A Revisit of Hamiltonian Monte Carlo Efficiency on Bayesian Neural Networks

Cuong N. Nguyen[1], **Lam Si Tung Ho**[2], **Vu C. Dinh**[3]
Georgios Karagiannis[1], **Cuong V. Nguyen**[1]

[1]Department of Mathematical Sciences, Durham University, UK
{ngoc.c.nguyen2, georgios.karagiannis, viet.c.nguyen}@durham.ac.uk

[2]Department of Mathematics and Statistics, Dalhousie University, Canada
lam.ho@dal.ca

[3]Department of Mathematical Sciences, University of Delaware, USA
vucdinh@udel.edu

Abstract

Bayesian neural networks (BNNs) offer principled uncertainty quantification, yet sampling from their posteriors via Hamiltonian Monte Carlo (HMC) remains challenging. While recent work theoretically identified the efficiency degradation of ReLU-based networks, smooth unbounded activations (*e.g.,* GELU, Swish, or Mish), which are prevalent in modern architectures, were assumed to be efficient due to their smoothness. In this work, we challenge this existing justification by deriving explicit local error formulas for the leapfrog integrator. Our analyses reveal that in addition to differentiability, the third-order curvature of the potential energy also significantly influences the HMC efficiency of BNN inference. Empirically, we find that even smooth unbounded activations may degrade sampling performance as severely as piecewise linear activations. Thus, optimally tuning the step size for these networks may require a comparable empirical step size scaling as their piecewise counterparts. This result sheds light on the implications for architecture design in Bayesian deep learning. Our code is available at: https://github.com/CuongNN218/bnn-hmc.

1 Introduction

Bayesian neural networks (BNNs) have emerged as an important research direction in recent years, offering principled uncertainty quantification in neural network predictions [Hernández-Lobato and Adams, 2015, Dhulipala et al., 2023, Liu et al., 2024, Papamarkou et al., 2024]. Unlike optimization-based approaches for neural networks, BNNs treat the parameters as random variables and infer a posterior distribution over the parameter space, allowing principled estimates of model confidence [Gal and Ghahramani, 2016, Wang et al., 2024] and robustness [Trinh et al., 2022]. However, sampling from the posterior of BNNs remains computationally challenging due to the high dimensionality of the parameter space and the non-convex nature of neural network loss landscapes.

Approximate posterior inference methods have been proposed to address this challenge. Variational inference [Blei et al., 2017, Ritter et al., 2018] offers computational scalability but suffers from variance underestimation [Blei et al., 2017, Giordano et al., 2018]. In contrast, Hamiltonian Monte Carlo (HMC) [Neal, 2011] provides a rigorous approach that asymptotically converges to the target distribution, offering more accurate uncertainty estimates. HMC leverages gradient information to generate distant, high-quality proposals in high-dimensional spaces, making it particularly suited for the parameter spaces of neural networks.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

While theoretical analyses of HMC have been developed for various probabilistic models [Nishimura et al., 2020, Dinh et al., 2017, Mohasel Afshar and Domke, 2015], the theoretical understanding of HMC applied to BNNs remains overlooked. A recent work by Dinh et al. [2024] shows that BNNs with ReLU activations exhibit degraded sampling efficiency compared to those using smooth activation functions. This degradation stems from the non-differentiability of ReLU at zero, which introduces a qualitatively different type of discretization error. Consequently, ReLU-based BNNs require smaller step sizes and yield substantially lower acceptance rates than their smooth counterparts. However, this study does not account for other widely used design choices in modern deep learning architectures [Vaswani et al., 2017, Dosovitskiy et al., 2021], most notably the class of smooth but unbounded activation functions. Recent state-of-the-art models [Grattafiori et al., 2024, Bai et al., 2023] increasingly employ this class of activation functions including GELU [Hendrycks and Gimpel, 2016], Swish [Ramachandran et al., 2018], and Mish [Misra, 2020]. These functions are theoretically smooth everywhere, suggesting they should allow HMC to operate efficiently with the standard $O(\varepsilon^3)$ local error bounds [Neal, 2011, Betancourt et al., 2017]. Nevertheless, existing analyses typically treat the gradient magnitude as a static constant within the error bound, focusing primarily on tuning the step size ε to optimize efficiency, rather than accounting for the dynamic behaviour of the potential energy landscape.

To address this critical gap, we explicitly characterize how activation functions impact the error of the leapfrog integrator in HMC for BNNs. By deriving explicit local error formulas for both smooth and piecewise functions, we reveal that the term describing the energy function landscape, often absorbed into error constants in classical analyses [Dinh et al., 2017, Beskos et al., 2013], can become the primary source of HMC error. Furthermore, we demonstrate that sampling efficiency is driven by the complex interplay between this landscape, gradient stability, and smoothness. *Surprisingly, even smooth activation functions can be as ineffective as their piecewise counterparts (Figure 3).* These findings indicate that the choice of activation function is a crucial design hyperparameter that fundamentally dictates MCMC efficiency in BNNs.

Our contributions. In this work, we make the following contributions to the theoretical understanding of HMC with the leapfrog integrator for BNN inference:

- We derive explicit constants in the leading order of local error formulas for HMC in BNNs, providing a theoretical analysis for smooth and piecewise activation regimes.
- By analyzing these constants, we derive bounds and empirically verify that unbounded activation functions generate large third-order curvature, directly amplifying leapfrog local error. This severe degradation in sampling efficiency challenges the classical belief that simply having a smooth activation function guarantees effective HMC.
- We empirically validate our theoretical framework on synthetic and real-world datasets, highlighting that even smooth activation functions might severely degrade sampling performance.

Related works. Classical HMC analyses assume the potential energy function $U(\mathbf{q})$ is smooth. Under this assumption and certain conditions, the leapfrog integrator preserves volume, and the algorithm has well-established theoretical guarantees. Beskos et al. [2013] showed that HMC scales favourably with dimension and attains an optimal acceptance rate of 0.651 while it outperforms both MALA (0.574) and random-walk Metropolis (0.234). Betancourt et al. [2017] established conditions for geometric ergodicity, guaranteeing exponential convergence to the target distribution.

When the energy function is non-smooth, direct application of HMC is not straightforward. Non-smooth and discontinuous energy functions arise naturally in preference learning [Afshar et al., 2015], probabilistic programming [Lunn et al., 2009], phylogenetic tree inference [Dinh et al., 2017], and population sizes inference [Nishimura et al., 2020]. In such settings, the leapfrog integrator fails to conserve the Hamiltonian, and numerical modifications such as surrogate functions [Dinh et al., 2017] or variable mapping [Pakman and Paninski, 2013] are required to recover correctness.

The theoretical foundations of HMC for BNNs remained largely unaddressed until Dinh et al. [2024] established that piecewise affine activations (e.g., ReLU) suffer a degeneration of local error to $\Omega(\varepsilon)$ due to gradient jumps at surfaces of non-differentiability. However, their analysis primarily treats curvatures as static and neglects the empirical impacts of large curvature on the efficiency of HMC for BNNs with unbounded smooth activations.

2 Preliminaries

Bayesian Neural Networks. We consider a general feed-forward neural network represented by a function $f(\boldsymbol{x}; \boldsymbol{q})$, where the parameter vector $\boldsymbol{q}$ contains the flattened and concatenated weight matrices and bias vectors of all layers. The network is structured as a composition of M layers, that is, $f = f_M \circ f_{M-1} \circ \cdots \circ f_1$. Here, each hidden layer $\{f_j\}_{j=1}^{M-1}$ applies a linear transformation followed by a pointwise nonlinear activation function φ, and f_M linearly maps the last hidden representation to a d_y dimensional output space $\mathcal{Y} \subseteq \mathbb{R}^{d_y}$. Formally, given a d_0-dimensional input $\boldsymbol{x} \in \mathbb{R}^{d_0}$ and denote $d_M = d_y$, the network output is computed via the recursive equations:

$$\begin{aligned} z_0 &= \boldsymbol{x}, \\ z_j &= \varphi(\boldsymbol{W}_j z_{j-1} + \boldsymbol{b}_j)\,,\ \forall j = 1, \ldots, M-1, \\ f(\boldsymbol{x}; \boldsymbol{q}) &= \boldsymbol{W}_M z_{M-1} + \boldsymbol{b}_M, \end{aligned}$$

where $\boldsymbol{W}_j \in \mathbb{R}^{d_j \times d_{j-1}}$ and $\boldsymbol{b}_j \in \mathbb{R}^{d_j}$ are the weight matrix and bias vector of the jth layer, respectively.

Unlike optimization-based approaches that yield a single point estimate, BNNs treat parameters $\boldsymbol{q}$ as random variables and infer a posterior distribution over the parameters $\boldsymbol{q}$ conditioned on the observed dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$. Here, x_i, y_i are the input and target value corresponding to the ith observation, which are independently and identically sampled from an unknown distribution. Assuming that $\boldsymbol{q}$ belongs to a parameter space $\mathcal{Q}$, the posterior is defined by Bayes' rule:

$$P(\boldsymbol{q} \mid \mathcal{D}) \ \propto\ \pi(\boldsymbol{q}) \prod_{i=1}^{N} \mathcal{L}(\boldsymbol{q} \mid \boldsymbol{x}_i, y_i), \tag{1}$$

where $\pi(\boldsymbol{q})$ is the prior density and $\mathcal{L}(\boldsymbol{q} \mid \boldsymbol{x}_i, y_i)$ is the likelihood function given a single data point. The posterior predictive distribution of target y^* at a new input $\boldsymbol{x}^*$ can be computed from the posterior expectation $P(y^* \mid \boldsymbol{x}^*, \mathcal{D}) = \int_{\boldsymbol{q} \in \mathcal{Q}} \mathcal{L}(\boldsymbol{q} \mid \boldsymbol{x}^*, y^*)\, P(\boldsymbol{q} \mid \mathcal{D})\, \mathrm{d}\boldsymbol{q}$. Computing this integral is generally intractable for complex neural networks; hence we can resort to Monte Carlo methods able to provide approximations $P(y^* \mid \boldsymbol{x}^*, \mathcal{D}) \approx \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\boldsymbol{q}_i \mid \boldsymbol{x}^*, y^*)$ given a sample $\{\boldsymbol{q}_i\}$ suitably simulated from the posterior distribution $P(\boldsymbol{q} \mid \mathcal{D})$. HMC is a widely used simulation method due to the availability of the gradients of the neural networks and its efficiency in exploring the parameter space.

Hamiltonian Monte Carlo (HMC). HMC is a Markov chain Monte Carlo method that leverages gradient information to generate distant, high-quality proposals in high-dimensional settings (*e.g.,* the parameter space of BNNs). To sample from a posterior distribution $P(\boldsymbol{q} \mid \mathcal{D})$, HMC augments the position $\boldsymbol{q} \in \mathbb{R}^d$ with an auxiliary momentum variable $\boldsymbol{p} \in \mathbb{R}^d$, typically drawn from a standard normal distribution for simplicity, $\boldsymbol{p} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. This defines a joint canonical distribution:

$$P(\boldsymbol{q}, \boldsymbol{p}) \propto \exp(-H(\boldsymbol{q}, \boldsymbol{p})),$$

where $H(\boldsymbol{q}, \boldsymbol{p}) = U(\boldsymbol{q}) + K(\boldsymbol{p})$ with $U(\boldsymbol{q}) = -\log P(\boldsymbol{q} \mid \mathcal{D})$ and $K(\boldsymbol{p}) = \frac{1}{2}\|\boldsymbol{p}\|^2$. Here, $H(\boldsymbol{q}, \boldsymbol{p})$, $U(\boldsymbol{q})$, and $K(\boldsymbol{p})$ are the total energy (*Hamiltonian*), the potential energy, and the kinetic energy of the system, respectively. The system evolves according to Hamilton's equations:

$$\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = \frac{\partial H}{\partial \boldsymbol{p}} = \boldsymbol{p}, \qquad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\frac{\partial H}{\partial \boldsymbol{q}} = -\nabla U(\boldsymbol{q}). \tag{2}$$

Because Hamiltonian dynamics are defined in continuous time, they must be discretised for practical implementation. The *leapfrog integrator* is the standard choice due to its reversibility and volume-preservation. Starting from the initial state $(\boldsymbol{q}_t, \boldsymbol{p}_t)$, the *leapfrog integrator* returns a new state $(\boldsymbol{q}_{t+\varepsilon}, \boldsymbol{p}_{t+\varepsilon})$ using the discrete steps:

$$\boldsymbol{p}_{t+\varepsilon/2} = \boldsymbol{p}_t - \tfrac{\varepsilon}{2} \nabla U(\boldsymbol{q}_t), \tag{3}$$

$$\boldsymbol{q}_{t+\varepsilon} = \boldsymbol{q}_t + \varepsilon\, \boldsymbol{p}_{t+\varepsilon/2}, \tag{4}$$

$$\boldsymbol{p}_{t+\varepsilon} = \boldsymbol{p}_{t+\varepsilon/2} - \tfrac{\varepsilon}{2} \nabla U(\boldsymbol{q}_{t+\varepsilon}). \tag{5}$$

HMC employs the above procedures to iteratively draw samples from the canonical distribution in two steps. In the first step, a momentum vector $\boldsymbol{p}_i$ is sampled from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, allowing the chain to explore different energy levels and discover the full parameter space. In the second step, given the current state $(\boldsymbol{q}_i, \boldsymbol{p}_i)$, HMC performs L leapfrog steps with step size ε to generate a new proposal

$(\boldsymbol{q}^*, \boldsymbol{p}^*)$. This proposal is accepted or rejected according to the Metropolis-Hastings criterion with probability $\alpha = \min\{1, \exp(-H(\boldsymbol{q}^*, \boldsymbol{p}^*) + H(\boldsymbol{q}_i, \boldsymbol{p}_i))\}$, resulting in a new state $(\boldsymbol{q}_{i+1}, \boldsymbol{p}_{i+1})$. This procedure allows HMC to suppress the random-walk behaviour of traditional sampling methods, offering a higher acceptance probability.

When the potential energy U is smooth, HMC has well-established theoretical properties that guarantee its correctness and efficiency, even with BNNs. In particular, the *local error* over a single step is of the order $O(\varepsilon^3)$. Since HMC is often tuned using a fixed traversal length $T = L\varepsilon$, the *global error* after L steps scales as $O(T\varepsilon^2)$ [Beskos et al., 2013].

Efficiency and Optimal Tuning of HMC. For a fixed target distribution, the practical performance of HMC depends critically on the step size ε and the trajectory length T. Beskos et al. [2013] defined the efficiency of HMC as

$$\text{eff}(\ell) = \ell \cdot \mathbb{E}[A(\boldsymbol{q}, \ell)]\,, \tag{6}$$

where ℓ is a tunable scaling parameter for the step size (*e.g.*, $\varepsilon = \ell d^{-1/4}$ for a d-dimensional target distribution) and $\mathbb{E}[A(\boldsymbol{q}, \ell)]$ is the expected acceptance probability. Maximizing this quantity is equivalent to minimizing the expected computational cost per accepted sample.[1]

For a target distribution composed of $d \gg 1$ smooth i.i.d. components, Beskos et al. [2013] showed that the step size must scale as $\varepsilon = \mathcal{O}(d^{-1/4})$ to maintain an $\mathcal{O}(1)$ acceptance probability, and that efficiency is maximised at an optimal acceptance rate of approximately 0.651. In the high-dimensional limit, the expected acceptance probability satisfies

$$\lim_{d\to\infty} \mathbb{E}[A(\boldsymbol{q}, \ell)] = 2\Phi\left(-\frac{\ell^2\sqrt{\Sigma}}{2}\right) =: a(\ell), \tag{7}$$

where Φ is the standard normal c.d.f. and Σ is a positive constant related to the leapfrog step size. Betancourt et al. [2017] subsequently relaxed the target to $a(\ell) \in [0.6,\ 0.9]$.

For BNNs with ReLU activations, Dinh et al. [2024] demonstrated that non-differentiability at the origin introduces a qualitatively different regime. Although HMC remains *correct*, provided that derivatives are defined consistently up to the second order and compatible with the chain rule, its *efficiency* degrades substantially: the local error grows from $O(\varepsilon^3)$ to $\Omega(\varepsilon)$. As a result, the step size must scale as $O(d^{-1/2})$, and the optimal acceptance rate reduces to approximately $[0.4, 0.75]$.

3 Errors of HMC for BNNs with Smooth Activations

3.1 Assumptions and Correctness

We formally state our assumption on the regularity of the loss function $\mathcal{L}(\mathbf{x}, y; \mathbf{q})$, which corresponds to the likelihood function:

Assumption 1 *The likelihood function $\mathcal{L}(\mathbf{q}|\mathbf{x}, y)$ is smooth i.e., infinitely differentiable.*

This assumption is justified in practice by the widespread adoption of the mean squared-error loss for regression and the cross-entropy loss for classification. In addition, in our analysis, we assume that the activation function is smooth and well-defined up to the third-order derivative. As a result, we only consider activation functions that belong to the C^3 class, which is stated in the following assumption:

Assumption 2 *The function $\varphi \in C^3(\mathbb{R})$, i.e., φ is at least three times continuously differentiable.*

We note that this assumption is realistic and covers many activation functions in practice, including Sigmoid, Tanh, GELU [Hendrycks and Gimpel, 2016], Swish [Ramachandran et al., 2018], and Mish [Misra, 2020]. Under the above assumptions, following Dinh et al. [2024], we can conclude that the leapfrog HMC applied to neural networks with smooth activation functions is correct, as the integrator remains reversible and volume-preserving. Traditional analyses of smooth Hamiltonian systems establish that the local error of the leapfrog integrator is of order $O(\varepsilon^3)$ [Neal, 2011, Betancourt et al., 2017]. These analyses treat the gradient's magnitude as a fixed, static constant

[1] This notion of efficiency is not invariant to changes in target scaling or parametrization. So it could be considered as a "local efficiency" that measures local integrator behaviour.

within the error bound, rather than a dynamic factor, and focus on how to tune the step size ε to attain the optimal efficiency. In addition, these analyses do not characterise how the local error depends on the neural network architecture, which could be attributed to activation functions. Next, we derive the explicit formula to reveal this dependency.

3.2 Explicit Local Error Formula

We consider an HMC sampler for BNNs with smooth activation functions to derive an explicit formula for the local error of a single leapfrog step. To give a demonstration of our result, we first consider a simple case where the particle moves following a single leapfrog update. The particle starts at $(\boldsymbol{q}_0, \mathbf{p}_0)$ and ends at $(\boldsymbol{q}_1, \mathbf{p}_1)$. Along the linear path from $\boldsymbol{q}_0$ to $\boldsymbol{q}_1 = \boldsymbol{q}_0 + \varepsilon \boldsymbol{p}_{1/2}$, the surface is smooth. The local error is $\Delta H = H(\boldsymbol{q}_1, \boldsymbol{p}_1) - H(\boldsymbol{q}_0, \boldsymbol{p}_0) = \Delta U + \Delta K$, where $\Delta U = U(\boldsymbol{q}_1) - U(\boldsymbol{q}_0)$ and $\Delta K = K(\boldsymbol{p}_1) - K(\boldsymbol{p}_0)$. Then, we derive an explicit formula for the local error in Theorem 1.

Theorem 1 (Single-step local error) *Consider a single leapfrog step starting at $(\boldsymbol{q}_0, \boldsymbol{p}_0)$ and ending at $(\boldsymbol{q}_1, \boldsymbol{p}_1)$. There exists $\xi \in (0, \varepsilon)$ such that the local approximation error of a single leapfrog step is given by:*

$$\Delta H = \underbrace{\frac{\varepsilon^2}{8}\left(\|\nabla U(\boldsymbol{q}_1)\|^2 - \|\nabla U(\boldsymbol{q}_0)\|^2\right)}_{\textit{gradient norm difference}} - \underbrace{\frac{\varepsilon^3}{12}\langle \nabla^3 U(\boldsymbol{q}_0 + \xi \boldsymbol{p}_{1/2}), \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2}\rangle}_{\textit{third-order curvature}}, \tag{8}$$

where $\otimes$ is the tensor product and $(\boldsymbol{u} \otimes \boldsymbol{v} \otimes \boldsymbol{w})_{ijk} = \boldsymbol{u}_i \boldsymbol{v}_j \boldsymbol{w}_k, \forall \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^d$. We denote $\langle \boldsymbol{T}, \boldsymbol{v} \otimes \boldsymbol{v} \otimes \boldsymbol{v}\rangle := \sum_{i,j,k=1}^{d} \boldsymbol{T}_{ijk} \boldsymbol{v}_i \boldsymbol{v}_j \boldsymbol{v}_k$ as the tensor contraction.

Theorem 1 identifies two distinct sources by which the activation function influences the local error of HMC for BNNs: (1) **Gradient stability**: The activation function directly controls the difference in gradient norms over a single leapfrog step; (2) **Energy function geometry**: The local error depends on the third-order derivatives of the potential energy U with respect to the network parameters, which in turn is affected by the activation function's output and the chain rule. Due to differences in the energy landscapes of neural networks with various activation functions, we hypothesize that the local error, which governs HMC efficiency, is primarily driven by third-order curvature. Our experiment in Section 5 will verify this claim. Proofs are provided in the appendix.

Example 1 (Regression). Consider a single-hidden-layer BNN with one neuron. The model parameters are $w, v \in \mathbb{R}$. The network output is given by $z = v\varphi(wx)$, where $\varphi \in C^3(\mathbb{R})$ is the activation function. Given a standard Gaussian prior $\mathcal{N}(0, 1)$ assigned independently on w, v and a squared error loss $(y - z)^2$, the potential energy function is $U(w, v) = \frac{1}{2}(w^2 + v^2) + (y - z)^2$. We can derive a local upper bound for the curvature:

$$\begin{aligned}\|\nabla^3 U(\boldsymbol{q})\|_\infty \leq\ & 6v^2\,|x|^3\,\|\varphi'\|_\infty\,\|\varphi''\|_\infty \\ & + 2|v|\,|x|^3\,|y - z|\,\|\varphi'''\|_\infty + 4|v|x^2\|\varphi'\|_\infty^2 + 2x^2\|\varphi''\|_\infty\,|y - 2z| + 4|x|\,|\varphi(u)|\,\|\varphi'\|_\infty,\end{aligned}$$

where $u = wx$ is the pre-activation. From the above inequality, the HMC efficiency is governed by the following elements: (1) *Data*: x, y; (2) *Parameter state*: w (indirectly) and v (directly); (3) *Activation gradients*: $\|\varphi'\|_\infty, \|\varphi''\|_\infty, \|\varphi'''\|_\infty$, which are bounded for several popular smooth activation functions; (4) *Activation output*: $\varphi(u)$. Unbounded activations may let the activation output $\varphi \to \infty$, leading to an uncontrollable local error. Interestingly, in Section 5, we show that Softplus achieves a sampling efficiency comparable to GELU despite having a significantly lower third-order derivative magnitude like Sigmoid ($\|\varphi'''\|_\infty = 0.125$). This observation confirms that smoothness is necessary but insufficient to guarantee HMC performance. Instead, the error characterized in Theorem 1 is fundamentally governed by the interplay between data, parameter state, and activation output. Full derivations are provided in the appendix.

Example 2 (Binary classification). We consider a binary classification problem with a one-hidden-layer BNN having a single neuron. In that case, we denote $w \in \mathbb{R}$ and $v \in \mathbb{R}$ as input-to-hidden and hidden-to-output weights, respectively. Let $u = wx$ be the pre-activation. The predictive function is given by $f(x; w) = \sigma(v\varphi(wx))$, where $\varphi \in C^3(\mathbb{R})$ is the activation function and σ is the Sigmoid function. Given a standard Gaussian prior $\mathcal{N}(0, 1)$ assigned on w, v and a cross-entropy loss $-\Big[y \log p + (1 - y) \log(1 - p)\Big]$, the potential energy function is $U(w, v) = \frac{1}{2}(w^2 + v^2) -$

$y(v\varphi(wx)) + \log\left(1 + e^{v\varphi(wx)}\right)$. Given the location of the network $\boldsymbol{q} = [w, v]$, the upper bound for the third-order curvature is given as:

$$\begin{aligned}\|\nabla^3 U(\boldsymbol{q})\|_\infty \leq C(&|v|^3 \|\varphi'\|_\infty^3 |x|^3 + v^2 \|\varphi'\|_\infty \|\varphi''\|_\infty |x|^3 + |v| \|\varphi'''\|_\infty |x|^3 + |\varphi(u)|^3 \\ &+ |v| |x| \|\varphi'\|_\infty \varphi(u)^2 + |x| \|\varphi'\|_\infty |\varphi(u)| + v^2 x^2 |\varphi(u)| \|\varphi'\|_\infty^2 \\ &+ |v| x^2 (\|\varphi'\|_\infty^2 + |\varphi(u)| \|\varphi''\|_\infty) + x^2 \|\varphi''\|_\infty),\end{aligned}$$

where C is a universal constant. As in the regression example, the third-order curvature for classification remains dependent on the *data*, the *parameter state*, the *activation gradients*, and the *activation output*. More importantly, the degree of $\varphi(u)$ in the previous inequality is of order three, suggesting that the activation output is more influential in the classification problem. The behaviour of Softplus in Section 5 is consistent with this interpretation. This example highlights how the boundedness of the activation function helps govern the local error of the leapfrog integrator in classification tasks. Detailed proofs are provided in the appendix.

3.3 Global Error over a Full Trajectory

Using Theorem 1 on the error of a single leapfrog step, we extend our results to cover the error of a full trajectory. Corollary 1 below states our finding.

Corollary 1 (Global error) *Under Assumptions 1 and 2, there exists $\xi_l \in (0, \varepsilon)$ for each lth step such that the accumulated error after L leapfrog steps with step size ε is given as:*

$$\Delta H_L = \frac{\varepsilon^2}{8}\left(\|\nabla U(\boldsymbol{q}_L)\|^2 - \|\nabla U(\boldsymbol{q}_0)\|^2\right) - \frac{\varepsilon^3}{12}\sum_{l=0}^{L-1}\langle \nabla^3 U(\boldsymbol{q}_l + \xi_l \boldsymbol{p}_{l+\frac{1}{2}}), \boldsymbol{p}_{l+\frac{1}{2}} \otimes \boldsymbol{p}_{l+\frac{1}{2}} \otimes \boldsymbol{p}_{l+\frac{1}{2}} \rangle$$

where $\boldsymbol{p}_{l+\frac{1}{2}} = \boldsymbol{p}_l - \frac{\varepsilon}{2}\nabla U(\boldsymbol{q}_l)$ is the momentum at the half step of the lth step.

Remark. The gradient norm terms $\|\nabla U(\boldsymbol{q}_l)\|^2$ telescope across steps, leaving only boundary differences. In contrast, third-order terms accumulate additively; this is why unbounded activations cause progressively worse errors for longer trajectories.

According to Corollary 1, the efficiency of HMC, which is intrinsically linked to the acceptance rate α, is fundamentally governed by the global error ΔH_L. When ΔH_L is large, proposals are frequently rejected, causing the efficiency $\text{eff}(\ell) = \ell \cdot \mathbb{E}[A(\boldsymbol{q}, \ell)]$ in Eq. (6) to degrade. In Section 5, we empirically demonstrate that smooth activation functions, particularly unbounded ones like Swish, can exhibit significantly large third-order curvature. Consequently, mathematically smooth functions suffer from poor sampling efficiency, potentially performing no better than piecewise linear counterparts. These results suggest that existing theoretical frameworks are insufficient to fully characterize the challenges of sampling from the posteriors of deep neural networks.

4 Errors of HMC for BNNs with Piecewise Smooth Activations

Building on our previous derivation for smooth activations, we now relax these smoothness constraints and turn our attention to the piecewise smooth activation functions. These functions remain continuous but non-differentiable at specific points. By accounting for these functions, we extend our error analysis to cover a range of widely used activation functions in deep learning.

4.1 Assumptions and Correctness

We relax Assumption 2 with the following weaker assumption to cover piecewise activations with a single point of non-differentiability at the origin:

Assumption 3 (Piecewise smooth activation) *The activation function φ is piecewise smooth at the origin. That is, it satisfies: (1) $\varphi \in C^0(\mathbb{R})$ (continuous everywhere); (2) $\varphi \in C^\infty(\mathbb{R} \setminus \{0\})$ (smooth everywhere except at 0); (3) $\varphi \notin C^1(\mathbb{R})$ (first-order derivative is not continuous at 0).*

This assumption is satisfied by activations such as ReLU and LeakyReLU with slope $\alpha \neq 1$. Under these conditions, the HMC leapfrog integrator for BNNs works correctly regardless of the derivative's

value at 0, as long as that value remains consistently defined. In addition, the potential energy function $U(\boldsymbol{q})$ is smooth almost everywhere on $\mathcal{Q}$, but exhibits jump discontinuities in its first derivative across a collection of surfaces in the parameter space. Lemma 3.2 in Dinh et al. [2024] characterises the geometry of these surfaces. Briefly, this surface $\mathcal{S}$ is the union of many smooth surfaces $\mathcal{S}_i$ such that $\mathcal{S}_i = \{\boldsymbol{q} : g_i(\boldsymbol{q}) = 0\}$ for analytic functions g_i, that is, $\mathcal{S} = \bigcup_{i\in\mathcal{I}} \mathcal{S}_i$.

As the particle traverses the parameter space following the leapfrog integrator, discontinuities are encountered whenever the trajectory crosses a surface in $\mathcal{S}$, corresponding to a change in the activation state of a neuron. Specifically, for a single leapfrog step of size ε, the linear path $\boldsymbol{q}(t) = \boldsymbol{q}_0 + t\boldsymbol{p}_{1/2}$ for $t \in [0, \varepsilon]$ intersects a finite number of surfaces $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ at points $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_k$. These crossings occur at discrete times $0 < \varepsilon_1 < \varepsilon_2 < \cdots < \varepsilon_k < \varepsilon$. Consequently, the path segments within the open intervals $(\varepsilon_i, \varepsilon_{i+1})$ are contained within smooth regions of U. At each intersection point $\boldsymbol{z}_i$, we denote by $\nabla U^-(\boldsymbol{z}_i)$ and $\nabla U^+(\boldsymbol{z}_i)$ the left and right limit of the gradient at $\boldsymbol{z}_i$ from the two smooth regions, respectively.

4.2 Explicit Local Error Formula

Next, we establish the explicit local error formula for BNNs utilising piecewise activations. Unlike the smooth case, the energy function here contains "kinks" that result in a discontinuous gradient. Due to these discontinuities, expressing ΔU through a single trapezoidal rule over a full interval $(0, \varepsilon)$ is no longer valid. To resolve this, we decompose the trajectory into finite subintervals where the surface of the energy function is locally smooth, so the trapezoidal approximation remains mathematically applicable. In particular, we consider a particle that crosses many surfaces of non-differentiability at $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_k$ at times $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k$. We apply the path integral on each interval $(\varepsilon_i, \varepsilon_{i+1})$ to derive the following theorem:

Theorem 2 *Consider a single leapfrog step starting at $(\boldsymbol{q}_0, \boldsymbol{p}_0)$ and ending at $(\boldsymbol{q}_1, \boldsymbol{p}_1)$ that crosses the surfaces of non-differentiability at $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_k$ at times $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k$. Let $\boldsymbol{z}_0 = \boldsymbol{q}_0$, $\boldsymbol{z}_{k+1} = \boldsymbol{q}_1$, $\varepsilon_0 = 0$, $\varepsilon_{k+1} = \varepsilon$. The local approximation error is given by:*

$$\Delta H = \boldsymbol{p}_{1/2} \cdot \sum_{i=0}^{k} \frac{\Delta\varepsilon_i}{2} \left(\nabla U^-(\boldsymbol{z}_{i+1}) + \nabla U^+(\boldsymbol{z}_i)\right) - \frac{\varepsilon}{2} \boldsymbol{p}_{1/2} \cdot \left(\nabla U(\boldsymbol{q}_0) + \nabla U(\boldsymbol{q}_1)\right)$$
$$+ \frac{\varepsilon^2}{8} \left(\|\nabla U(\boldsymbol{q}_1)\|^2 - \|\nabla U(\boldsymbol{q}_0)\|^2\right) - \sum_{i=0}^{k} \frac{\Delta\varepsilon_i^3}{12} \langle \nabla^3 U(\boldsymbol{q}_0 + \xi_i \boldsymbol{p}_{1/2}), \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \rangle,$$

where $\Delta\varepsilon_i = \varepsilon_{i+1} - \varepsilon_i$, and $\xi_i \in (\varepsilon_i, \varepsilon_{i+1})$ is an unknown time.

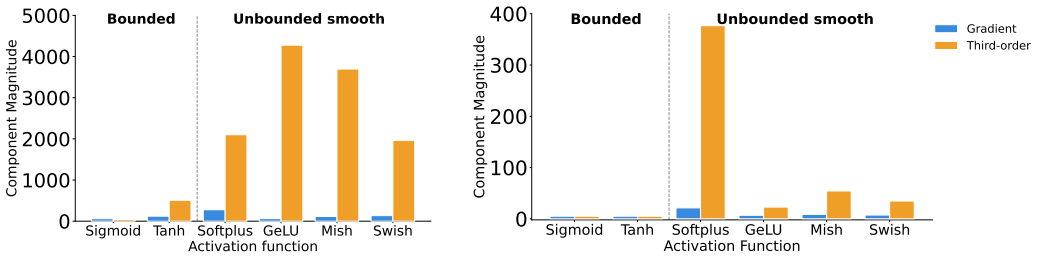
Compared to the local error of HMC for BNNs with smooth activations, Theorem 2 states that the local error in HMC for BNNs with non-smooth activations includes an additional term proportional to the gradient jump at each point of non-differentiability. This error is explicitly associated with the time that particles spend at the smooth region $\Delta\varepsilon$, which is proportional to ε, and the frequency of surface crossings. Specifically, as the linear path intersects more non-differentiable surfaces, the cumulative local error of the leapfrog step increases. As a result, the sampling efficiency of HMC becomes significantly harder to regulate in this regime, as the energy conservation depends on the discrete geometry of the energy landscape. Detailed proofs are provided in the appendix.

5 Simulations

5.1 Experiments on Toy Datasets

Datasets. We consider two synthetic datasets to isolate the effect of activation functions from confounding factors. Following Dinh et al. [2024], we generate $N = 100$ samples with inputs $x_i \sim \text{Uniform}(0, 4)$ and targets $y_i = \cos(2x_i) + \varepsilon_i$, where $\varepsilon_i \sim \mathcal{N}(0, 0.01)$ for regression. For classification, we use the `make_moons` dataset from `sklearn`, which is a two-dimensional binary classification dataset. We generate a dataset with 300 training and 100 test samples at a noise level of 0.1. Visualization of these datasets is given in the appendix.

Effects of third-order curvatures for smooth activations. To investigate the impact of third-order curvature on sampling efficiency, we conduct HMC inference using a standard leapfrog integrator



(a) Regression (b) Classification

Figure 1: Decomposition of the error factors contributing to the local error of the leapfrog integrator. The third-order curvature strongly influences the error, leading to a degradation in sampling efficiency.

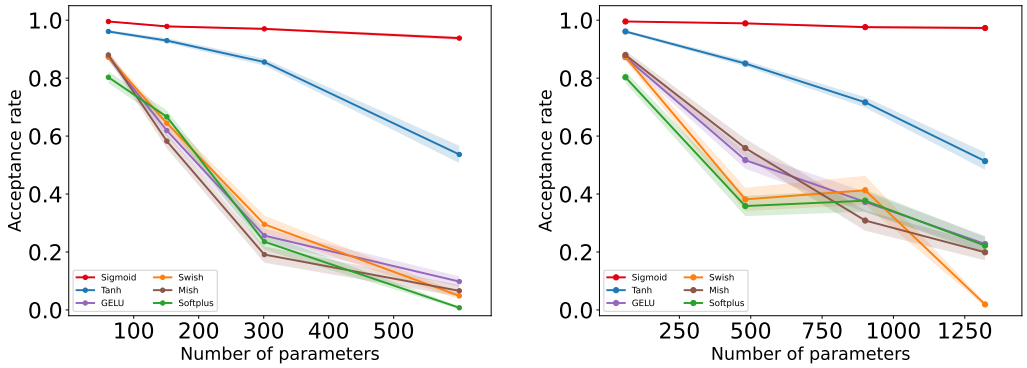
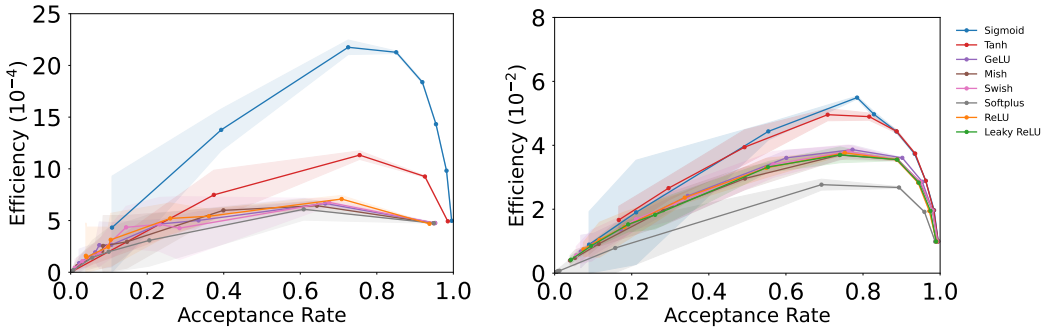


Figure 2: HMC acceptance rates as a function of the number of model parameters. *Left*: Single-hidden-layer BNNs with 20, 50, 100, and 200 hidden nodes. *Right*: BNNs with 2, 3, and 4 hidden layers of 20 nodes each. Although acceptance rates generally decrease as the number of parameters increases, networks with unbounded activations have substantially lower acceptance rates than those with bounded activations.

on both regression and classification datasets. We use a fixed step size $\varepsilon = 0.002$ with $L = 100$ leapfrog steps for regression, and $\varepsilon = 0.05$ with $L = 20$ steps for classification. We generate 3,000 samples, discarding the initial 1,000 for burn-in. To isolate the third-order contribution, we calculate its value by subtracting the gradient stability term, $\frac{\varepsilon^2}{8}(\|\nabla U(\boldsymbol{q}_L)\|^2 - \|\nabla U(\boldsymbol{q}_0)\|^2)$, from the total energy error ΔH. We use these settings for BNNs with a diverse set of smooth activation functions. We plot the averaged magnitudes of different terms across generated samples in Figure 1. We observe that curvature dynamics predominantly govern the local error for both regression and classification tasks. For regression, Sigmoid, characterized by its bounded nature and gradual slope, exhibits a minimal curvature term, resulting in a lower error than all the other activations tested. Notably, while Tanh is bounded, it incurs a higher error than Sigmoid due to its larger infinity norms of gradients and broader output range (*i.e.*,$(-1, 1)$), which will accumulate along the path. Interestingly, despite having the lowest $\|\varphi'''\|_\infty = 0.096$, Softplus exhibits a huge error due to its unbounded output, highlighting the effect of the unbounded activation functions on the approximation accuracy of the leapfrog integrator. Finally, unbounded activations, including GELU, Swish, and Mish, trigger an explosion in local error that severely degrades HMC performance due to the unboundedness of the activation output. We observe the same trend for classification, except Softplus, which exhibits the highest error among unbounded activations. These empirical results provide strong validation for our theoretical derivation in Theorem 1, confirming that we cannot ignore the network gradient for efficiency analysis of HMC. Details of the activation analyses are provided in the appendix.

Effects of activation choices on the acceptance rate. In this experiment, we examine how activation choices affect the acceptance rate. In particular, we consider a wide range of architectures by varying the width and depth of networks. For wide networks, we use a single-hidden-layer network featuring 20, 50, 100, and 200 hidden nodes. For deep networks, we consider multilayer perceptrons with



(a) Regression (b) Classification

Figure 3: Efficiency of HMC with respect to acceptance rate on BNNs with different activation functions. On both synthetic regression and classification datasets, HMC inference on BNNs with bounded and smooth activations is more efficient than with unbounded or non-smooth counterparts.

2, 3, and 4 hidden layers, with each layer containing 20 hidden nodes. All models are sampled on the toy regression dataset using the above HMC configuration with $L = 200$ and $\varepsilon = 0.001$. Figure 2 reports the acceptance rate of networks with different activation functions. Overall, Sigmoid maintains the highest acceptance rate across the tested architectures. Although Tanh's acceptance rate declines as the networks grow, it remains higher than that of the smooth unbounded activations. The unbounded activations generally show a sharper decline. This phenomenon can be attributed to the dominance of the third-order derivatives of the unbounded activations in comparison with the bounded ones, leading to a higher local error. Under the fixed HMC configuration, lower acceptance rates increase the expected computational cost per accepted proposal for BNNs with unbounded activations, making sampling less efficient. These simulations underscore the compound detrimental effect of the third-order curvature term, strongly aligning with our theoretical findings in Theorem 1 and the empirical evidence presented in Figure 1. In the appendix, we provide further analyses of how network architectures, preconditioning matrices, scaling factors, and posterior temperatures affect the acceptance rate and other MCMC diagnostics, including effective sample size, potential scale reduction factor, and integrated autocorrelation time.

Effect of activation choices on HMC efficiency. We investigate how activation choice influences the efficiency of the HMC sampler. In particular, we conduct an experiment with the above single-hidden-layer BNNs with 50 hidden units. Following Dinh et al. [2024], we fix the trajectory length $T = L\varepsilon$ and vary the step size, adjusting the number of leapfrog steps $L = \lceil T/\varepsilon \rceil$ accordingly. This protocol isolates the effect of step-size choice on sampling efficiency while maintaining constant total integration time. For regression, we set $T = 0.1$ and vary $\varepsilon \in \{0.0005, 0.001, \ldots, 0.004\}$. For classification, we set $T = 0.2$ and vary $\varepsilon \in \{0.01, 0.02, \ldots, 0.10\}$. For each configuration, we run HMC five times independently, generating 3,000 samples per run and discarding the first 1,000 as burn-in. To compare HMC efficiency across BNNs with different activation functions, we plot the efficiency curves (with their standard errors) against the corresponding acceptance rate in Figure 3. A higher curve indicates a lower expected computational cost per accepted sample and, therefore, greater computational efficiency.

As illustrated in Figure 3, the curves for some BNNs with smooth unbounded activations approach those of ReLU-based networks. At comparable acceptance rates, the curves for the bounded smooth activations, Sigmoid and Tanh, generally lie higher with their values approximately three to four times larger than those of the unbounded counterparts. This supports our findings that the bounded smooth activation is more efficient than the unbounded activations. In addition, our results generalize the findings of Dinh et al. [2024] to demonstrate that a comprehensive characterization of sampling performance must account for the boundedness of the activation functions, as this property directly influences the curvature term derived in our analysis. Furthermore, these results are consistent with the activation function error order observed in Figure 1. Finally, we observe that the optimal acceptance rates for Tanh and the unbounded activations are around 0.8. Despite exceeding the traditional optimal rate 0.651, this value remains consistent with the recommended range $[0.6, 0.9]$ proposed by Betancourt et al. [2017].

5.2 Experiments on Real-world Dataset

To investigate whether our theoretical findings hold in a real-data setting, we conduct an experiment using the TMDb movie dataset. This dataset provides comprehensive metadata for approximately 5,000 films, including financial metrics (*e.g.,* budget and revenue), creative attributes (*e.g.,* full cast and crew listings), and audience engagement markers (*e.g.,* user ratings). Our primary objective is to predict the average vote using five features as independent variables: budget, popularity, revenue, runtime, and vote count. In this experiment, we use a neural network with a single hidden layer containing 50 hidden units. We employ HMC with a fixed trajectory length $T = 0.005$ while varying the step size ε across the range $\{5 \times 10^{-5}, 2.5 \times 10^{-4}, \ldots, 1.25 \times 10^{-3}\}$. We ran HMC to generate 3,000 samples, discarding the first 1,000 samples for burn-in. For data preprocessing, we removed all rows with zero budget and zero revenue, and we applied z-normalization to the inputs. We randomly partitioned the dataset, using 80% for training and keeping 20% for testing. Finally, the training and testing sets contain 2,580 and 645 data points, respectively.

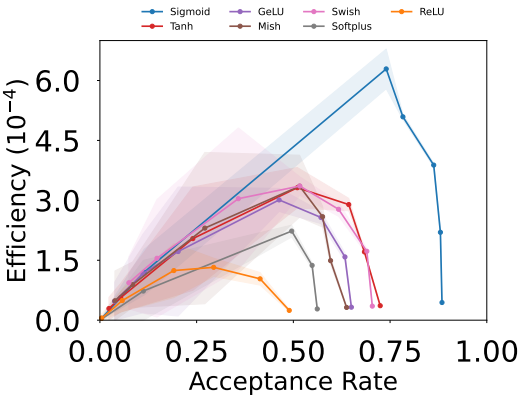


Figure 4: Efficiency of HMC with respect to acceptance rate on BNNs with different activations on TMDb Movie dataset. Sigmoid is more efficient than its counterparts.

Figure 4 reports the efficiency curves of networks with different activations (averaged over five runs). The results demonstrate that BNNs utilizing the Sigmoid activation function achieve the highest efficiency, which directly aligns with our previous toy dataset simulations. Conversely, the curves for unbounded activations consistently lie lower compared to Sigmoid's curve. Interestingly, the Tanh activation function shows a severe degradation in performance, falling to levels on par with unbounded activations. This phenomenon likely occurs because the influence of activation gradients is amplified within the complexities of a real-world data environment.

6 Conclusion

We derived explicit local error formulas for the leapfrog integrator of HMC for BNNs with smooth and non-smooth activation functions. Our analyses identify two error sources: gradient norm dynamics and third-order curvature, which are ignored in previous studies. Contrary to the classical assumption that smoothness guarantees HMC efficiency, we show that the boundedness of activation functions largely contributes to the efficiency of the HMC algorithm with BNNs. Extensive simulations on toy and real-world datasets verify our theoretical findings.

While the provided results establish a theoretical foundation, our current analysis and experiments are limited to shallow networks. Deeper architectures introduce residual connections, layer normalization, and depth-dependent gradient dynamics that may alter the relative contributions of the two error sources. Extending our explicit error formulas to these complex settings represents a natural next step. Furthermore, our analysis considers only the simplified case of an identity mass matrix, which could be generalized by preconditioning matrices such as the Fisher information matrix in RMHMC [Girolami and Calderhead, 2011]. Another promising direction is to investigate adaptive samplers like NUTS [Hoffman and Gelman, 2014], which may partially compensate for activation-induced curvature by shortening trajectories automatically. Quantifying this interaction remains an important open question. We leave these extensions to future work.

Acknowledgments and Disclosure of Funding

This work made use of the Hamilton HPC Service of Durham University and the facilities of the N8 Centre of Excellence in Computationally Intensive Research (N8 CIR) provided and funded by the N8 research partnership and EPSRC (Grant No. EP/T022167/1). The Centre is coordinated by the Universities of Durham, Manchester, and York. LSTH was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grants.

References

Hadi Afshar, Scott Sanner, and Ehsan Abbasnejad. Linear-time Gibbs sampling in piecewise graphical models. In *AAAI Conference on Artificial Intelligence*, 2015.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv*, 2023.

Alexandros Beskos, Natesh S Pillai, Gareth O Roberts, Jesus Sanz-Serna, and AM Stuart. Optimal tuning of the hybrid Monte Carlo algorithm. *Bernoulli*, 19:1501–1534, 2013.

Michael Betancourt, Simon Byrne, Sam Livingstone, and Mark Girolami. The geometric foundations of Hamiltonian Monte Carlo. *Bernoulli*, 2017.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112:859–877, 2017.

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. URL http://github.com/jax-ml/jax.

Somayajulu LN Dhulipala, Yifeng Che, and Michael D Shields. Efficient Bayesian inference with latent Hamiltonian neural networks in No-U-Turn Sampling. *Journal of Computational Physics*, 492:112425, 2023.

Vu Dinh, Arman Bilge, Cheng Zhang, and Frederick A Matsen IV. Probabilistic Path Hamiltonian Monte Carlo. In *International Conference on Machine Learning*, 2017.

Vu C Dinh, Lam S T Ho, and Cuong V Nguyen. Hamiltonian Monte Carlo on ReLU neural networks is inefficient. In *Annual Conference on Neural Information Processing Systems*, 2024.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016.

Ryan Giordano, Tamara Broderick, and Michael I Jordan. Covariances, robustness, and variational Bayes. *Journal of Machine Learning Research*, 19:1–49, 2018.

Mark Girolami and Ben Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 73:123–214, 2011.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units. *arXiv*, 2016.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, 2015.

Matthew D Hoffman and Andrew Gelman. The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15:1593–1623, 2014.

Chengcheng Liu, Yiru Wang, Chenyue Tao, Chung K Law, and Bin Yang. Artificial neural network-based Hamiltonian Monte Carlo for high-dimensional Bayesian inference of reaction kinetics models. *Proceedings of the Combustion Institute*, 40:105590, 2024.

David Lunn, David Spiegelhalter, Andrew Thomas, and Nicky Best. The BUGS project: Evolution, critique and future directions. *Statistics in Medicine*, 28:3049–3067, 2009.

Diganta Misra. Mish: A self regularized non-monotonic activation function. In *British Machine Vision Conference*, 2020.

Hadi Mohasel Afshar and Justin Domke. Reflection, refraction, and Hamiltonian Monte Carlo. In *Annual Conference on Neural Information Processing Systems*, 2015.

Radford M Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, pages 47–95, 2011.

Akihiko Nishimura, David B Dunson, and Jianfeng Lu. Discontinuous Hamiltonian Monte Carlo for discrete parameters and discontinuous likelihoods. *Biometrika*, 107:365–380, 2020.

Ari Pakman and Liam Paninski. Auxiliary-variable exact Hamiltonian Monte Carlo samplers for binary distributions. In *Annual Conference on Neural Information Processing Systems*, 2013.

Theodore Papamarkou, Maria Skoularidou, Konstantina Palla, Laurence Aitchison, Julyan Arbel, David Dunson, Maurizio Filippone, Vincent Fortuin, Philipp Hennig, Aliaksandr Hubin, et al. Position: Bayesian deep learning is needed in the age of large-scale AI. In *International Conference on Machine Learning*, 2024.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. In *International Conference on Learning Representations (ICLR) Workshop Track*, 2018.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable Laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

Trung Q Trinh, Markus Heinonen, Luigi Acerbi, and Samuel Kaski. Tackling covariate shift with node-based Bayesian neural networks. In *International Conference on Machine Learning*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Annual Conference on Neural Information Processing Systems*, 2017.

Yibin Wang, Haizhou Shi, Ligong Han, Dimitris Metaxas, and Hao Wang. BLoB: Bayesian low-rank adaptation by backpropagation for large language models. In *Annual Conference on Neural Information Processing Systems*, 2024.

A Additional Proofs

A.1 Proof of Theorem 1

We consider a general case where a particle starts at $(\boldsymbol{q}_0, \boldsymbol{p}_0)$ and ends at $(\boldsymbol{q}_1, \boldsymbol{p}_1)$ after a single leapfrog update. The new state $(\boldsymbol{q}_1, \boldsymbol{p}_1)$ is given by:

$$\boldsymbol{p}_{1/2} = \boldsymbol{p}_0 - \frac{\varepsilon}{2}\nabla U(\boldsymbol{q}_0) \tag{9}$$

$$\boldsymbol{q}_1 = \boldsymbol{q}_0 + \varepsilon \boldsymbol{p}_{1/2} \tag{10}$$

$$\boldsymbol{p}_1 = \boldsymbol{p}_{1/2} - \frac{\varepsilon}{2}\nabla U(\boldsymbol{q}_1). \tag{11}$$

Change in the kinetic energy. The difference in the kinetic energy is given by:

$$\begin{aligned}
\Delta K &= K(\boldsymbol{p}_1) - K(\boldsymbol{p}_0) \\
&= \frac{1}{2}(\|\boldsymbol{p}_1\|^2 - \|\boldsymbol{p}_0\|^2) \\
&= \frac{1}{2}\left(\|\boldsymbol{p}_0 - \frac{\varepsilon}{2}\nabla U(\boldsymbol{q}_0) - \frac{\varepsilon}{2}\nabla U(\boldsymbol{q}_1)\|^2 - \|\boldsymbol{p}_0\|^2\right) \\
&= \frac{1}{2}\Big(\|\boldsymbol{p}_0\|^2 + \frac{\varepsilon^2}{4}\|\nabla U(\boldsymbol{q}_0)\|^2 + \frac{\varepsilon^2}{4}\|\nabla U(\boldsymbol{q}_1)\|^2 \\
&\quad - \varepsilon \boldsymbol{p}_0 \cdot \nabla U(\boldsymbol{q}_0) - \varepsilon \boldsymbol{p}_0 \cdot \nabla U(\boldsymbol{q}_1) + \frac{\varepsilon^2}{2}\nabla U(\boldsymbol{q}_0) \cdot \nabla U(\boldsymbol{q}_1) - \|\boldsymbol{p}_0\|^2\Big) \\
&= -\frac{\varepsilon}{2}\,\boldsymbol{p}_0 \cdot (\nabla U(\boldsymbol{q}_0) + \nabla U(\boldsymbol{q}_1)) + \frac{\varepsilon^2}{4}\Big(\frac{1}{2}\,\|\nabla U(\boldsymbol{q}_0)\|^2 + \frac{1}{2}\,\|\nabla U(\boldsymbol{q}_1)\|^2 + \nabla U(\boldsymbol{q}_0) \cdot \nabla U(\boldsymbol{q}_1)\Big).
\end{aligned}$$

The equality between the second and third lines follows from equations (9) and (11). The first term is given by:

$$\begin{aligned}
&-\frac{\varepsilon}{2}\boldsymbol{p}_0 \cdot (\nabla U(\boldsymbol{q}_0) + \nabla U(\boldsymbol{q}_1)) \\
&= -\frac{\varepsilon}{2}\left(\boldsymbol{p}_{1/2} + \frac{\varepsilon}{2}\nabla U(\boldsymbol{q}_0)\right) \cdot (\nabla U(\boldsymbol{q}_0) + \nabla U(\boldsymbol{q}_1)) \\
&= -\frac{\varepsilon}{2}\,\boldsymbol{p}_{1/2} \cdot (\nabla U(\boldsymbol{q}_0) + \nabla U(\boldsymbol{q}_1)) - \frac{\varepsilon^2}{4}\left(\|\nabla U(\boldsymbol{q}_0)\|^2 + \nabla U(\boldsymbol{q}_0) \cdot \nabla U(\boldsymbol{q}_1)\right).
\end{aligned}$$

Substituting it into ΔK, we have:

$$\Delta K = -\frac{\varepsilon}{2}\,\boldsymbol{p}_{1/2} \cdot (\nabla U(\boldsymbol{q}_0) + \nabla U(\boldsymbol{q}_1)) + \frac{\varepsilon^2}{8}\left(\|\nabla U(\boldsymbol{q}_1)\|^2 - \|\nabla U(\boldsymbol{q}_0)\|^2\right). \tag{12}$$

Change in the potential energy. For a single step from $(\boldsymbol{q}_0, \boldsymbol{p}_0)$ to $(\boldsymbol{q}_1, \boldsymbol{p}_1)$ with a smooth activation function, we have:

$$\Delta U = U(\boldsymbol{q}_1) - U(\boldsymbol{q}_0) = \int_0^{\varepsilon} \frac{\partial U}{\partial q}(q_0 + t p_{1/2}) \cdot p_{1/2} dt.$$

We use the trapezoidal rule:

$$\int_a^b f(t)dt = \frac{(b-a)}{2}[f(b) + f(a)] - \frac{(b-a)^3}{12N^2} f''(\xi).$$

Following the above rule with $a = 0$, $b = \varepsilon$, $N = 1$ and the function $f(t) = \nabla U(\boldsymbol{q}_0 + t\boldsymbol{p}_{1/2}) \cdot \boldsymbol{p}_{1/2}$, we can express ΔU as:

$$\Delta U = \frac{\varepsilon}{2}\boldsymbol{p}_{1/2} \cdot (\nabla U(\boldsymbol{q}_0) + \nabla U(\boldsymbol{q}_1)) - \frac{\varepsilon^3}{12}\frac{d^2}{dt^2}\left(\nabla U(\boldsymbol{q}_0 + t\boldsymbol{p}_{1/2}) \cdot \boldsymbol{p}_{1/2}\right)\Big|_{t=\xi}.$$

The first derivative $f'(t)$ of $f(t)$ is given by:

$$\begin{aligned}
f'(t) &= \frac{d}{dt}\left(\nabla U(\boldsymbol{q}_0 + t\boldsymbol{p}_{1/2}) \cdot \boldsymbol{p}_{1/2}\right) \\
&= \frac{d}{dt}\left(\nabla U(\boldsymbol{q}_0 + t\boldsymbol{p}_{1/2})\right) \cdot \boldsymbol{p}_{1/2} \\
&= \left(\nabla^2 U(\boldsymbol{q}_0 + t\boldsymbol{p}_{1/2}) \cdot \boldsymbol{p}_{1/2}\right) \cdot \boldsymbol{p}_{1/2}.
\end{aligned}$$

We derive the second derivative $f''(t)$ of $f(t)$ using the chain rule:

$$\begin{aligned} f''(t) &= \frac{d}{dt}\left(\nabla^2 U(\boldsymbol{q}_0 + t\boldsymbol{p}_{1/2}) \cdot \boldsymbol{p}_{1/2}\right) \cdot \boldsymbol{p}_{1/2} \\ &= \left(\left(\nabla^3 U(\boldsymbol{q}_0 + t\boldsymbol{p}_{1/2}) \cdot \boldsymbol{p}_{1/2}\right) \cdot \boldsymbol{p}_{1/2}\right) \cdot \boldsymbol{p}_{1/2} \\ &= \sum_{i,j,k} \frac{\partial^3 U}{\partial q_i \partial q_j q_k}(\boldsymbol{q}_0 + t\boldsymbol{p}_{1/2}).(\boldsymbol{p}_{1/2})_i.(\boldsymbol{p}_{1/2})_j.(\boldsymbol{p}_{1/2})_k \\ &= \langle \nabla^3 U(\boldsymbol{q}_0 + t\boldsymbol{p}_{1/2}), \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \rangle. \end{aligned}$$

Thus, the change of ΔU is:

$$\Delta U = \frac{\varepsilon}{2}\boldsymbol{p}_{1/2} \cdot (\nabla U(\boldsymbol{q}_0) + \nabla U(\boldsymbol{q}_1)) - \frac{\varepsilon^3}{12}\langle \nabla^3 U(\boldsymbol{q}_0 + \xi\boldsymbol{p}_{1/2}), \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \rangle. \tag{13}$$

Combining (12) and (13), we have the local error $\Delta H = H(\boldsymbol{q}_1) - H(\boldsymbol{q}_0) = \Delta U + \Delta K$:

$$\Delta H = \frac{\varepsilon^2}{8}\left(\|\nabla U(\boldsymbol{q}_1)\|^2 - \|\nabla U(\boldsymbol{q}_0)\|^2\right) - \frac{\varepsilon^3}{12}\langle \nabla^3 U(\boldsymbol{q}_0 + \xi\boldsymbol{p}_{1/2})\ ,\ \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \rangle.$$

A.2 Proof of Regression Example in Section 3.2

We consider a single-hidden-layer BNN with one hidden neuron:

- Parameters: $w, v \in \mathbb{R}$
- Activation function: $\varphi : \mathbb{R} \to \mathbb{R}$ (smooth, C^3)
- Network output: $z = v\varphi(wx)$
- Data point: (x, y) with $x \in \mathbb{R}$, $y \in \mathbb{R}$
- Prior: $\pi(w), \pi(v) \sim \mathcal{N}(0, 1)$
- Residual: $r = y - z$
- Negative log-likelihood (squared loss): $(y - z)^2$
- Potential energy: $U(w, v) = \frac{1}{2}(w^2 + v^2) + (y - z)^2$

Let $u = wx$ be the preactivation, we pre-compute some reused facts:

$$\begin{aligned} \frac{\partial \varphi}{\partial w} &= x\varphi', \quad \frac{\partial \varphi}{\partial v} = 0, \\ \frac{\partial z}{\partial w} &= vx\varphi', \quad \frac{\partial z}{\partial v} = \varphi, \\ \frac{\partial r}{\partial w} &= -vx\varphi', \quad \frac{\partial r}{\partial v} = -\varphi. \end{aligned}$$

Note that for simplicity, we write φ instead of $\varphi(u)$. We derive the first-order derivatives with respect to w and v:

$$\begin{aligned} \frac{\partial U}{\partial w} &= w - 2rvx\varphi', \\ \frac{\partial U}{\partial v} &= v - 2r\varphi. \end{aligned}$$

We can derive the second-order derivatives as follows:

$$\begin{aligned} \frac{\partial^2 U}{\partial w\, \partial w} &= 1 - 2vx\left[\frac{\partial r}{\partial w}\varphi' + rx\ \varphi''\right] = 1 + 2v^2x^2\ (\varphi')^2 - 2rvx^2\ \varphi'', \\ \frac{\partial^2 U}{\partial v\, \partial v} &= 1 - 2\varphi\frac{\partial r}{\partial v} = 1 + 2\varphi^2, \\ \frac{\partial^2 U}{\partial w\, \partial v} &= -2x\varphi'(r - v\varphi) = -2x\varphi'\ (y - 2z). \end{aligned}$$

The third-order derivatives are given by:

$$\begin{aligned}
\frac{\partial^3 U}{\partial v\,\partial v\,\partial v} &= 0,\\
\frac{\partial^3 U}{\partial w\,\partial w\,\partial w} &= 6v^2x^3\varphi'\varphi'' - 2vx^3(y-z)\varphi''',\\
\frac{\partial^3 U}{\partial w\,\partial w\,\partial v} &= 4vx^2(\varphi')^2 - 2x^2\varphi''(y-2z),\\
\frac{\partial^3 U}{\partial w\,\partial v\,\partial v} &= 4x\varphi\varphi'.
\end{aligned}$$

Using the triangle inequality, we have:

$$\begin{aligned}
\|\nabla^3 U(\boldsymbol{q})\|_\infty \le \left|\frac{\partial^3 U}{\partial w\,\partial w\,\partial w}\right| + \left|\frac{\partial^3 U}{\partial w\,\partial w\,\partial v}\right| + \left|\frac{\partial^3 U}{\partial w\,\partial v\,\partial v}\right| &\le 6v^2\,|x|^3\,\|\varphi'\|_\infty\,\|\varphi''\|_\infty\\
+ 2|v|\,|x|^3\,|y-z|\,\|\varphi'''\|_\infty + 4|v|x^2\|\varphi'\|_\infty^2 + 2x^2\|\varphi''\|_\infty\,|y-2z| + 4|x|\,|\varphi|\,\|\varphi'\|_\infty&.
\end{aligned}$$

A.3 Proof of Classification Example in Section 3.2

We consider a single-hidden-layer BNN with a single hidden unit:

- Parameters: $w, v \in \mathbb{R}$
- Activation function: $\varphi : \mathbb{R} \to \mathbb{R}$ (smooth, C^3)
- Predictive function: $p = f(x; w) = \sigma(v\varphi(wx))$
- Data point: (x, y) with $x \in \mathbb{R}$, $y \in \{0, 1\}$
- Prior: $\pi(w), \pi(v) \sim \mathcal{N}(0, 1)$
- Negative log-likelihood (cross-entropy loss): $-[y \log p + (1-y)\log(1-p)]$
- Potential energy: $U(w, v) = \frac{1}{2}(w^2 + v^2) - y(v\varphi(wx)) + \log\left(1 + e^{v\varphi(wx)}\right)$

Let $u = wx$, $z = v\varphi(wx)$, $p = \sigma(z)$ and $q = (1-p)$. Again, to shorten the notations, we write φ instead of $\varphi(u)$. We pre-compute some repeatedly used facts:

$$\begin{aligned}
\frac{\partial U}{\partial z} &= p - y, \quad \frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z)) = pq,\\
\frac{\partial z}{\partial w} &= v\varphi' x, \quad \frac{\partial z}{\partial v} = \varphi,\\
\frac{\partial p}{\partial w} &= pqv\varphi' x, \quad \frac{\partial p}{\partial v} = pq\varphi,\\
\frac{\partial (pq)}{\partial z} &= \frac{\partial(\sigma(z)(1-\sigma(z)))}{\partial z} = pq(1-2p),\\
\frac{\partial (pq)}{\partial w} &= pq(1-2p)v\varphi'\,x, \quad \frac{\partial (pq)}{\partial v} = pq(1-2p)\varphi.
\end{aligned}$$

We have the first-order derivatives of U *w.r.t.* w and v:

$$\begin{aligned}
\frac{\partial U}{\partial w} &= (p-y)\cdot v\varphi' x + w,\\
\frac{\partial U}{\partial v} &= (p-y)\cdot \varphi + v.
\end{aligned}$$

For the second-order derivatives, we have:

$$\begin{aligned}
\frac{\partial^2 U}{\partial w\,\partial w} &= pqv^2(\varphi')^2x^2 \;+\; (p-y)v\varphi''x^2 \;+\; 1,\\
\frac{\partial^2 U}{\partial v\,\partial v} &= \;pq\,\varphi^2 \;+\; 1,\\
\frac{\partial^2 U}{\partial w\,\partial v} &= pq\,v\,\varphi\varphi'\,x \;+\; (p-y)\,\varphi'\,x.
\end{aligned}$$

Applying the chain rule, we derive the third-order derivatives *w.r.t.* w and v as follows:

$$\begin{aligned}
\frac{\partial^3 U}{\partial w \partial w \partial w} &= pq(1-2p)\, v^3 (\varphi')^3\, x^3 \;+\; 3\, pq\, v^2\, \varphi' \varphi''\, x^3 \;+\; (p-y)\, v\, \varphi'''\, x^3, \\
\frac{\partial^3 U}{\partial v\, \partial v\, \partial v} &= pq(1-2p)\, \varphi^3, \\
\frac{\partial^3 U}{\partial w\, \partial v\, \partial v} &= pq(1-2p) v x \varphi' \varphi^2 + 2pqx\varphi\varphi', \\
\frac{\partial^3 U}{\partial w\, \partial w\, \partial v} &= pq(1-2p) v^2 x^2 \varphi (\varphi')^2 + 2pqvx^2 (\varphi')^2 + pqvx^2 \varphi\varphi'' + (p-y) x^2 \varphi''.
\end{aligned}$$

An upper bound on the infinity norm of the third-order curvature is given as:

$$\begin{aligned}
\|\nabla^3 U(\boldsymbol{q})\|_\infty \le \left|\frac{\partial^3 U}{\partial w^3}\right| + \left|\frac{\partial^3 U}{\partial w \partial w \partial v}\right| + \left|\frac{\partial^3 U}{\partial w \partial v \partial v}\right| + \left|\frac{\partial^3 U}{\partial v^3}\right| \le C(|v|^3\, \|\varphi'\|_\infty^3\, |x|^3 \\
+ v^2\, \|\varphi'\|_\infty\, \|\varphi''\|_\infty\, |x|^3 + |v|\, \|\varphi'''\|_\infty\, |x|^3 + |\varphi|^3 + |v|\, |x|\, \|\varphi'\|_\infty\, \varphi^2 + |x|\, \|\varphi'\|_\infty\, |\varphi| \\
+ v^2 x^2 |\varphi|\, \|\varphi'\|_\infty^2 + |v| x^2 (\|\varphi'\|_\infty^2 + |\varphi|\, \|\varphi''\|_\infty) + x^2 \|\varphi''\|_\infty),
\end{aligned}$$

where C is a universal constant.

A.4 Proof of Corollary 1

We extend Theorem 1 for a longer trajectory containing L leapfrog steps. In particular, the particle starts from the initial point $(\boldsymbol{q}_0, \boldsymbol{p}_0)$, arrives at transition states $\{(\boldsymbol{q}_1, \boldsymbol{p}_1), (\boldsymbol{q}_2, \boldsymbol{p}_2), \ldots, (\boldsymbol{q}_{L-1}, \boldsymbol{p}_{L-1})\}$, and ends at $(\boldsymbol{q}_L, \boldsymbol{p}_L)$.

$$\begin{aligned}
\Delta H &= \Delta H_1 + \Delta H_2 + \cdots + \Delta H_L \\
&= \frac{\varepsilon^2}{8} \sum_{l=0}^{L-1} \left(\|\nabla U(\boldsymbol{q}_{l+1})\|^2 - \|\nabla U(\boldsymbol{q}_l)\|^2 \right) - \frac{\varepsilon^3}{12} \sum_{l=0}^{L-1} \langle \nabla^3 U(\boldsymbol{q}_l + \xi_l \boldsymbol{p}_{l+\frac{1}{2}}), \boldsymbol{p}_{l+\frac{1}{2}} \otimes \boldsymbol{p}_{l+\frac{1}{2}} \otimes \boldsymbol{p}_{l+\frac{1}{2}} \rangle \\
&= \frac{\varepsilon^2}{8} \left(\|\nabla U(\boldsymbol{q}_L)\|^2 - \|\nabla U(\boldsymbol{q}_0)\|^2 \right) - \frac{\varepsilon^3}{12} \sum_{l=0}^{L-1} \langle \nabla^3 U(\boldsymbol{q}_l + \xi_l \boldsymbol{p}_{l+\frac{1}{2}}), \boldsymbol{p}_{l+\frac{1}{2}} \otimes \boldsymbol{p}_{l+\frac{1}{2}} \otimes \boldsymbol{p}_{l+\frac{1}{2}} \rangle.
\end{aligned}$$

A.5 Proof of Theorem 2

Consider a leapfrog step that starts at $(\boldsymbol{q}_0, \boldsymbol{p}_0)$, ends at $(\boldsymbol{q}_1, \boldsymbol{p}_1)$, and crosses the surfaces of non-differentiability at $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_k$ at time $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k$.

Change in the kinetic energy.

$$\Delta K = -\frac{\varepsilon}{2}\, \boldsymbol{p}_{1/2} \cdot (\nabla U(\boldsymbol{q}_0) + \nabla U(\boldsymbol{q}_1)) + \frac{\varepsilon^2}{8} \left(\|\nabla U(\boldsymbol{q}_1)\|^2 - \|\nabla U(\boldsymbol{q}_0)\|^2 \right).$$

Change in the potential energy.

$$\begin{aligned}
\Delta U &= \sum_{i=0}^{k} \int_{\varepsilon_i}^{\varepsilon_{i+1}} \nabla U(\boldsymbol{q}_0 + t\boldsymbol{p}_{1/2}).\boldsymbol{p}_{1/2} dt \\
&= \sum_{i=0}^{k} \Bigg[\frac{\varepsilon_{i+1} - \varepsilon_i}{2} \big(\nabla U^+(\boldsymbol{z}_i) + \nabla U^-(\boldsymbol{z}_{i+1}) \big) \cdot \boldsymbol{p}_{1/2} \\
&\quad - \frac{(\varepsilon_{i+1} - \varepsilon_i)^3}{12} \Big\langle \nabla^3 U(\boldsymbol{q}_0 + \xi_i \boldsymbol{p}_{1/2}), \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \Big\rangle \Bigg] \\
&= \sum_{i=0}^{k} \left[\frac{\Delta\varepsilon_i}{2} \left(\nabla U^+(\boldsymbol{z}_i) + \nabla U^-(\boldsymbol{z}_{i+1}) \right) \cdot \boldsymbol{p}_{1/2} \;-\; \frac{(\Delta\varepsilon_i)^3}{12} \langle \nabla^3 U(\boldsymbol{q}_0 + \xi_i \boldsymbol{p}_{1/2}), \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \rangle \right].
\end{aligned}$$

Adding these two terms, we have:

$$\Delta H = \boldsymbol{p}_{1/2} \cdot \sum_{i=0}^{k} \frac{\Delta\varepsilon_i}{2} \left(\nabla U^-(\boldsymbol{z}_{i+1}) + \nabla U^+(\boldsymbol{z}_i)\right) - \frac{\varepsilon}{2}\, \boldsymbol{p}_{1/2} \cdot \left(\nabla U(\boldsymbol{q}_0) + \nabla U(\boldsymbol{q}_1)\right)$$
$$+ \frac{\varepsilon^2}{8} \left(\|\nabla U(\boldsymbol{q}_1)\|^2 - \|\nabla U(\boldsymbol{q}_0)\|^2\right) - \sum_{i=0}^{k} \frac{\Delta\varepsilon_i^3}{12} \langle \nabla^3 U(\boldsymbol{q}_0 + \xi_i \boldsymbol{p}_{1/2}), \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \otimes \boldsymbol{p}_{1/2} \rangle.$$

B Activation Analyses

Derivatives of activation functions. In this section, we analyze the smoothness of various activation functions by examining their higher-order derivatives. To simplify our notation, let σ denote the Sigmoid function, Φ the cumulative distribution function (c.d.f.) of the standard Gaussian distribution, and $\omega(u) = \tanh(\ln(1 + e^u))$ the core component of the Mish activation. The explicit formulas for these functions and their respective third-order derivatives are summarized in Table 1, while their calculated infinity norms are provided in Table 2. Our results indicate a clear correlation between the magnitude of $\|\varphi'''\|_\infty$ and the accumulation of error along the integration trajectory. As illustrated in Figure 1, GELU and Mish exhibit $\|\varphi'''\|_\infty$ values approximately double that of Swish. This increased value in the third-order derivative directly translates to higher discretization error within the HMC leapfrog integrator, underscoring the critical role of functional smoothness in sampling stability.

Table 1: Activation functions and their third-order derivatives in compact form. We omit the argument u of σ, $\tanh$ and ω for brevity.

Activation	**Formula** $\varphi(u)$	**Third-Order Derivative** $\varphi'''(u)$
Sigmoid (σ)	$\dfrac{1}{1+e^{-u}}$	$\sigma(1-\sigma)[1-6\sigma(1-\sigma)]$
Tanh	$\dfrac{e^u - e^{-u}}{e^u + e^{-u}}$	$2\big(1-\tanh^2\big)\big(3\tanh^2 - 1\big)$
Softplus	$\log(1+e^u)$	$\sigma(1-\sigma)(1-2\sigma)$
GELU	$u \cdot \Phi(u)$	$\phi(u)(u^3 - 4u)$
Swish	$u \cdot \sigma(u)$	$3\sigma(1-\sigma)\big[1-2\sigma\big] + u\sigma(1-\sigma)\big[1-6\sigma(1-\sigma)\big]$
Mish	$u \cdot \omega(u)$	$\sigma(1-\omega^2)\left[u\left(6\sigma^2\omega(\omega+1) - 6\sigma\omega - 3\sigma + 1\right) + 3(1-\sigma(2\omega+1))\right]$

Table 2: Infinity norms of first-, second- and third-order gradients of different activation functions and their corresponding output range.

Activation	$\|\varphi'(u)\|_\infty$	$\|\varphi''(u)\|_\infty$	$\|\varphi'''(u)\|_\infty$	**Range of** $\varphi(u)$
Sigmoid	0.250	0.0962	0.125	$(0, 1)$
Tanh	1.000	0.7699	2.000	$(-1, 1)$
Softplus	1.000	0.250	0.096	$(0, \infty)$
GELU	1.1289	0.5117	0.779	$[-0.17, \infty)$
Swish	1.0890	0.6051	0.308	$[-0.278, \infty)$
Mish	1.0998	0.4999	0.588	$[-0.308, \infty)$

Comparison among Softplus, GELU, Swish and Mish. We show that Softplus $> \max(0, u) \geq$ GELU, Swish and Mish since this order contributes to the error order in Figure 1. We first show that

Softplus $> \max(0, u)$. The Softplus function is given as:

$$\text{Softplus}(u) = \ln(1 + e^u) > \begin{cases} \ln(e^u) = u & u \geq 0, \\ \ln(1) = 0 & u < 0. \end{cases} \tag{14}$$

Next, we prove that $\max(0, u) \geq \text{GELU}, \text{Swish}, \text{Mish}$. These three activation functions share the following structure:

$$f(u) = u \cdot g(u), \tag{15}$$

where $g(u)$ is a gating function. For Swish, GELU and Mish, $g(u)$ is $\sigma(u)$ (sigmoid function), $\Phi(u)$ (Gaussian c.d.f.), and $\tanh(\ln(1 + e^u))$, respectively. Therefore, it is strictly bounded between 0 and 1. Consequently, $\max(0, u) \geq \text{GELU}, \text{Swish}, \text{Mish}$. This result can be combined with the classification example to explain the error behaviour of Softplus in Figure 1. We provide a visualization of activation function in Figure 5.

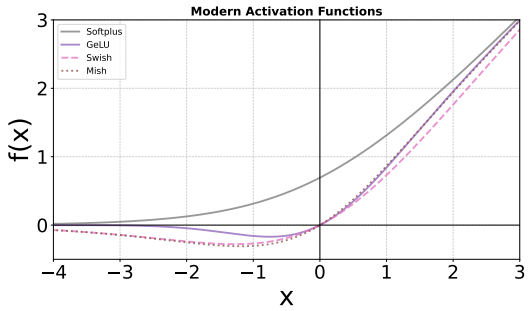
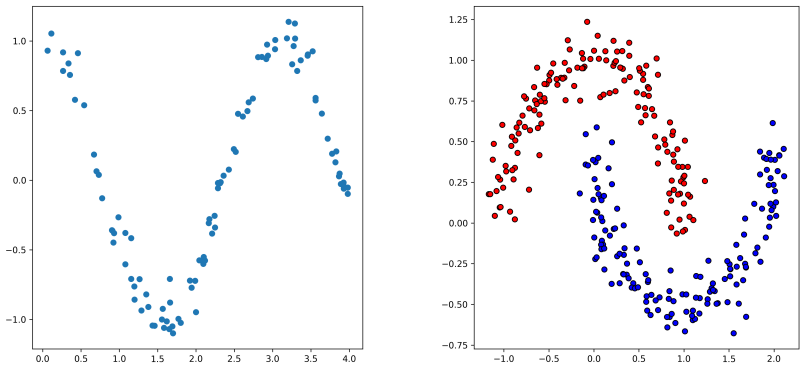


Figure 5: Visualization of different activation functions. Softplus is a pointwise upper bound for GELU, Swish and Mish.

C More Details on Experiment Settings in Section 5.1

Visualization of synthetic datasets. We visualize both synthetic datasets for regression and classification used in our simulations in Figure 6.



(a) Regression (b) Classification

Figure 6: Visualization of synthetic regression and classification datasets.

Network configuration. Both tasks employ a single-hidden-layer Bayesian feed-forward neural network with 50 hidden units. In summary, the total parameter count is $d = 151$ for regression and $d = 252$ for classification. We assume that the distribution of network parameters are fully independent, and place independent standard normal priors on the parameters, that is, $\pi(q) = \mathcal{N}(0, \boldsymbol{I}_d)$. The loss functions are mean squared error and binary cross-entropy for regression and classification, respectively. To examine the efficiency of HMC for BNNs with different activation functions, we investigate eight activation functions spanning three categories: *bounded smooth* (Sigmoid, Tanh),

unbounded smooth (GELU, Swish, Mish, Softplus), and *piecewise* (ReLU, LeakyReLU). This selection covers the most commonly used activations in modern architectures, including transformers [Vaswani et al., 2017] and convolutional networks [He et al., 2016]. Crucially, unbounded smooth activations satisfy Assumption 2 yet exhibit unbounded activation growth, making them the main focus of our study. Our code is implemented in JAX [Bradbury et al., 2018]. The experiments for toy datasets run on Google Colab with 12GB T4 GPU, while experiments on TMDb dataset use a fraction of an H200 GPU with 18GB VRAM.

D More Experiment Results

D.1 Error Decomposition

In Table 3, we provide the averaged values of each component in Figure 1. Clearly, the values for third-order curvature dominate the gradient value for most activations and strongly influence the efficiency of the HMC sampler.

Table 3: Error decomposition of different activation functions.

Activation	**Regression**		**Classification**	
	Gradient	Third-order	Gradient	Third-order
Sigmoid	0.05	2.47×10^{-6}	0.04	0.001
Tanh	121.52	507.11	0.05	0.03
Softplus	277.14	2099.58	21.28	376.59
GELU	61.83	4276.60	1.85	17.88
Mish	134.39	3696.57	3.52	49.31
Swish	114.57	1963.17	2.36	29.68

D.2 Multi-chain Diagnostics Under Common HMC Settings

We report below the effective sample size (ESS), integrated autocorrelation time (IACT), and multi-chain $\hat{R}$ for the log-posterior over 5 independent chains. Under the same HMC configuration in Section 5.1, Sigmoid yields better results than Softplus and Swish on all metrics. These results are consistent with our energy-error analysis: at the same step size and trajectory length, the bounded activation produces more stable trajectories and substantially better log-posterior mixing than the smooth unbounded activations. We note that the large $\hat{R}$ values for Softplus and Swish indicate these chains did not converge.

Table 4: Acceptance and log-posterior multi-chain diagnostics under common HMC settings. Values with $\pm$ are reported as mean $\pm$ standard deviation across five chains; $\widehat{R}$ is computed across chains.

Activation	Acceptance rate	ESS	IACT	$\widehat{R}$
Sigmoid	0.916 ± 0.008	45.3 ± 15.5	48.7 ± 17.0	1.012
Softplus	0.033 ± 0.037	8.6 ± 8.5	833.2 ± 821.5	2.722
Swish	0.195 ± 0.114	17.5 ± 13.9	303.5 ± 377.3	1.655

D.3 Sensitivity to a Constant Mass Matrix

We can extend Theorem 1 for the local error of HMC with an arbitrary constant matrix $\boldsymbol{M} \succ 0$ as below:

$$\Delta H = \frac{\varepsilon^2}{8}\Big[\|\nabla U(q_1)\|^2_{\boldsymbol{M}^{-1}} - \|\nabla U(q_0)\|^2_{\boldsymbol{M}^{-1}}\Big] - \frac{\varepsilon^3}{12}\nabla^3 U(q_\xi)\big[\boldsymbol{M}^{-1}p_{1/2}\big]^{\otimes 3}.$$

Thus, diagonal adaptation can correct coordinate-wise scale, while a full mass matrix can correct correlated or rotated directions. It may reduce the gap when the excess curvature is concentrated in

directions downweighted by $\boldsymbol{M}^{-1}$, but is not guaranteed: isotropic scaling changes the absolute but not the relative error. In addition, a fixed $\boldsymbol{M}$ cannot remove state-dependent curvature or a ReLU gradient jump.

We conduct an experiment to examine the effect of the mass matrix. We use the setting in Section 5.1 and find the adaptive pre-conditioning matrix during the warm-up phase as in Stan's implementation. We report the results for the identity, diagonal and dense mass matrices below. The dense matrix substantially reduces the error and restores high acceptance for Softplus and Swish, whereas identity and diagonal mass matrices fail to produce valid samples. However, dense estimates are highly ill-conditioned and yield low ESS for all three activations. Therefore, pre-conditioning mitigates but does not eliminate the activation-dependent gap.

Table 5: Acceptance rate, ESS, and IACT under different mass matrices.

Activation	Mass	Accept rate	ESS	IACT
Sigmoid	Identity	0.917 ± 0.003	43.58 ± 8.37	47.08 ± 9.32
	Diagonal	0.966 ± 0.003	27.67 ± 5.15	73.85 ± 12.70
	Dense	0.997 ± 0.001	3.81 ± 3.44	820.50 ± 500.39
Softplus	Dense	0.986 ± 0.004	7.54 ± 8.60	560.75 ± 394.41
Swish	Dense	0.983 ± 0.003	6.56 ± 5.36	544.07 ± 498.61

D.4 Activation-Rescaling Ablation

To assess the effect of activation scale, we repeat the toy regression experiment with $\phi_a(u) = a\phi(u)$ for $a \in \{0.25, 0.5, 1, 2\}$. Table 6 shows that the absolute scale affects acceptance rate for both activations. Softplus acceptance rate drops markedly between $a = 0.5$ and $a = 1$, whereas Sigmoid maintains substantially higher acceptance rate over the tested scales. The activation-dependent gap is therefore sensitive to scale; boundedness alone does not determine the outcome. Because convergence diagnostics were not reported for this ablation, we interpret the ESS and IACT values only as descriptive summaries.

Table 6: Acceptance and log-posterior chain summaries under activation rescaling in the toy regression setting. The scale a multiplies the activation output.

Scale a	Activation	Acceptance rate	ESS	IACT
0.25	Sigmoid	0.997	45.09	44.36
0.25	Softplus	0.926	32.66	61.24
0.50	Sigmoid	0.989	40.72	49.12
0.50	Softplus	0.603	68.27	29.29
1.00	Sigmoid	0.972	19.45	102.84
1.00	Softplus	0.001	1.21	1649.62
2.00	Sigmoid	0.791	44.65	44.80
2.00	Softplus	0.001	1.66	1202.25

D.5 A Residual-Network Check

Our findings are not limited to MLPs because they only depend on derivatives of the posterior potential rather than on a specific connectivity pattern. For instance, consider the one-block residual model $f_{\text{res}}(x) = v(x + \phi(wx))$ for the regression example. Its third derivative is $\frac{\partial^3 U_{\text{res}}}{\partial w^3} = 6v^2x^3\phi'(wx)\phi''(wx) - 2vx^3\left[y - v(x + \phi(wx))\right]\phi'''(wx)$. Compared with the plain model, the exact difference is $\frac{\partial^3 U_{\text{res}}}{\partial w^3} - \frac{\partial^3 U_{\text{plain}}}{\partial w^3} = 2v^2x^4\phi'''(wx)$. Thus, the skip connection changes the coefficient of the activation-induced curvature but does not entirely remove it. It may reduce the second term when the skip path improves the residual, although this is not guaranteed. We conduct an additional experiment to verify this claim, comparing MLPs with their residual counterparts at different depths. We report averaged results from 3 different runs in Table 7, where Sigmoid consistently yields more stable behaviour than Softplus and Swish for both MLP and residual networks.

Table 7: MLP and residual-network results at three depths. ESS and IACT are computed for the log posterior; values are reported as mean ± standard deviation over three runs.

Architecture	Depth	Activation	Acceptance rate	ESS	IACT
MLP	1	Sigmoid	0.872 ± 0.004	28.03 ± 6.02	73.37 ± 14.10
		Softplus	0.014 ± 0.012	1.868 ± 0.261	1081.0 ± 151.0
		Swish	0.125 ± 0.012	10.11 ± 5.75	246.3 ± 134.5
	2	Sigmoid	0.721 ± 0.048	34.76 ± 3.40	57.91 ± 5.67
		Softplus	0.028 ± 0.021	1.694 ± 0.680	1308.8 ± 491.9
		Swish	0.105 ± 0.011	1.883 ± 0.394	1090.5 ± 203.7
	3	Sigmoid	0.609 ± 0.083	24.14 ± 6.23	87.14 ± 25.16
		Softplus	0.013 ± 0.006	10.60 ± 14.98	800.3 ± 753.7
		Swish	0.023 ± 0.019	1.982 ± 1.133	1212.9 ± 542.2
Residual	1	Sigmoid	0.717 ± 0.054	19.78 ± 11.10	137.9 ± 101.5
		Softplus	0.055 ± 0.051	11.84 ± 7.10	206.0 ± 123.7
		Swish	0.179 ± 0.021	5.183 ± 4.955	645.0 ± 417.1
	2	Sigmoid	0.538 ± 0.034	20.27 ± 7.59	111.9 ± 53.18
		Softplus	0.018 ± 0.012	8.935 ± 1.053	225.8 ± 25.19
		Swish	0.079 ± 0.018	7.561 ± 9.848	754.1 ± 573.6
	3	Sigmoid	0.327 ± 0.081	16.88 ± 5.99	128.7 ± 44.22
		Softplus	0.007 ± 0.006	4.462 ± 3.256	771.1 ± 723.2
		Swish	0.018 ± 0.015	6.545 ± 4.150	382.4 ± 242.5

Table 8: Predictive performance and HMC diagnostics under posterior tempering. ESS and IACT are computed for the log posterior. Values are mean ± standard deviation over three seeds.

Activation	τ	Test MSE	Acceptance rate	ESS	IACT
Sigmoid	4.00	0.0207 ± 0.0003	0.998 ± 0.001	43.23 ± 15.55	50.76 ± 19.18
	2.00	0.0167 ± 0.0004	0.995 ± 0.001	58.66 ± 12.46	35.04 ± 6.72
	1.00	0.0135 ± 0.0003	0.993 ± 0.003	60.73 ± 22.39	36.13 ± 13.30
	0.50	0.0122 ± 0.0001	0.984 ± 0.004	54.90 ± 19.70	40.22 ± 16.10
	0.25	0.0115 ± 0.0001	0.961 ± 0.007	43.61 ± 30.53	89.28 ± 96.07
Softplus	4.00	0.0146 ± 0.0004	0.970 ± 0.007	51.47 ± 29.15	48.18 ± 25.98
	2.00	0.0130 ± 0.0001	0.931 ± 0.006	50.52 ± 19.95	45.05 ± 21.12
	1.00	0.0122 ± 0.0001	0.824 ± 0.049	33.37 ± 13.44	68.72 ± 33.25
	0.50	0.0119 ± 0.0002	0.476 ± 0.117	16.86 ± 12.74	265.03 ± 306.90
	0.25	0.0783 ± 0.1162	0.171 ± 0.198	15.41 ± 2.26	131.22 ± 19.28

This trend suggests that our conclusions are not specific to MLPs and can generalize to more complex networks.

D.6 Effect of Posterior Temperature

We examine how posterior temperature affects predictive performance and HMC sampling in the setting of Section 5. The tempered posterior is:

$$p_\tau(\mathbf{q} \mid \mathcal{D}) \propto p(\mathbf{q})\, p(\mathcal{D} \mid \mathbf{q})^{1/\tau}, \tag{16}$$

where $\tau = 1$ gives the nominal posterior and $\tau < 1$ gives a colder posterior. Table 8 reports the mean and standard deviation over three seeds. For Sigmoid, test MSE decreases as the posterior is cooled, while acceptance remains high. For Softplus, cooling from $\tau = 1$ to $\tau = 0.5$ slightly reduces test MSE but also substantially reduces acceptance and increases the reported IACT. At $\tau = 0.25$, Softplus results become highly variable. These results illustrate that predictive performance and sampling behavior need not improve together, and that the observed trade-off depends on the activation function.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Sections 3, 4 and 5.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 6.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: See Sections 3, 4 and the Appendix.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Section 5 and Appendix C.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have publicly released our code on GitHub. The repository link is provided in the abstract.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See Section 5 and Appendix C.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: See Section 5.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix C.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: We have reviewed the NeurIPS Code of Ethics and made an effort to ensure that every aspect of the work conforms with it.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The topic of our work is mainly theoretical with no direct societal impacts.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release any data or model.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cited the paper that introduced the Jax package used in our experiments in Section 5.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We used LLMs only for editing the paper.