol{\theta}_t)$ with previous values $\boldsymbol{h}(\boldsymbol{\theta}_{t-\epsilon}), \boldsymbol{h}(\boldsymbol{\theta}_{t-2\epsilon}), \dots$.

However, because $\mathbf{\Gamma}^{\alpha,t}$ is constructed from the derivatives of $G^{\alpha,t}$ w.r.t. the current parameters $\boldsymbol{\theta}_t$ and the explicit dependence of $G^{\alpha,t}$ on the current parameter vector is very weak, they find that $\mathbf{\Gamma}^{\alpha,t} = \mathcal{O}(1-\alpha)$ and is hence negligible for α sufficiently close to 1. But proceeding like this is simply not something that you can do in Riemannian Langevin dynamics where the metric is supposed to depend only on the current parameter vector. The main problem is, that in the limit of vanishing step sizes ϵ we have $\boldsymbol{V}_t \to \boldsymbol{h}(\boldsymbol{\theta}_t)$ and hence $G^{\alpha,t} \to G(\boldsymbol{\theta}_t)$. But $\frac{\partial}{\partial\boldsymbol{\theta}_{t,j}}\boldsymbol{V}_{t,l} = (1-\alpha)\frac{\partial}{\partial\boldsymbol{\theta}_{t,j}}\boldsymbol{h}_l(\boldsymbol{\theta}_t)$ and consequently

$$\mathbf{\Gamma}_i^{\alpha,t} := \sum_j \frac{\partial G_{i,j}^{\alpha,t}}{\partial\boldsymbol{\theta}_{t,j}} = \sum_{j,l} \frac{\partial\boldsymbol{V}_{t,l}}{\partial\boldsymbol{\theta}_{t,j}}\frac{\partial G_{i,j}^{\alpha,t}}{\partial\boldsymbol{V}_{t,l}} \tag{22}$$

$$= \sum_{j,l}(1-\alpha)\frac{\partial\boldsymbol{h}_l(\boldsymbol{\theta}_t)}{\partial\boldsymbol{\theta}_{t,j}}\frac{\partial G_{i,j}^{\alpha,t}}{\partial\boldsymbol{V}_{t,l}} \tag{23}$$

$$\to \sum_{j,l}(1-\alpha)\frac{\partial\boldsymbol{h}_l(\boldsymbol{\theta}_t)}{\partial\boldsymbol{\theta}_{t,j}}\frac{\partial G_{i,j}}{\partial\boldsymbol{h}_l(\boldsymbol{\theta}_t)} \tag{24}$$

$$= \sum_j(1-\alpha)\frac{\partial G_{i,j}(\boldsymbol{\theta}_t)}{\partial\boldsymbol{\theta}_{t,j}} = (1-\alpha)\mathbf{\Gamma}_i(\boldsymbol{\theta}_t). \tag{25}$$

Hence, in the limit of vanishing step sizes, the algorithm turns into the Itô diffusion

$$d\boldsymbol{\theta}_t = \frac{1}{2}G(\boldsymbol{\theta}_t)\nabla_{\boldsymbol{\theta}_t}u(\boldsymbol{\theta}_t)dt + \frac{1-\alpha}{2}\mathbf{\Gamma}(\boldsymbol{\theta}_t)dt + G(\boldsymbol{\theta}_t)^{\frac{1}{2}}d\boldsymbol{B}_t. \tag{26}$$

As is obvious, $\mathbf{\Gamma}(\boldsymbol{\theta}_t)$ is downscaled by a factor of $(1-\alpha)$ from what Riemannian Langevin dynamics would prescribe it to be.

The effect this has on the stationary density the process samples can be evaluated in closed form in one dimension (see Appendix A.1 for the derivation):

$$\pi(\theta) = Zp(\theta|\mathcal{D})G(\theta)^{-\alpha}. \tag{27}$$

The case where Γ was dropped entirely corresponds to the limit $\alpha \to 1$. While in some papers the convergence of the process with the dropped $\mathbf{\Gamma}$ was explicitly proven to be close to the stationary distribution of the process where this term is included, unfortunately, it was taken for granted that this stationary distribution is equal to the posterior while clearly the stationary distribution can deviate substantially from the posterior. In particular, since a typical metric will become very large for vanishing gradients, this will lead to the stationary density becoming very small around local **maxima** of the target distribution where the gradients necessarily need to vanish.

3.2 Convergence of Adam SGLD

The Adam SGLD algorithm does not fit into the framework of Riemannian Langevin dynamics and consequently, its convergence has to be analyzed separately. The algorithm was given by the updates:

$$\boldsymbol{V}_t = \alpha \boldsymbol{V}_{t-\epsilon} + (1-\alpha)\nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t) \odot \nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t) \tag{28}$$

$$G^{\alpha,t} = diag\left(\frac{1}{\lambda + \sqrt{\boldsymbol{V}_t}}\right) \tag{29}$$

$$\boldsymbol{m}_t = \beta \boldsymbol{m}_{t-\epsilon} + (1-\beta)\nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t), \tag{30}$$

$$\boldsymbol{\theta}_{t+\epsilon} = \boldsymbol{\theta}_t + \frac{\epsilon}{2}(\nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t) + aG^{\alpha,t}\boldsymbol{m}_t) + \boldsymbol{\mathcal{N}}_t(0, \epsilon\mathbb{I}). \tag{31}$$

The authors claim convergence to the posterior without any restrictions on the hyperparameter a as long as the step size ϵ is sufficiently small.
This however can not be the case, since for sufficiently large a and sufficiently small ϵ this algorithm just turns into regular Adam with step size $\frac{a\cdot\epsilon}{2}$.
In the limit $\epsilon \to 0$ we again have

$$G^{\alpha,t} \to diag\left(\frac{1}{\lambda + \sqrt{\nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t) \odot \nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t)}}\right) =: G(\boldsymbol{\theta}_t) \tag{32}$$

as well as

$$\boldsymbol{m}_t \to \nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t). \tag{33}$$

Consequently, the process turns into the Itô diffusion

$$d\boldsymbol{\theta}_t = \frac{1}{2}\left(1 + aG(\boldsymbol{\theta}_t)\right)\nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t)dt + d\boldsymbol{B}_t. \tag{34}$$

Evaluating the stationary density in the one-dimensional case yields (see Appendix A.2 for the derivation):

$$\pi(\theta) = Zp(\theta|\mathcal{D}) \exp\left(\int^{\theta} aG(x)\frac{d}{dx}\log p(x|\mathcal{D})dx\right). \tag{35}$$

In this case, the additional factor has the opposite effect and artificially sharpens the density as will be illustrated in the example of the next section.

3.3 Fixing the Convergence Problems of the Riemannian Langevin Dynamics-based Algorithms

As our analysis of the algorithms based on Riemannian Langevin dynamics demonstrates, the source of the convergence problems lies in the fact that the $\boldsymbol{\Gamma}$ term erroneously gets downscaled by a factor of $1-\alpha$ in the limit of vanishing step-sizes. Luckily, this can easily be fixed by rescaling this term by $1/(1-\alpha)$. A corrected version of the PSGLD algorithm is given by Algorithm 1. Note, that the stability constant was additionally moved inside the square root for numerical stability.

Algorithm 1 A corrected version of the PSGLD algorithm with correct stationary distribution for small step sizes ϵ

```
Inputs: $\epsilon$, $\alpha$, $\lambda$, $\boldsymbol{\theta}_0$, $N$
Outputs: $\{\boldsymbol{\theta}_{i\epsilon}\}_{i=1:N}$
Initialize: $\boldsymbol{V}_0 = \mathbf{0}$, $t = 0$
for $i = 1$ to $N$ do
    $t = t + \epsilon$
    $\boldsymbol{V}_t = \alpha \boldsymbol{V}_{t-\epsilon} + (1-\alpha)\nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t) \odot \nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t)$
    $G_t = diag\left(\frac{1}{\sqrt{\lambda^2 + \boldsymbol{V}_t}}\right)$
    $\boldsymbol{\mu}_t = \frac{1}{2}\left(G_t \nabla_{\boldsymbol{\theta}_t} u(\boldsymbol{\theta}_t) + \frac{\boldsymbol{\Gamma}(\boldsymbol{\theta}_t)}{1-\alpha}\right)$
    $\boldsymbol{\theta}_{t+\epsilon} = \boldsymbol{\theta}_t + \epsilon\boldsymbol{\mu}_t + G_t^{\frac{1}{2}}\boldsymbol{\mathcal{N}}_t(0, \epsilon\mathbb{I})$.
end for
```

4 EMPIRICAL EVALUATION

To demonstrate the correctness of the results derived in the previous section, we estimate the stationary densities of the discussed algorithms empirically in this section.
For PSGLD we use the official TensorFlow implementation linked on the first author's GitHub page. For SGRLD in Monge and Shampoo metric, we use the code made publically available by the authors. Because the authors don't provide an implementation where the $\boldsymbol{\Gamma}$ term is included, we run the experiments of those algorithms with the $\boldsymbol{\Gamma}$ term completely dropped. Note however, that in the PSGLD code in TensorFlow, the $\boldsymbol{\Gamma}$ term is included and hence **we run the full PSGLD algorithm with the $\boldsymbol{\Gamma}$ term included.**
Because we could not find the source code of Adam SGLD provided by the authors, we use our own reimplementation of that algorithm. The code for running all the experiments can be found at `https://github.com/TimRensmeyer/Convergence-Experiments/`.

4.1 The Experimental Setup

In this experiment, we empirically verify the derived results of the previous section, by attempting to sample from a standard normal distribution $p(\theta|\mathcal{D}) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\theta^2}{2}\right)$. In all experiments, we set $\alpha = 0.9$ and all stability constants to 10^{-1}.
Because there is a slight deviation in the PSGLD algorithm, where in the paper the gradients of the log prior are not included in the preconditioning but in the implementation they are, we simply assume a flat prior $p(\theta) = 1$ in this case.
The derivations for the stationary densities mentioned in the following are straightforward but somewhat tedious with the results from the last section and can be found in Appendix A.3. Utilizing the results from the previous section and determining the normalization constants via numerical integration, we get as the stationary density for PSGLD:

$$\pi(\theta) = \frac{1.1238}{\sqrt{2\pi}} exp(-\frac{1}{2}\theta^2)(10^{-1} + |\theta|)^{0.9} \tag{36}$$

and for SGRLD in Shampoo metric

$$\pi(\theta) = \frac{1.253}{\sqrt{2\pi}} exp(-\frac{1}{2}\theta^2)|\theta|. \tag{37}$$

For SGRLD in Monge metric we set $\beta = 1$ and get

$$\pi(\theta) = \frac{0.5}{\sqrt{2\pi}} exp(-\frac{1}{2}\theta^2)\left(1 - \frac{\theta^2}{1+\theta^2}\right)^{-1}. \tag{38}$$

Finally setting $a = 1$ and $\beta = 0.5$ in the Adam SGLD algorithm yields the stationary density

$$\pi(\theta) = \frac{2.0495}{\sqrt{2\pi}} exp(-\frac{1}{2}\theta^2) exp(-|\theta|)(|\theta| + 10^{-1})^{0.1}. \tag{39}$$

For $x \in [a, b)$ for a sufficiently small interval $[a, b)$ the stationary density for each algorithm can be estimated with the ergodic property via

$$\pi(x) \approx \frac{\mathbb{P}_\pi([a,b))}{b-a} = \frac{1}{b-a} \lim_{T\to\infty} \frac{1}{T} \int_{t=0}^{T} \mathbb{1}_{[a,b)}(\theta_t)dt \tag{40}$$

$$\approx \frac{1}{b-a} \frac{1}{T_{max}} \int_{t=0}^{T_{max}} \mathbb{1}_{[a,b)}(\theta_t)dt, \tag{41}$$

where T_{max} is chosen sufficiently large. We use this estimator with intervals of width 0.1 to estimate the stationary density by running each algorithm for 10^7 steps at a step size of 10^{-4}.

4.2 Results on the Original Algorithms

Even though the empirical estimator for the stationary density is somewhat primitive, the results in Figure 1 align well with the predictions made for the stationary densities of the algorithms derived in the previous sections and very clearly do not coincide with the target density the algorithm is supposed to converge to.

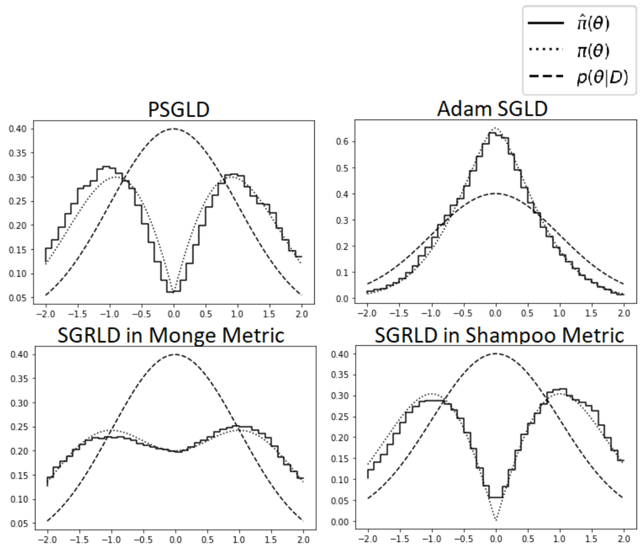

Figure 1: The stationary densities when attempting to sample from a standard normal distributed posterior with each of the algorithms. $p(\theta|\mathcal{D})$ denotes the target distribution, $\pi(\theta)$ are the stationary densities we derived in the previous sections and $\hat{\pi}(\theta)$ are the empirical estimates for the stationary densities.

4.3 Results for the Corrected PSGLD Algorithm

To demonstrate, that the proposed fix for the Riemannian Langevin dynamics-based algorithms works and that the source of the convergence problems illustrated in the previous subsection is indeed the downscaling of the $\mathbf{\Gamma}$ term, we estimate the stationary density of our proposed fix for PSGLD (Algorithm 1) on the same benchmark from the previous subsection with a step size of $2.5 \cdot 10^{-6}$ for $8 \cdot 10^8$ steps and $\lambda = 10^{-1}$. As can be seen in Figure 2, the fixed PSGLD algorithm now has the correct stationary distribution. The results of an additional experiment, where we estimate the stationary densities of the original and our fixed version of PSGLD with a decreasing step-size schedule can be found in Appendix B.

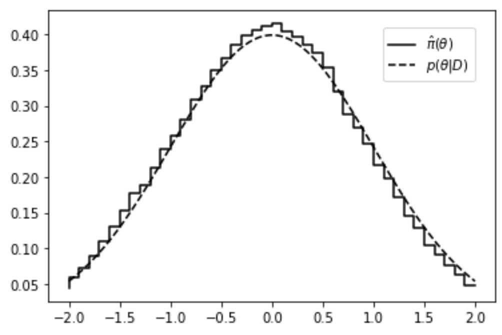

Figure 2: The estimated stationary density of the corrected PSGLD algorithm, where the $\mathbf{\Gamma}$ term has been rescaled by $(1-\alpha)^{-1}$. $p(\theta|\mathcal{D})$ denotes the target distribution which is a standard normal distribution and $\hat{\pi}(\theta)$ is the empirical estimate for the stationary density.

5 DISCUSSION AND OUTLOOK

For the algorithms under consideration here, we demonstrated both mathematically as well as empirically that they introduce a substantial bias in the sampled distribution to the point where the sampled distribution can have a deep local minimum at the global maximum of the target distribution. This directly disproves the theorems made by the respective authors about the convergence of their algorithms in their original works.
In particular, we directly contradict Theorem 1 and Corollary 2 in [Li et al., 2016] for PSGLD, Theorem 3.1 and 3.2 in [Yu et al., 2023] for SGRLD with Monge and Shampoo metric and Theorem 3.2 in Sehwan Kim and Liang [2022] for Adam SGLD.
Additionally, we showed that for PSGLD and SGRLD with Monge and Shampoo metric, even the full algorithm with the computationally costly $\mathbf{\Gamma}$ term still included already has the wrong stationary distribution because the $\mathbf{\Gamma}$ term is downscaled by a factor of $(1-\alpha)$ from what it should be. Importantly, the derivation in which we show this wrongful downscaling is valid for the case of arbitrary dimension. However, computing the quantitative effect of the downscaling for higher dimensional cases is challenging, due to the complexity of the associated Fokker-Planck equation. For this reason, we limited our further discussion to the one-dimensional case, which is sufficient for disproving the convergence claims for the original algorithms and demonstrating empirically that our derivation showing that the $\mathbf{\Gamma}$ term is downscaled is not based on a misunderstanding of the algorithms.

Due to the substantial bias of the algorithms discussed here, they clearly can not be considered sampling algorithms for the true Bayesian posterior distribution. Hence the main research question of this paper of whether the proposed algorithms converge to a distribution close to the true posterior can be conclusively answered with no they do not.
While they may still work well empirically as uncertainty quantification algorithms on some tasks, one should be aware of the biases these algorithms introduce to the sampled distribution.
Furthermore, as we showed, the convergence problems of PSGLD as well as SGRLD in Shampoo and Monge metric can actually be fixed by rescaling the $\mathbf{\Gamma}$ term by $\frac{1}{1-\alpha}$. The resulting algorithm then has the correct stationary distribution in the limit of vanishing step sizes. However, one of the apparent benefits of those algorithms, the fact that this term can be dropped completely, can not be upheld. Consequently, the updates of the corrected algorithms take roughly twice as long to compute when compared with the versions where the $\mathbf{\Gamma}$ term was dropped.

In regards to the question of what the consequences of using adaptive step sizes without a correction term are, we demonstrated that at least in one dimension, this leads to a change in the stationary distribution proportional to the inverse of the metric used. Further, we illustrated why, due to the ergodic properties of stochastic differential equations, it seems unlikely that adaptive step sizes similar to modern neural network optimizers can ever be incorporated into diffusion-based sampling methods without adding a computationally costly corrective drift term.

There are two related approaches to improving the convergence of diffusion-based sampling methods in the existing literature that we have not discussed so far in this paper.
The first one is a new approach by Lange et al. [2023] which uses the framework of Riemannian Langevin Dynamics to imitate the batch normalization operation. Unfortunately, they also follow the erroneous argument of Li et. al for dropping the corrective drift term. Hence, the algorithm can not be expected to converge to the correct distribution although the same fix of rescaling the corrective drift term as above can be applied. Furthermore, the resulting metric depends only on the current set of parameters and not their gradient. Due to this fact, it might be possible to calculate the $\mathbf{\Gamma}$ term more efficiently in this case, since no second-order derivatives w.r.t. the parameters would need to be calculated.
However, while this approach appears promising, it seems unlikely that an imitation of batch normalization alone will result in an improved convergence speed comparable to adaptive step sizes.

The second approach not discussed in this paper so far is the one introduced by Wenzel et al. [2020], who use adaptive step sizes which are not updated each time step but once every few thousand time steps. However, while this clearly would inhibit the deviation from the target distribution at finite step sizes, the limiting process one gets at vanishing step sizes is the same as the one that would result from updating the adaptive step sizes each time step. A natural question then arises, how much time there has to be between the updates of the adaptive step sizes in order for the sampled distribution to remain close to the target distribution? It appears unlikely that this time can be chosen small enough, that this procedure can be considered adapted to the local geometry around the current parameter vector. Hence it can probably not be used as a locally adaptive algorithm in any sense of the word. Wenzel et al. completely circumvent this issue by additionally using a single adaptive step size for each layer, which remains relatively constant along the entire trajectory of the process and hence is not very locally adaptive no matter how much time there is between updates of the adaptive step sizes.

Overall, it appears that the current approaches for making locally adaptive diffusion-based sampling methods less computationally demanding simply do not work and the ergodic theory of stochastic differential equations, as well as the results by Ma et al. [2015] make it seem very difficult to make any progress in that direction. If it is achievable at all, some new ideas are clearly needed.

6 ACKNOWLEDGEMENTS

This research as part of the project CoupltelT! is funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr which we gratefully acknowledge. dtec.bw is funded by the European Union – NextGenerationEU.

References

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76: 243–297, 2021. ISSN 1566-2535.

Andrei N Borodin and Paavo Salminen. *Handbook of Brownian motion: facts and formulae.* Springer, 2002.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, 2020.

Ginevra Carbone, Matthew Wicker, Luca Laurenti, Andrea Patane, Luca Bortolussi, and Guido Sanguinetti. Robustness of Bayesian neural networks to gradient-based attacks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2019.

Ethan Goan and Clinton Fookes. *Bayesian Neural Networks: An Introduction and Survey.* Case Studies in Applied Bayesian Data Science, 45–87, Springer International Publishing, 2020.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37:100270, 2020. ISSN 1574-0137.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

Susanna Lange, Wei Deng, Qiang Ye, and Guang Lin. Batch normalization preconditioning for stochastic gradient Langevin dynamics. *Journal of Machine Learning*, 2(1):65–82, 2023. ISSN 2790-2048.

Chunyuan Li, Changyou Chen, David E. Carlson, and Lawrence Carin. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1788–1794. AAAI Press, 2016.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

David Luengo, Luca Martino, Mónica F. Bugallo, Víctor Elvira, and Simo Särkkä. A survey of monte carlo methods for parameter estimation. *EURASIP Journal on Advances in Signal Processing*, 2020:1–62, 2020.

Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient mcmc. In *Advances in Neural Information Processing Systems*, volume 29, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. ISSN 00280836.

Bernt Øksendal. Stochastic differential equations : an introduction with applications. *Journal of the American Statistical Association*, 82:948, 1987.

Sam Patterson and Yee Whye Teh. Stochastic gradient Riemannian Langevin dynamics on the probability simplex. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2015.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2017.

Qifan Song Sehwan Kim and Faming Liang. Stochastic gradient Langevin dynamics with adaptive drifts. *Journal of Statistical Computation and Simulation*, 92(2):318–336, 2022.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

Florian Wenzel, Kevin Roth, Bastiaan S. Veeling, Jakub Swiatkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really? In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

J. Yao, W. Pan, S. Ghosh, and F. Doshi-Velez. Quality of uncertainty quantification for Bayesian neural network inference. In *Proceedings of the 36th International Conference on Machine Learning: Workshop on Uncertainty & Robustness in Deep Learning (ICML)*, 2019.

Hanlin Yu, Marcelo Hartmann, Bernardo Williams, and Arto Klami. Scalable stochastic gradient Riemannian Langevin dynamics in non-diagonal metrics. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Not Applicable]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Appendix

February 13, 2025

A Derivations

A.1 Derivation of the Stationary Density of Riemannian Langevin Dynamics with Downscaled Γ Term in One Dimension

Utilizing the general formula for the stationary density

$$\pi(\theta) = \frac{Z}{\sigma(\theta)^2} \exp\left(2\int^\theta \frac{\mu(x)}{\sigma(x)^2}dx\right),$$

of a one dimensional SDE $d\theta_t = \mu(\theta_t)dt + \sigma(\theta_t)dB_t$ and applying it to Riemannian Langevin dynamics with a downscaled Γ term

$$d\theta_t = \frac{1}{2}G(\theta_t)\frac{d}{d\theta_t}u(\theta_t)dt + \frac{1-\alpha}{2}\Gamma(\theta_t)dt + G(\theta_t)^{\frac{1}{2}}dB_t$$

yields

$$\begin{aligned}
\pi(\theta) &= \frac{Z}{G(\theta)} \exp\left(2\int^\theta \frac{G(x)\frac{d}{dx}u(x) + (1-\alpha)\Gamma(x)}{2G(x)}dx\right), \\
&= \frac{Z}{G(\theta)} \exp\left(\int^\theta \frac{G(x)\frac{d}{dx}\log p(x|\mathcal{D}) + (1-\alpha)\frac{d}{dx}G(x)}{G(x)}dx\right) \\
&= \frac{Z}{G(\theta)} \exp\left(\int^\theta \frac{d}{dx}\log p(x|\mathcal{D}) + \frac{d}{dx}\log(G(x)^{1-\alpha})dx\right) \\
&= \frac{Z}{G(\theta)} \exp\left(\log p(\theta|\mathcal{D}) + \log(G(\theta)^{1-\alpha})\right) \\
&= \frac{Zp(\theta|\mathcal{D})G(\theta)^{1-\alpha}}{G(\theta)} = Zp(\theta|\mathcal{D})G(\theta)^{-\alpha}.
\end{aligned}$$

The case in which the Γ term was dropped completely corresponds to the limit $\alpha \to 1$.

A.2 Derivation of the Stationary Density of Adam SGLD in One Dimension

As was discussed, in the limit of vanishing step sizes the Adam SGLD algorithm turns into the SDE

$$d\boldsymbol{\theta}_t = \frac{1}{2}\left(1 + aG(\boldsymbol{\theta}_t)\right)\nabla_{\boldsymbol{\theta}_t}u(\boldsymbol{\theta}_t)dt + d\boldsymbol{B}_t.$$

Using again the formula for for the stationary density for one-dimensional SDEs it follows that

$$\begin{aligned}
\pi(\theta) &= Z\exp\left(2\int^\theta \frac{1}{2}\left(1 + aG(x)\right)\frac{d}{dx}u(x)dx\right) \\
&= Z\exp\left(\int^\theta \left(1 + aG(x)\right)\frac{d}{dx}\log p(x|\mathcal{D})dx\right) \\
&= Zp(\theta|\mathcal{D})\exp\left(\int^\theta aG(x)\frac{d}{dx}\log p(x|\mathcal{D})dx\right).
\end{aligned}$$

A.3 Derivations of the Stationary Densities for the Experiments

Based on the results from Section 3, we expect the stationary density of all algorithms with the exception of Adam SGLD to be of the form

$$\pi(\theta) = \frac{Z}{\sqrt{2\pi}} \exp\left(-\frac{\theta^2}{2}\right) G(\theta)^{-\alpha}.$$

where $G(\theta)$ is the metric one gets in the limit $\epsilon \to 0$ and $\alpha = 1$ if the Γ term was dropped. For PSGLD and SGRLD in Shampoo metric $G(\theta) = \frac{1}{\lambda+\sqrt{u(\theta)^2}} = \frac{1}{\lambda+|\theta|}$ where $\lambda = 0$ for the Shampoo metric. For SGRLD in Monge metric we get $G(\theta) = 1 - \frac{\beta^2 u(\theta)^2}{1+\beta^2 u(\theta)^2} = 1 - \frac{\beta^2\theta^2}{1+\beta^2\theta^2}$.

Finally, for the Adam SGLD algorithm the stationary density is

$$\pi(\theta) = Zp(\theta|D) \exp\left(\int^{\theta} aG(x)\frac{d}{dx}\ln p(x|D)dx\right)$$

$$= Zp(\theta|D) \exp\left(a\int^{\theta} \frac{-x}{\lambda+|x|}dx\right)$$

$$= \frac{Z}{\sqrt{2\pi}} \exp\left(-\frac{\theta^2}{2}\right) \exp\left(-a|\theta|\right)|(|\theta|+\lambda)^{a\lambda}$$

The normalization constants Z were all determined via numerical integration of the densities.

B Experiments with Decreasing Learning Rate Schedule

For the main manuscript, we did the convergence experiments with a small but constant step size to investigate the stationary distribution in the limit of vanishing step sizes. Technically, the convergence results in the original work were claimed in the context of the time average when using a learning rate schedule ϵ_n that satisifies $\sum_{n=1}^{\infty} \epsilon_n = \infty$ and $\sum_{n=1}^{\infty} \epsilon_n^2 < \infty$. While this is equivalent due to the ergodic property we redid the experiments for the original and fixed version of the PSGLD algorithm here. For this we chose a learning rate schedule $\epsilon_n = \frac{0.1}{n^{0.55}}$. We ran the experiments for $4 \cdot 10^8$ steps. Due to the noise at larger step sizes with such a learning rate schedule, the convergence is worse than running at a small but constant step size at this time horizon.

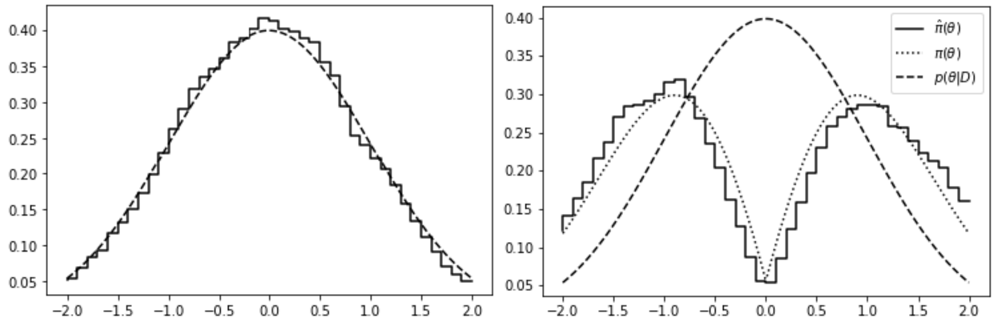

Figure 3: The stationary densities when attempting to sample from a standard normal distributed posterior with each of the algorithms and a decreasing learning rate schedule. $p(\theta|\mathcal{D})$ denotes the target distribution, $\pi(\theta)$ are the stationary densities we derived in the main manuscript and $\hat{\pi}(\theta)$ are the empirical estimates for the stationary densities. On the right the original PSGLD algorithm and on the left the fixed version we proposed in this paper