Controlled Langevin dynamics for sampling of feedforward neural networks trained with minibatches

OPEN ACCESS

Journal of Statistical Mechanics: Theory and Experiment
An IOP and SISSA journal

PAPER Interdisciplinary statistical mechanics

Controlled Langevin dynamics for sampling of feedforward neural networks trained with minibatches

Alessandro Zambon[1], **Francesca Caruso**[2], **Riccardo Zecchina**[2,*] **and Guido Tiana**[1,*]

[1] Department of Physics, Universitá degli Studi di Milano and INFN, via Celoria 16, 20133 Milano, Italy
[2] Department of Computing Sciences and Bocconi Institute for Data Science and Analytics (BIDSA), Bocconi University, 20136 Milano, Italy
E-mail: riccardo.zecchina@unibocconi.it and guido.tiana@unimi.it

Received 15 June 2026
Accepted for publication 15 June 2026
Published 9 July 2026

Online at stacks.iop.org/JSTAT/2026/073401
https://doi.org/10.1088/1742-5468/ae8246

Abstract. Sampling the parameter space of artificial neural networks according to a Boltzmann distribution provides insight into the geometry of low-loss solutions and offers an alternative to conventional loss minimization for training. However, exact sampling methods such as hybrid Monte Carlo (hMC), while formally correct, become computationally prohibitive for larger datasets because they require repeated evaluation of full–batch gradients. We introduce a pseudo–Langevin (pL) dynamics that enables efficient Boltzmann sampling of feed-forward neural networks trained with large datasets by using mini–batches in a controlled manner. The method exploits the statistical properties of mini–batch gradient noise and adjusts fictitious masses and friction coefficients to ensure that the induced stochastic process samples efficiently the desired equilibrium distribution. We validate numerically the approach by comparing its equilibrium statistics with those obtained from exact hMC sampling. Performance benchmarks demonstrate that, while hMC rapidly becomes inefficient as network

*Authors to whom any correspondence should be addressed.

size increases, the pL scheme maintains the best computational diffusion among exact and non–exact methods and scales favorably to networks with over one million parameters. Additionally, we show that sampling at intermediate temperatures yields optimal generalization performance, comparable to stochastic gradient descent, without requiring a validation set or early stopping procedure. Finally, we show that our sampling method can achieve optimal generalization on the MNIST dataset, independently of the architecture. These results establish controlled mini–batch Langevin dynamics as a practical and scalable tool for exploring and exploiting the solution space of large neural networks.

Keywords: artificial neural networks, Boltzmann measure

Contents

1. Introduction **2**

2. Methods **3**
2.1. The model and task 4
2.2. State–of–the–art MC algorithms 5
2.3. A pL algorithm 6
2.4. pL computational complexity 11

3. Results **12**
3.1. Numerical validation of the pL method 12
3.2. Better performance of the pL algorithm 14
3.3. Sampling with pL algorithm at intermediate temperatures provides optimal generalization properties 16
3.4. Test on realistic data 18

4. Discussion and conclusions **19**

Appendix **21**

References **25**

1. Introduction

The parameters that enable an artificial neural network (ANN) to make predictions are usually obtained by minimizing a loss function with a stochastic gradient descent (SGD) algorithm. However, exploring the low–loss manifold at large has proven useful for finding parameters with better generalization properties and for understanding the fundamental principles underlying the operation of the network. For example, sampling the space of parameters of a tree committee machine with hybrid Monte Carlo (hMC)

J. Stat. Mech. (2026) 073401

[1] elucidated that low–loss parameters are arranged according to a spiky topology close to the interpolation threshold [2]. A replica algorithm allowed the identification of wide minima of the loss function, which were shown to generalize more effectively than those identified using SGD [3–5]. Analogously, Bayesian learning, which requires sampling of the parameter space of the ANN, shows several advantages with respect to standard training methods [6], such as improved generalization capacity [7]. All these results suggest that sampling the solution space could be an alternative to straightforward loss minimization through SGD, even for the routine training of an ANN.

The main problem when sampling the parameter space of neural networks is the efficiency of the sampling algorithm. The hMC method employed in [2] has the virtue of sampling according to the desired Boltzmann distribution, while the elementary steps are guided by gradient–based molecular dynamics, which are critical to navigate such a high–dimensional space. However, this algorithm is difficult to use with training sets of larger size because calculating the gradient becomes lengthy. When the task is just loss minimization, the use of mini–batches tries to solve this problem; however, mini–batches cannot be applied in a straightforward manner to the Monte Carlo scheme as they would violate the detailed balance requirement.

An alternative approach is to devise efficient, albeit approximate, sampling schemes based on Langevin–like equations that use mini–batches to facilitate efficient exploration, even in the presence of large training sets. Langevin equations define a stochastic process in the space of parameters that converges to the Boltzmann distribution for a vanishing integration step size [8]. This approach has previously been employed to sample the posterior probability of parameters in the Bayesian training of ANNs [9–11], with various characterizations of the noise injected into the system during the dynamics.

We developed a Langevin–like approximate algorithm which samples efficiently the space of parameters of an ANN according to Boltzmann weight. This is achieved by controlling the noise generated when estimating the loss function with mini–batches. First, we validated the method against the prediction obtained through hMC sampling on a small–sized feedforward neural network. Next, we compared the diffusion properties of our algorithm with those of other exact and approximate sampling schemes at different network sizes. Furthermore, we investigated the thermodynamic properties of a large feedforward ANN, revealing a correlation between generalization and mini–batch noise variance. Finally, we tested our sampling method as an optimizer on real data, comparing its performance with that of standard SGD optimization techniques.

2. Methods

The following sections provide descriptions of the models used to validate the new sampling method and the task which defines the energy landscape. Next, we rapidly introduce the state–of–the–art sampling schemes used for comparison within this work. Lastly, we describe our new pseudo–Langevin (pL) algorithm.

J. Stat. Mech. (2026) 073401

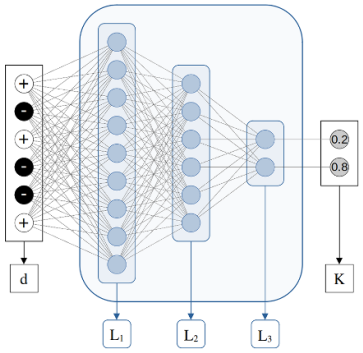

Figure 1. Sketch of the feed–forward ANN used in this work, with (from left to right) L_1, L_2 and L_3 neurons for the first, second and third layer respectively. On the left and on the right of the ANN, a possible couple of input and output vectors respectively, that is a d–dimensional spin vector and a normalized probability distribution on the $K = L_3$ classes.

2.1. The model and task

We studied a standard feed–forward ANN, made of three fully–connected layers of size L_1, L_2 and L_3, respectively, with ReLU activation functions for all neurons (figure 1). This can be described as

$$\begin{aligned} \boldsymbol{a}^{(l)} &= W^{(l,l-1)} \cdot \boldsymbol{z}^{(l-1)} + \boldsymbol{b}^{(l)} \\ \boldsymbol{z}^{(l)} &= \max\left[0, \boldsymbol{a}^{(l)}\right] \end{aligned} \tag{1}$$

with $W^{(l,l-1)} \in \mathbb{R}^{L_l \times L_{l-1}}$ and $\boldsymbol{b}^{(l)} \in \mathbb{R}^{L_l}$ (for $l = 1,2,3$) are the learnable weights of the network. In addition, $\boldsymbol{z}^{(0)} \equiv \boldsymbol{x}$ is the input vector, while $\hat{\boldsymbol{y}} = \mathrm{softmax}(\boldsymbol{z}^{(L_3)})$ is the output probability distribution. We will refer to the entire set of parameters of the network as the weight vector $\boldsymbol{w}$.

The task we want to address is a classification. We consider a dataset of examples $\mathcal{D} = \{(\boldsymbol{x}^\mu, y^\mu)\}_{\mu=1}^P$, where $\boldsymbol{x}^\mu \in \{\pm 1\}^d$ is a d–dimensional binary spin vector, and $y^\mu \in [0, 1, \ldots, K-1]$ is an integer, with K being the number of classes. Each class is associated with a reference vector $\boldsymbol{v}^{(k)}$, extracted from a flat distribution over the space $\{\pm 1\}^d$. Each input–label couple in the dataset is generated independently through a three–steps process: first, the label y^μ is extracted from a flat distribution over the classes indexes. Then, the input $\boldsymbol{x}^\mu$ is initialized as the reference spin vector $\boldsymbol{v}^{(y^\mu)}$ associated with the class y^μ. Lastly, each input component x_i^μ is flipped with probability p_f. The data generation process is summarized by

$$\begin{aligned} y^\mu &\sim K^{-1} \sum_{k=1}^{K} \delta_{y^\mu,k} \\ x_i^\mu &\sim (1-p_f)\, \delta_{x_i^\mu, v_i^{(y^\mu)}} + p_f\, \delta_{x_i^\mu, -v_i^{(y^\mu)}} \quad \forall i, \end{aligned} \tag{2}$$

J. Stat. Mech. (2026) 073401

where the notation $x \sim \rho(x)$ indicates that x is extracted from the distribution $\rho(x)$. In order to adapt the ANN presented in equation (1) to the current task, we will keep $L_2 = d$ and $L_3 = K$ fixed, while the dimension of the network N (i.e. the size of the weight vector $\boldsymbol{w}$) is determined by the number of neurons in the first hidden layer L_1.

Given the task at hand, we characterize each weight vector with a potential function composed of two terms, namely the mean cross–entropy loss function and a L^2 regularization term,

$$U(\boldsymbol{w}|\mathcal{D}) = -\frac{1}{P}\sum_{\mu=1}^{P}\sum_{k=1}^{K}\delta_{y^\mu,k}\ln\hat{y}_k^\mu + \frac{\lambda}{2N}\sum_{i=1}^{N}w_i^2. \tag{3}$$

In this study, we keep the ratio $\alpha = P/N$, called 'constrained density', fixed while increasing the size of the network N. For this reason, we considered an effective Lagrange multiplier λ/N, to make both the loss function and the regularization term independent of the scale of the system. Furthermore, another observable we can use to characterize each weight vector $\boldsymbol{w}$ is the fraction of misclassified input–output couples, that is,

$$\epsilon(\boldsymbol{w}|\mathcal{D}) = \frac{1}{P}\sum_{\mu=1}^{P}1-\delta_{y^\mu,\hat{k}^\mu}, \tag{4}$$

where $\hat{k}^\mu = \mathrm{argmax}_k\,\hat{y}_k^\mu$ is the most probable predicted class. From now on, we will no longer explicitly indicate dependence on the dataset $\mathcal{D}$ for the potential and error functions.

2.2. State–of–the–art MC algorithms

One way to sample the Boltzmann distribution over the space of parameters of an ANN is to implement the hMC algorithm [1, 2], which is exact in the limit of a large number of moves.

Operatively, at the nth iteration, the momenta $\boldsymbol{p}_n$ associated with the current weight vector $\boldsymbol{w}_n$ are extracted from a Gaussian $\mathcal{N}(0,\mathcal{M}T)$, where $\mathcal{M}$ is a diagonal matrix of (fictitious) masses and T is the temperature in the weight space (we also define $\beta \equiv T^{-1}$). Next, a time–reversible integration scheme is used to calculate an approximate solution of the equations of motion associated with the Hamiltonian

$$H(\boldsymbol{w},\boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}^T\mathcal{M}^{-1}\boldsymbol{p} + U(\boldsymbol{w}), \tag{5}$$

for a total time Δt, where the initial conditions are $(\boldsymbol{w}(0),\boldsymbol{p}(0)) \equiv (\boldsymbol{w}_n,\boldsymbol{p}_n)$. Lastly, the final weight vector $\boldsymbol{w}(\Delta t)$ is accepted as the new starting configuration with probability $\min[1,\exp(-\Delta H/T)]$, where $\Delta H = H[\boldsymbol{w}(\Delta t),\boldsymbol{p}(\Delta t)] - H[\boldsymbol{w}(0),\boldsymbol{p}(0)]$ is the total energy variation along the trajectory in the phase space.

The strength of this algorithm is that the energy gradient chooses the optimal direction for the proposed move, which is critical in such high–dimensional spaces. However,

J. Stat. Mech. (2026) 073401

this is also its main drawback, since the calculation of the gradient over the whole set of examples $\mathcal{D}$ has a large computational cost.

One way to mitigate this cost is to perform the dynamics using the potential energy computed on a single mini–batch of size $S \ll P$, that is a small subset of $\mathcal{D}$ selected at random at the beginning of each hMC step. At the end of the integration, the acceptance procedure would then be carried out using the kinetic energy difference derived from the integration and the potential energy difference calculated on the whole dataset, to guarantee the sampling of the correct distribution. We call this algorithm mini–batch hMC. However, the energy landscape defined by a stochastic mini–batch can only locally approximate the desired potential. As a consequence, a very small Δt is required to prevent the Metropolis acceptance probability from becoming negligible, which makes the mbhMC also inefficient (see section 3.2).

Over the years, numerous approximated sampling schemes have been proposed as an alternative to the exact hMC algorithm [9–11]. In this work, we consider two in particular: preconditioned Stochastic gradient Langevin dynamics (pSGLDs) [11] and stochastic gradient hMC (SGHMC) [10].

Like mbhMC, these two algorithms rely on computing the energy gradient on small stochastic mini–batches to guide the system's trajectory. However, unlike mbhMC, they implement Langevin–like dynamics (see section 2.3) instead of an MH acceptance criterion at the end of each move (which would guarantee the correctness of the sampling). The system moves within a viscous medium and additional noise, injected at each time step, determines the temperature at which the sampling is performed. The properties of this external noise result in significant performance differences between the two algorithms (see section 3.2). In this work, we implement these two schemes following the recommendation of the authors for gaining efficiency in practical cases, that is, we neglect the correction terms $\hat{B}_t$ for SGHMC [10] and $\Gamma(\boldsymbol{w})$ for pSGLD [11].

In the following section, we present our own approximated sampling scheme, pL.

2.3. A pL algorithm

As mentioned in section 2.2, another way of sampling the canonical ensemble is through Langevin dynamics, which is described by the following set of stochastic differential equations,

$$\begin{aligned} \mathrm{d}\boldsymbol{w}(t) &= \mathcal{M}^{-1}\boldsymbol{p}(t)\,\mathrm{d}t \\ \mathrm{d}\boldsymbol{p}(t) &= -\gamma\boldsymbol{p}(t)\,\mathrm{d}t - \nabla U(\boldsymbol{w}(t))\,\mathrm{d}t + \sqrt{2\mathcal{M}\gamma T}\mathrm{d}\boldsymbol{W}(t), \end{aligned} \tag{6}$$

where $\mathcal{M}$ is a diagonal mass matrix, $\gamma > 0$ is the friction coefficient and $\mathrm{d}W_i(t)$ is a Wiener process with $\langle \mathrm{d}W_i(t)\mathrm{d}W_j(t')\rangle = \delta_{i,j}\delta(t-t')$. An efficient way to solve equation (6) is to use the Bussi–Parrinello integrator [12]

$$\begin{aligned} \boldsymbol{w}(t+\delta t) &= \boldsymbol{w}(t) + \mathcal{M}^{-1}\boldsymbol{\Pi}(t)\,\delta t \\ \boldsymbol{\Pi}(t+\delta t) &= c_1^2\boldsymbol{\Pi}(t) - \frac{\left(1+c_1^2\right)\delta t}{2}\nabla U(\boldsymbol{w}(t+\delta t)) + \sqrt{1+c_1^2}C_2\boldsymbol{\mathcal{R}}(t+\delta t), \end{aligned} \tag{7}$$

J. Stat. Mech. (2026) 073401

where $\mathcal{R}_i(t) \sim \mathcal{N}(0,1)$ are independent gaussian random numbers, δt is the integration time step, $c_1 = \mathrm{e}^{-\gamma \delta t/2}$ accounts for the dissipation caused by the viscosity of the medium, $C_2^2 = (1-c_1^2)\mathcal{M}T$ is the variance matrix of the applied white noise at temperature T and $\mathbf{\Pi}(t)$ is a short–hand for

$$\mathbf{\Pi}(t) = c_1 \boldsymbol{p}(t) - \frac{\delta t}{2} \nabla U(\boldsymbol{w}(t)) + C_2 \boldsymbol{\mathcal{R}}(t). \tag{8}$$

However, this method still requires the computation of the energy gradient, so it suffers from the same computational inefficiency as the hMC. To address this issue, we evaluate the gradient on mini–batches in a controlled manner.

We define the set of example indexes of the dataset $\mathcal{D}$ as $\mathcal{I} = \{1, 2, \ldots, P\}$. For each given weight vector $\boldsymbol{w}$, we can associate with $\mathcal{I}$ a set $\mathcal{G}_i(\boldsymbol{w})$, containing the values assumed by the ith component of the energy gradient computed over all the examples in the dataset, that is

$$\mathcal{G}_i(\boldsymbol{w}) = \left\{ \nabla_i U^{(\mu)}(\boldsymbol{w}) \right\}_{\mu \in \mathcal{I}}, \tag{9}$$

where $U^{(\mu)}(\boldsymbol{w})$ is the potential function for a weight vector $\boldsymbol{w}$ evaluated on the μth example.

We now promote the example index μ to a stochastic variable extracted from a flat distribution over the set $\mathcal{I}$,

$$\mu \sim \mathrm{Unifom}(\mathcal{I}), \tag{10}$$

thus obtaining an estimate for the elements of the gradient on a single example

$$\nabla_i U^{(\mu)}(\boldsymbol{w}) \sim \mathrm{Unifom}(\mathcal{G}_i(\boldsymbol{w})). \tag{11}$$

Such an estimate satisfies the two following equations

$$\begin{aligned} &\langle \nabla_i U^{(\mu)}(\boldsymbol{w}) \rangle_\mu = \nabla_i U(\boldsymbol{w}) \\ &\mathrm{Var}\left[\nabla_i U^{(\mu)}(\boldsymbol{w}) \right] < \infty. \end{aligned} \tag{12}$$

The former is true because the function in equation (3) is intensive with respect to the number of examples P, and the latter is also true because the set $\mathcal{G}_i(\boldsymbol{w})$ is finite and bounded under the realistic assumption that U is differentiable almost everywhere.

A mini–batch b of size S can be regarded as a stochastic process $b = \{\mu_n | \mu_n \in \mathcal{I}\}_{n=1}^S$ and, analogously, the ith component of the energy gradient evaluated on such mini–batch is the mean of the other associated stochastic process $\{\nabla_i U^{(\mu_n)}(\boldsymbol{w}) | \mu_n \in \mathcal{I}\}_{n=1}^S$, that is,

$$\nabla_i U^{(b)}(\boldsymbol{w}) = S^{-1} \sum_{n=1}^{S} \nabla_i U^{(\mu_n)}(\boldsymbol{w}). \tag{13}$$

J. Stat. Mech. (2026) 073401

Although the set $\mathcal{G}_i(\boldsymbol{w})$ is generally composed of correlated values, the stochastic variables $\nabla_i U^{(\mu_n)}(\boldsymbol{w})$ are extracted independently from the same distribution $\mathrm{Unifom}\left(\mathcal{G}_i(\boldsymbol{w})\right)$. Together with the properties reported in equation (12), this justifies the application of the central limit theorem for $S \gg 1$; the ith component of the mini–batch gradient thus satisfies

$$\nabla_i U^{(b)}(\boldsymbol{w}) \sim \mathcal{N}\left(\nabla_i U(\boldsymbol{w}), \frac{\mathrm{Var}\left[\nabla_i U^{(\mu)}(\boldsymbol{w})\right]}{S}\right). \tag{14}$$

The result from equation (14) is valid for each component i and each weight vector $\boldsymbol{w}$, thus we can easily extend it to the vector $\nabla U^{(b)}$ by defining $\mathcal{V}(\boldsymbol{w}|S)$ as a diagonal matrix of the variances associated with each component of the mini–batch gradient. However, since the latter are not independent in principle, we also introduce the correlation matrix between the gradient components $\mathcal{C}(\boldsymbol{w}|S)$, such that $\mathcal{C}_{ii}(\boldsymbol{w}|S) = 0\ \forall i$. Therefore, the gradient of the potential energy calculated on a stochastic mini–batch b is distributed as follows,

$$\nabla U^{(b)}(\boldsymbol{w}) \sim \mathcal{N}\left(\nabla U(\boldsymbol{w}), \mathcal{V}(\boldsymbol{w}|S) + \mathcal{C}(\boldsymbol{w}|S)\right). \tag{15}$$

We can now substitute $\nabla U(\boldsymbol{w})$ in equation (7) with $\nabla U^{(b)}(\boldsymbol{w})$ by using the following identity,

$$\frac{\delta t}{2}\nabla U(\boldsymbol{w}) + \mathcal{N}\left(0, C_2^2\right) = \frac{\delta t}{2}\nabla U^{(b)}(\boldsymbol{w}) + \mathcal{N}\left[0, C_2^2 - \left(\frac{\delta t}{2}\right)^2 \left[\mathcal{V}(\boldsymbol{w}) + \mathcal{C}(\boldsymbol{w})\right]\right]. \tag{16}$$

The calculation can be made even more efficient. First of all, the estimation of the non–diagonal elements of mini–batch noise covariance matrix (i.e. $\mathcal{C}(\boldsymbol{w})$) can be computationally expensive for increasing N. Therefore, we can adjust the fictitious mass matrix $\mathcal{M}$ in such a way that

$$\mathcal{T}_i \equiv \left(\frac{\delta t}{2}\right)^2 \left[\left(C_2^2\right)^{-1}\mathcal{V}(\boldsymbol{w})\right]_{ii} = \frac{\delta t^2}{4(1-c_1^2)T}\left[\mathcal{M}^{-1}\mathcal{V}_\tau\right]_{ii} \ll 1 \quad \forall i \tag{17}$$

i.e. the error introduced by computing $\nabla U^{(b)}(\boldsymbol{w})$ instead of $\nabla U(\boldsymbol{w})$ is small compared to the order of magnitude of the total expected noise. We shall refer to this quantity $\mathcal{T}_i$ as 'temperatures ratio', and the condition expressed in equation (17) as the 'low temperatures ratio' (LTR) limit. This condition, along with the Cauchy–Schwarz inequality

$$\left|\mathcal{C}_{ij}(\boldsymbol{w})\right| \leqslant \sqrt{\mathcal{V}_{ii}(\boldsymbol{w})\mathcal{V}_{jj}(\boldsymbol{w})} \ll \sqrt{\left[\left(\frac{\delta t}{2}\right)^{-2} C_2^2 - \mathcal{V}(\boldsymbol{w})\right]_{ii}\left[\left(\frac{\delta t}{2}\right)^{-2} C_2^2 - \mathcal{V}(\boldsymbol{w})\right]_{jj}}, \tag{18}$$

allows us to approximate the expression in equation (16) to

$$\frac{\delta t}{2}\nabla U(\boldsymbol{w}) + \mathcal{N}\left(0, C_2^2\right) \approx \frac{\delta t}{2}\nabla U^{(b)}(\boldsymbol{w}) + \mathcal{N}\left[0, C_2^2 - \left(\frac{\delta t}{2}\right)^2 \mathcal{V}(\boldsymbol{w})\right], \tag{19}$$

J. Stat. Mech. (2026) 073401

which does not require the calculation of either $\nabla U(\boldsymbol{w})$ or $\mathcal{C}(\boldsymbol{w})$. Of course, this simplification comes at the cost of checking periodically that the condition of equation (17) is verified along the trajectory.

Moreover, we can remove the slowness associated with the fact that in principle the matrix $\mathcal{V}(\boldsymbol{w})$ changes along with the current weight vector and thus should be continuously updated. To do so, we regard the noise variances of mini–batches as constants for short periods during the trajectory,

$$\mathcal{V}(\boldsymbol{w}(t)) \approx \mathcal{V}(\boldsymbol{w}(\tau)) \equiv \mathcal{V}_\tau \quad \forall t \in [\tau, \tau + \Delta)\,, \tag{20}$$

where Δ is the period after which the expected variances of the noise $\mathcal{V}_\tau$ are updated. The reason for this approximation is that the mini–batch noise is a small fraction of the total noise injected into the system (see equation (17)). Therefore, temporarily neglecting fluctuations of the order $\mathcal{O}(\mathcal{V}_\tau)$ should not have a significant impact on the sampling.

In section 3.1, we shall analyze the properties of the 'mini–batch noise', defined as

$$\boldsymbol{\mathcal{R}}_\tau^{(b)}(\boldsymbol{w}(t)) \equiv \mathcal{V}_\tau^{-\frac{1}{2}}\left[\nabla U^{(b)}(\boldsymbol{w}(t)) - \nabla U(\boldsymbol{w}(t))\right], \tag{21}$$

that is the difference between the gradient computed on a stochastic mini–batch and on the whole dataset, scaled by the last updated standard deviation matrix, and we will verify that the approximations made in equations (19) and (20) are verified along the dynamics.

Taking advantage of the Gaussian character of the noise generated by mini–batches under the simplifications of equations (19) and (20), we obtain the pL integrator,

$$\begin{aligned}
\boldsymbol{w}(t+\delta t) &= \boldsymbol{w}(t) + \mathcal{M}^{-1}\boldsymbol{\Pi}(t)\,\delta t \\
\boldsymbol{\Pi}(t+\delta t) &= c_1^2\boldsymbol{\Pi}(t) - \frac{\left(1+c_1^2\right)\delta t}{2}\nabla U^{(b_{t+\delta t})}(\boldsymbol{w}(t+\delta t)) \\
&\quad + \sqrt{\left(1+c_1^2\right)\left[K_\tau^2 - \left(\frac{c_1\delta t}{2}\right)^2\mathcal{V}_\tau\right]}\,\boldsymbol{\mathcal{R}}(t+\delta t)\,,
\end{aligned} \tag{22}$$

where b_t is the mini–batch selected at time t, $K_\tau^2 \equiv C_2^2 - \left(\frac{\delta t}{2}\right)^2\mathcal{V}_\tau$ is the variance matrix of the white noise added on top of the correlated one associated with the mini–batch b_t and, similarly to equation (8),

$$\boldsymbol{\Pi}(t) = c_1\boldsymbol{p}(t) - \frac{\delta t}{2}\nabla U^{(b_t)}(\boldsymbol{w}(t)) + K_\tau\boldsymbol{\mathcal{R}}(t)\,. \tag{23}$$

which is an auxiliary variable employed during the integration in–between two updates to perform a single mini–batch gradient evaluation per–step.

J. Stat. Mech. (2026) 073401

We here make a couple of observations; first of all, one should note that equation (22) is valid only if

$$K_\tau^2 - \left(\frac{c_1 \delta t}{2}\right)^2 \mathcal{V}_\tau \geqslant 0 \Rightarrow \mathcal{T}_i \leqslant \frac{1}{1+c_1^2} \ \forall i, \tag{24}$$

meaning that, in general, $\mathcal{T}_i \leqslant 0.5$ (since $c_1 \in (0,1)$). Thus, this condition is actually already included in the LTR condition of equation (17). We would like to emphasize that equation (24) does not impose a constraint on the mini–batch size, but rather on the maximum value of the temperatures ratios given a certain viscosity of the medium. Reducing the mini–batch size clearly increases the variances associated to mini–batch gradients, but at the same time the variance of the injected white noise also increases proportionally according to the temperatures ratio (see equation (17)). This way, if the update period is adjusted properly, the approximations made in equations (19) and (20) remain valid.

Secondly, a key point is that, in principle, the Bussi–Parrinello integrator of equation (7) samples asymptotically the correct equilibrium distribution only in the limit $\delta t \to 0$. Thus, since the effective mini–batch variance scales quadratically with δt (see equation (16)), one could manually set the values of the mass matrix $\mathcal{M}$ to some arbitrary value and employ a very small integration time–step [10]. However, this choice does not take into account that the different components of the weight vector contribute differently to the total expected noise at a fixed temperature. A wiser choice is based on the observation that equation (7) is a good approximation of the stochastic differential equations equation (6) also if, given a finite step–size δt, the fictitious masses and the fictitious friction coefficient satisfy

$$\begin{aligned} &\frac{\delta t}{\sqrt{\mathcal{M}_{ii}}} \to 0 \ \forall i, \\ &\gamma \delta t \to 0. \end{aligned} \tag{25}$$

In terms of equation (22), these conditions can be expressed as

$$\begin{aligned} &\mathcal{T}_i \to 0 \ \forall i, \\ &c_1 \to 1, \end{aligned} \tag{26}$$

meaning that the limit from equation (17) not only guarantees the removal of the correlation between the components of the total noise, but also makes the dynamics produced by the pL algorithm to sample the correct equilibrium distribution (for a high–enough viscosity coefficient c_1).

In practice, at time t_0, the system is initialized; the matrix $\mathcal{V}_\tau$ (with $\tau = t_0$) is computed by extracting mini–batches of fixed size S until either the variance estimate converges for each gradient component or the maximum number of extractions is reached. Next, the initial values of the temperature ratios, $\mathcal{T} = \mathcal{T}_i$ (i.e. the same for each weight component), and the dissipative term, c_1, are set manually. The mass matrix is then calculated according to equation (17) using $\delta t = 1$. Finally, K_τ is computed and the

J. Stat. Mech. (2026) 073401

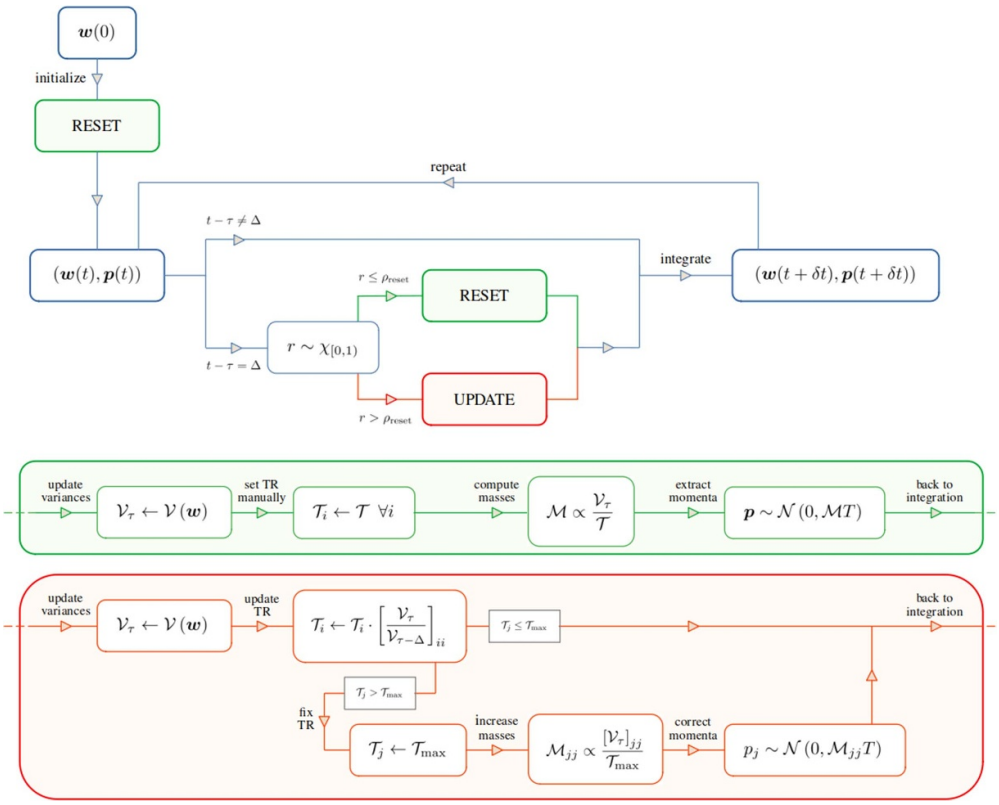

Figure 2. Scheme of the pseudo–Langevin (pL) algorithm.

momenta are extracted from the Gaussian distribution $\mathcal{N}(0, \mathcal{M}T)$. Once the initialization procedure has ended, a trajectory in phase space is generated using equation (22). At each time $t - \tau = \Delta$, the mini–batch noise variance matrix, $\mathcal{V}_\tau$, is adjourned using the same procedure as at time t_0. Then the system undergoes an 'update' or a 'reset' procedure, with probabilities $1 - \rho_{\text{reset}}$ and ρ_{reset}, respectively. In the first case, the current temperature ratios are updated (based on the new mini–batch variances) and the LTR condition is checked using a threshold value, i.e. a maximum temperature ratio, $\mathcal{T}_{\max}$, such that $\mathcal{T}_i \leqslant \mathcal{T}_{\max}\ \forall i$. If this inequality is not satisfied for a certain component j, its associated mass is increased until $\mathcal{T}_j = \mathcal{T}_{\max}$. The corresponding momentum is then extracted from the correct Gaussian distribution. In the event of a reset, the initialization procedure is carried out according to the last adjourned $\mathcal{V}_\tau$, with each temperature ratio being manually set to an user determined value. The algorithm pipeline is presented schematically in figure 2.

2.4. pL computational complexity

For a network of size N and a dataset of size P, the time complexity of the pL algorithm consists of two main contributions, both of which depend on the cost of estimating the mini–batch gradient, denoted as $c(S|N, P)$.

At each step, the cost of updating $(\boldsymbol{w}, \boldsymbol{p})$ is $\mathcal{O}(N + c(S|N, P))$. The first term accounts for operations involving N–dimensional vectors and the generation of white

J. Stat. Mech. (2026) 073401

noise with a zero mean and unit variance (see equation (22)). The latter refers to the computation of the mini–batch gradient for the current weight configuration.

In addition, we must consider the time required for the periodic update of the mini–batch noise variance matrix, $\mathcal{V}_\tau$, and the other related parameters (e.g. $\mathcal{T}$ and $\mathcal{M}$). We define n as the maximum number of mini–batch extractions permitted during estimation. In the worst–case scenario, when the variance estimate does not converge for each network component, the time complexity is at most $\mathcal{O}\left(\frac{n}{\Delta}\left(N + c(S|N,P)\right) + \frac{1}{\Delta}N\right)$. The $\frac{n}{\Delta}N$ term takes into account all the operations necessary to refine the estimate of the variances as the number of extractions increases, $\frac{n}{\Delta}c(S|N,P)$ represents the cost of computing the n mini–batch gradients to adjourn the variance matrix, while $\frac{1}{\Delta}N$ stands for the cost of updating the temperatures ratios, the masses and (eventually) the momenta (cf section 2.3) after a period Δ.

Therefore, given that usually $n \gg 1$, the average per–iteration complexity of the pL scheme is $\mathcal{O}\left((1+\frac{n}{\Delta})(N + c(S|N,P))\right)$, where the ratio $\frac{n}{\Delta}$ mainly depends on the state of the simulation. If the system is out of equilibrium, the variance matrix requires more frequent updating, meaning that in practice $\frac{n}{\Delta} \approx 1$. Conversely, when the system is at (or close to) equilibrium, the update interval can be extended, since slower changes are expected in the mini–batch gradient variances. In such cases, the ratio can be set so that $\frac{n}{\Delta} \ll 1$. In conclusion, the computational complexity of the pL scheme can generally be approximated to $\mathcal{O}\left(N + c(S|N,P)\right)$.

3. Results

We studied three feed–forward ANN models (figure 1) of size $N = 11\,160$, $N = 101\,610$ and $N = 1006\,110$, respectively. For each size, we generated a dataset of random spin vectors of dimension $d = 100$, divided in $K = 10$ classes (cf section 2.1). To make the classification problem non–trivial, we chose a spin–flip probability of $p_\mathrm{f} = 0.355$, so that the fraction of spin vectors closer to the reference vector of another class than to their own is approximately 0.1. We studied each model near the interpolation threshold, with a training dataset composed of $P = 5N$ examples. Lastly, the norm of the weights is controlled by the Lagrange multiplier $\lambda \approx 10$ (see equation (3)), to avoid uncontrolled diffusion.

3.1. Numerical validation of the pL method

We first used the smallest architecture ($N = 11\,160$) to validate the approximations described in section 2.3. For this purpose, we performed three independent short pL simulations at temperature $T = 10^{-6}$, mini–batch size $S = 10^{-2}P$ and temperatures ratio $\mathcal{T} = 3 \cdot 10^{-2}$. During each simulation, we periodically stopped the integration process to calculate the full–batch gradient and extract 250 mini–batch gradients $\{\boldsymbol{\mathcal{R}}_\tau^{(b)}(\boldsymbol{w}(t))\}$. Each simulation starts from a configuration obtained minimizing the loss using Adam as optimizer for 5000 epochs with mini–batch size $s = 128$ and initial learning rate $\eta = 10^{-4}$ (scaled by a factor of $\gamma = 0.8$ every 1000 epochs).

We observe that indeed the distribution of the noise generated by the use of mini–batches on the gradient follows a Gaussian distribution and that the true variance

J. Stat. Mech. (2026) 073401

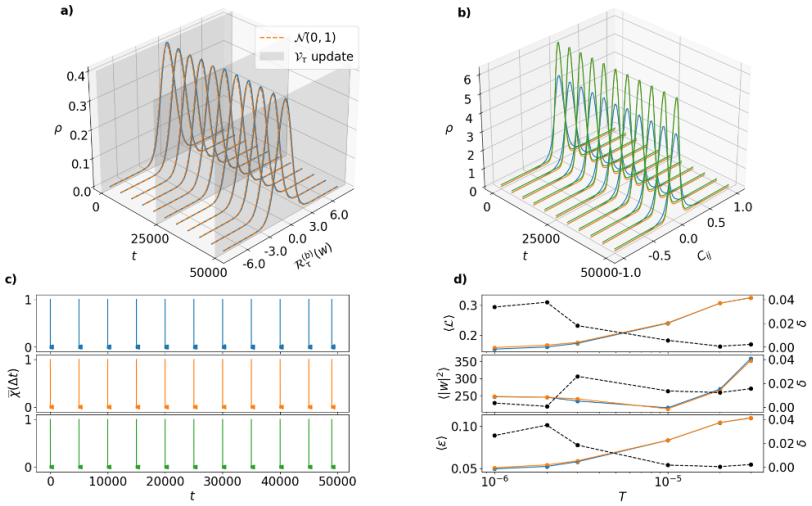

Figure 3. (a) The distribution ρ of the components of the mini–batch noise $\boldsymbol{\mathcal{R}}_\tau^{(b)}(\boldsymbol{w}(t))$ during a pL simulation as a function of time t (blue curves), compared to expected distribution $\mathcal{N}(0,1)$ (orange curves). The gray planes indicate the times at which the values of the mini–batch noise variance matrix $\mathcal{V}_\tau$ are updated. (b) The distribution ρ of the elements of the correlation matrix C_{ij} during a pL simulation as a function of time t, computed among the components of the mini–batch noise $\boldsymbol{\mathcal{R}}_\tau^{(b)}(\boldsymbol{w}(t))$ (blue curves), of the white noise $\boldsymbol{\mathcal{R}}(t)$ (orange curves) and of the weighted sum of the two (green curves). The curves are slightly shifted along the z–axis to make them distinguishable. (c) The average of the component–wise noise autocorrelation $\overline{\chi}(\Delta t)$ for the mini–batch noise (upper panel, blue curves), for the white noise (middle panel, orange curves) and for their weighted sum (lower panel, green curves), computed for small time intervals during a pL simulation parametrized by the time t. (d) The average of the mean cross–entropy $\mathcal{L}$ (upper plot), the squared norm $|\boldsymbol{w}|^2$ (center plot) and the training error ϵ (lower plot) as a function of the temperature T obtained from hMC (blue curves) and pL (orange curves) for the smallest model $N = 11\,160$. The relative error between the estimates is also reported (gray dotted line).

matrix $\mathcal{V}(\boldsymbol{w}(t))$ along the trajectory can be approximated, for sufficiently small time periods, with the last updated $\mathcal{V}_\tau$ (figure 3(a)). At each time t and for each gradient component i, we performed a Kolmogorov–Smirnov (KS) test on the distribution of extracted mini–batch noises $\{\boldsymbol{\mathcal{R}}_\tau^{(b)}(\boldsymbol{w}(t))\}$ (see equation (21)) to the standard normal distribution $\mathcal{N}(0,1)$. A fraction of 0.946 of the tests (averaged over t and i) displays a p–value > 0.05, which is a result comparable to what one would get if the KS tests were performed on truly Gaussian–distributed random variables.

Secondly, the LTR condition stated in equation (17) is sufficient to eliminate the correlation between the components of the total noise produced during integration. We calculated periodically the correlation matrix C_{ij} between the components of the mini–batch noise $\{\boldsymbol{\mathcal{R}}_\tau^{(b)}(\boldsymbol{w}(t))\}$ and the weighted sum of the latter with white noises $\{\boldsymbol{\mathcal{R}}(t)\}$, that is $\{\frac{\delta t}{2}\mathcal{V}_\tau^{\frac{1}{2}}\boldsymbol{\mathcal{R}}^{(b)}(\boldsymbol{w}(t)) + K_\tau\boldsymbol{\mathcal{R}}(t)\}$. The distribution of the elements of C_{ij} for

J. Stat. Mech. (2026) 073401

the white noise and for the weighted sum are indistinguishable (figure 3(b)), meaning that the components of the total noise used during integration are uncorrelated.

Moreover, the mini–batch noise is not self–correlated in time, which is another necessary property of the noise used within a Langevin scheme (figure 3(c)). In fact, the autocorrelation

$$\overline{\chi}(\Delta t) = N^{-1}\sum_{i=1}^{N}\langle w_i(t+\Delta t)\, w_i(t)\rangle - \langle w_i(t+\Delta t)\rangle\langle w_i(t)\rangle,$$

of the mini–batch noise as a function of the time interval Δt, averaged over the components of the weight vector and over the starting time t, drops to zero essentially at the first time step (figure 3(c)).

Finally, we verified that the probability distribution sampled by the pL algorithm is compatible with that of the hMC, assumed as ground truth. For this purpose, we equilibrated each of the trained vectors at different temperatures using both sampling methods, and we measured at each temperature the mean and the standard deviation for the loss function, for the squared norm of the weights and for the training error ϵ as predicted by both algorithms.

All the statistical properties of the model are very similar to each other (blue and orange curves in figure 3(d). We also calculated the relative error with respect to the hMC prediction, that is $|\langle O\rangle_T^{\mathrm{pL}} - \langle O\rangle_T^{\mathrm{hMC}}|/\langle O\rangle_T^{\mathrm{hMC}}$, for each observable O and temperature T. The prediction error is always of the order $\approx 2\cdot 10^{-2}$ and increases only slightly at low temperatures with respect to the mean loss function, where the system can get trapped in local minima, making the equilibration of the system problematic and the results dependent on the specific simulations.

3.2. Better performance of the pL algorithm

We compared the performance of the pL scheme with that of other sampling methods (see section 2.2), either exact (hMC and mbhMC) or approximate (SGHMC and pSGLD), for networks of different sizes (see section 3). Performance was quantified by measuring the diffusion of the weight vector at equilibrium in wall–clock time. The simulations were performed on a dedicated 24–core Intel Xeon W5-3423 processor equipped with a Nvidia RTX 4500 Ada graphics card for this test only.

For each sampling algorithm and for each model of size N, we performed several simulations at low temperature ($T = 10^{-6}$), at which the system learns the input data well (cf figure 3(d) and lower plot of figure 5), starting from different pre–equilibrated weight configurations. We scanned the values of the hyper–parameters of each algorithm and selected those that maximize $\overline{d^2}(\Delta t_{\mathrm{W}})$, which is the mean squared distance computed between all the pairs of weight vectors separated by a wall–clock time Δt_{W}. For the approximate sampling methods, we impose the additional constraint that the estimated average loss function lies within 2% from the hMC–predicted value (see figure A.1, panels in the first row).

Then, we performed three simulations with the optimal set of hyper–parameters, each starting from a different pre–equilibrated weight vector. All the results except for

J. Stat. Mech. (2026) 073401

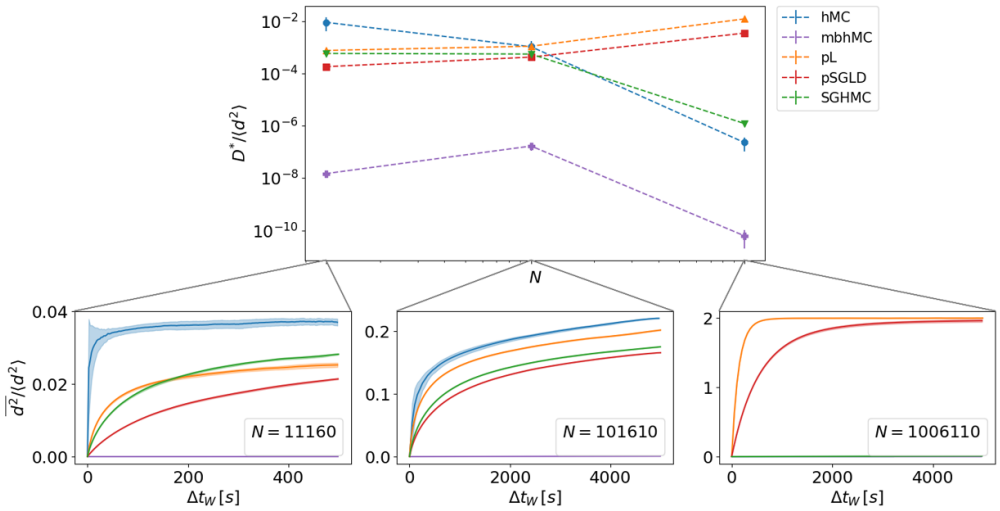

Figure 4. In the upper plot, the diffusion coefficient D^* scaled by the predicted squared distance $\langle d^2 \rangle$ at equilibrium at $T = 10^{-6}$ as a function of N, for the three studied sizes $N = 11\,160$, $N = 101\,610$ and $N = 1006\,110$ and for different sampling methods, hMC (blue curve), mbhMC (purple curve), pL (orange curve), pSGLD (red curve) and SGHMC (green curve). In the lower plots, the mean squared distance between sampled vectors $\overline{d^2}$ scaled by the predicted squared distance $\langle d^2 \rangle$ at equilibrium at $T = 10^{-6}$ as a function of the wall–clock time interval Δt_{W}, for the three sizes and for all the sampling methods listed above. The mean values and the standard deviations of the squared distance are calculated over three independent simulations starting from different equilibrated weight vectors. The error bars (in the upper plot) and the shaded areas around each curve (in the lower plots) represent the standard deviations. In some of the curves, the latter are so small that they cannot be seen with the naked eye.

hMC were obtained using a mini–batch size of $S = 10^{-2}P$. The results for smaller S relative to the network with $N = 1006110$ parameters are shown in figure A.2.

The hMC scheme shows a computational advantage for small network sizes (figure 4, lower left and lower center panels) because for small datasets the calculation of the full–batch gradient can be run in parallel, strongly reducing the advantage of calculating the gradient on a mini–batch of size $S \ll P$. All the approximated sampling schemes (i.e. SGHMC, pSGLD and pL) show comparable performance, with differences mainly due to how well the average value of the loss function is conserved with respect to the prediction given by hMC.

However, as N increases, the pL scheme clearly takes the lead (see figure 4, lower–right panel), followed by the pSGLD algorithm. Both algorithms reach a stationary state, whereas the other sampling schemes cannot move appreciably from their starting point. Moreover, we note that the mean squared norm obtained from pSGLD at $T = 10^{-6}$ is about 30% higher than that given by pL (see figure A.1, lower–right panel). Lastly, the mbhMC scheme performs the worst at every scale (figure 4, purple curves).

J. Stat. Mech. (2026) 073401

The computational efficiency of the algorithms can be quantified with a 'computational diffusion coefficient' D^*, which can be defined as the numerical derivative of the $\overline{d^2}$ curve for $\Delta t_{\mathrm{W}} \to 0$. Operatively, we measure

$$D^* = \frac{\overline{d^2}\left(\Delta t_{\mathrm{W}}^{\min}\right) - \overline{d^2}\,(0)}{\Delta t_{\mathrm{W}}^{\min}}$$

where $\Delta t_{\mathrm{W}}^{\min}$ is the minimum wall–clock time interval between sampled vectors, different for each algorithm (see table A1 in the appendix). As we can see in the upper panel of figure 4, for hMC and SGHMC the scaled computational diffusion coefficient $D^*/\langle d^2 \rangle$ drops rapidly as a function of the model size N, while it increases for pL and pSGLD.

These results suggest that the pL scheme can be used efficiently to sample the parameter space when the network size is increased.

3.3. Sampling with pL algorithm at intermediate temperatures provides optimal generalization properties

We investigated the generalization properties of the weights sampled with the pL algorithm at different temperatures, comparing them with those obtained from a SGD. We focused our attention to the largest network ($N = 1006\,110$), since it is the most interesting case in practical applications.

We calculated at different temperatures the average of the generalization error ϵ_{g} (cf equation (4)) on a test set of size $P_{\mathrm{g}} \approx 0.18P$, generated as described in section 2.1 using the same spin–flip probability and the same reference vectors as for the training set.

We observe that the average loss $\langle \mathcal{L} \rangle$ displays a sudden increase, reminiscent of a first–order phase transition, at $T_{\mathrm{f}} \approx 2 \cdot 10^{-7}$ (figure 5(b)). Interestingly, this transition seems to affect also the generalization properties of the sampled vectors, as the average generalization error is larger at very low temperatures and displays a sudden drop at the critical temperature at which the average loss increases (figure 5(a)). The average generalization error then increases, assuming its minimum value just above the transition temperature.

The minimal value $\langle \epsilon_{\mathrm{g}} \rangle \approx 0.113$ is very close to the lower bound that the generalization error can assume based on geometric arguments. As anticipated, for $p_{\mathrm{f}} = 0.355$, the probability that a spin vector of class μ is closer to the reference vector of another class ν than to its own is approximately ≈ 0.1, which implies that approximately a tenth of the spin vectors do not display the properties that define their own class (and thus cannot be classified correctly without explicit training on such data).

In the low–temperature phase, the system loses some of its generalization capabilities. For example, at $T = 10^{-7}$, the average prediction error in the training set is $\langle \epsilon \rangle \approx 0.014$ while it is $\langle \epsilon_{\mathrm{g}} \rangle \approx 0.17$ in the test set (figure 5(a)) suggesting that at these temperatures the system overfits the training data.

We also compared the results from the samplings with those obtained by training three independent models by SGD using the Adam algorithm (with a learning rate $\eta = 10^{-5}$) and early–stopping procedure (on a validation subset of the training data with size $P_{\mathrm{v}} \approx 0.18P$). Starting from random weights, both techniques minimize their error on

J. Stat. Mech. (2026) 073401

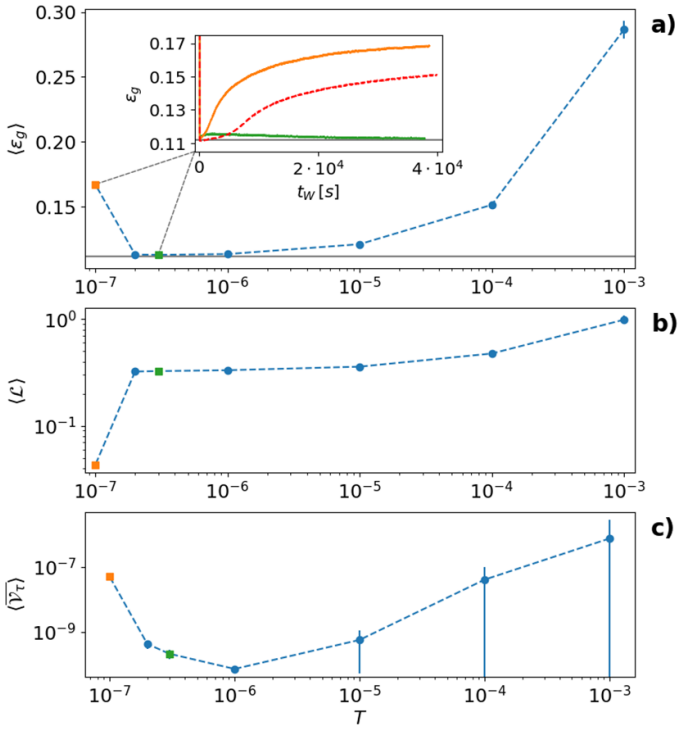

Figure 5. The mean value and the standard deviation of the generalization error ϵ_g (a), the mean cross–entropy function $\mathcal{L}$ (b) and the average of all the variances of the mini–batch gradient components in the first two layers $\overline{\mathcal{V}_\tau}$ (c) sampled at equilibrium at different temperatures T using the pL scheme. In the inserted plot, the generalization error ϵ_g as function of the wall–clock time $t_{\rm W}$ during two simulations starting from initialized models at two different temperatures, $T = 1.0 \cdot 10^{-7}$ (orange curve) and $T = 3.0 \cdot 10^{-7}$ (green curve), and during an Adam training beyond early–stopping (red dotted curve). The gray straight lines reported in the upper plot and in the inserted one represent the best mean generalization error found with Adam training. All values of $\epsilon_{\rm g}$ have been computed on the same test dataset.

the test set very rapidly (inset of figure 5(a)), as both descents are of the order of seconds. The best models found through standard training techniques have a mean generalization error $\langle \epsilon_{\rm g} \rangle_{\rm Adam} \approx 0.112$, but the latter increases markedly afterwards (inset of figure 5(a), red dotted curve) if early–stopping is not applied. Also, the optimal generalization error found by the Adam algorithm is similar to that of the pL sampling at intermediate temperature. However, the pL scheme does not require an early–stopping procedure (and thus a validation set) to avoid overfitting, as the generalization properties of the sampled weights are uniquely determined by the temperature T.

In fact, during the training (inset of figure 5(a)), the low–temperature ($T = 1.0 \cdot 10^{-7}$) simulation starting from random initialized vectors reaches a minimum of generalization error and rapidly departs from it, as the Adam algorithm does. Vice versa, during the sampling at intermediate temperature ($T = 3.0 \cdot 10^{-7}$), the system fluctuates steadily around the low equilibrium value without overfitting.

J. Stat. Mech. (2026) 073401

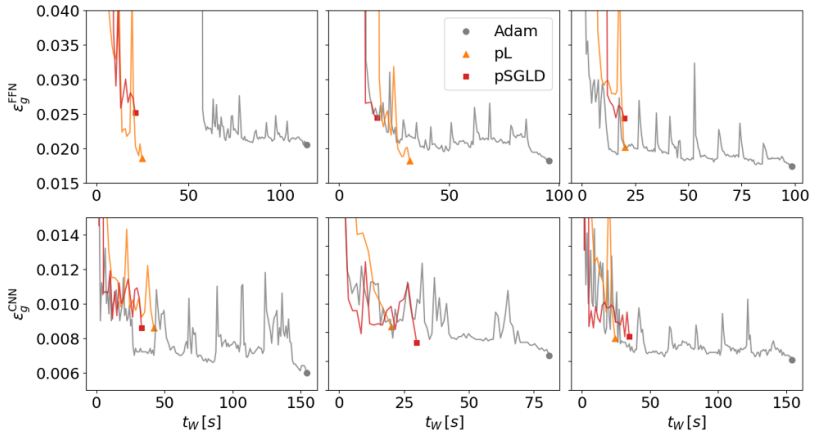

Figure 6. The generalization error ϵ_g computed on the test set of the MNIST dataset as a function of the wall–clock time t_W during Adam trainings (gray curves), pL samplings (orange curves) and pSGLD samplings (red curves) for three different initialization seeds (first, second and third column) and for two network architectures, feedforward (upper row) and convolutional (lower row). The curves in each panel stop at the minimum value in generalization error obtained during either the training or the sampling. The minimum test error is highlighted with a different marker for each algorithm.

Lastly, we observe a correlation between the generalization error $\langle \epsilon_g \rangle$ and the average of all the variances associated to weights in the first two layers $\langle \overline{\mathcal{V}_\tau} \rangle$ (which account for 99.4% of the network weights) as functions of the temperature T (figures 5(a) and (c)). The minimum values for each observable are reached in a temperature interval going from $T = 2 \cdot 10^{-7}$ (right above the phase transition) up to $T = 10^{-6}$, where the generalization error has a flat shape at approximately $\langle \epsilon_g \rangle \approx 0.113$.

Finally, in figure A.4, we report three simulations from different random initialized weight vector for the two most interesting temperatures, $T = 10^{-7}$ and $T = 3.0 \cdot 10^{-7}$ (figure 5, upper panel, orange and green dot, respectively), as proof of thermalization.

3.4. Test on realistic data

We tested the two sampling algorithms with the best diffusion properties, i.e. pL and pSGLD, on the MNIST dataset, composed of (28×28) images ($P = 60000$ for training and validation, and $P_g = 10000$ for evaluation) and $K = 10$ classes. We studied their optimization performance for two types of architectures, namely a feedforward (as described in section 2.1, with $L_1 = 784$) and a convolutional neural network (as the one described in [10]). For each of them, we performed samplings at $T = 5.0 \cdot 10^{-7}$ starting from three stochastically generated weight vectors, using a mini–batch size $S = 10^{-2}P$ and a regularization coefficient $\lambda/N = 10^{-5}$. For each seed, we also trained the model using Adam as optimizer along with the same regularization coefficient used for pL and pSGLD.

Both sampling schemes are basically equivalent (figure 6) in terms of time–efficiency and, in the CNN case, they both reach configurations with comparable generalization

J. Stat. Mech. (2026) 073401

properties ($\overline{\epsilon_{\rm g}^{\rm pSGLD}} = 7.87 \cdot 10^{-3}$ and $\overline{\epsilon_{\rm g}^{\rm pL}} = 8.20 \cdot 10^{-3}$, where $\overline{\epsilon_{\rm g}}$ stands for the minimum average test error over the optimizations performed). Conversely, pL shows a clear edge when used with the FFN architecture, where $\overline{\epsilon_{\rm g}^{\rm pL}} = 1.89 \cdot 10^{-2}$ and $\overline{\epsilon_{\rm g}^{\rm pSGLD}} = 2.46 \cdot 10^{-2}$.

For both architecture, the Adam algorithm is more time–consuming, but on average reaches configurations with lower generalization errors ($\overline{\epsilon_{\rm g}^{\rm Adam}} = 1.87 \cdot 10^{-2}$ for the FNN and (especially) $\overline{\epsilon_{\rm g}^{\rm Adam}} = 6.17 \cdot 10^{-3}$ for the CNN).

The same conclusions can be drawn from observing the optimization performance of the three algorithms on the synthetic spin dataset and for the largest three–layer feedforward ANN model used in section 2.1. The results of these optimizations for three independent runs are shown in figure A.5.

4. Discussion and conclusions

Sampling the parameter space of ANN can be useful in many ways. As well as providing insights into the topology of the solution space and how the system functions, sampling can be used to obtain sets of parameters with good predictive capabilities. Exploring the space of parameters according to a specified probability distribution can also be useful in connection with Bayesian networks, which have a marked tendency to remove overfitting as well [7].

Our findings indicate that the pL scheme seems the most well–fitted for sampling a large ANN according to Boltzmann probability. Monte Carlo techniques guarantee that the sampling converges to the desired distribution, but usually suffer the high dimensionality of the space, where random moves tend to lead to high–energy conformations, thus reducing drastically the acceptance probability. The hMC [1] amends this problem, proposing moves that are guided by the gradient and still obeys the principle of detailed balance. However, the exact calculation of the gradient is computationally demanding and limits the application of the hMC to systems whose training dataset is much smaller than those used in everyday applications. While loss minimization techniques solve the problem by using mini–batches, in hMC the estimation of the gradient on small subsets of the training data results in a negligibly small acceptance probability. On the other hand, using mini–batches to compute the gradient in approximate sampling methods does not necessarily lead to the desired performance for larger network sizes, as we demonstrated for the SGHMC scheme [10].

The pL algorithm combines the ability to sample asymptotically the correct distribution with the computational efficiency given by the use of mini–batches. We have shown that the system can diffuse efficiently in the manyfold of low–loss parameters, even for large systems trained on massive datasets. A key observation which allows the algorithm to work well is that the noise induced by the approximate estimation of gradients is Gaussian for mini–batches of nontrivial numerosity, thanks to the central limit theorem. Moreover, we can control the effect of this noise in the Langevin equations

J. Stat. Mech. (2026) 073401

through the optimal selection of fictitious masses and fictitious friction coefficients associated with the system's parameters. This allows us to sample the Boltzmann weight associated with the desired temperature without using a negligible timestep.

Interestingly, we observed a significant correlation between the variance of the gradient over mini–batches and the ability of the network to generalize. We can speculate that this is because the variance of the gradient in a given region of parameter space correlates with the flatness of the energy landscape; this, in turn, was shown to determine the generalization properties of the system [3], in the sense that parameters belonging to wider minima display better generalization properties than those in narrower ones, even with similar loss. The reason why we expect that the variance of the gradient correlates with flatness is that the energy profile of each stochastic mini–batch in flat minima share similar properties across mini–batches, including the gradient (as shown in [13]). As a matter of fact, the only other sampling scheme capable of providing significant diffusion is pSGLD [11], which uses explicitly the information about the flatness of the energy landscape to guide the trajectory in the weight space.

Using the pL algorithm, we have shown that sampling the Boltzmann distribution at intermediate temperatures yields sets of parameters with optimal generalization error, eliminating the need for early stopping procedures and the usage of a portion of the training dataset for validation. One possible problem when using pL as a training tool is the need to select 'intermediate' temperatures. In the low–temperature regime, the system overfits in a manner similar to that observed in gradient–descent minimization. At high temperatures, however, it fails to learn the training dataset. Nonetheless, at least in the case analyzed, the range of 'intermediate' temperatures spans several orders of magnitude, therefore should be easily identified.

Overall, our results suggest that controlled mini–batch Langevin dynamics provides a principled and scalable route to explore and exploit the low–loss manifold of large ANN, replacing optimization with equilibrium statistical mechanics.

The codes and the data can be downloaded freely from the following github repository.

J. Stat. Mech. (2026) 073401

Appendix

Table A1. The minimum wall–clock time interval between sampled vectors $\Delta t_{\mathrm{W}}^{\min}$ for all model sizes N (that is, $N = 11160$, $N = 101610$ and $N = 1006110$) and for all sampling schemes (i.e. hMC, mbhMC, pL, pSGLD and SGHMC). For pL, the first value within the parenthesis represents the time needed for a single integration, while the second one is the time spent to extract 100 mini–batches for the estimate of $\mathcal{V}_\tau$. With the exception of hMC, all the values reported are relative to a mini–batch of size $S = 10^{-2}P$.

N	hMC [s]	mbhMC [s]	pL [s]	pSGLD [s]	SGHMC [s]
11 160	$2.8 \cdot 10^{-3}$	$2.3 \cdot 10^{-3}$	$(1.6 + 1.1) \cdot 10^{-3}$	$1.7 \cdot 10^{-3}$	$1.4 \cdot 10^{-3}$
101 610	$5.1 \cdot 10^{-2}$	$2.6 \cdot 10^{-3}$	$(1.6 + 1.3) \cdot 10^{-3}$	$1.8 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$
1006 110	$3.2 \cdot 10^{0}$	$6.5 \cdot 10^{-2}$	$(3.3 + 3.3) \cdot 10^{-2}$	$3.3 \cdot 10^{-2}$	$3.3 \cdot 10^{-2}$

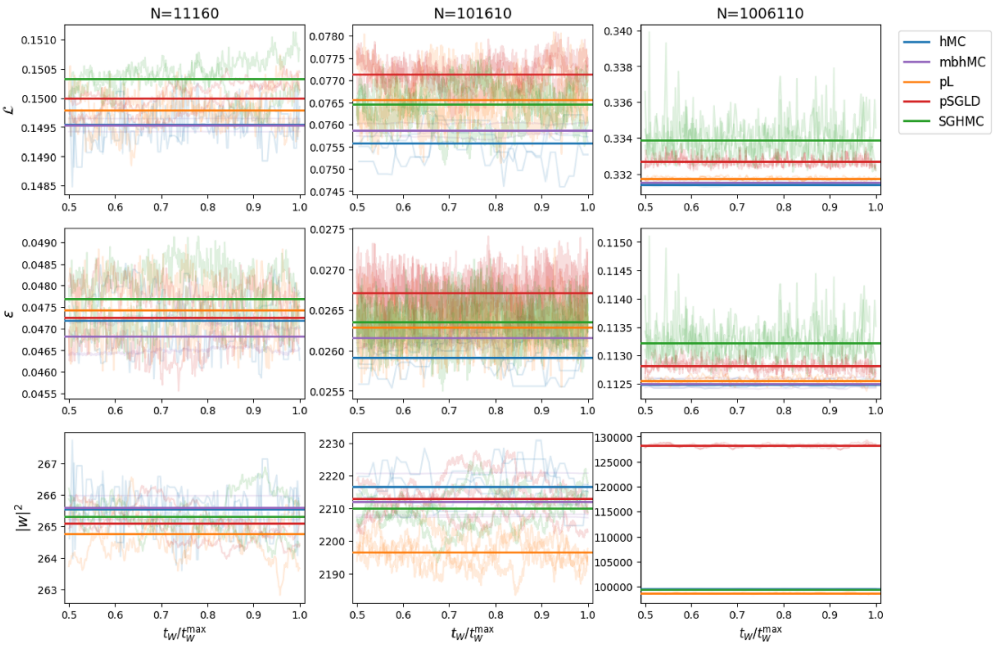

Figure A.1. The average value (straight lines) of the mean cross–entropy $\mathcal{L}$ (top row), the training error ϵ (middle row) and the squared norm $|\boldsymbol{w}|^2$ (bottom row) at $T = 10^{-6}$ for the three model sizes, $N = 11160$ (left column), $N = 101610$ (middle column) and $N = 1006110$ (right column) and for each sampling method, hMC (blue), mbhMC (purple), pL (orange), pSGLD (red) and SGHMC (green). For each observable and model size, the average value predicted by each algorithm is obtained by averaging the second half of the data collected during three simulations each starting from a different equilibrated vector at $T = 10^{-6}$. These three simulations are shown for each scheme as faded lines, plotted against wall–clock time t_{W} scaled by the total simulation time $t_{\mathrm{W}}^{\max}$.

J. Stat. Mech. (2026) 073401

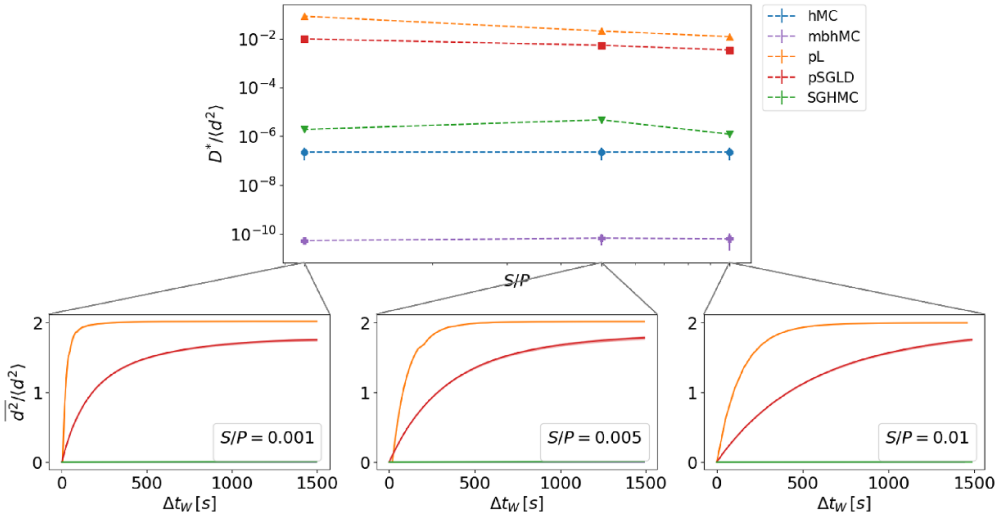

Figure A.2. In the upper plot, the diffusion coefficient D^* scaled by the predicted squared distance $\langle d^2 \rangle$ at equilibrium at $T = 10^{-6}$ as a function of the mini–batch fraction S/P for the biggest model size $N = 1006\,110$ and for different sampling methods, hMC (blue curve), mbhMC (purple curve), pL (orange curve), SGHMC (green curve) and pSGLD (red curve). In the lower plots, the mean squared distance between sampled vectors $\overline{d^2}$ scaled by the predicted squared distance $\langle d^2 \rangle$ at equilibrium at $T = 10^{-6}$ as a function of wall–clock time interval Δt_{W}, for three mini–batch fractions (i.e. $S/P = 0.001$ (lower left), $S/P = 0.005$ (lower center) and $S/P = 0.01$ (lower right), and for all the sampling methods listed above. The mean values and the standard deviations of the squared distance are calculated over three independent simulations starting from different equilibrated weight vectors. The error bars (in the upper plot) and the shaded areas around each curve (in the lower plots) represent the standard deviations. In some of the curves, the latter are so small that they cannot be seen with the naked eye. The hMC values are reported as reference and they are the same for each mini–batch fraction S/P.

J. Stat. Mech. (2026) 073401

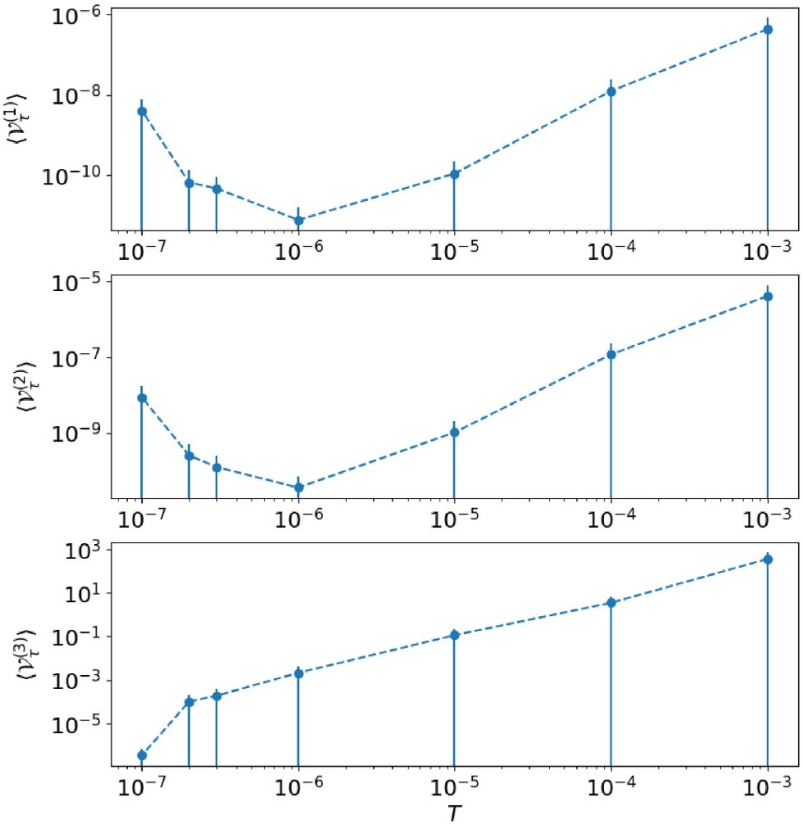

Figure A.3. Mean and standard deviation of the average of all the variances in the first layer (upper plot), in the second layer (middle plot) and in the third layer (lower plot) for the weight vectors relative to the biggest model $N = 1006\,110$ sampled at equilibrium at different temperatures T using the pL scheme.

J. Stat. Mech. (2026) 073401

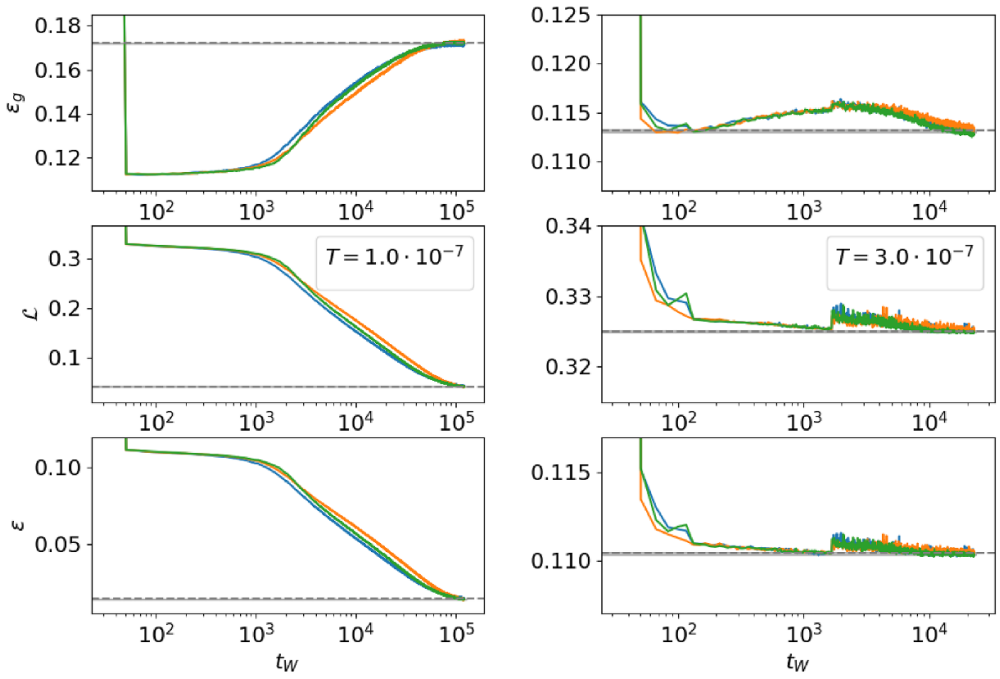

Figure A.4. The generalization error ϵ_{g} (upper row), the mean cross–entropy loss function $\mathcal{L}$ (middle row) and the training error ϵ (lower row) as functions of the wall–clock time t_{W} at two temperatures, $T = 10^{-7}$ (left column) and $T = 3.0 \cdot 10^{-7}$ (right column) for three simulations each starting from a different initialized weight vector. In each panel, the predicted average and standard deviation are shown as a straight dotted line and as a shaded area around it, respectively. Different colors refer to different initialization seeds.

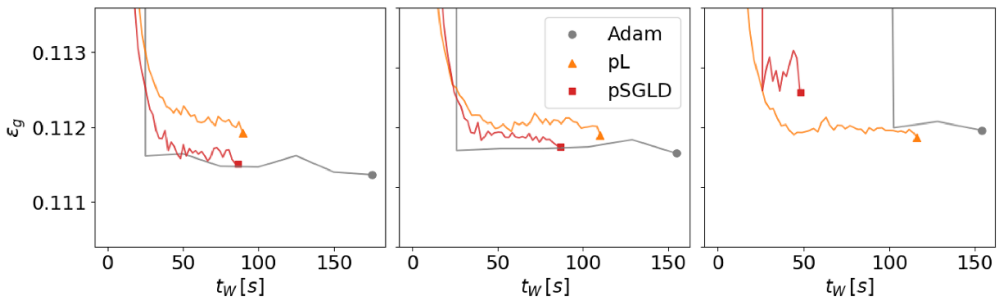

Figure A.5. The generalization error ϵ_{g} computed on the test set of the synthetic spin dataset as a function of the wall–clock time t_{W} during Adam trainings (gray curves), pL samplings (orange curves) and pSGLD samplings (red curves) for three different initialization seeds (first, second and third column) obtained using the largest three–layer feedforward neural network of $N = 1006\,110$ parameters. The curves in each panel stop at the minimum value of the generalization error, as obtained during either the training or the sampling. The minimum test error is highlighted with a different marker for each algorithm.

J. Stat. Mech. (2026) 073401

ORCID iDs

Riccardo Zecchina 0000-0002-1221-5207
Guido Tiana 0000-0001-9868-1809

References

[1] Duane S, Kennedy A D, Pendleton B J and Roweth D 1987 *Phys. Lett. B* **195** 216–22
[2] Zambon A, Malatesta E M, Tiana G and Zecchina R 2025 *Phys. Rev. E* **112** 045303
[3] Baldassi C, Ingrosso A, Lucibello C, Saglietti L and Zecchina R 2015 *Phys. Rev. Lett.* **115** 128101
[4] Baldassi C, Borgs C, Chayes J T, Ingrosso A, Lucibello C, Saglietti L and Zecchina R 2016 *Proc. Natl Acad. Sci.* **113** E7655–62
[5] Pittorino F, Lucibello C, Feinauer C, Perugini G, Baldassi C, Demyanenko E and Zecchina R 2021 *J. Stat. Mech.* 124015
[6] MacKay D J C 1992 *Neural Comput.* **4** 448–72
[7] Welling M, Bren D and Teh Y W 2011 Bayesian Learning via Stochastic Gradient Langevin Dynamics *Proc. 28th. Conf. on Machine Learning (ICML-11)*
[8] van Kampen N G 2007 *Stochastic Processes in Physics and Chemistry* (Elsevier)
[9] Ma Y A, Chen T and Fox E B 2015 arXiv:1506.04696
[10] Chen T, Fox E B and Guestrin C 2014 arXiv:1402.4102
[11] Li C, Chen C, Carlson D and Carin L 2015 arXiv:1512.07666
[12] Bussi G and Parrinello M 2007 *Phys. Rev. E* **75** 56707
[13] Feng Y and Tu Y 2021 *Proc. Natl Acad. Sci. USA* vol **18** e2015617118

J. Stat. Mech. (2026) 073401