CAN MICROCANONICAL LANGEVIN DYNAMICS LEVERAGE MINI-BATCH GRADIENT NOISE?

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling inference methods such as Markov chain Monte Carlo to high-dimensional models remains a central challenge in Bayesian deep learning. A promising recent proposal, microcanonical Langevin Monte Carlo, has shown state-of-the-art performance across a wide range of problems. However, its reliance on full-dataset gradients makes it prohibitively expensive for large-scale problems. This paper addresses a fundamental question: Can microcanonical dynamics effectively leverage mini-batch gradient noise? We provide the first systematic study of this problem, revealing two critical failure modes: a limitation due to anisotropic gradient noise and numerical instabilities in complex high-dimensional posteriors. We resolve both issues by proposing a principled gradient noise preconditioning scheme and developing a novel, energy-variance-based adaptive tuner that automates step size selection and informs dynamical numerical guardrails. The resulting algorithm is a robust and scalable microcanonical Monte Carlo sampler that consistently outperforms strong stochastic gradient MCMC baselines on challenging high-dimensional inference tasks like Bayesian neural networks. Combined with recent ensemble techniques, our work unlocks a new class of stochastic microcanonical Langevin ensemble (SMILE) samplers for large-scale Bayesian inference.

1 INTRODUCTION

The quest for more efficient and robust Markov chain Monte Carlo (MCMC) samplers is central to advancing high-dimensional Bayesian inference. For years, Hamiltonian Monte Carlo (HMC; Neal, 2011) has been the dominant paradigm for navigating the complex posterior landscapes of modern machine learning models. However, the recently proposed microcanonical HMC (Robnik et al., 2023) and its Langevin-based counterpart, the microcanonical Langevin Monte Carlo (MCLMC; Robnik & Seljak, 2024) sampler, have proven to be a powerful new alternative. By simulating dynamics on a constant-energy surface, MCLMC is uniquely equipped to explore challenging posteriors, such as those of Bayesian neural networks (BNN), much faster than traditional HMC-based methods (Robnik et al., 2024; Sommer et al., 2025).

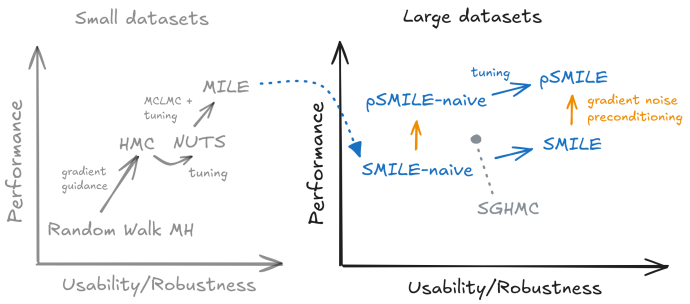


Figure 1: A qualitative contextualization of the newly proposed and explored methods (in blue) relative to prior work (in grey).

Despite its promising results, MCLMC faces a critical limitation that has so far confined its impact and application: its reliance on full-dataset gradients. This makes it computationally infeasible for large-scale problems omnipresent in modern machine learning. Although a mature ecosystem of stochastic gradient MCMC (SGMCMC) methods exists to handle large datasets (Welling & Teh, 2011; Chen et al., 2014; Girolami & Calderhead, 2011), The challenge is that mini-batching already introduces gradient noise, which resembles the Langevin-type noise added to MCLMC even in the full-batch setting.

Robnik & Seljak (2024) showed that such Langevin noise in MCLMC does not require a corresponding damping term for convergence and does not affect the stationary distribution. This observation suggests that mini-batched MCLMC may work *without* any additional noise injection. This stands in contrast to stochastic HMC or Langevin samplers, which in practice are typically run in a heavily overdamped regime, likely at the cost of efficiency. This research gap motivates the central question of our work: *Can microcanonical dynamics leverage mini-batch gradient noise, and thereby be made scalable to modern deep learning?*

This paper presents the **first systematic study of stochastic microcanonical Langevin dynamics**, making several contributions, which are contextualized in Figure 1 relative to prior work.

Our contributions

- We first demonstrate that naive adaptations of MCLMC with stochastic gradients are insufficient.
- Based on theoretical considerations, we then identify a critical pitfall—sampler bias induced by anisotropic gradient noise—and propose a principled gradient noise preconditioning scheme to resolve it.
- Algorithmically, we address numerical instability of mini-batch MCLMC in high dimensions by introducing a novel energy-variance-based adaptive tuner that ensures robust performance and reduces hyperparameter sensitivity.
- Taking these findings together, we propose a scalable and effective mini-batch version of MCLMC, *(preconditioned) stochastic microcanonical Langevin ensembles*, short *(p)SMILE*.
- Finally, our empirical evaluation across a diverse set of BNN applications validates that (p)SMILE is robust and well-working in high dimensions, often outperforming strong SGMCMC baselines.

2 Background & related work

We consider the general problem of Bayesian inference for high-dimensional parametric models such as BNNs. The goal is to infer the posterior distribution over the model's parameters $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$. Given a prior density $p(\boldsymbol{\theta})$, and observed data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N \in (\mathcal{X} \times \mathcal{Y})^n$, the posterior density $p(\boldsymbol{\theta}|\mathcal{D})$ is given by Bayes' rule: $p(\boldsymbol{\theta}|\mathcal{D}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})/p(\mathcal{D})$. Using the posterior predictive density (PPD), we can quantify the uncertainty of a prediction $\boldsymbol{y}^* \in \mathcal{Y}$ for a new data point $\boldsymbol{x}^* \in \mathcal{X}$ by integrating over the posterior distribution of the parameters $p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) = \int_\Theta p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})\,\mathrm{d}\boldsymbol{\theta}$. This integral is analytically intractable for most models, necessitating approximation methods.

2.1 Monte Carlo sampling

Monte Carlo sampling provides a practical way to approximate the posterior and PPD by numerically approximating its integral via samples from $p(\boldsymbol{\theta}|\mathcal{D})$. Given a set of $S \cdot K$ MCMC samples $\{\boldsymbol{\theta}^{(k,s)}, k \in [K], s \in [S]\}$ from K independent chains, the PPD is approximated by its empirical counterpart $p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) \approx \frac{1}{K \cdot S} \sum_{k=1}^K \sum_{s=1}^S p\left(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}^{(k,s)}\right)$.

Full-batch Sampling Full-batch MCMC methods, such as HMC and its tuned variant, the No-U-Turn Sampler (NUTS; Hoffman & Gelman, 2014), are considered the gold standard for high-dimensional sampling (Štrumbelj et al., 2024). They leverage gradients of the full-data likelihood to explore the posterior efficiently. However, these methods are computationally expensive as each step requires calculating gradients over the entire dataset. This makes them impractical for large-scale datasets. Furthermore, a significant challenge even for powerful HMC-based samplers is the difficulty of navigating complex and often highly multimodal loss landscapes of neural networks. While ensembles of many short and warm-started chains have been shown to improve exploration and efficiency (see, e.g., Sommer et al., 2025), they still rely on full-batch gradients.

MCLMC Recently, MCLMC has emerged as a state-of-the-art full-batch sampler, outperforming alternatives like the popular Hamiltonian Monte Carlo-based NUTS in analytical benchmarks (Robnik et al., 2024; Robnik & Seljak, 2024; Robnik et al., 2025), cosmological inference (Simon-Onfroy et al., 2025), and BNN inference (Sommer et al., 2025). MCLMC chooses a specific Hamiltonian

$H(\boldsymbol{\theta}, \boldsymbol{\Pi})$ such that marginalizing the momentum $\boldsymbol{\Pi} \in \mathbb{R}^d$ at a fixed total energy yields the desired stationary distribution of $\boldsymbol{\theta} \in \mathbb{R}^d$. This dynamic is described by the following stochastic differential equation (SDE):

$$\mathrm{d}\boldsymbol{\theta} = \boldsymbol{u}\,\mathrm{d}t, \qquad \mathrm{d}\boldsymbol{u} = \left(1 - \boldsymbol{u}\boldsymbol{u}^\top\right)\left((d-1)^{-1}\nabla \log p(\boldsymbol{\theta}|\mathcal{D})\,\mathrm{d}t + \eta\,\mathrm{d}\boldsymbol{W}\right), \tag{1}$$

where $\boldsymbol{u} = \boldsymbol{\Pi}/||\boldsymbol{\Pi}||$ is the momentum direction, $\boldsymbol{W}$ is the standard Wiener process, and η is a free parameter that determines the distance traveled before momentum decoherence.

In practice, this SDE is solved using numerical integrators such as the Velocity Verlet algorithm (Leimkuhler & Matthews, 2015), which introduces numerical errors with each step. Since the ideal MCLMC dynamic conserves total energy E, the change in total energy per step

$$\Delta E = \Delta \log p(\boldsymbol{\theta}|\mathcal{D}) - (d-1)\log(\cosh\delta + \boldsymbol{e}^\top \boldsymbol{u}\sinh\delta), \tag{2}$$

serves as a useful proxy for this numerical error, where $\boldsymbol{e} = -\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{D})/||\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{D})||$, $\delta = \Delta t||\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{D})||/(d-1)$, and Δt is the integration step size.

2.2 Mini-batch sampling

To overcome the scalability limitations of full-batch methods, stochastic gradient MCMC (SGMCMC) algorithms were developed. These methods use gradients computed on mini-batches of data, offering a significant speedup, similar to stochastic gradient descent in standard optimization of neural networks. Prominent examples include stochastic gradient Langevin dynamics (SGLD; Welling & Teh, 2011) and stochastic gradient Hamiltonian Monte Carlo (SGHMC; Chen et al., 2014).

Among the most effective extensions is scale-adapted SGHMC (Springenberg et al., 2016), which is widely recognized as a state-of-the-art SGMCMC baseline (see e.g. Shi et al., 2025; Andrade & Sato, 2025, for recent studies). Scale-adapted SGHMC incorporates diagonal preconditioning (Girolami & Calderhead, 2011), similar to the mechanism in the RMSprop optimizer, to automatically adjust the step size for each parameter. This adaptation is motivated by the complex geometry of a neural network's loss landscape, empirically significantly enhancing both convergence and the ability to explore the posterior distribution. Due to its rather robust performance and efficient scaling, we select scale-adapted SGHMC as our primary SGHMC baseline method. Unless otherwise specified, all references to SGHMC in our experiments refer to this enhanced version.

Mini-batch samplers offer a more scalable approach, but their theoretical guarantees and sampling efficiency often differ considerably from their full-batch counterparts. A possible stochastic version of MCLMC faces several challenges, many distinct from those in traditional SGLD or SGHMC.

2.3 Improving sampling for neural networks

One particularly challenging application for sampling-based inference is BNNs. Due to their complex and often highly multimodal loss surface, traversing the posterior with a sampler is challenging. To mitigate these problems, recent approaches propose to use optimized solutions as warmstarts for sampling (instead of, e.g., sampling from the chosen prior), lifting the sampling into regions of higher probability (Paulin et al., 2025). To tackle the multimodality of the posterior, ensembling methods using multiple chains from different starting locations have proven effective. Such approaches have been proposed both for HMC and for MCLMC (Duffield et al., 2025; Sommer et al., 2025). These methods, also called Bayesian deep ensembles (BDE), have shown state-of-the-art performance for BNN uncertainty quantification, one of our main interests for large-scale and high-dimensional applications. We will thus not only study stochastic variants of MLCMC, but also microcanonical Langevin ensembles (MILE).

3 Stochastic microcanonical Langevin dynamics

In order to develop and study stochastic MCLMC and its ensemble variant MILE, we begin by analyzing its pitfalls and derive the necessary remedies to establish it as a practical and scalable sampler for modern applications. In the following, we will therefore propose different variants of stochastic MILE, with its most basic version denoted as SMILE-naive. Our later proposed extensions pSMILE-naive and pSMILE build on this basic version.

3.1 SAMPLING WITHOUT EXPLICIT NOISE INJECTION

To develop stochastic MCLMC, Eq. (1) needs to be computed using mini-batches. This will introduce an error in the gradient estimation, differing from the averaged full-batch gradient. For a random mini-batch $\mathcal{B} \subseteq \mathcal{D}$ of the data $\mathcal{D}$ with $|\mathcal{B}| =: B \leq N$, assuming the gradient difference for sample $\mathcal{D}_i$ is $\boldsymbol{\varepsilon}_{\mathcal{D}_i} = \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{D})/N - \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{D}_i)$, the mini-batch gradient at fixed $\boldsymbol{\theta}$ is the sum of the single sample gradients,

$$\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{B} \subset \mathcal{D}) = {}^{B}/_{N} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{D}) + \textstyle\sum_{i \in \mathcal{B}} \boldsymbol{\varepsilon}_{\mathcal{D}_i}.$$

A typical assumption for the gradient noise (see, e.g., Ma et al., 2015; Zhu et al., 2019; Ziyin et al., 2022) is $\sum_{i \in \mathcal{B}} \boldsymbol{\varepsilon}_{\mathcal{D}_i} \sim \mathcal{N}(0, \mathbf{V}(\boldsymbol{\theta}))$, with the correlation $\mathbf{V}$ having an unknown dependence on $\boldsymbol{\theta}$.

A naive stochastic version of MILE (referred to as **SMILE-naive**) can be defined as

$$\mathrm{d}\boldsymbol{\theta} = \boldsymbol{u}\,\mathrm{d}t, \qquad \mathrm{d}\boldsymbol{u} = {}^{N}/_{B}\big(1 - \boldsymbol{u}\boldsymbol{u}^{\top}\big)(d-1)^{-1} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{B})\,\mathrm{d}t. \tag{3}$$

Here we omit the explicit noise injection because the noise already exists in $\nabla \log p(\boldsymbol{\theta}|\mathcal{B})$, which is typically rather large for a small batch size. Thus, additional noise is likely to slow down the convergence. In practice, the second-order Minimal-Norm integrator (Omelyan et al., 2002; 2003) is used to numerically solve (3) for all experiments in this work.

Preliminary finding: Stochastic microcanonical Langevin dynamics can potentially be implemented with implicit mini-batch gradient noise injection instead of explicit noise injection.

3.2 THE PITFALL OF ANISOTROPIC STOCHASTIC GRADIENT NOISE

Robnik & Seljak (2024) show that in continuous time, the stationary distribution of MCLMC is the target posterior $p(\boldsymbol{\theta}|\mathcal{D})$ for any amplitude of injected isotropic noise. In Appendix A we show that this is also the case for continuous-time SMILE-naive if the minibatching noise can in the limit be modeled as an isotropic Wiener process. However, the noise from mini-batching often has a position-dependent covariance matrix $\mathbf{V}(\boldsymbol{\theta})$, which **alters the stationary distribution**, see Appendix A. In Table 1, we verify and quantify this effect by comparing the second-moment bias between samples and analytical expectations (Hoffman & Sountsov, 2022), $b^2 = (\mathbb{E}[\boldsymbol{\theta}^2_{\text{sample}}] - \mathbb{E}[\boldsymbol{\theta}^2])^2/\mathrm{Var}(\boldsymbol{\theta}^2)$, under different scenarios of explicit noise injection. This leads to the important finding that the sample bias of SMILE-naive increases substantially across all settings under anisotropic noise.

Noise preconditioning When $\mathbf{V}(\boldsymbol{\theta})$ is known, one can standardize the mini-batching noise by preconditioning the stochastic gradient at each step. Given the Cholesky decomposition of the covariance $\mathbf{V}(\boldsymbol{\theta}) = \mathbf{L}(\boldsymbol{\theta})\mathbf{L}(\boldsymbol{\theta})^{\top}$, the preconditioning is $\boldsymbol{\theta}' = \mathbf{L}(\boldsymbol{\theta})^{\top}\boldsymbol{\theta}$. This preconditioning defines a new dynamic in a reparameterized space $\boldsymbol{\theta}'$. The relationship between the preconditioned gradient and the original gradient is

$$\nabla_{\boldsymbol{\theta}'} \log p(\boldsymbol{\theta}'|\mathcal{B}) = \mathbf{L}(\boldsymbol{\theta})^{-1} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{B}) = \mathbf{L}(\boldsymbol{\theta})^{-1} \left({}^{B}/_{N} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{D}) + \textstyle\sum_{i \in \mathcal{B}} \boldsymbol{\varepsilon}_{\mathcal{D}_i} \right),$$

where the transformed noise term $\mathbf{L}(\boldsymbol{\theta})^{-1} \sum_i \boldsymbol{\varepsilon}_{\mathcal{D}_i} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is now isotropic. Consequently, the sample distribution for the dynamics of $\boldsymbol{\theta}'$ **correctly converges to the target** posterior $p(\boldsymbol{\theta}'|\mathcal{D})$, and multiplying $\boldsymbol{\theta}'$ by any non-zero constant preserves the convergence.

In practice, estimating $\mathbf{V}(\boldsymbol{\theta})$ is computationally infeasible for large-scale models. Here we propose to only use diagonal preconditioning based on moving averages to make computations tractable. Thus, the reparameterized variable is $\boldsymbol{\theta}' = \left(\sqrt{d}/\|\boldsymbol{\sigma}(\sum_{i \in \mathcal{B}} \boldsymbol{\varepsilon}_{\mathcal{D}_i})\|\right) \boldsymbol{\theta} \odot \boldsymbol{\sigma}(\sum_{i \in \mathcal{B}} \boldsymbol{\varepsilon}_{\mathcal{D}_i})$, where $\odot$ denotes the Hadamard product and we only need to estimate the gradient standard deviation $\boldsymbol{\sigma}(\sum_{i \in \mathcal{B}} \boldsymbol{\varepsilon}_{\mathcal{D}_i})$. The normalizing constant $\sqrt{d}/\|\boldsymbol{\sigma}(\sum_{i \in \mathcal{B}} \boldsymbol{\varepsilon}_{\mathcal{D}_i})\|$ is chosen such that $\boldsymbol{\theta}' = \boldsymbol{\theta}$ for isotropic noise. In addition to the estimation of the gradient standard deviation, we also require an estimate $\bar{\boldsymbol{g}}$ for the expected gradient, which is also computed using a moving average:

$$\bar{\boldsymbol{g}}^{(t+1)} \leftarrow (1-\alpha)\bar{\boldsymbol{g}}^{(t)} + \alpha \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{B}),$$
$$\boldsymbol{\sigma}^{(t+1)} \leftarrow \sqrt{(1-\alpha)(\boldsymbol{\sigma}^{(t)})^2 + \alpha(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{B}) - \bar{\boldsymbol{g}}^{(t+1)})^2}.$$

α is a smoothing factor (fixed to $\alpha = 0.01$ throughout the paper) to ensure stable and low-variance estimates. We call this method preconditioned SMILE-naive (pSMILE-naive), as it preconditions the stochastic gradient to ensure the effective noise is isotropic. In principle, extra strong noise injection (Chen et al., 2014) and Fisher scoring (Ahn et al., 2012) would also be applicable to reduce the bias with potential sacrifice in efficiency.

Analytical benchmarks Table 1 compares the squared bias b^2, of single-chain SMILE-naive, pSMILE-naive, SGLD, and vanilla SGHMC across several anisotropic noise scenarios. In all settings, we assume the average noise magnitude is larger than that of the true gradient. We define three anisotropic noise scenarios based on the noise covariance matrix: (i) **Diagonal**, where the covariance matrix is a constant ill-conditioned diagonal matrix; (ii) **Correlated**, where the covariance matrix is the randomly rotated 'Diagonal' covariance; and (iii) **Spatially-varied**, where the covariance matrix is the 'Correlated' covariance times $\exp(-\theta_2/\sigma(\theta_2))$ with θ_2 being the second element of $\boldsymbol{\theta}$.

The results show that although pSMILE-naive does not fully match the performance of SMILE-naive under ideal isotropic noise, it significantly reduces the bias in all three anisotropic settings, particularly for an ill-conditioned Gaussian (ICG) posterior. Furthermore, pSMILE-naive consistently outperforms both SGLD and SGHMC in these challenging scenarios.

Table 1: Bias $b^2(\downarrow)$ of different SGMCMC samplers on three 10-d analytical posteriors with explicitly injected Gaussian noise. The reported bias is the average over 10 independent chains, and the standard deviation is the standard deviation of the average value obtained with bootstrap. The *Baseline* is full-batch MCLMC performance with optimal noise parameter η.

Target	Noise Type	SMILE-naive	SGLD	SGHMC	pSMILE-naive
ICG (Baseline: *0.0001*)	Isotropic	$\mathbf{0.003} \pm 0.001$	0.033 ± 0.006	0.095 ± 0.033	0.006 ± 0.001
	Diagonal	0.245 ± 0.027	0.184 ± 0.021	0.186 ± 0.020	$\mathbf{0.038} \pm 0.008$
	Correlated	0.502 ± 0.194	0.189 ± 0.029	0.235 ± 0.067	$\mathbf{0.055} \pm 0.007$
	Spatially-varied	0.157 ± 0.010	0.328 ± 0.011	0.331 ± 0.011	$\mathbf{0.093} \pm 0.019$
Rosenbrock (Baseline: *0.0003*)	Isotropic	$\mathbf{0.002} \pm 0.001$	0.005 ± 0.001	0.004 ± 0.002	0.004 ± 0.001
	Diagonal	0.302 ± 0.111	0.085 ± 0.007	0.160 ± 0.027	$\mathbf{0.046} \pm 0.002$
	Correlated	0.265 ± 0.042	0.074 ± 0.007	0.085 ± 0.014	$\mathbf{0.070} \pm 0.005$
	Spatially-varied	$\mathbf{0.048} \pm 0.005$	0.079 ± 0.013	0.095 ± 0.013	0.052 ± 0.005
Funnel (Baseline: *0.004*)	Isotropic	$\mathbf{0.014} \pm 0.005$	0.141 ± 0.019	0.128 ± 0.019	0.021 ± 0.005
	Diagonal	0.283 ± 0.146	0.063 ± 0.017	0.077 ± 0.039	$\mathbf{0.042} \pm 0.012$
	Correlated	0.453 ± 0.231	0.147 ± 0.034	0.138 ± 0.039	$\mathbf{0.004} \pm 0.002$
	Spatially-varied	0.023 ± 0.008	0.241 ± 0.043	0.218 ± 0.034	$\mathbf{0.012} \pm 0.003$

3.3 A NAIVE SMILE IN PRACTICE

To further validate our proposed adaptation in a more complex setting, we evaluate our methods on a BNN regression benchmark where MILE is considered the current gold standard.

The benefit of gradient noise preconditioning We first run stochastic samplers using a *batch-wise sampling* approach, i.e., we produce one sample for every mini-batch step. This provides all stochastic samplers with the same number of total gradient computations as the MILE baseline. The results (Table 2, first four rows) provide strong empirical support for our theory as both SMILE-naive and pSMILE-naive consistently, and notably outperform the SGHMC baseline in both log pointwise predictive density (LPPD) as a measure for the evaluation of the approaches' uncertainty quantification and root mean squared error (RMSE) to measure their prediction performance. Critically, pSMILE-naive markedly improves upon SMILE-naive, confirming that correcting for gradient noise anisotropy is crucial for the optimal performance of stochastic microcanonical dynamics.

Closing the gap to full-batch MCMC The batch-wise sampling comparison still implies that MILE makes more passes over the entire dataset than the stochastic samplers if both are run for the same number of iterations. To provide a fairer comparison, we also run *epoch-wise* sampling. This ensures that the stochastic samplers make the same number of full passes over the dataset as MILE. Under this condition, pSMILE-naive closes the remaining performance gap entirely, matching the performance of the full-batch MILE (Table 2, last three rows). This result is significant: it demonstrates that with proper preconditioning, stochastic microcanonical dynamics can achieve

Table 2: Mean RMSE (↓) and LPPD (↑) results (± standard deviation across 3 train-test splits) for a 3 hidden-layer fully-connected neural network on regression tasks. All methods are DE initialized and use 10 chains. Batch-wise sampling refers to the equivalent budget of sampling steps with respect to MILE (the gold standard), and epoch-wise sampling equalizes the number of passes through the dataset compared to MILE.

	LPPD (↑)			**RMSE** (↓)		
Dataset	Airfoil	Bikesharing	Energy	Airfoil	Bikesharing	Energy
Full-batch Gold Standard						
MILE	$\mathbf{0.665 \pm 0.062}$	$\mathbf{0.226 \pm 0.043}$	$\mathbf{2.204 \pm 0.024}$	$\mathbf{0.152 \pm 0.014}$	$\mathbf{0.229 \pm 0.016}$	$\mathbf{0.032 \pm 0.002}$
Stochastic (Batch-wise Sampling)						
SGHMC	-0.176 ± 0.023	-0.092 ± 0.029	0.062 ± 0.034	0.265 ± 0.020	$\mathbf{0.250 \pm 0.015}$	0.113 ± 0.009
SMILE-naive	0.280 ± 0.008	$\mathbf{0.132 \pm 0.046}$	1.191 ± 0.015	$\mathbf{0.185 \pm 0.006}$	$\mathbf{0.236 \pm 0.017}$	0.045 ± 0.001
pSMILE-naive	$\mathbf{0.438 \pm 0.042}$	$\mathbf{0.185 \pm 0.037}$	$\mathbf{1.666 \pm 0.017}$	$\mathbf{0.172 \pm 0.007}$	$\mathbf{0.231 \pm 0.014}$	$\mathbf{0.041 \pm 0.001}$
Stochastic (Epoch-wise Sampling)						
SGHMC	0.177 ± 0.037	0.145 ± 0.059	1.063 ± 0.020	0.214 ± 0.004	$\mathbf{0.236 \pm 0.019}$	0.053 ± 0.002
SMILE-naive	0.497 ± 0.049	0.167 ± 0.058	1.287 ± 0.020	0.166 ± 0.007	$\mathbf{0.233 \pm 0.018}$	0.046 ± 0.001
pSMILE-naive	$\mathbf{0.633 \pm 0.052}$	$\mathbf{0.221 \pm 0.028}$	$\mathbf{2.018 \pm 0.060}$	$\mathbf{0.151 \pm 0.005}$	$\mathbf{0.230 \pm 0.013}$	$\mathbf{0.038 \pm 0.003}$

state-of-the-art sampling performance, effectively mitigating the performance gap often observed between stochastic and full-batch MCMC.

One should note that the performance of these naive methods is rather sensitive to their hyperparameters. As shown in the Appendix (Fig. 4), the step size crucially depends on the gradient noise, which is turn related to the batch size. This sensitivity becomes more pronounced on more complex architectures. Further experiments on a LeNet architecture (62k parameters) for Fashion-MNIST classification (Table 6, Appendix D.2) reinforce the findings of the UCI benchmark. While SMILE-naive fails to sample meaningfully, pSMILE-naive performs exceptionally well, indicating that our gradient noise preconditioning remedy enables scaling to larger architectures.

Pitfall: Anisotropic mini-batch gradient noise breaks the stationarity of MCLMC.
Remedy: Gradient noise preconditioning resolves the problem, facilitating state-of-the-art performance both in simulated examples and on small to medium-scale BNNs.

4 Scaling stochastic microcanonical Langevin

While gradient noise preconditioning resolves theoretical inconsistencies of stochastic microcanonical dynamics, a critical practical barrier remains: Naive implementations with fixed step sizes, performing very well for smaller models, break down when applied to modern architectures such as ResNets or transformers. In the following, we discuss the cause of this problem and present our proposed solution, an adaptive tuning scheme that yields the algorithms SMILE and pSMILE .

4.1 The challenge of scaling: numerical instability

When running previously proposed samplers in very high-dimensional and complex parameter spaces, performance notably degrades. Analyzing this result, we find that the microcanonical dynamics is very sensitive to the step size: a too large step size causes the sampler's trajectory to diverge rapidly, while one that is too small leads to prohibitively slow exploration.

This failure is demonstrated in Fig. 2, where we apply SMILE-naive and pSMILE-naive to a ResNet-7 (428k parameters) on CIFAR-10. Despite preconditioning, tuning over various step sizes, and using cosine decay-type step size schedulers, both methods fail to produce meaningful results, diverging into unstable regions. In contrast, SGHMC remains stable, and our later introduced fully-tuned SMILE and pSMILE variants perform competitively. This highlights that simply correcting for gradient noise anisotropy is insufficient, and a robust yet lightweight tuning mechanism is needed.

Pitfall: The naive application of stochastic microcanonical Langevin Dynamics to large modern network architectures fails due to stability issues.
Remedy: Energy-variance-based numerical guardrails and adaptive step size scheduling enable robust and competitive scaling to large network architectures.

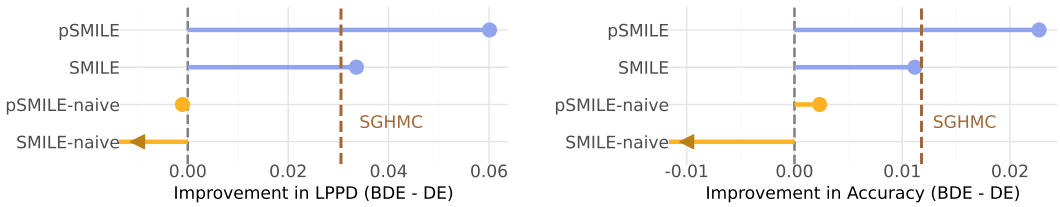


Figure 2: Differences between the BDE performance of naive (orange) and tuned (blue) SMILE variants and a DE baseline for a ResNet-7 (428k parameters) on the CIFAR10 dataset. The x-axis is truncated at -0.01 for readability. For all samplers, we report the best performance of an ensemble of 8 chains over a grid of explored step sizes. Standard deviations over replications are comparable to those reported for the larger-scale setting reported in Table 3. A more detailed plot covering different step and batch sizes is given in Fig. 6 (Appendix).

4.2 Energy error-based tuning

To address the numerical instabilities encountered at scale, a key challenge lies in managing errors introduced by the numerical integrator. As discussed in the background, the MCLMC dynamic provides a natural mechanism for this: while the ideal dynamic conserves total energy, any change in energy, ΔE, serves as a direct proxy for the numerical error per integration step. This insight motivates a tuning scheme based on monitoring the energy error ΔE. Practically, it requires only a single user-specified hyperparameter (an initial step size, typically close to the optimizer's learning rate used in the warmstart), introduces no additional noise injection, and operates efficiently in high-dimensional SGMCMC regimes. The full procedure is given in Algorithm 1 and described below.

Modeling energy error To create robust guardrails and dynamic step-size adjustments, we need to assess whether a given energy error is typical or an outlier. This requires modeling the underlying distribution of the energy error to compute meaningful adaptive thresholds.

To this end, we model the distribution of $|\Delta E|$ in an online fashion using a Gamma distribution, which is well-suited for positive, skewed data. For this, exponential moving averages (EMAs) of its mean $\mu^{(t)}_{|\Delta E|}$ and standard deviation $\sigma^{(t)}_{|\Delta E|}$ are computed

$$\mu^{(t+1)}_{|\Delta E|} \leftarrow (1-\beta)\mu^{(t)}_{|\Delta E|} + \beta|\Delta E|, \quad \sigma^{(t+1)}_{|\Delta E|} \leftarrow \sqrt{(1-\beta)(\sigma^{(t)}_{|\Delta E|})^2 + \beta(|\Delta E| - \mu^{(t+1)}_{|\Delta E|})^2}, \tag{4}$$

where β is the EMAs' smoothing parameter (set to 0.01 throughout the paper). To counteract the bias towards zero in the early stages of the tuning, we further apply a standard correction factor to the variance estimate, helping it converge more rapidly. In order to provide a tractable estimate for the energy error distribution, we then dynamically fit a Gamma distribution $\mathcal{G}a(\gamma^{(t)}_{\text{shape}}, \gamma^{(t)}_{\text{scale}})$ using moment-matching based on the empirical parameters

$$\gamma^{(t)}_{\text{shape}} \leftarrow {}^{(\sigma^{(t)}_{|\Delta E|})^2}\!/_{\mu^{(t)}_{|\Delta E|}}, \quad \gamma^{(t)}_{\text{scale}} \leftarrow \left({}^{\mu^{(t)}_{|\Delta E|}}\!/_{\sigma^{(t)}_{|\Delta E|}}\right)^2. \tag{5}$$

Quantiles from this dynamically fitted distribution as a measure of extremity are then efficiently approximated using the Wilson-Hilferty transform (Wilson & Hilferty, 1931). We denote the approximate quantile function of the Gamma distribution by $\mathcal{G}a^{-1}(\cdot, \gamma^{(t)}_{\text{shape}}, \gamma^{(t)}_{\text{scale}})$.

This online fitting procedure is highly practical because the sampler begins its run from an optimized starting point (a high-likelihood region). Given a meaningful step size (e.g., the final learning rate from the warmstart optimization), the initial energy errors are naturally contained within a reasonable range—meaning they are not excessive and do not significantly degrade performance. This initial stability provides a reliable "golden window" for the EMA estimates and the resulting Gamma fit to stabilize quickly (in our experiments only after a few mini-batch steps, see Fig. 7), We can then robustly estimate quantile that inform the subsequent adaptive tuning and define numerical guardrails throughout the entire sampling process.

Numerical guardrails Numerical instability can cause the chain's trajectory to diverge, leading it into regions of extremely low likelihood from which recovery is computationally expensive and slow. To avoid this, we define a quantile threshold κ based on a high quantile (0.98 by default) of the Gamma distribution. If $|\Delta E|$ exceeds this threshold, we reject the proposed step, resetting the

Algorithm 1 Energy variance-based tuning

Require: $\boldsymbol{\theta}^{(t)}$, $\boldsymbol{u}^{(t)}$, $\mu^{(t)}_{|\Delta E|}$, $\sigma^{(t)}_{|\Delta E|}$, β, κ, a and δ.

1: **Integrate:** $\boldsymbol{\theta}^{(t+1)}, \boldsymbol{u}^{(t+1)}, \Delta E \leftarrow \textsc{IntegratorStep}(\boldsymbol{\theta}^{(t)}, \boldsymbol{u}^{(t)}, \eta^{(t)})$
2: Estimate the current Gamma distribution via moment matching (Eq. (5)).
3: Update the exponential moving averages $\mu^{(t+1)}_{|\Delta E|}$ and $\sigma^{(t+1)}_{|\Delta E|}$ as detailed in Eq. (4).
4: **Apply adaptive numerical guardrails** (Eq. (6)).
5: **Adapt step size** $\eta^{(t)}$ according to Eq. (7).
6: **return** $\boldsymbol{\theta}^{(t+1)}, \boldsymbol{u}^{(t+1)}, \eta^{(t+1)}$

position to its previous state and zeroing the momentum as

$$(\boldsymbol{\theta}^{(t+1)}, \boldsymbol{u}^{(t+1)}) \leftarrow (\boldsymbol{\theta}^{(t)}, \mathbf{0}), \quad \text{if } |\Delta E| > \mathcal{G}a^{-1}(\kappa, \gamma^{(t)}_{\texttt{shape}}, \gamma^{(t)}_{\texttt{scale}}). \tag{6}$$

By preventing the chain from wasting time and computation in distant, low-probability regions, these guardrails ensure that a higher proportion of the samples are drawn from the high-posterior density regions. Without this guardrail, the sampler's performance catastrophically degrades (cf. Table 4).

Adaptive step sizes While the guardrails prevent catastrophic failure, a more nuanced mechanism is needed to ensure efficient and robust exploration. For this, we adapt the step size multiplicatively based on where $|\Delta E|$ lies relative to the Gamma quantiles: when errors are too large, the step sizes are decreased, and increased when they are unusually small (with defaults $\delta = 0.02, a = 0.1$):

$$\eta^{(t+1)} = \begin{cases} \eta^{(t)}(1+\delta) & \text{if } |\Delta E| < \mathcal{G}a^{-1}(\frac{a}{3}, \gamma^{(t)}_{\texttt{shape}}, \gamma^{(t)}_{\texttt{scale}}) \\ \eta^{(t)}(1-\delta) & \text{if } |\Delta E| > \mathcal{G}a^{-1}(1-\frac{2a}{3}, \gamma^{(t)}_{\texttt{shape}}, \gamma^{(t)}_{\texttt{scale}}) \\ \eta^{(t)} & \text{otherwise} \end{cases} \tag{7}$$

where a parameterizes the probability of adaptation. The update is asymmetric, giving stronger incentives to shrink the step size and thus biasing the dynamics toward stability. This design connects to adaptive MCMC (Andrieu & Moulines, 2006; Haario et al., 2006; Roberts & Rosenthal, 2009), but is specialized for the high-dimensional SGMCMC setting, where classical Metropolis-Hastings adjustments and detailed balance cannot be leveraged effectively (Garriga-Alonso & Fortuin, 2021).

4.3 Sampling of contemporary Bayesian neural network architectures

Equipped with our adaptive tuning scheme, we now evaluate SMILE and pSMILE on contemporary BNN architectures, demonstrating their scalability and performance against strong baselines.

Table 3: Image classification task on CIFAR-10 using a ResNet-18 with 11.2M parameters. Mean accuracy (↑) and LPPD (↑) results (± standard deviation) are reported. Numbers in brackets indicate ensemble members.

Method	Accuracy (↑)	LPPD (↑)
Laplace	0.8915 ± 0.0036	-0.4525 ± 0.0345
IVON	0.8735 ± 0.0083	-1.6163 ± 0.0122
DE (8)	0.8995 ± 0.0024	-0.3026 ± 0.0056
SGHMC (8)	$\mathbf{0.9104 \pm 0.0015}$	-0.2908 ± 0.0021
SMILE (8)	$\mathbf{0.9101 \pm 0.0019}$	-0.2763 ± 0.0027
pSMILE (8)	$\mathbf{0.9116 \pm 0.0010}$	$\mathbf{-0.2659 \pm 0.0034}$

Image classification As shown in Table 3, on a ResNet-18 (11.2M parameters) for CIFAR-10 classification, both SMILE and pSMILE outperform SGHMC in LPPD. pSMILE achieves the best overall performance, demonstrating the synergistic benefit of both of our proposed remedies. All sampling-based methods attain similar, high accuracy.

Language modeling In the experiments corresponding to Fig. 3 and Table 10 (Appendix D.5), we test robustness on a nanoGPT model (10.8M parameters, Karpathy, 2022). We ablate the initial step size over four orders of magnitude. For reference, the well-working learning rate of AdamW for the DE optimization is $2 \cdot 10^{-4}$, which also coincides with the best performing step size for all considered SGMCMC samplers. The results reveal a key strength of our method: robustness. While SGHMC's performance degrades catastrophically with a misspecified step size, pSMILE

Table 4: Ablation on reset quantile κ of SMILE comparing predictive and UQ performance metrics for a single replication of the ResNet-18 experimental setup on CIFAR10 of Table 3.

κ	Accuracy	Brier Score	NLL	F1 Score	AUROC	AURC	LPPD
0.90	0.9084	0.1328	0.3939	0.9082	0.9950	0.0123	-0.2704
0.95	**0.9127**	**0.1280**	**0.3931**	**0.9126**	**0.9954**	**0.0112**	**-0.2614**
0.98	0.9108	0.1375	0.4159	0.9105	0.9948	0.0125	-0.2863
0.99	0.9053	0.1444	0.4449	0.9051	0.9946	0.0132	-0.3044
No Guardrail	0.0998	0.9384	12.3519	0.0182	0.5528	0.8828	-2.5710

consistently improves upon the strong DE baseline across all tested step sizes. This increased robustness to the initial step size is a significant practical advantage, alleviating the need for costly hyperparameter sweeps common to many SGMCMC methods.

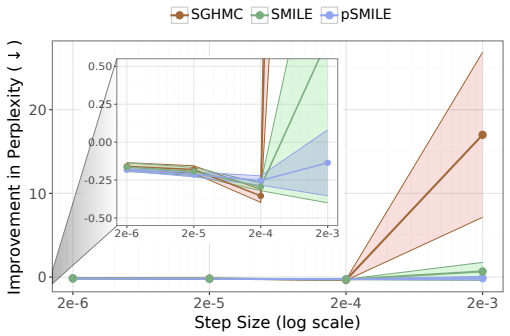


Figure 3: Robustness assessment: Perplexity improvement (smaller is better, std. dev. as shaded area) of MCMC sampling over the optimized warmstart across samplers and step sizes for the nanoGPT model with 10.8M parameters on the `modern-shakespeare` dataset.

4.4 Analysis and Ablations of the Tuning Mechanism

We conduct a series of ablations to better understand the behavior of our proposed adaptive scheme. Our hyperparameter robustness analyses (Tables 4 and 9 in the Appendix) show that the energy-variance-based tuning is robust across a range of settings. Notably, the ablation on our guardrail mechanism (Table 4) highlights its necessity: without it, naive sampling fails catastrophically. The influence of the batch size (Fig. 5, Appendix) is also explored; while larger batches improve performance, the gains diminish, confirming that our method remains effective with moderate batch sizes. Finally, our analyses of the tuning dynamics (Figs. 7 and 8) reveal that the adaptive step size and Gamma distribution parameters rapidly converge to a stable regime. This is a key user-friendly feature, as it eliminates the need for manual tuning, which poses a non-trivial challenge in many traditional MCLMC implementations. Furthermore, the empirically observed energy errors appear to be well described by the proposed Gamma distribution (Appendix D.6).

Takeaway: 1) Gradient noise preconditioning in combination with 2) energy variance-based tuning enables the robust and successful application of stochastic microcanonical Langevin dynamics to modern CNN and GPT-style architectures, often outperforming strong SGMCMC baselines.

5 Discussion

This work bridges the gap between the strong performance of full-batch microcanonical Langevin Monte Carlo and the scalability needs of Bayesian inference. We show that naive stochastic adaptations fail due to gradient noise bias and instability, and propose two key solutions: a principled preconditioning scheme for correctness and an energy-variance-based tuner for stability. Our resulting method, (p)SMILE, matches and often outperforms strong baselines, demonstrating that the benefits of microcanonical dynamics can be realized in the stochastic regime. This establishes (p)SMILE as a practical and powerful addition to the SGMCMC toolkit for complex models like BNNs. Further, we provide efficient code at `https://anonymous.4open.science/r/SMILE2026iclr/`.

Limitations and future work Our method relies solely on mini-batch gradient noise to drive dynamics. While effective, we did not study reintroducing explicit noise as in the original MCLMC SDE (Eq. (1)), which could enhance exploration but adds further tuning complexity. Exploring this trade-off is a promising direction. Future work could also consider richer preconditioning to capture gradient geometry or alternative integrators within our adaptive framework.

REFERENCES

Sungjin Ahn, Anoop Korattikara Balan, and Max Welling. Bayesian posterior sampling via stochastic gradient fisher scoring. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Daniel Andrade and Koki Sato. On the effectiveness of partially deterministic bayesian neural networks. *Computational Statistics*, 40(5):2491–2518, 2025.

Christophe Andrieu and Éric Moulines. On the ergodicity properties of some adaptive mcmc algorithms. 2006.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

Alberto Cabezas, Adrien Corenflos, Junpeng Lao, and Rémi Louf. Blackjax: Composable Bayesian inference in JAX, 2024.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, 2014.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Samuel Duffield, Kaelan Donatella, Johnathan Chiu, Phoebe Klett, and Daniel Simpson. Scalable bayesian learning with posteriors. In *The Thirteenth International Conference on Learning Representations*, 2025.

Hadi Fanaee-T. Bike Sharing Dataset. UCI Machine Learning Repository, 2013.

Adrià Garriga-Alonso and Vincent Fortuin. Exact langevin dynamics with stochastic gradients. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2021.

A. Gelman, J. Hwang, and A. Vehtari. Understanding predictive information criteria for Bayesian models. *Statistics and Computing*, 24(6):997–1016, 2014.

Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2): 123–214, 2011.

Richard D.P. Grumitt, Biwei Dai, and Uroš Seljak. Deterministic langevin monte carlo with normalizing flows for bayesian inference. In *Advances in Neural Information Processing Systems*, 2022.

Heikki Haario, Marko Laine, Antonietta Mira, and Eero Saksman. Dram: efficient adaptive mcmc. *Statistics and computing*, 16(4):339–354, 2006.

Matthew D Hoffman and Andrew Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15:1351–1381, 2014.

Matthew D Hoffman and Pavel Sountsov. Tuning-free generalized hamiltonian monte carlo. In *International conference on artificial intelligence and statistics*, pp. 7799–7813. PMLR, 2022.

Pavel Izmailov, Sharad Vikram, Matthew D. Hoffman, and Andrew Gordon Wilson. What Are Bayesian Neural Network Posteriors Really Like? In *Proceedings of the 38th International Conference on Machine Learning, PMLR 139,*, 2021.

Andrej Karpathy. NanoGPT. `https://github.com/karpathy/nanoGPT`, 2022.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

Benedict Leimkuhler and Charles Matthews. *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods*. Interdisciplinary Applied Mathematics. Springer, United Kingdom, May 2015.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient mcmc. *Advances in neural information processing systems*, 28, 2015.

Radford M. Neal. *MCMC Using Hamiltonian Dynamics*. Chapman & Hall / CRC Press,, 2011.

I. P. Omelyan, I. M. Mryglod, and R. Folk. Optimized Verlet-like algorithms for molecular dynamics simulations. *Physical Review E*, 65(5):056706, May 2002. Publisher: American Physical Society.

I. P. Omelyan, I. M. Mryglod, and R. Folk. Symplectic analytically integrable decomposition algorithms: classification, derivation, and application to molecular dynamics, quantum and celestial mechanics simulations. *Computer Physics Communications*, 151(3):272–314, April 2003.

Daniel Paulin, Peter A. Whalley, Neil K. Chada, and Benedict J. Leimkuhler. Sampling from Bayesian Neural Network Posteriors with Symmetric Minibatch Splitting Langevin Dynamics. In *The 28th International Conference on Artificial Intelligence and Statistics*, 2025.

Gareth O Roberts and Jeffrey S Rosenthal. Examples of adaptive mcmc. *Journal of computational and graphical statistics*, 18(2):349–367, 2009.

Jakob Robnik and Uroš Seljak. Fluctuation without dissipation: Microcanonical langevin monte carlo. In *Symposium on Advances in Approximate Bayesian Inference, Vienna, Austria, 21 July 2024*, volume 253 of *Proceedings of Machine Learning Research*, pp. 111–126, 2024.

Jakob Robnik, G Bruno De Luca, Eva Silverstein, and Uroš Seljak. Microcanonical hamiltonian monte carlo. *The Journal of Machine Learning Research*, 24(1):14696–14729, 2023.

Jakob Robnik, Reuben Cohn-Gordon, and Uroš Seljak. Black-box unadjusted Hamiltonian Monte Carlo. *arXiv*, art. arXiv:2412.08876, December 2024.

Jakob Robnik, Reuben Cohn-Gordon, and Uroš Seljak. Metropolis Adjusted Microcanonical Hamiltonian Monte Carlo. In *Advances in Neural Information Processing Systems*, 2025.

Yuesong Shen, Nico Daheim, Bai Cong, Peter Nickl, Gian Maria Marconi, Bazan Clement Emile Marcel Raoul, Rio Yokota, Iryna Gurevych, Daniel Cremers, Mohammad Emtiyaz Khan, and Thomas Möllenhoff. Variational learning is effective for large deep networks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 44665–44686. PMLR, 21–27 Jul 2024.

Xinxing Shi, Xiaoyu Jiang, and Mauricio A Álvarez. Neighbour-driven gaussian process variational autoencoders for scalable structured latent modelling. In *Forty-second International Conference on Machine Learning*, 2025.

Hugo Simon-Onfroy, François Lanusse, and Arnaud de Mattia. Benchmarking field-level cosmological inference from galaxy redshift surveys. *arXiv*, April 2025.

Saurabh Singh and Shankar Krishnan. Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11237–11246, 2020.

Emanuel Sommer, Jakob Robnik, Giorgi Nozadze, Uroš Seljak, and David Rügamer. Microcanonical Langevin Ensembles: Advancing the Sampling of Bayesian Neural Networks. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Athanasios Tsanas and Angeliki Xifara. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, 49:560–567, 2012.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 681–688, 2011.

Florian Wenzel, Kevin Roth, Bastiaan Veeling, Jakub Swiatkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the Bayes posterior in deep neural networks really? In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10248–10259, 13–18 Jul 2020.

Edwin B. Wilson and Margaret M. Hilferty. The distribution of chi-square. *Proceedings of the National Academy of Sciences of the United States of America*, 17(12):684–688, 1931.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, 2017.

Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning. In *Proceedings of the Eighth International Conference on Learning Representations*, 2020.

Tim G Zhou, Evan Shelhamer, and Geoff Pleiss. Asymmetric duos: Sidekicks improve uncertainty. *arXiv preprint arXiv:2505.18636*, 2025.

Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7654–7663. PMLR, 09–15 Jun 2019.

Liu Ziyin, Kangqiao Liu, Takashi Mori, and Masahito Ueda. Strength of minibatch noise in SGD. In *International Conference on Learning Representations*, 2022.

Erik Štrumbelj, Alexandre Bouchard-Côté, Jukka Corander, Andrew Gelman, Håvard Rue, Lawrence Murray, Henri Pesonen, Martyn Plummer, and Aki Vehtari. Past, Present and Future of Software for Bayesian Inference. *Statistical Science*, 39(1):46 – 61, 2024.

A STATIONARY DISTRIBUTION IN THE CONTINUOUS-TIME LIMIT

A.1 FULL-BATCH

Let's start by reviewing the full-batch continuous-time MCLMC. We denote the variables as $\boldsymbol{z} = (\boldsymbol{\theta}, \boldsymbol{u})$, where $\boldsymbol{z} \in \mathbb{R}^d \times S^{d-1}$, i.e., the velocity is normalized to the unit sphere. MCLMC dynamics can be expressed as a Stratonovich degenerate diffusion on the manifold,

$$\mathrm{d}\boldsymbol{z} = B(\boldsymbol{z})\,\mathrm{d}t + \eta \sum_{i=1}^{d} \sigma_i(\boldsymbol{z}) \circ \mathrm{d}W_i. \tag{8}$$

The notation convention is as in Robnik & Seljak (2024), and we briefly summarize the notation here: W_i are independent $\mathbb{R}$-valued Wiener processes, $\circ$ denotes that the SDE is to be interpreted in the Stratonovich sense, σ_i are vector fields, in coordinates expressed as

$$\sigma_i(\boldsymbol{\vartheta}) = g^{\mu\nu}(\boldsymbol{\vartheta}) \frac{\partial u_i}{\partial \vartheta^\nu}(\boldsymbol{\vartheta}) \frac{\partial}{\partial \vartheta^\mu}(\boldsymbol{\vartheta}). \tag{9}$$

Given that $\boldsymbol{u}$ lives in the unit sphere, $\boldsymbol{u}$ are parametrized with spherical coordinates,

$$\boldsymbol{u}(\boldsymbol{\vartheta}) = (\cos\vartheta_1,\, \sin\vartheta_1 \cos\vartheta_2, ...,\, \sin\vartheta_1 \sin\vartheta_2 ... \cos\vartheta_{d-1},\, \sin\vartheta_1 \sin\vartheta_2 ... \sin\vartheta_{d-1}), \tag{10}$$

and we denote $\boldsymbol{\vartheta} = (\vartheta_1, \vartheta_2, ..., \vartheta_{d-1})$. The metric tensor on the sphere is

$$g_{\mu\nu} = \sum_{i=1}^{d} \partial_\mu u_i(\vartheta) \partial_\nu u_i(\vartheta) \tag{11}$$

We will use Greek letter indices to denote parameters on the sphere and Latin letter indices to denote parameters in the Euclidean space. We will adopt Einstein convention, which implies a sum whenever there are repeated upper and lower indices.

Drift vector field is

$$B(\boldsymbol{\theta}, \boldsymbol{u}) = (\boldsymbol{u},\, \mathbf{P}(\boldsymbol{u}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})/(d-1)), \tag{12}$$

and $\mathbf{P}(\boldsymbol{u}) = \mathbf{I} - \boldsymbol{u}\boldsymbol{u}^\top$ is the projection tensor.

The Fokker-Planck equation corresponding to Equation 8 is

$$\dot{\rho} = -\left(\nabla_i(\rho B^i) + \nabla_\mu(\rho B^\mu)\right) + \frac{1}{2} \nabla_\mu \nabla_\nu (D^{\mu\nu} \rho), \tag{13}$$

where ∇ is the covariant derivative, and the diffusion tensor is

$$D^{\mu\nu} = \eta^2 \sum_{i=1}^{d} \sigma_i^\mu \sigma_i^\nu = \eta^2 \sum_{i=1}^{d} \partial^\mu u_i\, \partial^\nu u_i = \eta^2 g^{\mu\nu}, \tag{14}$$

so the diffusion term in the Fokker-Planck equation becomes the Laplacian. It is shown in Robnik & Seljak (2024) that the stationary distribution of the Fokker-Planck equation has the form of $\rho_\infty \propto p(\boldsymbol{\theta}|\mathcal{D})\sqrt{g}$ because both terms on RHS of Eq. (13) vanish with ρ_∞. Here g is the determinant of the metric.

A.2 ISOTROPIC MINI-BATCHING NOISE

Now we consider a mini-batching noise on the gradient. We will be interested in the limit of step size going to zero. Mini-batching noise can then be modeled as a Wiener process. Note, however, that in the limit, the process converges to the Itô's SDE, not Stratonovich's SDE. This is because the realization of the mini-batching noise is always taken at the beginning of the step. For Gaussian mini-batching noise with isotropic covariance matrix, $\boldsymbol{\Sigma} = \eta^2 \mathbf{I}$, we thus obtain

$$\mathrm{d}\boldsymbol{z} = B(\boldsymbol{z})\,\mathrm{d}t + \eta \sum_{i=1}^{d} \sigma_i(\boldsymbol{z})\,\mathrm{d}W_i, \tag{15}$$

which is equivalent to the following Stratonovich SDE:

$$\mathrm{d}\boldsymbol{z} = B_{\mathrm{SG}}(\boldsymbol{z})\,\mathrm{d}t + \eta \sum_{i=1}^{d} \sigma_i(\boldsymbol{z}) \circ \mathrm{d}W_i, \tag{16}$$

where the drift obtains an additional term:

$$B_{\mathrm{SG}} = B - \frac{1}{2}\sum_{i=1}^{d} \nabla_{\sigma_i}\sigma_i \tag{17}$$

Lemma 1. *σ_i are geodesic vector fields. Specifically,*

$$\nabla_{\sigma_i}\sigma_i = \lambda_i(\boldsymbol{\vartheta})\sigma_i \qquad \lambda_i(\boldsymbol{\vartheta}) = -u_i(\boldsymbol{\vartheta}).$$

Proof. To calculate the parallel transport of the vector field along itself, we will use the Gauss formula, by which the Levi–Civita connection of a submanifold is the tangential projection of the ambient derivative. Viewing the sphere S^{d-1} as a submanifold of the Euclidean space $\mathbb{R}^d$, we can use this to calculate

$$\nabla_X Y = \mathbf{P}\mathbf{D}[X]Y,$$

where X and Y are any smooth vector fields in the Euclidean space, which are tangential to the sphere when restricted to the sphere. $\mathbf{P} = (\mathbf{I} - \boldsymbol{u}\boldsymbol{u}^T)$ here again is the projection operator and $\mathbf{D}[X]$ denotes the Jacobian matrix. $\sigma_i = \mathbf{P}\boldsymbol{e}_i$, where $\boldsymbol{e}_i$ is a unit vector in direction i is such a vector field. Its Jacobian is

$$D[\sigma_i] = -\mathbf{I}(\boldsymbol{e}_i \cdot \boldsymbol{u}) - \boldsymbol{u} \otimes \boldsymbol{e}_i,$$

where $\otimes$ is the Kronecker product. Using the projected ambient derivative gives:

$$\nabla_{\sigma_i}\sigma_i = -\mathbf{P}(\mathbf{I}(\boldsymbol{e}_i \cdot \boldsymbol{u}) + \boldsymbol{u} \otimes \boldsymbol{e}_i)\mathbf{P}\boldsymbol{e}_i = -(\boldsymbol{e}_i \cdot \boldsymbol{u})\sigma_i,$$

where we have used that $\mathbf{P}^2 = \mathbf{P}$ and $\mathbf{P}(\boldsymbol{u} \otimes \boldsymbol{e}_i) = 0$. □

Using this lemma, we can see that the additional drift term vanishes, so $B_{\mathrm{SG}} = B$:

$$[B_{\mathrm{SG}}]_\mu = B_\mu - \frac{1}{2}\sum_{i=1}^{d}[\nabla_{\sigma_i}\sigma_i]_\mu = B_\mu + \frac{1}{2}\sum_{i=1}^{d} u_i \frac{\partial u_i}{\partial \vartheta^\mu} = B_\mu + \frac{1}{2}\frac{\partial}{\partial \vartheta^\mu}\sum_{i=1}^{d} u_i^2 = B_\mu. \tag{18}$$

Therefore, the Fokker-Planck equation is still Eq. (13) and the stationary distribution is unchanged.

A.3 Anisotropic mini-batching noise

Let the mini-batching noise now have some general covariance matrix $\boldsymbol{\Sigma}$. Without loss of generality, we may assume that $\boldsymbol{\Sigma} = \mathrm{Diag}(\eta_1^2, \eta_2^2, \ldots, \eta_d^2)$, by rotating the coordinate system if it is not in this form. We now get

$$\mathrm{d}\boldsymbol{z} = B_{\mathrm{SG}}(\boldsymbol{z})\,\mathrm{d}t + \sum_{i=1}^{d} \eta_i \sigma_i(\boldsymbol{z}) \circ \mathrm{d}W_i, \tag{19}$$

and

$$B_{\mathrm{SG}} = B - \frac{1}{2}\sum_{i=1}^{d} \eta_i^2 \nabla_{\sigma_i}\sigma_i = B + \frac{1}{2}\sum_{i=1}^{d} \eta_i^2 u_i \sigma_i, \tag{20}$$

where we have used the Lemma from the previous section. The diffusion tensor is now

$$D_{\mathrm{SG}}^{\mu\nu} = \sum_{i=1}^{d} \eta_i^2 \sigma_i^\mu \sigma_i^\nu = \sum_{i=1}^{d} \eta_i^2 \partial^\mu u_i\, \partial^\nu u_i \tag{21}$$

Putting these together, the Fokker-Planck equation becomes

$$\dot{\rho} = -\left(\nabla_i(\rho B^i) + \nabla_\mu(\rho B^\mu)\right) + \frac{1}{2}\sum_{i=1}^{d} \eta_i^2 \left(-\nabla_\mu(\rho u_i \sigma_i^\mu) + \nabla_\mu\nabla_\nu(\rho \partial^\mu u_i \partial^\nu u_i)\right). \tag{22}$$

The Fokker-Planck equation has now changed, implying that the stationary distribution has also changed.

B EXPERIMENTAL SETUP AND GENERAL DETAILS

B.1 SOFTWARE AND COMPUTING ENVIRONMENT

Experiments were implemented in Python using `jax` (Bradbury et al., 2018), `BlackJAX` (Cabezas et al., 2024), and extensions of the codebase of Sommer et al. (2025), with selected baselines from `posteriors` (Duffield et al., 2025). Computations were performed on two NVIDIA RTX A6000 or four NVIDIA A100 GPUs and a 64-core AMD Ryzen™ Threadripper™ CPU; CPU parallelism was used for smaller tasks, while large models like CNNs were trained on GPUs. A comprehensive codebase is available at `https://anonymous.4open.science/r/SMILE2026iclr/`.

B.2 DATASETS & OPTIMIZATION

Table 5 summarizes the benchmark datasets utilized in our BNN experiments. For all tabular benchmarks, unless specified otherwise, we use a 70% train, 10% validation, and 20% test split together with a fully connected model architecture of three hidden layers with 16 neurons each. For image classification benchmarks, we adopt the standard train/test split and employ CNN/ResNet-type architectures of varying size. Before training the nanoGPT model, we translated the `tiny-shakespeare` dataset available from Karpathy (2022) into more modern English using Gemini 2.5 Pro to facilitate a more accessible assessment of the quality of the generated text. This was done using the prompt "`Please translate the attached file into simplified modern English. Keep the structure of the text as is (new line for each speaker, speaker name followed by a colon, then the sentence in a new line)`", together with an attached `txt` file of the original text. We call this dataset `modern-shakespeare` and provide it within our public code repository for reproducibility. Furthermore, we use ADAM with decoupled weight decay (Loshchilov & Hutter, 2019) for all DEs and vanilla optimizations. Unless stated otherwise, we assume a standard Gaussian prior, $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$, as is common practice.

Table 5: Datasets used in the Bayesian Deep Learning experiments.

Dataset	Size	Features	Source
Airfoil	1503	5	Dua & Graff (2017)
Bikesharing	17379	13	Fanaee-T (2013)
Energy	768	8	Tsanas & Xifara (2012)
F(ashion)-MNIST	60000	28x28	Xiao et al. (2017)
CIFAR-10	60000	28x28	Krizhevsky et al. (2009)
`modern-shakespeare`	39890 (rows)	65 (CharacterTokenizer)	adapted from Karpathy (2022)

B.3 EVALUATION

We evaluate predictive performance using a range of metrics, with specific metrics chosen based on the task type (e.g., classification, regression, or language modeling). For evaluating the quality of the full predictive distribution and uncertainty, we utilize the log pointwise predictive density (LPPD) on a held-out test set $\mathcal{D}_{\text{test}}$, as advocated by Gelman et al. (2014). The LPPD is defined as:

$$\text{LPPD} = \frac{1}{n_{\text{test}}} \sum_{(\boldsymbol{y}^*, \boldsymbol{x}^*) \in \mathcal{D}_{\text{test}}} \log \left(\frac{1}{K \cdot S} \sum_{k=1}^{K} \sum_{s=1}^{S} p\left(\boldsymbol{y}^* | \boldsymbol{\theta}^{(k,s)}(\boldsymbol{x}^*) \right) \right) \tag{23}$$

Where $\boldsymbol{\theta}^{(k,s)}$ are the obtained posterior samples from K chains of S samples each. This metric measures how well the predictive distribution covers the observed labels, with higher values indicating a better fit.

For classification tasks, we assess point predictions using accuracy, which measures the proportion of correct predictions. Following Zhou et al. (2025) we also consider the Brier score, which quantifies the mean squared error of the predicted probabilities, and the Area Under the Receiver Operating Characteristic (AUROC), which evaluates the model's discriminative ability. Furthermore, we examine calibration with the Area Under the Reliability Curve (AURC).

For regression tasks, the Root Mean Squared Error (RMSE) is employed to evaluate the accuracy of point predictions.

Further, for models, such as our nanoGPT model, we report the Negative Log-Likelihood (NLL) or Perplexity to assess how well the model predicts the test data. Perplexity, a common metric for language models, is a function of the average NLL (more specifically Perplexity $:= \exp(\text{NLL})$) and indicates the effective number of choices the model has at each step, with a lower value representing better performance. The specific metrics used in each experiment are indicated in the respective results sections.

B.4 LLM USAGE

The only use of Large Language Models (LLMs) was for minor language, grammar, and stylistic edits, as well as trivial coding support (such as plotting scripts). No part of the scientific work or core implementation was generated by LLMs.

C ANALYTICAL BENCHMARK

In Table 1 we use four SGMCMC samplers, SMILE-naive , SGLD, vanilla SGHMC and pSMILE-naive, to sample three 10-dimentional analytical posteriors with explicit noise injection. The three analytical posteriors are

1. **Ill-Conditioned Gaussian (ICG)** The distribution is $\mathcal{N}(0, \mathbf{\Sigma} = \mathbf{R}^\top \mathbf{\Lambda} \mathbf{R})$, where $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues equally sampled in log space from $1/100$ to 100 and $\mathbf{R}$ is a random rotation matrix.
2. **Rosenbrock** This is a product of 5 banana-shaped posterior from Grumitt et al. (2022). The posterior is $\log p(\boldsymbol{\theta}) = -\sum_{i=1}^{d/2}[(\theta_{2i-1}^2 - \theta_{2i})^2/Q + (\theta_{2i-1} - 1)^2]$ with $Q = 0.1$ in our setting.
3. **Neal's Funnel** This is a hierarchical model with $\theta_1 \sim \mathcal{N}(0, 3)$ and $\theta_i \sim \mathcal{N}(0, e^{\theta/2}), \quad i \in [2, ..., 10]$.

The explicit noise injection is realized by adding a noise term in the $\log p(\boldsymbol{\theta})$,

$$\log p(\boldsymbol{\theta})_{\text{noise}} = \log p(\boldsymbol{\theta}) + \boldsymbol{\epsilon}^\top \boldsymbol{\theta} \tag{24}$$

where $\boldsymbol{\epsilon}$ is a 10-dimensional Gaussian noise with covariance matrix $\mathbf{V}(\boldsymbol{\theta})$. In 'Isotropic' case, $\mathbf{V}_{\text{iso}} = 256\mathbf{I}$; in 'Diagonal' case, $\mathbf{V}_{\text{diag}} = \mathbf{V}_{\text{iso}}\mathbf{\Lambda}$ and the $\mathbf{\Lambda}$ is the same one used in ICG posterior; in 'Correlated' case, $\mathbf{V}_{\text{corr}} = \mathbf{R}'^\top \mathbf{V}_{\text{diag}} \mathbf{R}'$ and $\mathbf{R}'$ is another random rotation matrix different from the rotation matrix used in ICG posterior; in 'Spatially-Varied' case, the covariance matrix is $\mathbf{V}_{\text{spa}}(\boldsymbol{\theta}) = \mathbf{V}_{\text{corr}} \exp(-\theta_2/\sigma(\theta_2))$, with θ_2 being the second element of $\boldsymbol{\theta}$. In each scenario, we initialize the position at the mean of the posterior and run 10 chains for 10^6 samples.

The optimal step size for all SGMCMC samplers are determined form grid search. We first run benchmark for $\Delta t = 10^i, \quad i \in \{-6, -5, ..., 0\}$, then for optimal i_{opt} we evaluate the performance for 15 step sizes spaced equally from $i_{\text{opt}} - 1$ to $i_{\text{opt}} + 1$ and report the bias in Table 1. For ICG and Rosenbrok, the **average bias** over all 10 dimensions are reported, and we use **maximum bias** for Neal's Funnel, because in practice the bias of θ_0 is significantly larger than other parameters. We employ a bootstrapping technique to estimate the standard deviation of the mean bias by repeatedly drawing 10 chains with replacement from the original 10 chains, and report the standard deviation in Table 1.

D BAYESIAN NEURAL NETWORK EXPERIMENTS

D.1 UCI BENCHMARK

For the UCI benchmark presented in Table 2, we fit classical mean regression to the different tasks corresponding to the datasets described in Table 5. In the process, we always use a fully-connected feed-forward neural network with three hidden layers of size 16 each, resulting in about 700 total

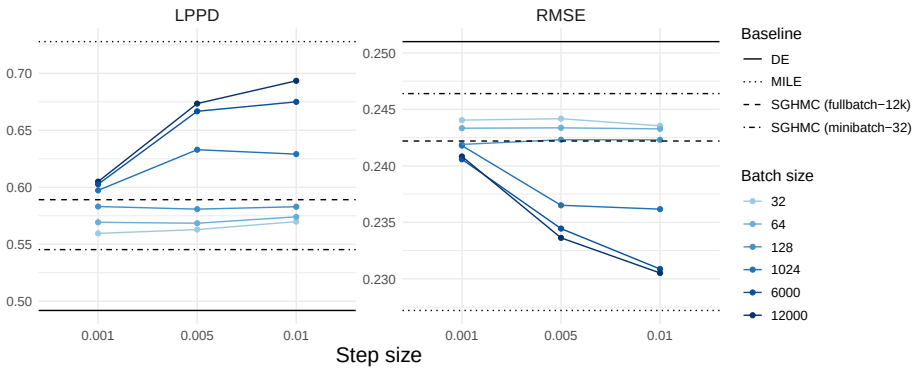


Figure 4: The performance of the SMILE-naive algorithm across various batch and step sizes in comparison with a few baselines on a distributional regression task for the `bikesharing` dataset. The SGHMC's step size was tuned and the best performance is displayed

model parameters (depending on the input dimension). When sampling from the posterior, we use 1000 samples per ensemble member (chain), which is set to 10 members if not specified otherwise.

For the recently proposed Microcanonical Langevin Ensemble (MILE) method, we follow the setup of Sommer et al. (2025). Specifically, the DEs are optimized with the Adam optimizer with decoupled weight decay (Loshchilov & Hutter, 2019) and memberwise early stopping, after which sampling employs the auto-tuning strategy of MILE with 50k steps before providing 1k samples (following thinning of 10k samples). This leads to 60k full-batch sampling steps.

For all SGMCMC variants, we fix the step size of 0.001, which was individually verified to work best in terms of performance by a grid over both smaller and larger step sizes. For SMILE-naive, we even explored decaying step size schedulers, which did not improve the performance significantly. Furthermore, we both consider epoch-wise and batch-wise sampling. For the batch-wise case, we allow the stochastic variants to have the same computational budget as MILE in terms of sampling steps. As with the same number of mini-batch steps, the SGMCMC variants have not passed the data as often as MILE, we also decided to compare with the epoch-wise sampling. This results in a number of batches times more sampling steps as we only collect a sample after a full pass through the dataset. For hardware that can still handle full-batch updates well, this results in a stark computational overhead compared with the batch-wise and full-batch approaches, but might be considered a fairer comparison. For all SGMCMC experiments, we use a batch size of 256.

Further, each method in Table 2 is evaluated using three distinct train-test splits to assess the robustness of its performance.

For the batch and step size ablations summarized in Fig. 4, we adopted the same setup as for Table 2, altering just the batch and step size of SMILE-naive and also considering one mini-batch and one full-batch configuration of scale-adapted SGHMC as baselines (for which we determined a suitable step size via grid search, and only the best performing step size 0.001 is reported).

D.2 Image classification (LeNet)

For these experiments, we adapt the CNN (v2) architecture from Sommer et al. (2025), which is a LeNet type of architecture (Lecun et al., 1998). We run an ensemble of 8 independent MCMC chains, each configured with 5,000 warmup steps followed by 10,000 sampling steps. Applying a thinning interval of 100 yields 100 samples per chain, for a total of 800 final posterior samples used in the Bayesian model average and evaluation. Our analysis focuses on two key aspects: the empirically superior performance of pSMILE-naive against strong baselines and SMILE-naive (Table 6) and the robustness of SMILE to variations in mini-batch size (Fig. 5). While larger batch sizes appear beneficial, the tested configurations suggest the improvements may exhibit diminishing returns. This suggests that SMILE can operate effectively even with moderate batch sizes, preserving much of its computational efficiency.

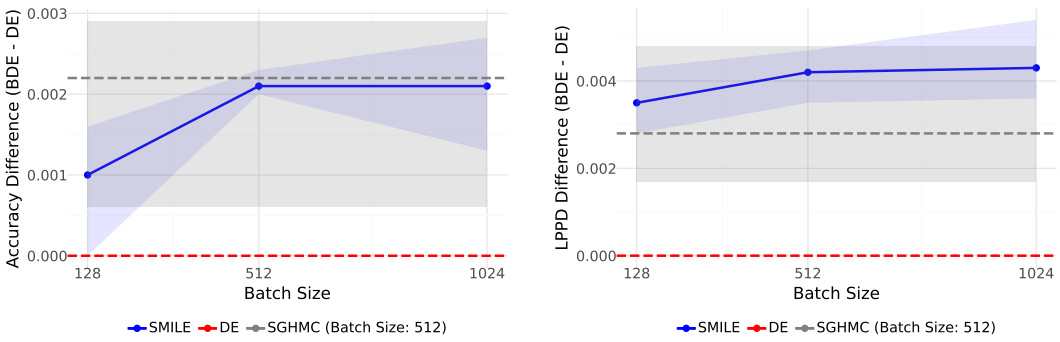


Figure 5: Relative performances with respect to a Deep Ensemble baseline of a Bayesian Deep Ensemble of LeNets (62k parameters) on the Fashion-MNIST dataset using different sampling routines. The shaded areas represent the minimal and maximal performance across three replications for the respective method. For both SGHMC and SMILE , we performed a grid search of suitable step sizes, and for both methods, 0.001 performed best.

Table 6: Relative performances with respect to a Deep Ensemble baseline of a Bayesian Deep Ensemble of LeNets (62k parameters) on the Fashion-MNIST dataset using different sampling routines. For all methods, we ablated over the step sizes of $[0.01, 0.001, 0.0001]$ and both for SGHMC and SMILE 0.001 performed best, for SMILE-naive 0.0001 and 0.01 for pSMILE-naive. The average performance across 3 replications is reported.

	SGHMC	SMILE-naive	pSMILE-naive	SMILE
Δ Accuracy (↑)	0.0022	−0.0027	**0.0082**	0.0021
Δ LPPD (↑)	0.0028	−0.0362	**0.0101**	0.0042

D.3 Image classification (ResNet-7)

The following details correspond to the results provided in Fig. 2 and Fig. 6. The architecture is a custom ResNet-7 with 428k parameters and Filter Response Normalization (FRN; Singh & Krishnan, 2020) instead of BatchNorm due to the critiques of BatchNorm in combination with sampling (Wenzel et al., 2020; Shen et al., 2024). Details on the architecture can be found in Table 7. We use an ensemble of 8 with 5k warmup steps, 10k sampling steps, thinning of 100, batch size 512 (if not indicated otherwise), standard normal isotropic priors, as well as various step sizes, and in the case of SMILE-naive, we also try out a cosine decay step size schedule.

Table 7: The Custom ResNet-7 Architecture with 428k trainable parameters. The output shape is specified for a sample input tensor of size $3 \times 32 \times 32$. All convolutional layers use a 3×3 kernel, stride 1, and 'SAME' padding, and are followed by Filter Response Normalization (FRN; Singh & Krishnan, 2020).

Stage	**Layer Operation(s)**	**Filters**
`input`	Image	-
`stem`	Conv-FRN	32
`body`	Conv-FRN → MaxPool	64
	Conv-FRN	64
	Conv-FRN → MaxPool	128
	Conv-FRN → MaxPool	128
`res_block`	*(Identity Shortcut from previous output)*	128
	↪ Conv-FRN	128
	↪ Add	128
`head`	Global Average Pool	-
	Fully Connected	-

D.4 Image classification (ResNet-18)

The following details correspond to the results provided in Table 3 and Tables 9 and 4. The architecture is a ResNet-18 with 11.2M parameters and Filter Response Normalization (FRN; Singh

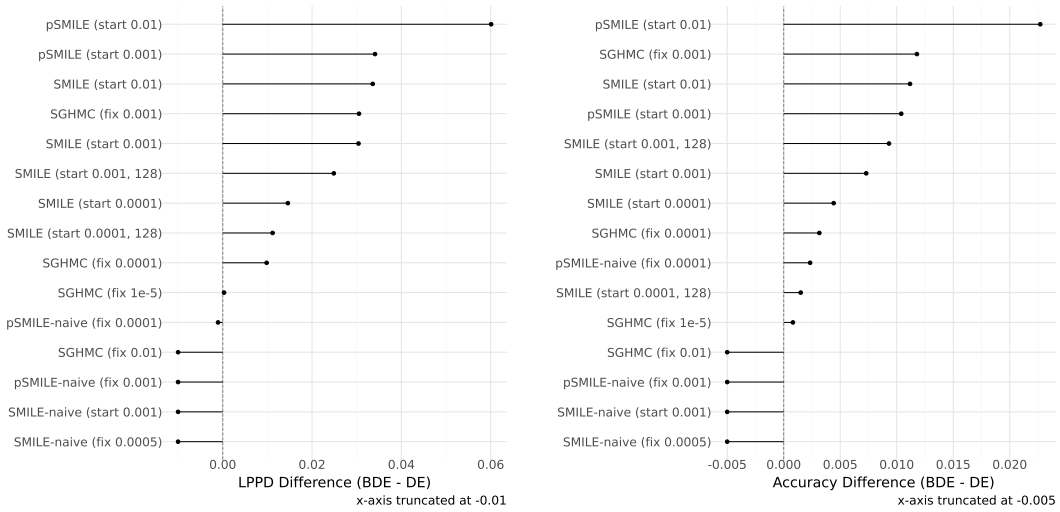


Figure 6: Relative performances difference between different Bayesian deep ensemble approaches and a deep ensemble baseline for a ResNet-7 (428k parameters) on the CIFAR10 dataset. The content of the brackets indicated whether a dynamic schedule with an initial step size or a constant step size schedule was used. If not indicated otherwise in the brackets, a batch size of 512 is employed. In each case, the performance of an ensemble of 8 chains is evaluated. Standard deviations over replications are comparable to those reported for the larger-scale setting reported in Table 3.

Table 8: The ResNet-18 Architecture using Filter Response Normalization (FRN; Singh & Krishnan, 2020) and 11.2M trainable parameters. The output shape is specified for a sample input tensor of size $3 \times 32 \times 32$. Each residual block (in square brackets) consists of two 3×3 convolutional layers. The first block of stages 2-4 uses a stride of 2 for downsampling.

Stage	**Layer Operation(s)**	**Filters**
`input`	Image	-
`stem`	3×3 Conv-FRN $\rightarrow$ MaxPool	64
`stage1`	$\begin{bmatrix} 3 \times 3 \text{ Conv-FRN} \\ 3 \times 3 \text{ Conv-FRN} \end{bmatrix} \times 2$	64
`stage2`	$\begin{bmatrix} 3 \times 3 \text{ Conv-FRN, stride=2} \\ 3 \times 3 \text{ Conv-FRN} \end{bmatrix} \times 2$	128
`stage3`	$\begin{bmatrix} 3 \times 3 \text{ Conv-FRN, stride=2} \\ 3 \times 3 \text{ Conv-FRN} \end{bmatrix} \times 2$	256
`stage4`	$\begin{bmatrix} 3 \times 3 \text{ Conv-FRN, stride=2} \\ 3 \times 3 \text{ Conv-FRN} \end{bmatrix} \times 2$	512
`head`	Global Average Pool Fully Connected	- -

& Krishnan, 2020) instead of BatchNorm due to the critiques of BatchNorm in combination with sampling (Wenzel et al., 2020; Shen et al., 2024) as for the ResNet-7 model. Details on the architecture can be found in Table 8. We use an ensemble of 8 with 5k warmup steps, 10k sampling steps, thinning of 100, batch size 512, standard normal isotropic priors, as well as a step size of 0.001 and momentum decay 0.05 for scale-adapted SGHMC and a step size of 0.01 for SMILE. Both step sizes were determined by a shared grid over step sizes of 5 orders of magnitude.

We further provide two optimization-based baselines in Table 3, namely IVON (Shen et al., 2024) and the Laplace approximation. For the Laplace approximation, the `posteriors` package (Duffield et al., 2025) is used. We run it for 300 epochs, a learning rate of 0.001, and a weight decay of 0.02. For IVON, we use the defaults of the accompanying code repository and suggestions of Shen et al. (2024) with the single Monte Carlo sample configuration.

Table 9: Ablation on adaptation probability a of SMILE comparing predictive and UQ performance metrics for a single replication of the ResNet-18 experimental setup on CIFAR10 of Table 3.

a	Accuracy	Brier Score	NLL	F1 Score	AUROC	AURC	LPPD
0.00	0.9071	0.1475	0.4704	0.9069	0.9947	0.0130	-0.3154
0.05	0.9065	0.1543	0.5144	0.9062	0.9945	0.0134	-0.3379
0.10	0.9054	0.1391	0.4172	0.9053	0.9948	0.0128	-0.2881
0.20	**0.9096**	**0.1331**	**0.3879**	**0.9096**	**0.9950**	**0.0123**	**-0.2732**
0.40	0.9046	0.1422	0.4060	0.9046	0.9944	0.0137	-0.2941

D.5 Language Modeling (NanoGPT)

The following details correspond to the results provided in Table 10, Fig. 3 and the qualitative examples in Appendix E.3. The task is character-level language modeling on the `modern-shakespeare` dataset. The architecture is a 6-layer, 6-head GPT-style transformer with a context length of 256 and an embedding size of 384, with dropout disabled, adapted from Karpathy (2022). For model initialization, we employ a warmstart phase, training the model for 30 epochs using the Adam optimizer with decoupled weight decay with a batch size of 64 and a weight decay of 0.05. The learning rate is annealed linearly from $3\text{e}{-4}$ to $2.5\text{e}{-4}$. For the subsequent sampling phase, we use an ensemble of 4 chains with a batch size of 128. We run 200 warmup steps and collect 1000 posterior samples, thinned by a factor of 100. We use standard normal isotropic priors and a grid of the step sizes: $\{2\text{e}{-6}, 2\text{e}{-5}, 2\text{e}{-4}, 2\text{e}{-3}\}$.

Table 10: Performance results for NanoGPT (10.8M) trained on `modern-Shakespeare` using the scale-adapted SGHMC, pSMILE and SMILE sampler.

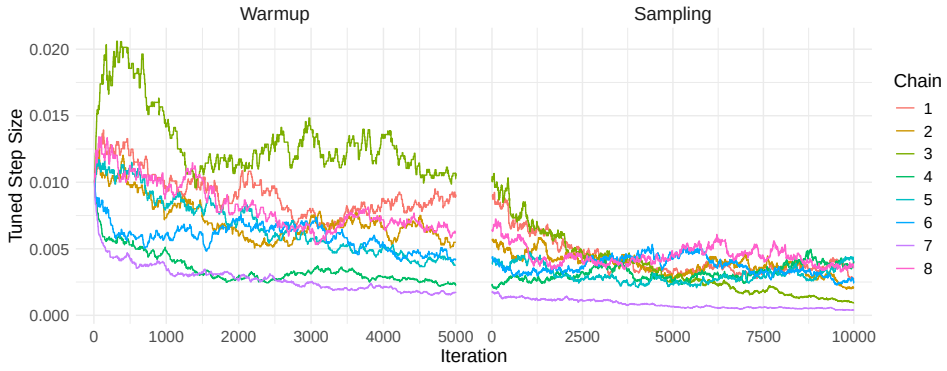
Sampler/ Step size	Accuracy (↑)				Perplexity (↓)			
	DNN	BDE(1)	DE(4)	BDE(4)	DNN	BDE(1)	DE(4)	BDE(4)
SGHMC								
0.000002	0.5298	0.5337	0.5443	**0.5495**	5.0693	4.9094	4.4855	**4.3710**
0.000020	0.5290	0.5357	0.5451	**0.5506**	5.0547	4.8741	4.4503	**4.3427**
0.000200	0.5282	0.5441	0.5441	**0.5563**	5.0717	4.7170	4.4460	**4.2601**
0.002000	0.5302	0.2472	**0.5470**	0.1529	5.0227	22.0208	**4.4194**	10.9048
SMILE								
0.000002	0.5289	0.5345	0.5428	**0.5508**	5.0803	4.9154	4.4955	**4.3724**
0.000020	0.5296	0.5381	0.5434	**0.5526**	5.0324	4.8393	4.4330	**4.2980**
0.000200	0.5296	0.5400	0.5458	**0.5538**	5.0779	4.7828	4.4663	**4.3045**
0.002000	0.5309	0.4900	0.5458	**0.5527**	5.0238	5.6973	**4.4180**	4.8501
pSMILE								
0.000002	0.5290	0.5343	0.5464	**0.5496**	5.0678	4.8855	4.4744	**4.3475**
0.000020	0.5309	0.5376	0.5453	**0.5531**	5.0828	4.8699	4.4928	**4.3540**
0.000200	0.5290	0.5407	0.5457	**0.5562**	5.0374	4.7843	4.4507	**4.3047**
0.002000	0.5294	0.5318	0.5448	**0.5530**	5.0582	4.9221	4.4653	**4.4454**

D.6 Ablations and analysis of the adaptive tuning

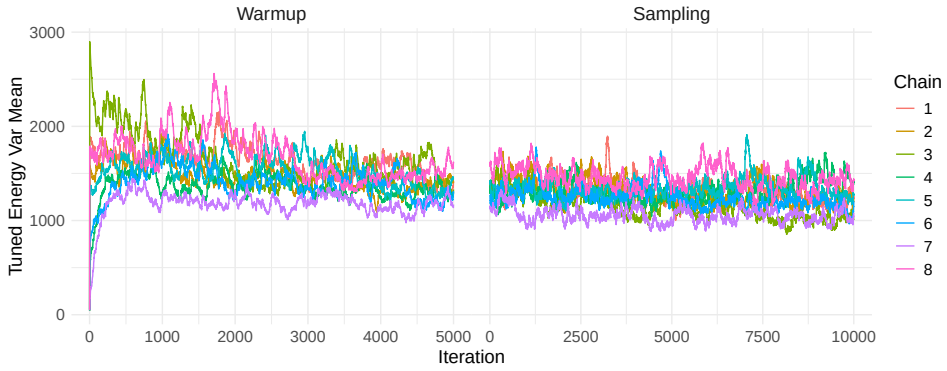
Tuning evolution and goodness of fit Figures 7 and 8 show that the adaptive step size and the parameters of the fitted Gamma distribution quickly converge to a stable regime. The system automatically finds and maintains a suitable level of energy error, which can differ by orders of magnitude between models. This highlights a key advantage: our method is more user-friendly in the context of BNNs than common MCLMC implementations that require manually setting a target energy error (Cabezas et al., 2024), as the user only needs to provide a reasonable initial step size like the optimizer's learning rate. The visualizations also confirm that the step size remains dynamic, biased toward decay for stability but capable of increasing to facilitate exploration. The effectiveness of this tuning is underpinned by the Gamma distribution's goodness of fit. With a target κ of 0.02, the empirical reset frequency measured over tens of thousands of update steps for the NanoGPT model was 0.02314 ± 0.00626 and for the ResNet-18 model was 0.01778 ± 0.00086. This close alignment

with the target confirms that the Gamma distribution provides a robust working model for the energy error.

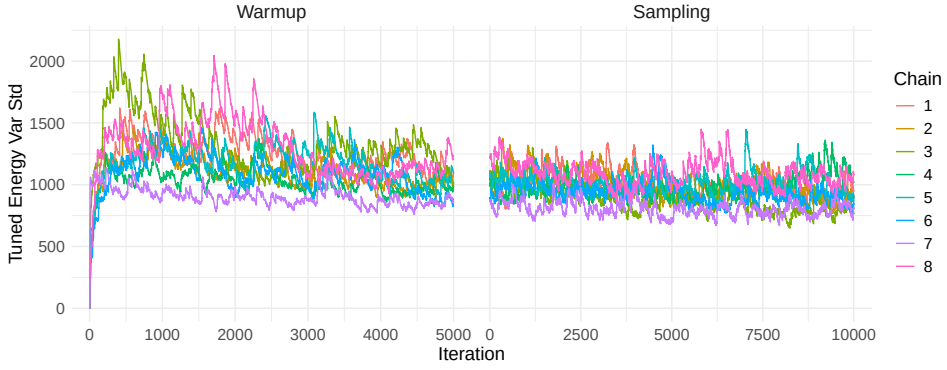
Hyperparameter robustness analyses Both Table 9 and Table 4 confirm that the energy-variance-based tuning works robustly for a range of meaningful settings of the two core hyperparameters of Algorithm 1 κ and adaptation probability a, with optimal performance achieved with moderate guardrailing and adaptation of the step size, even indicating that the strong performance of SMILE reported in Table 3 can be further improved. Notably, Table 4 clearly highlights the necessity to put the guardrailing via κ into place, as without this important component the naive sampling completely diverges and performs poorly.



(a) Evolution of the step sizes over time for the SMILE sampler.

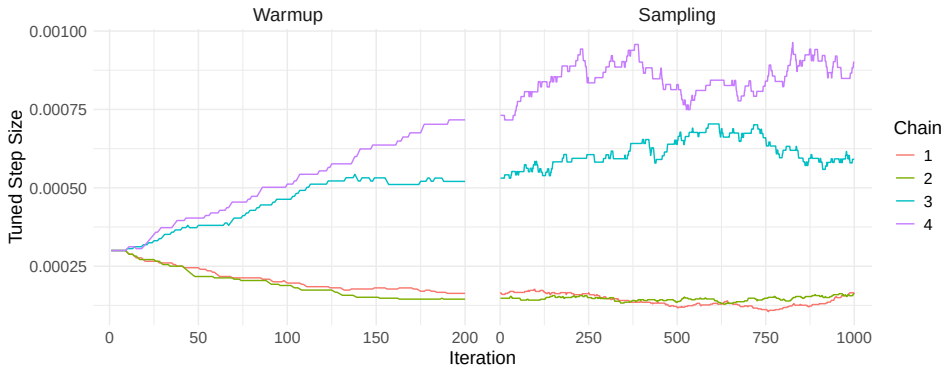


(b) Evolution of ΔE over time for the SMILE sampler.

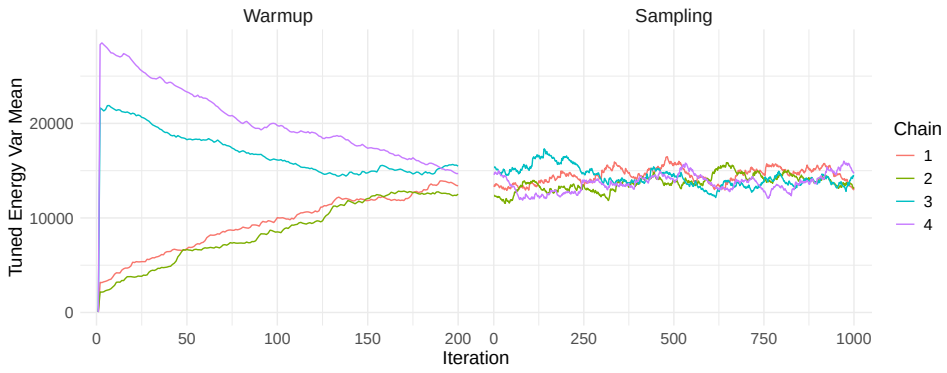


(c) Evolution of $SD(\Delta E)$ over time for the SMILE sampler.

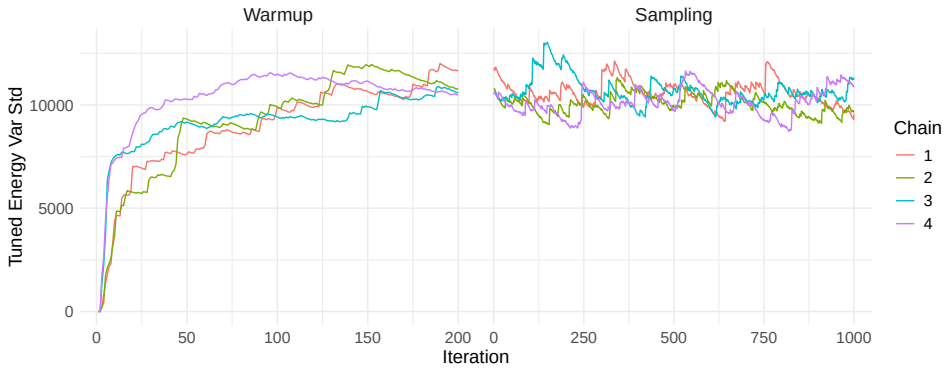
Figure 7: Evolution of the key adapted quantities over time during the sampling of a ResNet-18 via SMILE , respectively, for 8 independent chains. The evolutions are depicted for the SMILE model in Table 3.



(a) Evolution of the step sizes over time for the SMILE sampler.



(b) Evolution of ΔE over time for the SMILE sampler.



(c) Evolution of $SD(\Delta E)$ over time for the SMILE sampler.

Figure 8: Evolution of the key adapted quantities over time during the sampling of NanoGPT via SMILE , respectively, for 4 independent chains. The evolutions are depicted for the best performing SMILE model in Table 10.

E Supplementary Information

E.1 Conceptual Similarity with cyclical SGLD

The resulting step size schedule of Algorithm 1 also shares conceptual similarities with cyclical SGLD (cSGLD Zhang et al., 2020): both introduce systematic variation of the step size to balance exploration and exploitation. However, our approach is stochastic rather than deterministic, producing diverse trajectories across chains, and decays only in expectation. This stochasticity enhances robustness and supports broader posterior exploration, while on average reproducing the empirically

well-working decaying schedule of cSGLD in high-dimensional SGMCMC sampling. Further, as we rely on parallel ensembles, we effectively parallelize the sequential exploration and exploitation cycles of cSGLD and therefore increase effectiveness.

E.2 Runtimes

A brief note on computational cost. For the UCI benchmark (Table 2), the sampling runtime is on the same order of magnitude as the initial optimization time (with the only exception being the epoch-wise sampling, which is discussed above), a characteristic analyzed in Sommer et al. (2025). For our larger-scale experiments, we find that this practical feasibility holds. Sampling with the NanoGPT model took approximately 2.6 times longer than optimization. For the ResNet-18 on CIFAR-10, the sampling phase was only about 10% more costly than optimization. In both cases, the additional runtime invested into sampling is not excessive as in other works on sampling-based inference for BNNs (Izmailov et al., 2021), placing it in the same ballpark as optimization-based training while consistently yielding superior performance.

E.3 Qualitative generation examples

Below, we provide an example of a qualitative generation from one of the many fitted nanoGPT instances, demonstrating that it has learned meaningful syntactic and semantic structures despite the small vocabulary size (SingleChar Tokenizer).

Model configuration: `vocab_size = 65, context_len = 256, emb_size = 384, n_blocks = 6, n_heads = 6`

Prompt I: `'KIN'`

Generated Text:

```
KING EDWARD IV:
No; that must I do well it mildle with of what
hors all my brother Angelo.  This grated is bourned
to the cowdich to save my heavinous, and lords,
whose I clim by your kindrow's doom!


DUKE OF AUMERLE:
As every me.  Duke of confidens bo my oweard.
I have smo not raitful and with the hour
```

Prompt II: `'Now is your time.  Speak loud and kneel before him.'`

Generated Text:

```
Now is your time.  Speak loud and kneel before him.
DUCHESS OF YORK:
My lord come him.
Have your an mistivon!  What do you fremaid?


DUKE OF YORK:
Up the matter of the fight givive watch gentle Brotheren.
If you shall sham so my southern in or prevents;
some that find which coulding of the fast.


DUCHES OF YORK:
I have done the lords; and what often
```

Figure 9: Example texts generated by a trained nanoGPT instance (DNN) of Table 10 given the respective prompt and generating 300 new tokens autoregressively via sampling from the predicted Categorical distribution over the vocabulary. The model produces coherent theatrical formatting, consistent character names, and dialogue structure, indicating learned domain-specific patterns.