Question 1 Part 1 Report

Yan Kin Chi
yan_kin.chi@unsw.edu.au

I. Introduction

Efficient Bayesian Inference with Latent Hamiltonian Neural Networks in No-U-Turn Sampling is a paper that builds upon traditional Hamiltonian neural networks (HNNs) but introduces the following novel contributions:

1. Introducing latent variable layers within traditional HNNs to enhance the neural network's expressivity and reduce integration errors.
2. Proposed a threshold-based monitoring scheme that monitors the Hamiltonian gradients inferred by the HNN which reverts to manual integration when errors are large.

II. Paper Critique

While the paper is well written, there are various aspects that are unclear that are essential to the understanding/ reproduction of the presented results. Below we break down the missing/confusing details into three sections corresponding to the three stages of method. They are generation of the L-HNN training data, training the L-HNN, and conducting Bayesian inference using No-U-Turn sampling (NUTS).

A. *Generating training data for L-HNN*

- **Stochastic Coverage**: Diversity in trajectories has been emphasized to be essential to avoid training L-HNNs on limited regions of the target density. However, when sampling trajectories, q(0) is defined as the final position of the previous trajectory. This does not maximise the training data's stochastic coverage of the target density and may cause the training data to be biased towards the neighbourhood of a specific mode.
- **Notation Inconsistencies**: Defining the amount of training data to be based on "numerical gradient evaluations" without explicitly defining its meaning makes the exact training details ambiguous. This is made worse with the provided code for generating training samples only based on sample count and trajectory length.
- **Distribution Specific Parameters**: During our testing, parameters for generating training samples are extremely varied for different distributions. However, the paper only explores the critical length (trajectory length) for the 3-D degenerate Rosenbrock density. This makes investigation extremely difficult.
-

B. *Training the L-HNN*

- **Latent Variables λ**: The paper does not establish a clear structural relationship between the latent variables and the underlying Hamiltonian system. Furthermore, it lacks an explanation of any novel methodology for integrating latent variables into a standard multilayer perceptron framework. As a result, the proposed L-HNN effectively functions as a HNN with an output layer of λ-dimensions.

  Within the paper, there is also no ablation study investigating the effect of modifying λ's dimensionality on performance. Therefore, the level of contribution of the

- **1-D Gaussian mixture density typo:**

  Eq. 18 (1-D Gaussian mixture density) is written as:

$$f(q) \propto 0.5\exp\left(\frac{(q-1)^2}{2\times 0.35^2}\right) + 0.5\exp\left(\frac{(q+1)^2}{2\times 0.35^2}\right)$$

  and is missing a negative sign within each exponent to match the Gaussian density form. Corrected:

$$f(q) \propto 0.5\exp\left(-\frac{(q-1)^2}{2\times 0.35^2}\right) + 0.5\exp\left(-\frac{(q+1)^2}{2\times 0.35^2}\right)$$

C. *Conducting Bayesian Inference using NUTS*

- **Excessive use of deterministic parameters**: While the inclusion of algorithm 4 and 5 provides a clear structural representation of the recursive build tree process for NUTS, the use of too many deterministic parameters makes it challenging to evaluate the generalizability of the approach. For example, thresholds Δ_{max}^{hnn} and Δ_{max}^{lf} being set to 10 and 1000 definitely may trigger unnecessary fallbacks (threshold too strict) or fail to detect errors in dynamic regions (too lenient).

III. Code Implementation

In this section we present investigations results using a code implementation that utilises TensorFlow.

A. *Methodology and Steps to Ensure Correctness:* To minimize implementation errors given the system's complexity, we closely replicated the original code's structure and logic using the TensorFlow framework. While alternative approaches such as leveraging TensorFlow Probability's HMC method exist and warrant future exploration, in this case we focus on maintaining fidelity to the original design.

B. *Sampling Experiments*

The method's sampling capabilities are mainly demonstrated by sampling the 1-D Gaussian mixture distribution. When training we trained 100 trajectory samples each with $T = 50$ and $\Delta t = 0.05$. To ensure the sample count sufficiently covers all regions of the 1-D Gaussian density, we plotted the position and momentum values for all training samples as shown from in Figure 1.

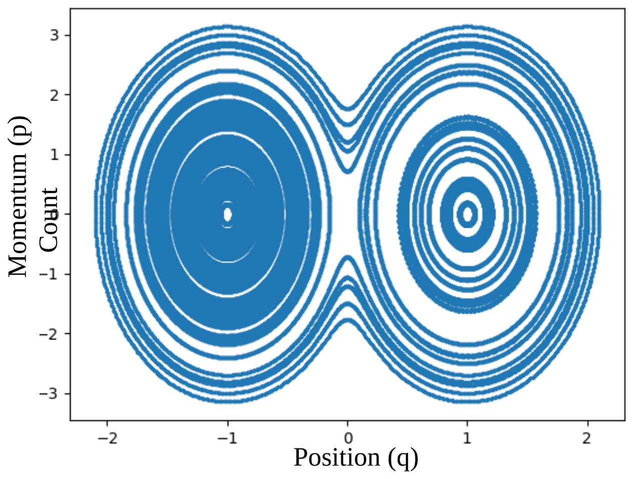

Figure 1: Phase Plot for all training samples

It can be seen that all regions of the density have been adequately covered.

Next, we generated 8000 samples from the posterior distribution using traditional HMC and L-HNN respectively and plotted the histograms for comparison.

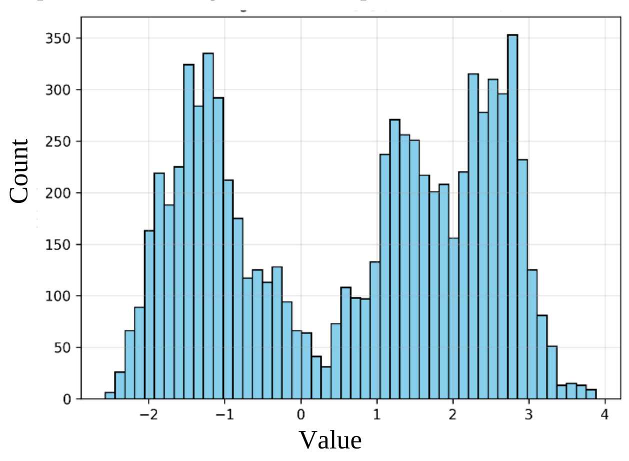

Figure 2: Sampled Distribution (L-HNN)

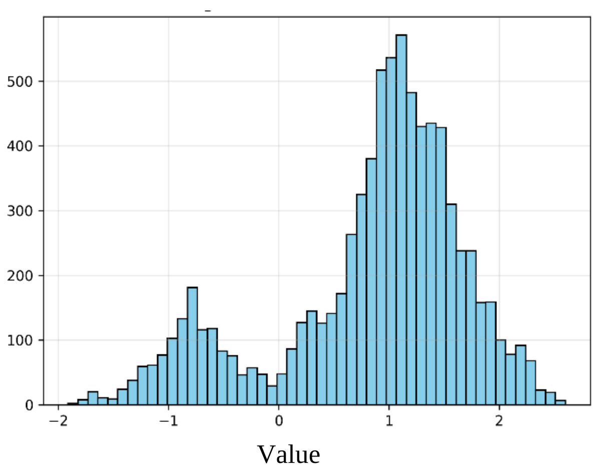

Figure 3: Sampled Distribution (HMC)

For this experiment we were unable to fully replicate the results shown in Figures 2a, 2b, and 2c of the original paper which show the histograms of HMC and L-HNN closely matching. Specifically, there are a few notable observations:

- **Stochastic Coverage**: During both the generation of training samples for L-HNN as well as during the execution of NUTS, one mode consistently dominates in terms of sample count This behaviour is expected because NUTS generates new samples by inheriting the previous position (q) while only drawing a new momentum (p) from a standard Gaussian distribution. This means transitioning from one mode to another often requires traversing regions of low posterior density, effectively "climbing" over energy barriers. To mitigate this issue, multiple chains could be initiated from each available mode, improving diversity in exploration.