Journal of Machine Learning Research 21 (2020) 1-33 Submitted 4/19; Revised 1/20; Published 7/20

Posterior sampling strategies based on discretized stochastic differential equations for machine learning applications

Frederik Heber FREDERIK.HEBER@GMAIL.COM
Alan Turing Institute,
96 Euston Rd,
Kings Cross,
London NW1 2DB, UK

Žofia Trsťanová Z.TRSTANOVA@CRITEO.COM
Criteo Labs,
32 Rue Blanche,
75009 Paris, France

Benedict Leimkuhler B.LEIMKUHLER@ED.AC.UK
School of Mathematics,
University of Edinburgh,
Edinburgh EH9 2NX, UK

Editor: Francois Caron

Abstract

With the advent of GPU-assisted hardware and maturing high-efficiency software platforms such as TensorFlow and PyTorch, Bayesian posterior sampling for neural networks becomes plausible. In this article we discuss Bayesian parametrization in machine learning based on Markov Chain Monte Carlo methods, specifically discretized stochastic differential equations such as Langevin dynamics and extended system methods in which an ensemble of walkers is employed to enhance sampling. We provide a glimpse of the potential of the sampling-intensive approach by studying (and visualizing) the loss landscape of a neural network applied to the MNIST data set. Moreover, we investigate how the sampling efficiency itself can be significantly enhanced through an ensemble quasi-Newton preconditioning method. This article accompanies the release of a new TensorFlow software package, the Thermodynamic Analytics ToolkIt, which is used in the computational experiments.

Keywords: Bayesian posterior sampling, Markov Chain Monte Carlo, Neural network training, software platforms for machine learning, ensemble sampling strategies

1. Introduction

The fundamental role of neural networks (NNs) is readily apparent from their widespread use in machine learning in applications such as natural language processing (Young et al. (2017)), social network analysis (Gómez et al. (2015)), medical diagnosis (Bone et al. (2015); Johnson et al. (2018)), vision systems (Sebe et al. (2005)), and robotic path planning (LeCun et al. (2015)). The greatest success of these models lies in their flexibility, their ability to represent complex, nonlinear relationships in high-dimensional data sets, and the availability of frameworks that allow NNs to be implemented on rapidly evolving GPU platforms (Krizhevsky

et al. (2012); He et al. (2016)). The industrial appetite for deep learning has led to very rapid expansion of the subject in recent years, although, as pointed out by Dunson (2018), at times the mathematical and theoretical understanding of these methods has been swept aside in the rush to advance the methodology.

The potential impact on society of machine learning algorithms demands that their exposition and use be subject to the highest standards of clarity, ease of interpretation, and uncertainty quantification. Typical NN training seeks to optimize the parameters of the network (biases and weights) under the constraint that the training data set is well approximated (Hardt et al. (2016); Goodfellow et al. (2014)). In the Bayesian setting, the parameters of a neural network are defined by the observations, but only in the probabilistic sense, thus specific parameter values are only realized as modes or means of the associated distribution, which can require substantial computation. Bayesian approaches based on exploration of the posterior probability distribution have been discussed throughout the development of neural networks (MacKay (1992); Neal (1992); Hinton and Van Camp (1993); Barber and Bishop (1998)), and underpin much of the work in this field, but they are less commonly implemented in practice for "big data" applications due to legitimate concerns about efficiency (Welling and Teh (2011)). While the idea of sampling (or partially sampling) the posterior of large scale neural networks is not new, improvements in computers continually render this goal more plausible. For a recent discussion, see the PhD thesis of Gal (2016), which again champions the use of a (Bayesian) statistical framework, mentioning among other aims the prospect for meaningful uncertainty quantification in deep neural networks. Posterior sampling has the potential for broad impact in several research areas related to NN construction, including: (i) the relationship between network architecture and parameterization efficiency (Safran and Shamir (2016); Livni et al. (2014); Haeffele and Vidal (2017); Soltanolkotabi et al. (2019)), (ii) the visualization of the loss manifold and the parameterization process (see Draxler et al. (2018); Im et al. (2017); Li et al. (2018); Goodfellow et al. (2014)), and (iii) the assessment of the generalization capability of networks (see Dinh et al. (2017); Hochreiter and Schmidhuber (1997); Kawaguchi et al. (2020); Hoffer et al. (2017)).

We have recently developed the Thermodynamic Analytics ToolkIt (TATi), a python framework whose purpose is to facilitate the sampling of the posterior parameter distribution of automated machine learning systems with a balance of ease of use and computational efficiency, by leveraging highly optimised computational procedures within *TensorFlow*.[1] Although still at an early stage in its development, this software package makes possible the exploration of sampling-based approaches in the high-dimensional setting. This article provides the motivation for TATi by illustrating that it facilitates analysis of the loss manifold in a way which would be impossible using standard optimization methods.

The standard approach to Bayesian sampling in high dimensions relies on Markov Chain Monte Carlo methods, but these can be difficult to scale to large system size. In the context of deep learning, we use the term *large* to refer both to large data set size (which translates into expensive likelihood computations) and to large numbers of parameters (usually the consequence of adopting a *deep learning* paradigm). We base our software on discretized stochastic differential equations which offer a reliable and accurate means of computing

1. Installation of TATI is as simple as `pip install tati` from the unix command line; the software includes a readable user guide and programmer's manual, see `https://pypi.org/project/tati/`.

Markov Chain Monte Carlo (MCMC) sampling paths while providing rigorous results with statistical error bounds (Leimkuhler et al. (2015)). An underlying assumption in using software like this is that the exploration is limited to a bounded region of the parameter space by some structural features of the problem and/or its regularization. The methodology we describe could also be extended to include an explicit localization scheme to implement a quasi-stationary (spatially restricted) distribution Collet et al. (2012), although we do not discuss this here. Our approach is instead to rely on the inclusion of a temperature parameter which allows tuning the sampling process to retain compatibility with commonly used optimization methods (and consistent with other recent approaches based on "cold posteriors" for machine learning applications Wenzel et al. (2020); Baldock and Marzari (2019)).

As an indication of the possible scope for practical posterior sampling in large scale machine learning, we note that Markov Chain Monte Carlo methods like those implemented in TATi have been successfully deployed in the more mature setting of very large scale molecular simulations in computational sciences for applications in physics, chemistry, material science and biology, with variables numbering in the millions or even billions (Klein and Shinoda (2008); Zhao et al. (2013); Kadau et al. (2004)). The goal in molecular dynamics is similarly the sampling of a target probability distribution, although the distribution typically arises from semi-empirical modelling of interactions among atoms of the substances of interest. The scientific communities in molecular sciences are developing highly efficient "enhanced sampling" procedures to tackle the computational difficulties of large scale sampling (see the surveys (Elber (2016); Bernardi et al. (2015)) for some examples of the wide variety of methods being used in biomolecular applications). In analogy with molecular dynamics, the focus on posterior sampling makes possible the elucidation of reduced descriptions of neural networks through concepts such as free energy calculation (Leliévre et al. (2010)) and transition path sampling (Bolhuis et al. (2002)). Although we reserve detailed study of these ideas for future work, the fundamental tool in their construction and practical implementation is the ability to compute sample sequences efficiently and reliably using MCMC paths. Further potential benefits of the posterior sampling approach lie in the fact that it offers a means of high-dimensional uncertainty quantification through standard statistical methodology (Smith (2013); Gal (2016)).

We favor schemes based on underdamped Langevin dynamics (with canonical momentum variables). Such methods have excellent properties in terms of accuracy of statistical averages and convergence rates (Leimkuhler et al. (2015); Leliévre et al. (2010)). A variety of other methods are available which are also based on second order dynamics (Chen et al. (2014); Patterson and Teh (2013); Shang et al. (2015); Matthews and Weare (2018)). One of the purposes of the TATi software is to provide a relatively simple mechanism for the implementation and evaluation of sampling strategies of statistical physics in machine learning applications. [2]

2. While *TensorFlow*'s graph programming structure is well explained and motivated by its developers and provides efficient execution on GPUs, it is not straightforward for numerical analysts, statisticians and statistical physicists to modify in order to test their methods. TATi therefore uses a simplified interface structure that allows algorithms to be coded directly in pure *Python* and then linked to *TensorFlow* for efficient calculation of gradients for arbitrary *TensorFlow* models. As a consequence, building and testing a new sampling scheme in TATi requires little knowledge of the underpinnings of *TensorFlow*.

Another goal we have had in mind in this study is to gain insight into the loss landscape (given by the log of the posterior density) so as to better understand its structure vis a vis the performance of parameterization algorithms. The loss is not convex in general ((LeCun et al., 1998, sect. 4)), and its corrugated ("metastable") structure has been compared to models of spin-glasses (Choromanska et al. (2015)): there is a band of minima close to the global minimum as lower bound and whose multitude diminishes exponentially for larger loss values. The availability of an efficient sampling scheme gives a means of better understanding the loss landscape and its relation to the network (and properties of the data set). We illustrate some of the potential for such studies in Section 3 of this article, where we examine the loss landscape of an MNIST classification problem.

Finally, we note that the statistical perspective underpins many optimization schemes in current use in machine learning, such as the Stochastic Gradient Descent (SGD) method. The stochasticity enters into this method through the subsampling of the dataset at every iteration of the optimization algorithm, due to 'minibatching'. Several stochastic optimisation methods have been proposed which aim to improve the computational efficiency or reduce generalisation error, e. g., RMSprop (Hinton et al. (2014)), AdaGrad (Duchi et al. (2011)), Adam (Kingma and Ba (2014)), entropy-SGD (Chaudhari et al. (2019)). Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh (2011)), and these, similarly, can be given a statistical (sampling) interpretation. Indeed, when its stepsize is held fixed and under simplifying assumptions on the character of the gradient noise, it can be viewed as a first-order discretization of overdamped Langevin dynamics. We discuss SGD and SGLD in Sec 2, in order to motivate SDE sampling methods.

The remainder of this paper is structured as follows: Section 2 provides an overview of MCMC, Langevin dynamics schemes and other sampling strategies and the concepts from numerical error analysis that underpin our approach. We also outline there the Ensemble Quasi-Newton method (Leimkuhler et al. (2018)). Subsequently, in Section 3, we discuss the accuracy of various schemes as well as their convergence behavior in the context of neural network posterior sampling. We also elaborate on how the approach we describe can be used to handle moderate-dimensional sampling on the MNIST dataset. The last part of this section consists of a demonstration of the convergence acceleration of the EQN sampler in the MNIST application. The numerical results presented in the paper represent a proof-of-concept for the utility of the posterior sampling paradigm and the TATi software which implements it.

2. Markov Chain Monte Carlo Methods and Stochastic Differential Equations

In this section, we provide a brief introduction to the MCMC methods we have used for sampling high dimensional distributions. These methods include schemes based on discretization of stochastic differential equations, especially Langevin dynamics. We discuss in some detail the construction of numerical schemes in order to control the finite time stepsize bias. Moreover, we also describe an ensemble quasi-Newton method implemented in TATi that adaptively rescales the dynamics to enhance sampling of poorly conditioned target posteriors.

Assume that we are given a dataset $\mathcal{D} = (\mathcal{X}, \mathcal{Y}) = \{(x_i, y_i)\}_{i=1}^M$ comprising inputs and outputs of an unknown functional relationship. We also assume we are given a neural network defined by a vector of parameters $\theta \in \Omega \subset \mathbb{R}^N$ which acts on an input $x \in \mathcal{X}$ to produce output $f(x, \theta)$. The goal of training the network is then typically formulated as solving an optimization problem over the parameters given the dataset:

$$\min_{\theta \in \mathbb{R}^N} L(\theta, \mathcal{D}), \tag{1}$$

where the function $L(\theta, \mathcal{D})$ is the total loss function associated to the dataset $\mathcal{D}$, defined by

$$L(\theta, \mathcal{D}) = \sum_{(x,y)\in\mathcal{D}} l(f(x, \theta), y) = \sum_{i=1}^{M} l(f(x_i, \theta), y_i). \tag{2}$$

The loss function $l(\hat{y}, y)$ depends on the metric choice, for example the squared error, logarithmic loss, cross entropy loss, etc (Rosasco et al. (2004)). The total loss is in general not a convex function of the parameters θ even though l may be convex as a function of $y, \hat{y}$. When the loss landscape is rough there are likely to be many local minima and flattened intermediate states in the loss manifold $L(\theta, \mathcal{D})$, as well as many saddles, see Baldi and Hornik (1989); Choromanska et al. (2015).

The parametrization procedure is based on optimization algorithms, which generate a sequence of parameter vectors $\theta_0, \theta_1, \ldots, \theta_k, \ldots$, converging to a (local) minimizer of (1) as $k \to \infty$. The basic optimization algorithm is the Gradient Descent (GD), which uses the negative gradient to update the parameter values, i. e.,

$$\theta_{k+1} = \theta_k - \varepsilon \nabla L(\theta_k), \tag{3}$$

where ε is the stepsize (or learning rate), which is either constant or may be varied during the computation. GD converges for a convex function and for a smooth non-convex total loss function it converges to the nearest local minimum (Nocedal and Wright (1999)).

Gradient calculations are the primary computational burden when training neural networks, i. e., the computational cost of a gradient can be taken as a reasonable measure of computational work. As the total loss L implicitly depends on the whole dataset, one natural idea that reduces the computational cost is to exploit the redundancy in the dataset by estimating the gradient of the average loss from a subset of the data, that is, to replace the gradient in each parameterization step by the approximation

$$\nabla \widetilde{L}(\theta) \approx \tfrac{M}{m} \sum_{i \in S_k} \nabla l(f(x_i, \theta), y_i).$$

Where S_k represents a randomized data subset (re-randomized throughout the training process) of dimension m, this method, which has many variants, is referred to as SGD.

We are interested in the Bayesian inference formulation, where θ is the parameter vector and we wish to sample from the posterior distribution $\pi(\theta \mid \mathcal{D})$ of the parameters given a dataset of size $M > 0$,

$$\pi(\theta \mid \mathcal{D}) \propto \pi_0(\theta) \prod_{i=1}^{M} \pi((x_i, y_i) \mid \theta),$$

with prior probability density $\pi_0(\theta)$, and likelihood $\pi((x, y) \mid \theta)$.

Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh (2011)) generates a step based on a subset of the data and injects additional noise. The additive noise creates a controllable stochastic model with known ergodic properties and it improves the numerical stability and convergence properties of the training algorithm. For additional discussion of these points see Leimkuhler et al. (2019), where small injected noise has been shown to substantially accelerate and improve robustness of the training process for neural network models.

The SGLD parameter update is using a sequence of stepsizes $\{\varepsilon_n\}$ and reads

$$\theta_{k+1} = \theta_k - \varepsilon_k \nabla \tilde{L}(\theta_k) + \sqrt{2\beta^{-1}\varepsilon_k} G_k, \tag{4}$$

with

$$\nabla \tilde{L}(\theta) = -\beta^{-1} \nabla \log\big(\pi_0(\theta)\big) - \beta^{-1} \frac{M}{m} \prod_{i \in S_k} \nabla \log\big(\pi((x_i, y_i) \mid \theta)\big)$$

where $\beta > 0$ is a constant (in physics, it would be associated with reciprocal temperature) and again S_k is a random subset of indices $1, \ldots, M$ of size m. The original algorithm uses a diminishing stepsize sequence, however, Vollmer et al. (2016) showed that fixing the stepsize has the same efficiency, up to a constant.

Under the assumption that the gradient noise is uncorrelated and identically distributed from step to step, it is easy to demonstrate that SGLD with fixed stepsize is an Euler-Maruyama (first order) discretization of overdamped Langevin dynamics (Welling and Teh (2011)):

$$d\theta = -\nabla \tilde{L}(\theta) dt + \sqrt{2\beta^{-1}} dW_t. \tag{5}$$

This creates a natural starting point for our approach: we simply change the perspective to focus on the sampling of the posterior distribution, rather than the identification of its mode.

In case a full gradient is used, dynamics (5) preserves the Boltzmann distribution with measure proportional to an exponential function of the negative loss:

$$\pi(d\theta) \propto \mathrm{e}^{-\beta L(\theta)} d\theta. \tag{6}$$

Given a generic process providing samples θ_k asymptotically distributed with respect to a defined target measure π, it is possible to use sampling paths to estimate integrals with respect to π. Define the finite time average of a C^∞ function φ by

$$\widehat{\varphi}(K) := \frac{1}{K+1} \sum_{k=0}^{K} \varphi\left(\theta_k\right).$$

For an ergodic process, we have

$$\lim_{K \to \infty} \widehat{\varphi}(K) = \int_\Omega \varphi(\theta)\, \pi(d\theta). \tag{7}$$

The convergence rate of the limit above is given by the Central Limit Theorem (CLT). Given a generic process generating samples θ_k from a target distribution with density π, the variance of an observable φ behaves, asymptotically for large K, as

$$\operatorname{var} \widehat{\varphi}(K) \sim \tau_\varphi \operatorname{var} \varphi / K$$

where τ_φ is the *integrated autocorrelation time*, see Goodman and Weare (2010, sect. 3). τ_φ can be viewed as a measure of the redundancy of the sampled values or the number of steps until the sampled values of φ decorrelate, thus $\tau_\varphi = 1$ is optimal, i. e., immediately stepping from one independent state to the next. The Integrated Autocorrelation Time (IAT) τ_φ can also be calculated as

$$\tau_\varphi = 1 + 2\sum_{1}^{\infty} \frac{C_\varphi(k)}{C_\varphi(0)} \quad \text{with} \quad C_\varphi(k) = \operatorname{cov}[\varphi\big(\theta_k\big), \varphi\big(\theta_0\big)], \tag{8}$$

where the covariance is averaged over the initial condition.

In practice, the process of discretization of the SDE introduces asymptotic bias, which is controlled by the stepsize, however it is nonetheless possible and useful to compute the IAT in such a case to describe the convergence to the asymptotically perturbed equilibrium distribution.

Langevin Dynamics. Sampling from (6) is one of the main challenges of computational statistical physics. There are three popular alternative approaches to sample from (6): discretization of continuous stochastic differential equations (SDEs), Metropolis-Hastings based algorithms and deterministic dynamics, as well as combinations among the three groups.

Langevin dynamics is an extended version of (5):

$$\begin{cases} d\theta_t = M^{-1} p_t \, dt, \\ dp_t = -\nabla L(\theta_t) \, dt - \gamma M^{-1} p_t \, dt + \sigma \, dW_t, \end{cases} \tag{9}$$

where p is the momentum variable and $\gamma > 0$ is the friction. The fluctuation-dissipation relation $\sigma^2 = \frac{2\gamma}{\beta}$ ensures that the extended (canonical) distribution with density

$$\pi_{\text{ext}} \propto \exp\left[-\beta \left(p^T M^{-1} p/2\right)\right] \pi(\theta)$$

is preserved; the target distribution is recovered by marginalization.

Whereas the momentum is a physical variable in statistical mechanics, it is introduced as an artificial auxiliary variable in the machine learning application. M in (9) is a positive definite mass matrix, which can in many cases be taken to the identity matrix.

The discretization of stochastic dynamics introduces bias in the invariant distribution (see below, for discussion). Although the bias can be removed through the incorporation of a Metropolis-Hastings (MH) step, such methods do not always scale well with the dimension Beskos and Stuart (2009); Robert (2015); and the MH test is often neglected in practice. For example the "unadjusted Langevin algorithm" of Durmus and Moulines (2019) tolerates the presence of small stepsize-dependent bias, in order to obtain faster convergence (reduced asymptotic variance or lower integrated autocorrelation time for observables of interest) and better overall efficiency.

Systematic design of schemes. The mathematical foundation for the construction of discretization schemes for (9) by splitting of the generator of the dynamics is now well understood (Leimkuhler et al. (2015)). Although different choices can be made, the usual

starting point is an additive decomposition of the generator of dynamics (9) into three operators $\mathcal{L} = A + B + O$, where

$$A := M^{-1}p \cdot \nabla_\theta, \qquad B := -\nabla L(\theta) \cdot \nabla_p, \qquad O := -\gamma M^{-1}p \cdot \nabla_p + \sigma \Delta_p \,, \tag{10}$$

The main idea is that each of these sub-dynamics can be resolved exactly in the weak (distributional) sense. Note that the ("O" step) dynamics

$$dp_t = -\gamma M^{-1} p_t \, dt + \sigma \, dW_t, \tag{11}$$

has an analytical solution

$$p_t = \alpha_t p_0 + \sigma \int_0^t \alpha_{t-s} dW_s \,, \quad \alpha_t := \mathrm{e}^{-\gamma M^{-1} t}. \tag{12}$$

A Lie-Trotter splitting of the elementary evolution generated by A, B, and O provides six possible first-order splitting schemes of the general form

$$P_\varepsilon^{Z,Y,X} = \mathrm{e}^{\varepsilon Z} \, \mathrm{e}^{\varepsilon Y} \, \mathrm{e}^{\varepsilon X},$$

with all possible permutations (Z, Y, X) of (A, B, O), and second-order splitting schemes are then obtained by a Strang splitting of the elementary evolutions generated by A, B, and O.

Using the notation for the three operators of the sub-dynamics (10), we define the following updates which will be combined in the full scheme for the discretization of the Langevin dynamics with stepsize ε:

$$\begin{aligned} &A_\varepsilon : \theta \to \theta + \varepsilon p, \\ &B_\varepsilon : p \to p - \varepsilon \nabla L(\theta), \\ &O_\varepsilon : p \to \alpha p + \sqrt{\beta^{-1}(1-\alpha^2)} R, \end{aligned} \tag{13}$$

with $\alpha = \mathrm{e}^{-\gamma\varepsilon}$ and $R \sim \mathcal{N}(0,1)$. A numerical methods is easily specified by a string such as 'ABO'. This is an instance of the Geometric Langevin Algorithm (Bou-Rabee and Owhadi (2010)). We also consider symmetric compositions of several basic steps. The 'BAOAB' scheme is in this notation $B_{\varepsilon/2} A_{\varepsilon/2} O_\varepsilon A_{\varepsilon/2} B_{\varepsilon/2}$. This method can be written out in detail as a step from (θ_k, p_k) to (θ_{k+1}, p_{k+1}) as follows:

$$\begin{aligned} p_{k+1/2} &= p_k - \tfrac{\varepsilon}{2} \nabla L(\theta_k), \\ \theta_{k+1/2} &= \theta_k + \tfrac{\varepsilon}{2} M^{-1} p_{k+1/2}, \\ \tilde{p}_{k+1/2} &= \alpha p_{k+1/2} + \sqrt{\beta^{-1}(1-\alpha^2)} R_k, \\ \theta_{k+1} &= \theta_{k+1/2} + \tfrac{\varepsilon}{2} M^{-1} \tilde{p}_{k+1/2}, \\ p_{k+1} &= \tilde{p}_{k+1/2} - \tfrac{\varepsilon}{2} \nabla L(\theta_{k+1}). \end{aligned}$$

Given the sequence of samples (θ_k, p_k) determined using such a method, we can approximate expected values of a C^∞ function of θ and p using the standard estimator based on trajectory averages

$$\widehat{\varphi}_K = \frac{1}{K+1} \sum_{k=0}^{K} \varphi\left(\theta_k, p_k\right). \tag{14}$$

It can be shown in certain cases that these discrete averages converge to an ensemble average,

$$\lim_{K\to\infty} \widehat{\varphi}_K = \int_\Omega \varphi(\theta,p)\; d\pi_\varepsilon\,(\theta,p) = \mathbb{E}_{\pi_\varepsilon}(\varphi)\,,$$

where π_ε represents the stationary density of the (biased) discrete process. Under specific assumptions on the splitting scheme (Leimkuhler et al. (2015)), an expansion may be made in the time stepsize of the invariant measure of the splitting scheme which guarantees that the error in an ergodic approximation of an observable average is bounded relative to ε^q, where q depends on the detailed structure of the numerical method. Thus

$$\mathbb{E}_{\pi_\varepsilon}(\varphi) = \mathbb{E}_{\pi_{\text{ext}}}(\varphi) + O(\varepsilon^q).$$

Schemes such as "ABO" can be shown to be first order ($q = 1$), whereas BAOAB and ABOBA are second order. Delicate cancellations imply that BAOAB can exhibit an unexpected fourth order of accuracy in the "high friction" limit ($\gamma \to \infty$) when the target is sampling of configurational (θ-dependent) quantities. The latter method also has remarkable features with respect configurational averages in harmonic systems and near-harmonic systems Leimkuhler and Matthews (2015).

All explicit Langevin integrators are subject to stability restrictions which require that the product of stepsize and the frequency of the fastest oscillatory mode is bounded.

Hybrid Monte Carlo Hybrid Monte-Carlo, also called Hamiltonian Monte Carlo, is a MCMC method based on the Metropolis-Hastings algorithm that allows one to sample directly from the target distribution π. Starting from an initial position (θ, p), the momenta are re-sampled from $\mathcal{N}\left(0, M\beta^{-1}\right)$ and a proposal $(\tilde{\theta}, \tilde{p})$ is obtained by evaluating K_{HMC} times the Verlet method, i.e., $B_{\varepsilon/2}A_\varepsilon B_{\varepsilon/2}$. The proposal is then accepted with a probability given by the Metropolis ratio:

$$\min\left(1, \mathrm{e}^{-\beta\left(H(\tilde{\theta},\tilde{p})-H(\theta,p)\right)}\right)\,, \tag{15}$$

where the energy H is given by $H(\theta,p) = L(\theta,\mathcal{D}) + \frac{1}{2}p^T M^{-1} p$. In case the proposal is rejected, the state is reset to the starting point (θ, p). The number of steps K_{HMC} is often randomized. The parameterization of this method depends on the trade-off between the larger time stepsizes ε and the number of steps K_{HMC}, implying higher rejection rates for the proposals and smaller stepsizes leading to potentially slow exploration of the parameter space. To our knowledge, there is currently no extension of HMC to the case of noisy gradients which retains the exact sampling property.

Multiple walkers: the ensemble quasi-Newton method We next comment on more exotic sampling schemes which can give higher sampling efficiency. One such scheme is the preconditioned method developed in Leimkuhler et al. (2018) which we refer to as the "ensemble quasi-Newton" (EQN) method. The idea of this scheme is easily motivated by reference to a simple harmonic model problem in two space dimensions in which the stiffness (or frequency of oscillation) is very high in one direction and not in the other. The stepsize for stable simulation using a Langevin dynamics strategy will be determined by the high frequency term meaning that the exploration rate suffers in the slowest direction. Effectively

the integrated autocorrelation time in the "slow" direction is large compared to the timestep of dynamics.

In the harmonic case it is easy to rescale the dynamical system in such a way that sampling proceeds rapidly. In the more complicated setting of nonlinear systems in multiple dimensions, the idea that a few directions may restrict the progress of the sampling still has merit, but we no longer have direct access to the underpinning frequencies. The idea is to determine this rescaling adaptively and dynamically during simulation. There are several ways to do this in practice (see LeCun et al. (1991) for an early related work on using Hessian information to speed up convergence of GD).

As a simple illustration consider a Gausssian mixture model as shown in Figure 1. Here, a poor choice of initial condition has led to a naive sampling path (here generated using Euler-Maruyama and shown in white) getting stuck for a long time in a poorly scaled basin. Eventually the sampler proceeds to the deeper (and more relevant) basin, but not before a great deal of useless computational effort has been expended. Note that the stepsize threshold for stable integration of the SDE is inversely proportional to the frequency of the largest normal mode of oscillation in the local basin, thus it is not possible to simply 'step over' the irrelevant intermediate region by using large stepsizes. This process mimics the behavior of many optimization schemes as well.

In the paper of Leimkuhler et al. (2018) modified dynamics ("Ensemble Quasi-Newton" or EQN for short) is based on construction of the covariance matrix of the walker collection. This matrix is computed during simulation to determine the appropriate dynamical rescaling. The implementation is somewhat involved, thus making it an excellent demonstration of TATi's versatility and robustness.

We next briefly describe the EQN scheme; for more detail, the reader is referred to Leimkuhler et al. (2018) (and, in particular, the *Python* code referenced within it). Suppose we have L walkers (replicas). Denote by θ_i the position vector of the ith walker and by p_i the corresponding momentum vector, each of which is a vector in $\mathbb{R}^N$, where N is the number of parameters of the network. We assume that the mass matrix of the underlying system is the identity matrix, for simplicity. The equations of motion for the ith walker then take the form

$$d\theta_i = B_i(\theta)p_i dt, \tag{16}$$

$$dp_i = -B_i(\theta)^T \nabla L(\theta_i)dt + \text{div}\left(B_i(\theta)^T\right)dt - \gamma p_i dt + \sqrt{2\beta^{-1}\gamma}dW_i. \tag{17}$$

Here dW_i has the previous interpretation of a Wiener increment and div is the tensor divergence. The matrix B_i estimates the matrix square root of the covariance matrix which depends on the locations of all the walkers.

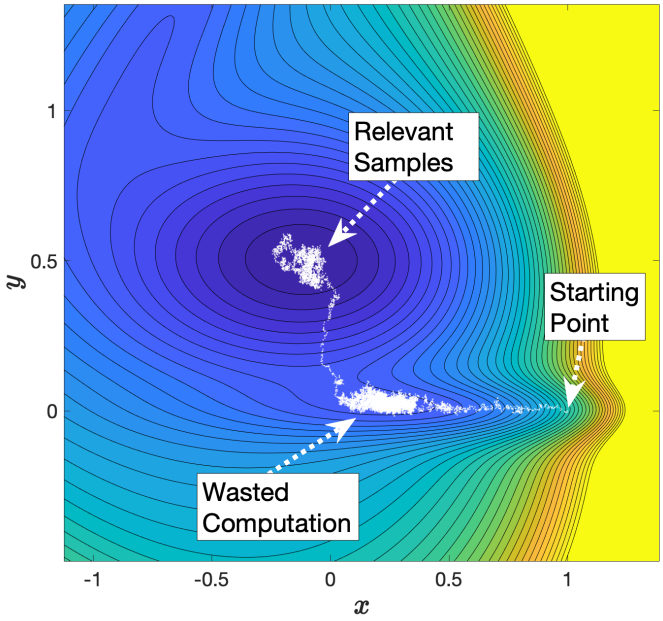


Figure 1: Slow exploration of a complex landscape using a (naive) canonical sampler. Here Euler-Maruyama has been applied to overdamped Langevin dynamics to sample a Gaussian mixture model on a two-dimensional state space. A poor initialization in this case leads to the extended sampling of an irrelevant intermediate basin (it contributes little to the sampling of the overall canonical measure). The consequence of poor scaling of the intermediate region is wasted computational effort. The goal of the enhanced sampling procedures is to accelerate the exploration in poorly scaled domains.

Under discretization, we compute steps in configurations $\theta_{k,i}$ and momenta $p_{k,i} \in \mathbb{R}^N$ at iteration step k with walker index i. We use a variant of the BAOAB discretization applied to walker i:

$$p_{k+1/2,i} = p_{k,i} - \frac{\varepsilon}{2} B_{k,i} \nabla L(\theta_{k,i}), \tag{18a}$$

$$\theta_{k+1/2,i} = \theta_{k,i} + \frac{\varepsilon}{2} B_{k+1/2,i}\, p_{k+1/2,i}, \tag{18b}$$

$$\hat{p}_{k+1/2,i} = \alpha p_{k+1/2,i} + \frac{(\alpha+1)\varepsilon}{2} \mathrm{div}\Big(B^T_{k+1/2,i}\Big) + \sqrt{\frac{1-\alpha^2}{\beta}} R_{k,i}, \tag{18c}$$

$$\theta_{k+1,i} = \theta_{k+1/2,i} + \frac{\varepsilon}{2} B_{k+1/2,i}\, \hat{p}_{k+1/2,i}, \tag{18d}$$

$$p_{k+1,i} = \hat{p}_{k+1/2,i} - \frac{\varepsilon}{2} \nabla L(\theta_{k+1,i}). \tag{18e}$$

As in the code of Leimkuhler et al. (2018), we make the calculation in (18b) explicit by updating $B_{k,i}$ only infrequently, typically every 1,000 steps, and we define $B_{k,i}$ by

$$B_{k,i} = \sqrt{\mathbb{1} + \eta\, \mathrm{cov}\left(\theta_{k,[i]}, \theta_{k,[i]}\right)}, \tag{19}$$

In the above, the matrix square root is understood in the sense of a Cholesky factorization, $\eta \geq 0$ is a covariance blending constant, and $\theta_{k,[i]}$ is the set of all walker positions at timestep k, excluding those of walker i itself. Note that for moderate η the choice (19) is always positive definite even if $L < N$. However, with (19) the affine invariance property, an elegant feature of the original system, no longer strictly holds. Nonetheless, the simplified method has been shown to work very well in practice (Leimkuhler et al., 2018, sect. 4).

3. Numerical Experiments

In this section, we first look at the convergence rates and accuracy of the samplers described previously for the simplified case of a harmonic oscillator, comparing them with analytically known rates. Next we turn to a very simple clustering problem to further explore the error (bias) introduced by SDE schemes; we show that a particular discretization ("BAOAB") of underdamped Langevin dynamics offers extraordinarily high accuracy compared to several alternative, mimicing observations about this scheme in the molecular dynamics setting. Subsequently, we use the MNIST dataset on single-layer and multi-layer perceptrons to illustrate an enhanced loss landscape visualization technique that obtains its projection directions from sampling trajectories. Finally, we present results on the ensemble quasi-Newton (EQN) method described in Section 2 for a Gaussian model and for a single layer perceptron applied to the MNIST dataset.

3.1 Sampler Properties and Error Analysis

In this error analysis, we explore a state $\{\theta, p\}$ with position θ and momentum p of a single degree of freedom. The kinetic energy is defined as $\varphi(\theta, p) = \frac{1}{2}p^T p$, where we have set the mass matrix to unity. Its asymptotic value in the canonical distribution is given by the number of parameters N and the (inverse) temperature β as $\frac{1}{2\beta}$ (Leimkuhler and Matthews, 2012, sect. 6.1.5). The virial is defined as $\varphi(\theta, p) = \sum_i^N \theta_i \cdot \nabla_i L(\theta) = \theta \cdot \nabla L(\theta)$, where $\nabla_i L(\theta)$ is the derivative of the loss function with respect to the parameter θ_i. Its asymptotic value is two times that of the kinetic energy, as a consequence of the virial theorem. For this theorem to hold we need a potential that is unbounded from above and grows sufficiently rapidly at infinity, see appendix A.1 for a derivation.

Averages of the kinetic energy or of the virial are examples of time integrals of (7) that can be used to assess the accuracy of the chosen dynamics and discretization, since their asymptotic values are known. As we are interested in discretized integrals over finite number of time steps, which are accessible in numerical computations in the absence of analytical solutions, we look at estimators (14) of the following form

$$\hat{\varphi}_N = \frac{1}{N} \sum_{n=0}^{N-1} \varphi(q_n, p_n).$$

The total error with respect to the expected value $E_\mu(\varphi)$ for the invariant probability measure μ decomposes as

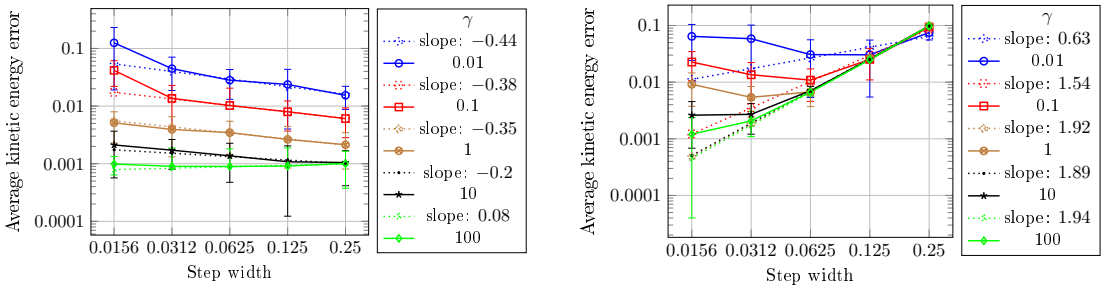
$$E\left(|\hat{\varphi}_N - E_\mu(\varphi)|^2\right) = \left(E(\hat{\varphi}_N) - E_\mu(\varphi)\right)^2 + E\left(|\hat{\varphi}_N - E(\hat{\varphi}_N)|^2\right), \tag{20}$$

i.e. it is a combination of the *discretization error*, from the finite step size when discretizing the dynamics (9), and the *sampling error* that results from the inability to sample over an infinite time or to generate an infinite number of steps. The first source is also sometimes referred to as the perfect sampling bias, emphasizing its presence even in the limit of infinitely many samples.

In the extreme case of a vanishing gradient, only the sampling error is present. This error will decay so that its variance is proportional to $1/N$. A more interesting case is that of a "harmonic potential" $I(\theta) = a\theta^2/2$, in which case the truncation error is nontrivial.

3.1.1 Truncation Error for Harmonic Potential

We have contributions to the total error (20) from both the sampling error and the discretization error. We will see that the latter may dominate when the scale a of the potential and therefore the average gradients are sufficiently large. We concentrate here on the empirical results; refer to Leimkuhler and Matthews (2012, sect. 7.4) for a general discussion of harmonic problems in the context of Langevin dynamics. We use a single-layer perceptron with a single input node and a single output node with linear activation and zero bias, i. e. $f_\theta(x_i) = \theta_1 x_i$. We use the mean squared loss function. Then, such a harmonic potential can be easily introduced by a dataset with the square root of the prefactor as its single feature and a zero label, i. e. the only dataset item is $(X, Y) = (\sqrt{a}, 0)$. We use three different factors $a \in \{0.01, 1, 4\}$. Moreover, we use the following sets of parameters for $\gamma \in \{0.01, 0.1, 1, 10, 100\}$, $\beta = 10$, and $\varepsilon \in \{2^{-2}, 2^{-3}, 2^{-4}, 2^{-5}, 2^{-6}\}$. Here, we employ BAOAB as the sampler with $N = 10^6$ sampling steps.



(a) Small prefactor $a = 0.01$

(b) Large prefactor $a = 4$

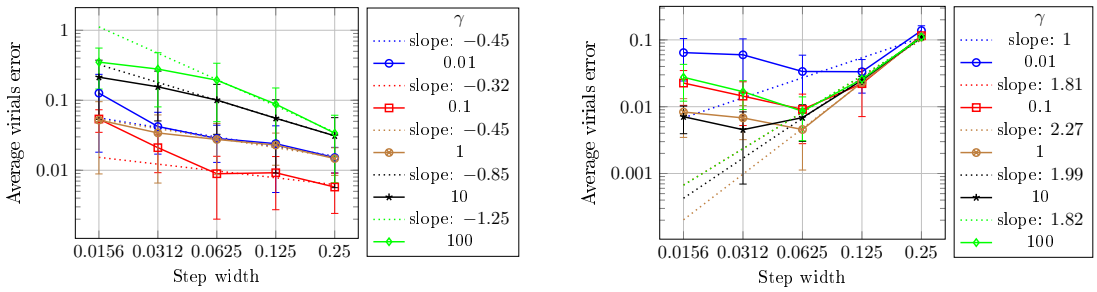
Figure 2: Highlighting the sampling error with respect to the step size by showing the relative error of the average kinetic energy with respect to its asymptotic value for the "quadratic potential" case. (a) If the gradients are small, the system performs a pure random walk and results are independent of the step size. (b) Given large enough gradients relative to the temperature the convergence order of the integration method with the step size becomes apparent, here second order for BAOAB in the momenta.

Examining Figure 2a where we use a small prefactor a, we notice that the error decreases with increasing step size because α becomes smaller and therefore we obtain more random walk-like behavior. However, it does not entirely depend on α, but also to some extent on the step size ε. For the highest value of $\gamma = 100$ we again have a flat line due to the lower bound enforced by the CLT.

In Figure 2b with a large prefactor a for large step sizes all of the curves coincide regardless of γ and the behavior has reversed: now the error becomes smaller for smaller step sizes. Naturally, the reason for this change is the discretization error that arises because of substantial non-zero gradients, and that the error depends on the step size ε. Measuring the slope in the domain where all curves overlap for $a = 4$, we obtain values of up to 2, i. e. second order convergence in the discretization error as expected from BAOAB. In place of the prefactor a we could also have varied the inverse temperature β to the same effect that only depends on the scale of the noise relative to the scale of the gradients.

Using a higher-order sampler allows for a smaller error at a given step size or to use larger step sizes (and therefore sample more space) for a given error threshold. Note that the step size is bounded from above by a stability threshold. The benefit of the trade-off between accuracy and computational effort is limited however, and it is likely that very high order schemes (beyond the second order splittings discussed here) are not efficacious in TATi, in keeping with previous studies in molecular dynamics (Leimkuhler and Matthews (2015)).

For comparison, we also look at the average virials in Figure 3. However, they depend only on the position and *not* on momentum. Note that the BAOAB scheme has nil perfect sampling bias for purely configurational quantities such as virials. Here, we are instead using the Geometric Langevin Algorithm (GLA) 2nd order sampler but keep all the other aspects of the method unchanged.



(a) Small prefactor $a = 0.01$

(b) Large prefactor $a = 4$

Figure 3: Relative error of the average virial with respect to its asymptotic value for the "quadratic potential" case using GLA2 sampler. Again, we see second order convergence, here in the positions, given the gradients are large enough relative to the noise, see also Figure 2.

We obtain the same qualitative picture for the average virials sampling with GLA2 as we got with the average kinetic energy sampling with BAOAB. Again, for a large enough prefactor the discretization error dominates. Inspecting the slopes in the doubly logarithmic plots, we find values around 2 that peak for $\gamma = 1$.

3.1.2 Langevin Sampler Performance in a Two-Cluster Classification Problem

In the following we will be inspecting the average virial, obtained over sampled, finite trajectories in order to assess the accuracies of positions obtained from various samplers. We will be investigating the following samplers: Stochastic Gradient Langevin Dynamics (SGLD), Geometric Langevin Algorithm (GLA) 1st and 2nd order, and BAOAB.

Very long runs are needed to bring forth the different convergence order of the discretization error because of the involved statistical errors. Therefore, we still use a very simple dataset: it consists of 500 points drawn from two Gaussians in two dimensions, one centered at $[2, 2]$ with label 1, the other centered at $[-2, -2]$ with label -1. The points are additionally perturbed by 0.1 relative noise.

We use a single layer perceptron with two input nodes, a single output node with linear activation and mean squared loss. Therefore, the network has $N = 3$ parameters in total, two weight degrees and one bias degree.

The parameters are first equilibrated for 2,000 steps with a learning rate of 0.03 with Gradient Descent (GD). Next, we perform sampling runs from the resulting position for 10^6 steps at various step sizes ε. In the case of a sampler based on Langevin dynamics, we use a friction constant $\gamma = 10$ and an inverse temperature $\beta = 10$. Note that, because we start at an equilibrated position with zero gradients and because the potential function is squared, positions are only rescaled when using other temperatures if the same random number sequence is used. We use 100 different seeds and average the last average value per trajectory over all seeds.

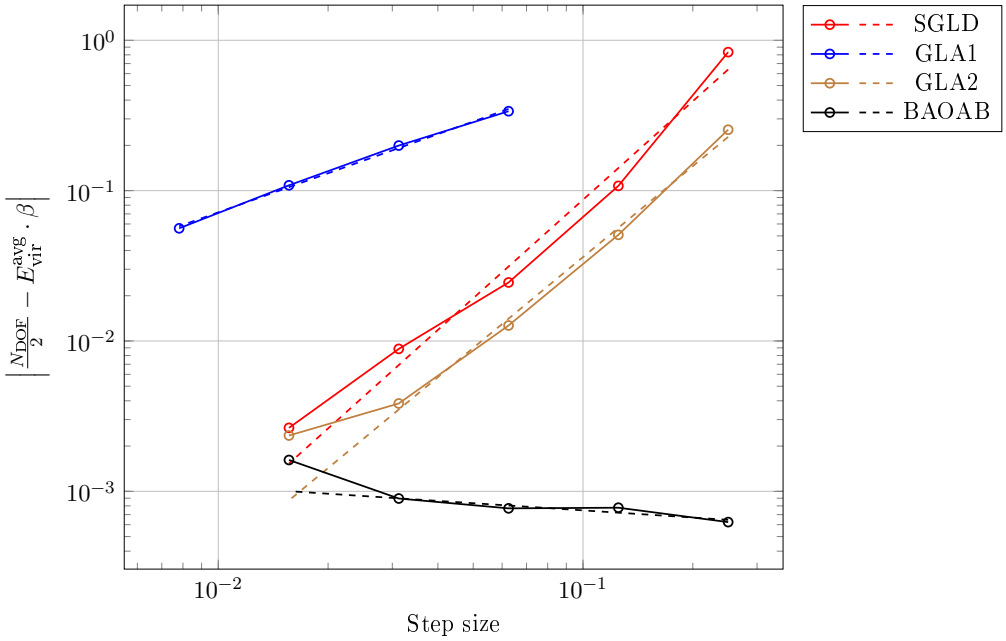


Figure 4: Order of convergence for the discretization error for four samplers of the average average virial for the simple two clusters dataset. Dynamics become unstable if the largest step sizes are increased by a factor of two. GLA1 has first order (slope 0.86). SGLD (slope 2.18) and GLA2 (slope 2.01) have second order. BAOAB's accuracy on the position marginal is so good that its second to fourth order convergence cannot be seen against the lower bound of the CLT.

In Figure 4 we give the absolute error between the average virial and its asymptotic value scaled by the inverse temperature β per sampler for each step size employed. Note again that the virial depends on the positions and the gradient. Moreover, we remark that increasing the largest step size shown for each sampler individually by a factor of two would

cause the dynamics to become unstable. Naturally, the exact threshold depends on the magnitude of the gradients and therefore on the dataset.

The slopes have been obtained from least squares regression fits where the data point to the smallest step size have been weighted by $\frac{1}{10}$.

We make the following observations: GLA2 has second order convergence in the average virial, GLA1 has first-order convergence. BAOAB shows such great accuracy at this finite trajectory length that its second to fourth order convergence does not show as it reaches the CLT limit. These results are in absolute agreement with results from an analysis on harmonic problems, see (Leimkuhler and Matthews, 2012, sect. 7.4.1). Note that SGLD exhibits second-order convergence in the virial in this example; actually this is an artifact of the way the data has been graphed: the SGLD 'step size' can be viewed as the square root of the step size $\sqrt{\varepsilon}$ of the other samplers, see (Leimkuhler and Matthews, 2012, p. 36), i.e. it is really a first order scheme when expressed in the standard way.

From these results the sampling method of choice seems to be BAOAB which has superior accuracy in the positions and general second order convergence at little computational overhead and extra memory requirement, compared to SGLD. Note that SGLD is based on Brownian dynamics, thus lacking momenta, and therefore is expected to be less efficient in exploration for multimodal problems than BAOAB or other Langevin dynamics samplers Leimkuhler and Matthews (2015).

A similar study on the MNIST dataset is impeded by its wide-spread covariance eigenvalue spectrum, shown in the next section. There we encounter both large and small gradients and therefore have a mixture of the "flat potential" and "harmonic potential" cases which obscures the covergence orders. However, choosing a high-accuracy integration method is nonetheless very important there as well, as the presence of large gradients (or large directional derivatives) will dominate the exploration.

3.2 Application: Loss Manifold Analysis for MNIST dataset

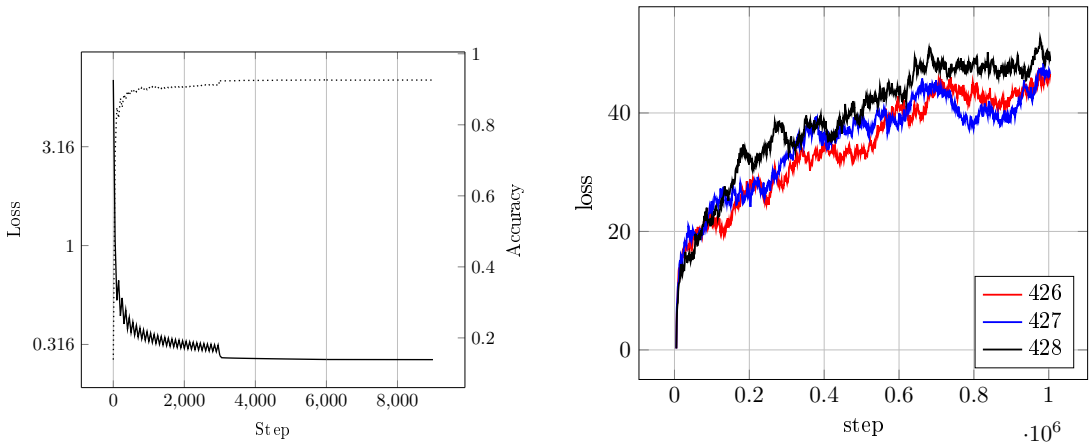
We now consider the MNIST training data set of 70,000 grey-scale images of 28x28 pixels, see LeCun. We have divided these into a validation set of 10,000 images, a test set of 5,000 images and a training data set of 55,000 images.

The simplest network to tackle this classification problem is a single layer perceptron with 784 input nodes and 10 output nodes. We use linear output activation and the softmax cross-entropy as loss function. The accuracy is quantified by the strongest output (argmax).

In a first experiment, we use SGD as optimizer with a batch size of 550, i. e. 1 % of the dataset size, for 9000 steps and an initial learning rate of 0.5 that is reduced to 0.05 while the batch size is increased to 5,500 after 3,000 steps and finally down to 0.01 with no more mini-batching after 6,000 steps. With this particular training scheme, we obtain a loss of 0.265 and 92.6% accuracy on the training dataset and a loss of 0.271 and 92.6% accuracy on the test dataset, c. f. 91.6% and 92.4% in LeCun et al. (1998).

The resulting loss per step is given in Figure 5a. There are fluctuations due to the stochastic gradients that decrease with the learning rate and with the batch size.

Note that we stop the optimization after a finite number of steps and not when the gradients has a zero norm. Therefore, the points encountered in the loss landscape during



(a) Loss (solid) and accuracy (dotted) along the optimization trajectory consisting of three parts (blue, gray and black) where only every 10th step is shown. The optimization yields 92.6% test and training set accuracy.

(b) Loss values along the sampled trajectories where only every 10th step is shown where trajectories obviously deviate rapidly from the initially same starting position. Each run only differs by the random number seed, hence being subject to different thermal noise.

Figure 5: Loss values along the optimization and three sampling trajectories.

optimization are just critical points. Nonetheless, we will refer to them as "quasi-minima" in the following as the gradient norm is very small.

In order to inspect the quality of the quasi-minimum found, we need to look at the resulting loss manifold in a neighborhood. As the network has 7850 degrees of freedom in total, we need appropriate techniques for its visualization. Several of these are discussed in Li et al. (2018). The current state-of-the-art is to project onto two random directions, as proposed in Goodfellow et al. (2014), that frequently only shows little variation (Li et al. (2018)).

With the sampling approach proposed here, there is a different alternative. Starting in at or near a quasi-minimum, a walker generates a cloud of points iteratively by following the chosen dynamics. Therefore, sampling provides us with a cloud of points that expands within that basin according to rules of these dynamics: if the basin is flat in certain directions, the cloud's expansion will prefer these over directions where the walls are steep. The strength of the preference is controlled by the (inverse) temperature parameter β that allows the method to overcome walls to a certain steepness and height.

If we analyze the sampled point cloud's principal components, using the eigensystem of the covariance matrix, then it's major principal component points along the direction where the basin is flattest.

In the following, we compare two components associated with a) the largest eigenvalue, and b) an arbitrarily chosen small eigenvalue of the covarance matrix. This will allow us to get a notion of the extent of the flat part of the basin.

The covariance matrix is computed from a single sampling run of 10^6 steps using BAOAB as the sampler with a step size $\varepsilon = 0.125$, $\gamma = 10$, $\beta = 10$, and a batch size of 550.

In the following we have performed three of these sampling runs starting from the same position, equilibrated with SGD with the same batch size of 550, for 5000 steps with a learning rate of 0.5. The only point in using multiple runs is to exclude the possibility that subsequent results depend simply on a specific, rare combination of samples taken. Therefore, each run differs only by the random number seed and is thus only subject to different thermal noise. Note that the specific initial position will not have an effect as the temperature is high enough to bring the walker quickly to a completely different position: see Figure 5b for loss values of all trajectories. There, the loss increases from its initial value because of the small value chosen for β, i. e. a high sampling temperature. The walker typically assumes positions far away from equilibrium.

For reasons of computational efficiency we note the following: to obtain the covariance matrix of very high-dimensional networks the trajectories from several (parallel) runs can be combined to overcome the computational burden. Moreover, a truncated eigendecomposition, e. g., using the power method and shift-and-invert, would fully suffice to obtain the largest and a small eigenvalue within a certain range and their associated eigenvectors, taking advantage of symmetry and positive semi-definiteness.

The resulting computed eigenvalue spectra are given in Figure 6. Note that we have truncated the spectrum here to 1000 non-zero eigenvalues.

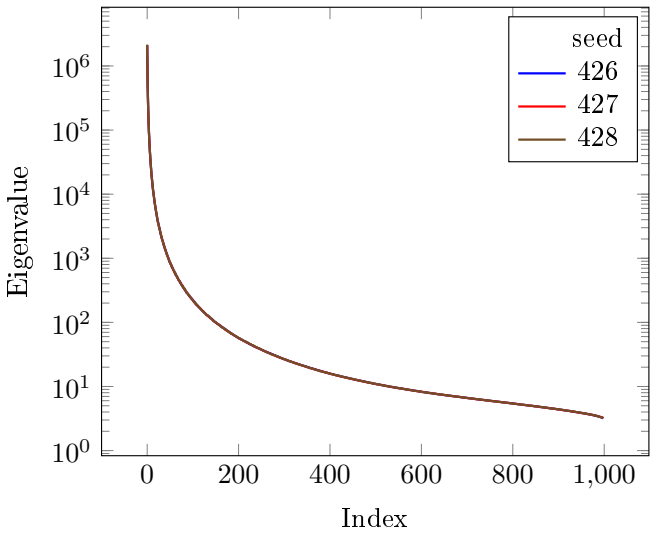


Figure 6: Logarithmic depiction of the eigenvalues of the covariance matrix of three sampled trajectories. Each legend entry refers to the (different) random number seed employed. Spectra are in good agreement with about 1% deviation between the three random number seeds.

All three curves in Figure 6 coincide approximately in the logarithmic scale, with a deviation of about 1%. This leads us to assume that the sampling runs have indeed been long enough for the system to lose memory of its initialization and have produced representative thermodynamic samples from the target ensemble. We hypothesize that this eigenvalue spectrum is representative of the general covariance structure at large scale for MNIST and for the particular model chosen.

Observe that there is a strong decay of the eigenvalue magnitudes[3].

The decay in the eigenvalue spectrum expresses itself as few eigenvectors with very large eigenvalues and many eigenvectors whose eigenvalues have at least 3 or 4 orders of magnitude smaller eigenvalues. We judge this variation in scale as the spectrum's major feature.

To highlight the extent of the flat part, we pick the first and 64th eigenvalues and use their associated eigenvectors as the directions v_1 and v_0 in which we plot the loss manifold. The direction v_1 represents the large covariance, while v_0 represents the small covariance.[4]

The strong decay indicates that random directions are poorly suited for the loss visualization (at least for MNIST and for the chosen model) as, on average, these will not relate to directions of strong covariance. Hence, the flattest manifold parts will be missed on average when choosing random directions.

For visualizing the loss manifold, we sample it on an equidistant grid with 41 samples per axis along the two chosen directions $v_0 = (W_0, b_0)$, $v_1 = (W_1, b_1)$, split into weight components $W_j \in \mathbb{R}^{784\times10}$ and bias components $b_j \in \mathbb{R}^{10}$. We evaluate $\tilde{L}(c_0, c_1) = \sum_{i=1} \sigma(\tilde{f}(c_0, c_1), y_i)$ with the softmax cross-entropy σ and $\tilde{f}(c_0, c_1) = (W + c_0 W_0 + c_1 W_1) \cdot x + B + b_0 + b_1$, i. e. the loss constrained to the two-dimensional subspace and centered at the obtained (local) quasi-minimum at (W, B). We use different intervals per axis of $[-10^i, 10^i]$ with $i \in \{1, 0, \ldots, -4, -5\}$, endpoints included. For visualizing the previously obtained optimization trajectories, we re-evaluate them using the full training dataset (i. e. no mini-batches) per step and project them onto the two chosen directions. Note that also each sampled point of the loss manifold is evaluated using the full training dataset.

In Figure 7 we look at four of these manifold plots where we give the sampled manifold and the projected optimization trajectory. The x and y axes correspond to the two chosen directions, the z axis gives the loss as $\ln|\,\tilde{L}(c_0, c_1) - L_0)\,|$, with respect to the lowest loss L_0 value found overall, at the specific point $(c_0, c_1 0)$ in case of the sampled manifold and the true loss in case of the projected trajectory. Because of the projection the trajectory steps will not lie on the manifold itself.

In Figure 7a we recognize a large funnel where the c_0 direction is associated with the large eigenvalue and the c_1 direction is associated with the small eigenvalue. We see that the optimization trajectory gradually enters the funnel in (b). However, in (c) we realize that the previously obtained optimization trajectory ends prematurely, not in the possibly global minimum but stuck in one of the lower minima on the funnel wall in (d).

The intuition therefore is that the loss manifold resembles a large funnel whose extension though differs significantly in each direction. Its walls are corrugated with many local minima, see Figure 8, especially at its bottom. Note that this funnel is not a product of the logarithmic scale, again see Figure 8 where a normal scale is used. This corresponds well with the observation in Goodfellow et al. (2014) that optimization runs never encounter serious obstacles and that there is a "sea of minima" in a small band of the loss bounded from below as proposed in Choromanska et al. (2015) from translating results on spherical spin-glasses. Note though that the latter was derived for a different loss function.

This study hints at the usefulness of second-order methods also for optimization, see Bottou et al. (2018) for a review.

3. The decay resembles the Marchenko-Pastur distribution which describes the singular values of random matrices. However, we do not pursue this further in the scope of this article.
4. Picking the 1st and the 100th eigenvalue would have given an essentially equivalent visulization.

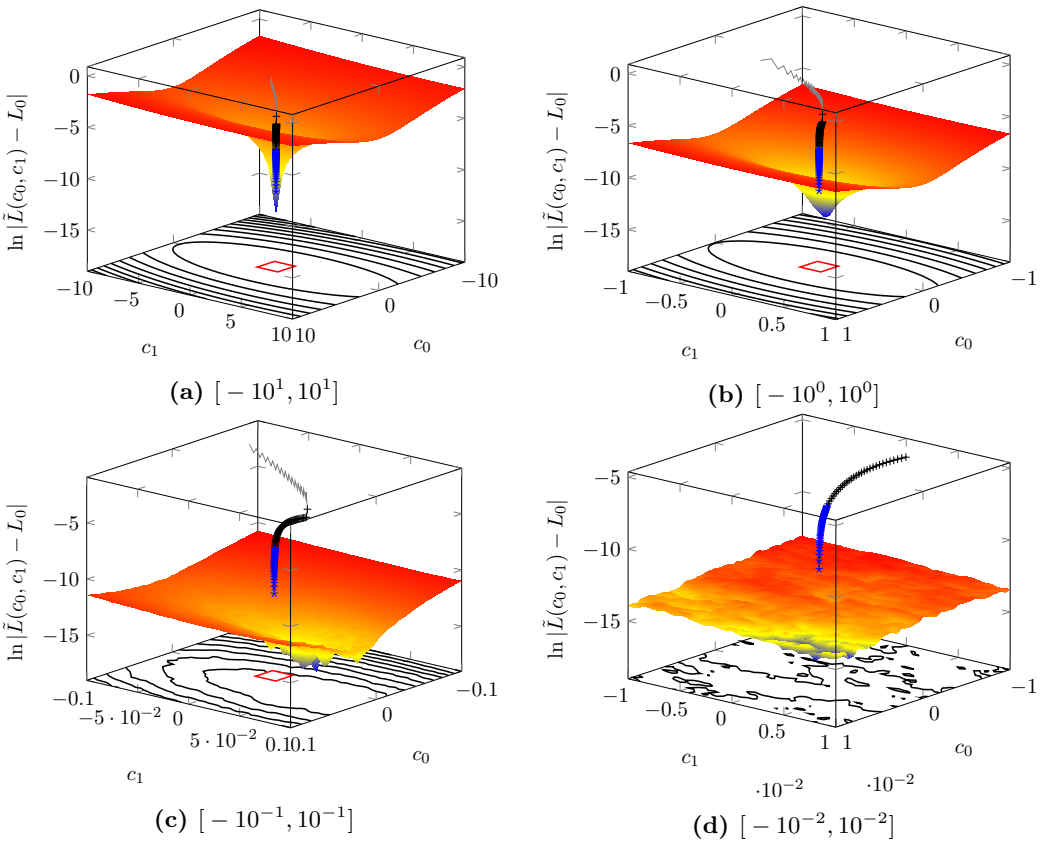


Figure 7: Visualization of the regularly sampled loss manifold for the MNIST dataset of single-layer perceptron with softmax cross-entropy loss function and linear activations. The two coordinate directions c_0, c_1 correspond to the 1st and 64th eigenvalues (descending) of a covariance matrix sampled from a very long run. The z-axis, in *log scale*, corresponds to $\ln(\tilde{L}(c_0, c_1) - L_0)$, where L_0 is the smallest loss encountered overall. The value for z is also used to color the manifold. Additionally, an optimization trajectory using SGD consisting of three consecutive parts is given where after each leg the learning rate is reduced and batch size is increased. The terminal point of its last leg provides the point of origin. Only every 50th step is shown. Note that we show the exact loss for the whole training dataset for both manifold and trajectory. A red square shows the subsequent plot's domain.

Even in the limited scope of the two-dimensional projection we have seen that a minimum associated with an even smaller loss would have been attainable. However, the differences in the loss values are marginal, namely less than 10^{-5}. Naturally, this analysis is not complete. There are possibly multiple funnels from multiple minima with high barriers in between, as in the disconnectivity graphs by Ballard et al. (2017). Recent work (Draxler et al. (2018)), however, suggests the contrary and corroborates our finding of a single funnel with many minima at its bottom. We conclude by remarking that this type of analysis can also easily be extended to multi-layer perceptrons, which we briefly touch in the next section, and potentially to more advanced network architectures. Note that for multi-layer perceptron,

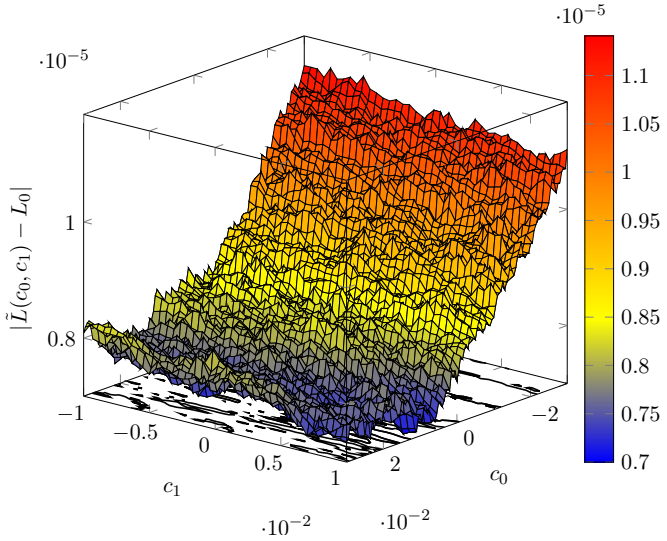


Figure 8: Sweep in the direction c_0 over five times the domain length with respect to Figure 7d. The walls of the funnel's bottom are dented with many local minima. Note that the figure is in linear scale for comparison to the log-scale before.

scale invariance needs to be accounted for, see Li et al. (2018) for a normalization scheme. There, multiple minima may be encountered due to symmetries.

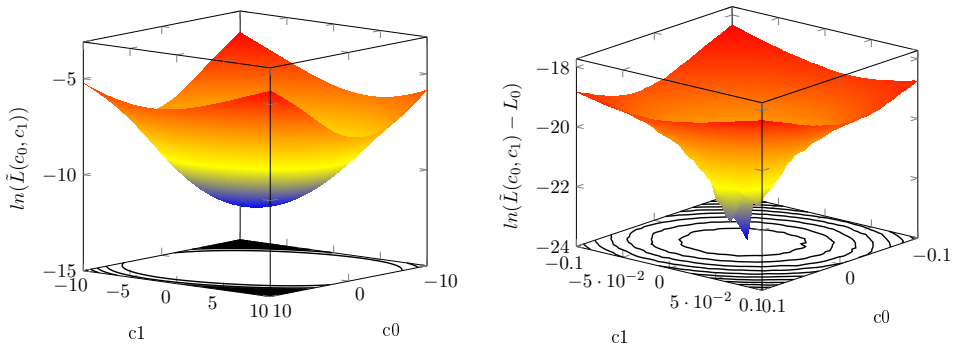
3.3 Application: More Complex Networks for MNIST Dataset

The previous model with no hidden layer used linear activations; only its loss function was non-linear. In this subsection, we consider very briefly the same dataset with a multi-layer perceptron with a single hidden layer of 100 nodes and sigmoidal hidden activation functions. This is to illustrate that the technique is not limited to linear models. We use again linear activations in the output layer and softmax cross-entropy as loss functions. This model has 79,510 degrees of freedom, i. e. an order of magnitude more than the linear model used before. We will not distinguish in the following between degrees of freedom associated to the first hidden layer and those associated to the output layer.

We have implemented GD with a Barzilei-Borwein step width choice (BBGD), see Tan et al. (2016), using the full gradient information. We use 3000 BBGD steps with an initial learning rate of 0.05. This advanced optimization method is not required for the sampling's starting point, there SGD would suffice. However, it yields an improved origin of the loss landscape in the following visualization.

Covariance directions have been obtained through sampling with the same parameters as with the linear model. Then, centered around the located quasi-minimum and using the 1st and 100th covariance direction sorted by their associated eigenvalue, we again sample on an equidistant grid in the subspace spanned by these two directions v_1 and v_0. Again, the sampling still finds parametrizations θ with smaller loss than located during optimization whose lowest serves as reference L_0.

In Figure 9 we give the resulting subspace manifold of the loss landscape for the sigmoid hidden activations function and the softmax cross-entropy loss. For the network using a 100



(a) $[-10^1, 10^1]$ **(b)** $[-10^{-1}, 10^{-1}]$

Figure 9: Visualization of the regularly sampled loss manifold for the MNIST dataset of multi-layer perceptron with 100 hidden nodes, softmax cross-entropy loss function, sigmoid hidden activations and linear output activations. The two coordinate directions c_0, c_1 correspond to directions associated with the 1st and 100th eigenvalues (descending) of a covariance matrix sampled from a very long run. The value for z, also used to color the manifold, is as follows. (a): The z-axis, in *log scale*, corresponds to $\ln(\tilde{L}(c_0, c_1))$ with $c_0, c_1 \in [-10, 10]$, i. e. the absolute logarithmic loss on the largest sampled interval in the two chosen covariance directions. (b): The z-axis, also in *log scale*, corresponds to $\ln(\tilde{L}(c_0, c_1) - L_0)$ with $c_0, c_1 \in [-0.1, 0.1]$, where L_0 is the smallest loss encountered overall.

hidden nodes we have found a quasi-minimum during optimization that achieves a perfect accuracy of 1.0 on the training set. Its accuracy on the test set is slightly lower with 0.975.

In Fig. 9a we look at the overall shape of the loss landscape in logarithmic representation at a length scale of 10. We find a very smooth funnel. On a smaller scale of length 0.1 in Fig. 9b, where we look again at the logarithmic difference to the smallest loss encountered overall, we notice that there are multiple local minima close to the funnel's bottom. Not shown is the sampled domain of length scale 0.01 where we then would see a heavily corrugated landscape.

This suggests that we see a similar landscape as with the linear model before where the bottom of the funnel is corrugated and that there is a plethora of states with small loss values.

3.4 Application: Ensemble Quasi-Newton Method

Returning to the analysis of the linear model, we have seen there is a large degree of anisotropy in the funnel observed in the loss manifold for the MNIST dataset. This results in significant variation in the projections of the gradient into different subspaces and is precisely the situation which motivated the development of the ensemble quasi-Newton scheme of Section 2.

TATi facilitates the implementation of a scheme such as the ensemble quasi-Newton method (see Section 2) by its multiple walker framework. In Appendix B, we describe the TATi implementation. Next, we investigate the method's qualities in a simple, well understood model before returning to the MNIST dataset.

3.4.1 Gaussian Model

A prototypical setting whose analytical properties are well-known is given by sampling from the Gaussian model,

$$\exp\left(-w^T C w\right) \tag{21}$$

with a covariance matrix $C \in \mathbb{R}^{d\times d}$, where d is the dimension of the parameter space. Here, we naturally encounter directions that are "slow" to sample, identified by large eigenvalues in C. In fact, when we (only) look at the covariance structure of the MNIST loss manifold, we replace it by an effective Gaussian model of that particular covariance matrix.

Transfering this model to the setting of sampling loss manifolds of neural networks is straight-forward: We use the mean squared loss $l_\theta\big(f(\theta, x_i), y_i\big) = \big(f(\theta, x_i) - y_i\big)^2$ with the network's prediction $f(\theta, x_i)$. Then, inserting into (2) in the case of n-dimensional input data $x_i \in \mathbb{R}^n$, single-dimensional output $y_i \in \mathbb{R}$, and a single-layer perceptron, i.e., $f_\theta(x) = w\cdot x + b$ with the parameters $\theta = \{w, b\}$ in the form of weights $w \in \mathbb{R}^n$ and of a bias $b \in \mathbb{R}$, we obtain

$$L(\theta, D) = \sum_i (w \cdot x_i + b - y_i)^2.$$

Setting the bias b and all outputs y_i to zero, we get

$$L(\theta, D) = \sum_i (w \cdot x_i)^2 = \sum_i \sum_{l,m=1}^{n} w_l (x_{l,i} x_{m,i}) w_m = \sum_{l,m=1}^{n} w_l \Big(\sum_i x_{l,i} x_{m,i}\Big) w_m.$$

Note that we sample from the canonical Gibbs distribution $\exp\big(-\beta L(\theta, D)\big)$. Therefore, the dataset needs to consist of rank-1 factors x_i that represent the chosen covariance matrix with components $C_{lm} = \sum_i x_{l,i} x_{m,i}$ in order to match this with (21). These factors can be obtained for example through an eigendecomposition $C = V \Lambda V^T$ as the eigenvectors V_i times the square root of their associated eigenvalue $\Lambda_{i,i}$. Naturally, any other (even non-orthogonal) decomposition into rank-1 factors would be admissible, too.

In order to produce random covariance matrices C of a certain structure, we resort to the following approach: We generate a random symmetric matrix, compute its eigendecomposition and modify the diagonal matrix D to consist of values picked from an equidistant spacing of the interval $[1, 100]$, where endpoints are included. This way we obtain a set of orthogonal vectors pointing uniformly randomly in $\mathbb{R}^n$, see Mezzadri (2007), and we make sure to generate both slow (eigenvalues close to 100) and fast (eigenvalues close to 1) directions.

Having generated a random covariance matrix C for dimensions $n \in \{2, 4, 8, 16, 32, 64, 128\}$ and having created the resulting dataset as its rank-1 factors, we sample the mean squared loss manifold of the single-layer perceptron using the BAOAB sampler. We use 50,000 steps with a time step size $\varepsilon = 0.125$, inverse temperature $\beta = 1$, friction constant $\gamma = 1$, and covariance blending factor of $\eta = 10$.

We measure the exploration speed in the Gaussian model by looking at the IAT τ per random direction, that we know from the random matrix' eigendecomposition, by projecting it onto each eigenvector and measure the IAT using the package *acor* (Goodman (2009)).

In Figure 10 we see that using the EQN scheme with 8 or 16 walkers significantly improves the Integrated Autocorrelation Time ε for the slow directions. We note that the fast

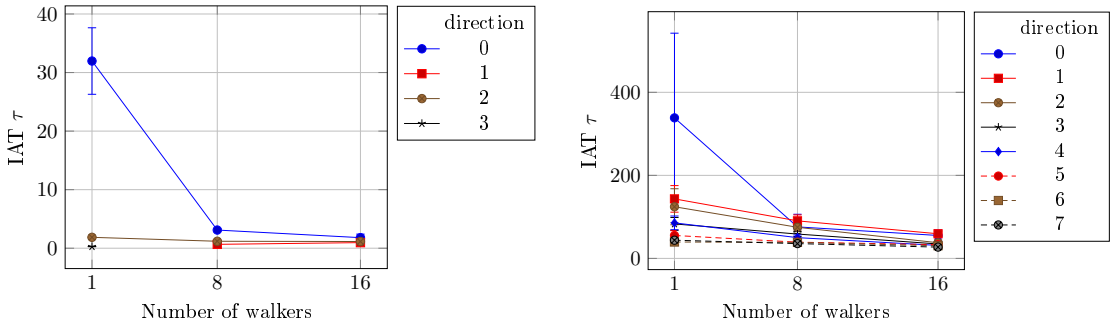


Figure 10: Integrated Autocorrelation Time (IAT) ε over the number of walkers for Gaussian models of dimensions $n \in \{4, 64\}$ where only up to the first 8 directions are shown. Five Sampling runs have been completed for each random covariance matrix and average IAT and standard deviation is shown. With a single walker standard BAOAB sampling is employed, with multiple walkers the EQN method is used.

directions are unaffected. We remind the reader that each walker samples its own trajectory. Hence, we generate up to 16 trajectories in parallel and do so ten times more efficiently because of the reduction in the IATs.

3.4.2 MNIST

We now turn to the MNIST dataset again for a real-world application of the EQN method. We constrain the training dataset to two classes, namely the digits 7 and 9. This results in 11,169 training dataset items for this two-class problem. We employ the same single-layer perceptron as before.

For the IAT computation we have extracted distinct covariance matrices per walker $C_i = \mathrm{cov}(\theta_i^{(n)}, \theta_i^{(n)})$, obtained the eigendecomposition $C = V\Lambda V^T$, extracted the eigenvectors V_j of the 20 dominant eigenvalues, and projected the walker's trajectory onto these, $\pi_{i,j}(n) = \theta_i^{(n)} \cdot V_j$. Finally, we calculated the IAT of each $\pi_{i,j}(n)$ using the `acor` package and averaged over all walkers i.

Using walker-individual covariance matrices does not generally change results compared to a single covariance matrix obtained from averaging the trajectory over all walkers; however, it makes them more stable. Because of the high dimensionality of the parameter space $\mathbb{R}^{1,568}$ already small perturbations may cause vectors to become orthogonal[5]. This phenomena is explained by the "Concentration of Measure", see Ledoux (2001). The eigenvalue spectra themselves are stable for each walker, bounded in deviation by the Bauer-Fike theorem. Note further that the preconditioning matrix $B_i^{(n)}$ is also uniquely defined for each walker.

Due to computations necessary for the additional preconditioning matrix, there is a slight walker-dependent overhead compare to the case of single walker: For 8 walkers we measured about 40% overhead per walker, for 16 walkers we obtained 50%.

5. The more dimensions a space has, the more likely it becomes for two random vectors to be orthogonal to each other, see also (Li et al., 2018, sect. 7.1).

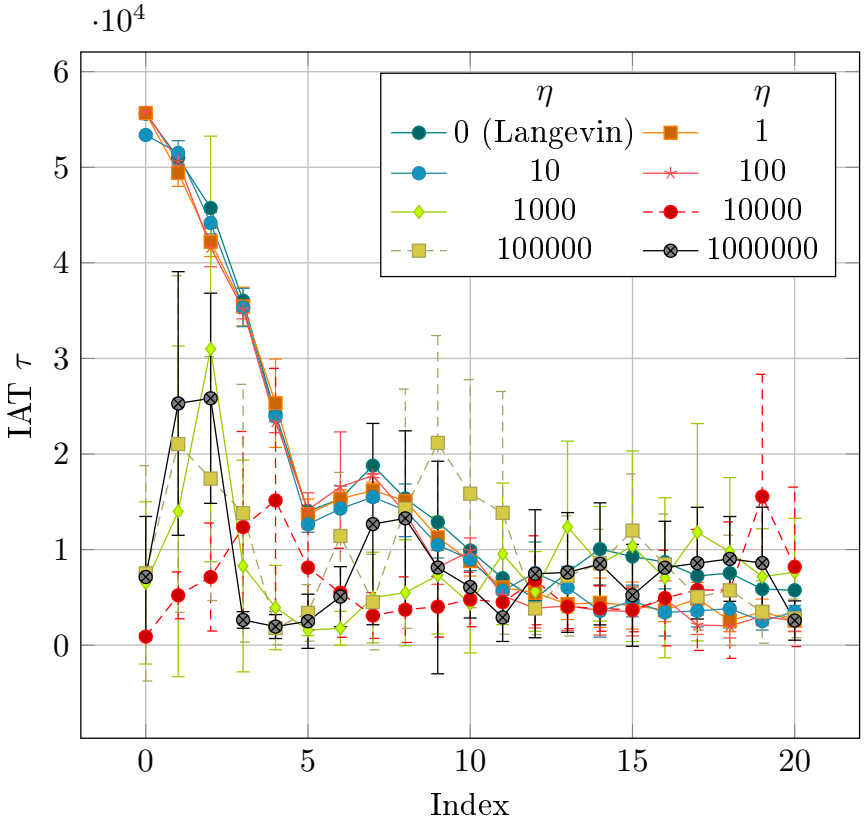


Figure 11: Integrated Autocorrelation Time (IAT) τ for the first 20 covariance eigenvectors over the number of walkers for the MNIST two class problem. We observe a strong improvement of up to a factor of 4 for the slowest Integrated Autocorrelation Time (IAT) which relates to a similar increase in exploration speed, e. g., when using η of 10,000 (EQN)compared to $\eta = 0$ (standard Langevin).

In Figure 11 we then look at the IAT over the first 20 covariance eigenvectors for various values of the covariance blending constant η. We used a fixed number of 8 walkers, a batch size of 550, an inverse temperature constant of $\beta = 10$, the friction constant set to $\gamma = 10$ and a step size $\varepsilon = 0.125$ with the BAOAB sampler. All runs are started from an equilibrated position using SGD with batch size of 550 and a learning rate of 0.1 for 5,000 steps. We recompute the covariance matrix after 10,000 steps. The trajectories were stored with only every 100th sampling step. Hence, the ε values in the Figure have been rescaled appropriately.

As there is a scaling invariance with respect to the biases of the output layer due to the `argmax` function, we have fixed the biases for the sampling to the values obtained from a prior optimization. At the moment, the EQN implementation cannot deal with such invariances. They represent "flat valleys" in the loss landscape and the walkers will be pushed by the preconditioning along the valley in vain search for its bounds. Such a valley can be hypothesized from the spectrum of the covariance matrix, see Figure 6, where the first eigenvalue with $2.1 \cdot 10^6$ is unusually high, and when the EQN does not effectively

reduce the IATs although the eigenvalue spectrum indicates it, c. f. (Leimkuhler et al., 2018, p. 281).

We generally observe that especially the first five IAT values are dramatically reduced and see an improvement of up to a factor of 4. The covariance blending can be chosen robustly, up to very high values. This indicates that the covariance matrix itself, despite $L < N$, is already positive definite.

4. Conclusion

We have discussed comparisons of sampling strategies based primarily on stochastic differential equations, with a focus on issues such as sampling efficiency and accuracy. All the methods discussed are implemented in the TATi software. Relying on *TensorFlow*, the TATi implementation efficiently runs in parallel and also on GPU-assisted hardware.

We have looked in our evaluation at the MNIST loss manifold for a single-layer perceptron and a multi-layer perceptron using softmax cross-entropy. We find that it resembles an anisotropic funnel on the large scale combined with many local minima at its bottom matching well the band of minima bounded from below and exponentially decaying in density with higher loss values predicted in Choromanska et al. (2015). This motivated an ensemble method employing a number of so-called walkers to obtain a local approximation of the covariance that, when turned into a preconditioner, results in a significant improvement of the sampling speed.

The focus on posterior sampling using SDEs provides us with a starting point for a wide range of improvements. For example we are currently exploring schemes based on simulated tempering Marinari and Parisi (1992); Martinsson et al. (2019) and diffusion maps Trstanova et al. (2020); the latter approach allows simulation data obtained using e.g. Langevin dynamics to be distilled into a few collective variables which succinctly describe the progress of transition between neighboring local minima. We are also exploring the use of the sampling paradigm as a tool to design neural networks with improved sparsity and generalizability.

Acknowledgments

The authors acknowledge funding through a Rutherford fellowship from the Alan Turing Institute in London (R-SIS-003, R-RUT-001) and EPSRC grant no. EP/P006175/1 (Data Driven Coarse Graining using Space-Time Diffusion Maps, B. Leimkuhler PI), as well as B. Leimkuhler's Turing Fellowship (The Alan Turing Institute is supported by EPSRC grant EP/N510129/1). The computing resources were provided by a Microsoft Azure Sponsorship award (MS-AZR-0143P).

In relation to the development of the TATi software, the authors would also like to express their gratitude to the Research Software Engineering Team of the Alan Turing Institute led by James Hetherington and especially to Martin O'Reilly for support in using the Azure platform. Charles Matthews provided valuable comments on a preliminary draft of the article, leading to a much improved exposition.

Appendix A. Virial Theorem

The virial is defined as $G = \sum_i^N \theta_i p_i$ with positions θ and momenta p. The virial theorem states that $\frac{dG}{dt} = 0$ which would imply through

$$\frac{dG}{dt} = \sum_i^N p_i p_i + \sum_i^N \theta_i \frac{\partial L(\theta)}{\partial \theta_i} \tag{22}$$

that the first term, twice the kinetic energy, equals the negative of the second.

Let us inspect the second term and look at its average over the whole domain using the Gibbs measure with a single degree of freedom ($N = 1$),

$$\frac{\int_{\mathbb{R}} \theta \cdot \nabla L(\theta) \exp\left(-\beta L(\theta)\right) d\theta}{\int_{\mathbb{R}} \exp\left(-\beta L(\theta)\right) d\theta},$$

where β is the inverse temperature.

Let us ignore the denominator for the moment and integrate the nominator by parts. We obtain with the derivative $\frac{\partial}{\partial \theta} \exp\left(-\beta L(\theta)\right) = -\beta \exp\left(-\beta L(\theta)\right) \frac{\partial L(\theta)}{\partial \theta}$,

$$\int_{\mathbb{R}} \theta \cdot \nabla L(\theta) \exp\left(-\beta L(\theta)\right) d\theta = \left[\theta \cdot \tfrac{-1}{\beta} \exp\left(-\beta L(\theta)\right)\right] - \int_{\mathbb{R}} 1 \cdot \tfrac{-1}{\beta} \exp\left(-\beta L(\theta)\right) d\theta.$$

If the boundary term vanishes, we obtain

$$\frac{\int_{\mathbb{R}} \frac{1}{\beta} \exp\left(-\beta L(\theta)\right) d\theta}{\int_{\mathbb{R}} \exp\left(-\beta L(\theta)\right) d\theta} = \tfrac{1}{\beta}.$$

In other words, the average virial, the second term, would be identical to two times the average kinetic energy $\frac{1}{\beta}$, the first term in (22).

Hence, all that remains is to show that $\left[\theta \cdot \exp\left(-\beta L(\theta)\right)\right]_{-\infty}^{\infty} = 0$. Naturally, this holds if $\lim_{|\theta| \to \infty} L(\theta) \to \infty$ to the effect that $\exp\left(-\beta L(\theta)\right) \to 0$ faster than $|\theta| \to \infty$, noting that the exponential increases faster than any polynomial.

In other words, the potential $L(\theta)$ needs to be unbounded and to increase faster than $|\theta|$ for the virial theorem to hold.

A.1 Virial and MNIST

If for the MNIST dataset, a single-layer perceptron with a softmax cross-entropy function is employed, then the virial theorem does not hold.

The output of the single-layer perceptron is $f_i(\theta) = \sum_j W_{i,j} x_j + b_i$ with weight matrix θ and bias vector b, i. e. $\theta = (W, b)$.

As the cross entropy is $-\sum_i y_i \log p_i(f(\theta))$ and the softmax function is $p_i(f(\theta)) = \frac{\exp f_i(\theta)}{\sum_i \exp f_i(\theta)}$, we have $\exp\left(\beta \sum_i y_i \log p_i(f(\theta))\right) = \prod_i p_i(f(\theta))^{\beta y_i}$.

Let us set all parameters components to zero except for a single weight component $W_{i,j}$ where at least for one item in the dataset we have $x_j \neq 0$. Then we obtain $W_{i,j}(W_{i,j} x_j)^{\beta y_i}$ as the integrand for this data item (x_i, y_i) and the boundary integral will not converge (to zero) in this case.

Note that this issue could be addressed by the addition of an L_2 regularization strategy.

Appendix B. Ensemble Quasi-Newton Implementation in TATi

In Listing 1 we provide a rapid prototype of the algorithm using TATi's `simulation` module.

Listing 1: Example implementation of EQN using TATi's `simulation` module. We require for the number of walkers $L > 1$. We have skipped (...) the details of instantiation of the interface class `tati` setting the options.

```
import math
import numpy as np
import tensorflow
import TATi.simulation as tati

nn = tati(
    ....
)
options = nn.get_options()

def baoab_update_step(nn, momenta, old_gradients, preconditioner, step_width, beta, gamma,
    walker_index):
    def B(step_width, gradients):
        nonlocal momenta
        momenta -= .5 * step_width * np.dot( np.transpose(preconditioner), gradients)

    def A(step_width, momenta):
        nn.parameters[walker_index] += .5 * step_width * preconditioner.dot(momenta)

    def O(step_width, beta, gamma):
        nonlocal momenta
        alpha = math.exp(-gamma * step_width)
        momenta = alpha * momenta + \
                  math.sqrt((1. - math.pow(alpha, 2.)) / beta) * np.random.standard_normal(momenta
                      .shape)

    B(step_width, old_gradients)
    A(step_width, momenta)
    O(step_width, beta, gamma)
    A(step_width, momenta)
    gradients = nn.gradients(walker_index=walker_index)
    B(step_width, gradients)

    return gradients, momenta

preconditioner = [np.identity((nn.num_parameters())) for i in range(options.number_walkers)]
normalization = 1./(float(nn.num_walkers()) - 1.)

def update_preconditioner():
    for walker_index in range(nn.num_walkers()):
        means = normalization*np.add.reduce([nn.parameters[i] for i in range(nn.num_walkers()) if
            i != walker_index])
        covariance = normalization*np.add.reduce(\
            [np.outer(nn.parameters[i] - means, nn.parameters[i] - means) \
            for i in range(nn.num_walkers()) if i != walker_index])
        preconditioner[walker_index] = np.linalg.cholesky( \
            options.covariance_blending * covariance + np.identity((nn.num_parameters())))

momenta = [np.zeros((nn.num_parameters())) for i in range(options.number_walkers)]
old_gradients = [np.zeros((nn.num_parameters())) for i in range(options.number_walkers)]

def perform_step():
    for walker_index in range(nn.num_walkers()):
        old_gradients[walker_index], momenta[walker_index] = baoab_update_step(
            nn, momenta[walker_index], old_gradients[walker_index],
            preconditioner=preconditioner[walker_index],
            step_width=options.step_width,
            beta=options.inverse_temperature, gamma=options.friction_constant, walker_index=
                walker_index)

for step in range(options.max_steps):
    if (step) % options.covariance_after_steps:
        update_preconditioner()
    perform_step()
```

The full implementation with *TensorFlow* in TATi requires special care with the conditionals for the infrequent updates of B_i, needs to copy the network parameters to avoid changes within the parallel execution, and needs to compute the covariance matrices. All implemented samplers have been adapted in a similar way as in (18) to allow for precondi-

tioning. For performance reasons the computation of $B_i^{(n)}$ could be done entirely through rank-1 updates.

References

P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.

R. Baldock and N. Marzari. Bayesian neural networks at finite temperature. 2019. arXiv:1904.04154.

A.J. Ballard, R. Das, S. Martiniani, D. Mehta, L. Sagun, J.D. Stevenson, and D.J. Wales. Energy landscapes for machine learning. *Phys. Chem. Chem. Phys.*, 19:12585–12603, 2017.

D. Barber and C. M. Bishop. Ensemble learning in bayesian neural networks. *Neural Networks and Machine Learning*, 168:215–238, 1998.

R. Bernardi, M. Melo, and K. Schulten. Enhanced sampling techniques in molecular dynamics simulations of biological systems. *Biochimica et Biophysica Acta*, 1850(5):872 – 877, 2015.

A. Beskos and A. Stuart. Computational complexity of Metropolis-Hastings methods in high dimensions. In Pierre L' Ecuyer and Art B. Owen, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2008*, pages 61–71. Springer, 2009.

P.G. Bolhuis, D. Chandler, C. Dellago, and P.L. Geissler. Transition path sampling: throwing ropes over rough mountain passes, in the dark. *Annual Review of Physical Chemistry*, 53(1):291–318, 2002.

D. Bone, M. Goodwin, M. Black, C.-C. Lee, K. Audhkhasi, and S. Narayanan. Applying machine learning to facilitate autism diagnostics: pitfalls and promises. *J. Autism Dev. Disord.*, 45:1121–1136, 2015.

L. Bottou, F. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

N. Bou-Rabee and H. Owhadi. Long-run accuracy of variational integrators in the stochastic context. *SIAM J. Num. Anal.*, 48(1):278–297, 2010.

P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina. Entropy-sgd: biasing gradient descent into wide valleys. *J. Stat. Mech.*, 2019(12):124018, 2019.

T. Chen, E. Fox, and C. Guestrin. Stochastic gradient hamiltonian monte carlo. In *31st International Conference on Machine Learning*, pages 1683–1691, 2014.

A. Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, and Y. LeCun. The loss surfaces of multilayer networks. *J. Mach. Learn. Res.*, 38:192–204, 2015.

P. Collet, S. Martinez, and J. San Martin. *Quasi-stationary distributions: Markov chains, diffusions and dynamical systems*. Springer, 2012.

L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. Sharp minima can generalize for deep nets. In D. Precup and Y.-W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1019–1028, 2017.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1309–1318, 2018.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. of Mach. Learn. Res.*, 12:2121–2159, 2011.

D.B. Dunson. Statistics in the big data era: Failures of the machine. *Stat. Probab. Lett.*, 136:4–9, 2018.

A. Durmus and E. Moulines. High-dimensional Bayesian inference via the Unadjusted Langevin Algorithm. *Bernoulli*, 25(4):2854–2882, 2019.

R. Elber. Perspective: computer simulations of long time dynamics. *J. Chem. Phys.*, 144: 060901, 2016.

Y. Gal. *Uncertainty in deep learning*. PhD thesis, Cambridge University, 2016.

I.J. Goodfellow, O. Vinyals, and A.M. Saxe. Qualitatively characterizing neural network optimization problems. In *Int. Conf. Learn. Represent.*, 2014.

J. Goodman. *ACOR package*, 2009. URL `http://www.math.nyu.edu/faculty/goodman/software/`.

J. Goodman and J. Weare. Ensemble samplers with affine invariance. *Commun. Appl. Math. Comput. Sci.*, 5(1):65–80, 2010.

C.L. Gómez, I. Santos, J.G. de la Puerta, P.G. Bringas, and P. Galán-García. Supervised machine learning for the detection of troll profiles in twitter social network: application to a real case of cyberbullying. *Logic Journal of the IGPL*, 24(1):42–53, 10 2015.

B.D. Haeffele and R. Vidal. Global optimality in neural network training. *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, 2017-Janua(3):4390–4398, 2017.

M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *33rd International Conference on Machine Learning, New York, NY*, 2016. JMLR: Workshops and Conference Proceedings, Vol. 48.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, pages 770–778, 2016.

G. Hinton, N. Srivastava, K. Swervsky, and T. Tieleman. rmsprop: Divide the gradient by a running average of its recent magnitude, 2014. unpublished.

G.E. Hinton and D. Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM, 1993.

S. Hochreiter and J. Schmidhuber. Flat Minima. *Neural Comput.*, 9(1):1–42, 1997.

E. Hoffer, I. Hubara, and D. Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Adv. Neur. Inf. Proc. Sys. 31*, 2017.

D.J. Im, M. Tao, and K. Branson. An empirical analysis of the optimization of deep network loss surfaces. 2017. arXiv:1612.04010.

K. Johnson, J. Torres Soto, B. Glicksberg, K. Shameer, R. Miotto, M. Ali, E. Ashley, and J. Dudley. Artificial intelligence in cardiology. *Journal of the American College of Cardiology*, 71:2668–2679, 2018.

K. Kadau, T.C. Germann, and P.S. Lomdahl. Large scale molecular dynamics simulation of 19 billion particles. *International Journal of Modern Physics C*, 15(01):193–201, 2004.

K. Kawaguchi, L. Kaelbling, and Y. Bengio. Generalization in deep learning. 2020. arXiv:1710.05468v6.

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

M.L. Klein and W. Shinoda. Large-scale molecular dynamics simulations of self-assembling systems. *Science*, 321(5890):798–800, 2008.

A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. In *Adv. Neur. Inf. Proc. Syst. 25*, pages 1097–1105, 2012.

Y. LeCun. MNIST dataset. URL `http://yann.lecun.com/exdb/mnist/`. accessed 2018.

Y. LeCun, I. Kanter, and S.A. Solla. Eigenvalues of covariance matrices: Application to neural-network learning. *Phys. Rev. Lett.*, 66(18):2396–2399, 1991.

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

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

M. Ledoux. *The concentration of measure phenomenon*. Mathematical surveys and monographs ; no. 89. American Mathematical Society, Providence, R.I., 2001.

B. Leimkuhler and C. Matthews. Rational construction of stochastic numerical methods for molecular sampling. *Appl. Math. Res. eXpress*, 2013(1):34–56, 2012.

B. Leimkuhler and C. Matthews. *Molecular Dynamics*. Springer International Publishing, Heidelberg, 1st edition, 2015.

B. Leimkuhler, C. Matthews, and G. Stoltz. The computation of averages from equilibrium and nonequilibrium Langevin molecular dynamics. *IMA J. Numer. Anal.*, 36(1):13–79, 2015.

B. Leimkuhler, C. Matthews, and J. Weare. Ensemble preconditioning for Markov chain Monte Carlo simulation. *Stat. Comput.*, 28(2):277–290, 2018.

B. Leimkuhler, C. Matthews, and T. Vlaar. Partitioned integrators for thermodynamic parameterization of neural networks. *Foundations of Data Science*, 1:457–489, 2019. doi: 10.3934/fods.2019019.

T. Leliévre, M. Rousset, and G. Stoltz. *Free Energy Computations*. Imperial College Press, 2010.

H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. In *Adv. Neur. Inf. Proc. Sys. 32*, 2018.

R. Livni, S. Shalev-Shwartz, and O. Shamir. On the computational efficiency of training neural networks. In *Adv. Neur. Inf. Proc. Syst. 27*, pages 855–863, 2014.

D.J. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4:448–472, 1992.

E Marinari and G Parisi. Simulated tempering: a new monte carlo scheme. *Europhysics Letters (EPL)*, 19(6):451–458, jul 1992.

Anton Martinsson, Jianfeng Lu, Benedict Leimkuhler, and Eric Vanden-Eijnden. The simulated tempering method in the infinite switch limit with adaptive weight learning. *J. Stat. Mech.*, 2019(1):013207, jan 2019.

C. Matthews and J. Weare. Langevin Markov Chain Monte Carlo with stochastic gradients. 2018. arXiv:1805.08863.

F. Mezzadri. How to generate random matrices from the classical compact groups. *Notices of the AMS*, 54:592–604, 2007.

R.M. Neal. Bayesian training of backpropagation networks by the hybrid monte carlo method. Technical Report CRG-TR-92-1, Dept. of Computer Science, University of Toronto, 1992.

J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1. edition, 1999.

S. Patterson and Y.-W. Teh. Stochastic gradient Riemannian Langevin dynamics on the probability simplex. In *Adv. Neur. Inf. Proc. Syst. 26*, pages 3102–3110, 2013.

C.P. Robert. *The Metropolis-Hastings algorithm*, pages 1–15. American Cancer Society, 2015.

Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same? *Neural Computation*, 16(5):1063–1076, 2004.

Itay Safran and Ohad Shamir. On the quality of the initial basin in overspecified neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 774–782. JMLR.org, 2016.

N. Sebe, I. Cohen, A. Garg, and T.S. Huang. Machine learning in computer vision. In *Computational Imaging and Vision*, 2005.

X. Shang, Z. Zhu, B. Leimkuhler, and A. Storkey. Covariance-controlled adaptive Langevin thermostat for large-scale Bayesian sampling. In *Adv. Neural Inf. Process. Syst. 28*, pages 37–45, 2015.

R. Smith. *Uncertainty quantification: theory, implementation, and applications.* SIAM, 2013.

M. Soltanolkotabi, A. Javanmard, and J.D. Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 65(2):742–769, Feb 2019.

Conghui Tan, Shiqian Ma, Yu-Hong Dai, and Yuqiu Qian. Barzilai-Borwein step size for stochastic gradient descent. In *Adv. Neural Inf. Process. Syst. 29*, 2016.

Z. Trstanova, B. Leimkuhler, and T. Lelièvre. Local and global perspectives on diffusion maps in the analysis of molecular systems. *Proceedings of the Royal Society A*, 476(2233): 20190036, 2020.

S.J. Vollmer, K.C. Zygalakis, and Y.-W. Teh. Exploration of the (non-) asymptotic bias and variance of stochastic gradient langevin dynamics. *J. Mach. Learn. Res.*, 17(1):5504–5548, 2016.

M. Welling and Y.-W. Teh. Bayesian learning via stochastic gradient langevin dynamics. *Proc. 28th Int. Conf. Mach. Learn.*, pages 681–688, 2011.

F. Wenzel, K. Roth, B.S. Veeling, J. Swiatkowski, L. Tran, S. Mandt, J. Snoek, T. Salimans, R. Jenatton, and S. Nowozin. How good is the bayes posterior in deep neural networks really? In *Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119*, 2020.

T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing. *CoRR*, abs/1708.02709, 2017.

G. Zhao, J.R. Perilla, E.L. Yufenyuy, X. Meng, B. Chen, J. Ning, J. Ahn, A.M. Greenborn, K. Schulten, C. Aiken, and P. Zhang. Mature hiv-1 capsid structure by cryo-electron microscopy and all-atom molecular dynamics. *Nature*, 497:643–646, 2013.