Scaling Hamiltonian Monte Carlo Inference for Bayesian Neural Networks with Symmetric Splitting

Adam D. Cobb[1] **Brian Jalaian**[1]

[1]US Army Research Laboratory, Adelphi, Maryland, USA

Abstract

Hamiltonian Monte Carlo (HMC) is a Markov chain Monte Carlo (MCMC) approach that exhibits favourable exploration properties in high-dimensional models such as neural networks. Unfortunately, HMC has limited use in large-data regimes and little work has explored suitable approaches that aim to preserve the entire Hamiltonian. In our work, we introduce a new symmetric integration scheme for split HMC that does not rely on stochastic gradients. We show that our new formulation is more efficient than previous approaches and is easy to implement with a single GPU. As a result, we are able to perform full HMC over common deep learning architectures using entire data sets. In addition, when we compare with stochastic gradient MCMC, we show that our method achieves better performance in both accuracy and uncertainty quantification. Our approach demonstrates HMC as a feasible option when considering inference schemes for large-scale machine learning problems.

1 INTRODUCTION

To this day, Hamiltonian Monte Carlo remains the gold standard for inference in Bayesian Neural Networks (BNNs) [Duane et al., 1987, Neal, 1995]. However, the challenge of scaling HMC to applications involving large data sets limits its wide-scale use. Instead, approaches that utilise stochastic gradients are preferred due to their ability to better scale with data set size. The challenge for these stochastic gradient approaches is often finding a compromise between scalability and the modelling of uncertainty. However, if we cannot afford to compromise on uncertainty performance, then any feasible way of performing HMC would be extremely attractive. This would allow us to leverage the properties of HMC in modern deep learning architectures that are already starting to play a key part in safety-critical applications such as in medical diagnosis [Leibig et al., 2017], self-driving vehicles [Filos et al., 2020], and disaster response [Lu et al., 2020].

The two common approaches for performing Bayesian inference in large-scale models are stochastic variational inference (e.g. Graves [2011], Blundell et al. [2015], Gal and Ghahramani [2016]) and Markov chain Monte Carlo (MCMC). The latter MCMC approach only became practical for large data with the introduction of Stochastic Gradient Langevin Dynamics (SGLD) [Welling and Teh, 2011]. The appeal of MCMC (including the stochastic gradient variant) is that once the samples have converged to the target distribution, we can be confident that we are sampling from the distribution of interest and not from an approximate variational distribution. As a result there now exist multiple stochastic gradient MCMC schemes for inference in BNNs [Chen et al., 2014, Ding et al., 2014, Zhang et al., 2020]. In comparison to traditional implementations of MCMC, stochastic gradient approaches avoid using both a full likelihood model as well as a Metropolis-Hastings step. Instead, they tend to use a decaying learning rate and an approximation of the full likelihood. In contrast, we look to the original formulation of HMC and augment the Hamiltonian such that we can perform HMC over entire data sets.

In this work, we introduce a novel symmetric splitting integration scheme for HMC that is more robust than previous approaches and easy to implement as part of a Python package.[1] Our approach allows us to take advantage of the superior high-dimensional exploration of HMC, by letting chains with long trajectory lengths explore the parameter space of neural networks. We show how we are able to perform HMC without stochastic approximations, and achieve results that are more robust to large data sets. In addition to improving on previous proposed splitting formulations, we introduce a

[1]For the code, please refer to `https://github.com/AdamCobb/hamiltorch`.

realistic application of vehicle classification from acoustic data and show our novel symmetric split HMC inference scheme is also able to outperform its stochastic counterparts. In particular, our extensive analysis of uncertainty quantification shows the value of our approach over the stochastic MCMC baselines. In summary, our contributions are as follows:

- We introduce a new symmetric integration scheme for split HMC that does not rely on stochastic gradients.
- Our approach outperforms all previous proposed splitting schemes by achieving a larger effective sample size and a higher acceptance rate.
- We include a real-world example where our symmetric split scheme for HMC is able to provide better uncertainty estimates compared to SGD, SGLD, and SGHMC.
- We show that preserving the full Hamiltonian is still a viable option, even when it is necessary to split the data into smaller batches (e.g. due to memory limitations).

Our paper is structured as follows. Section 2 describes the related work. Section 3 covers previous theory on HMC and split HMC, enabling us to introduce our new approach of symmetric split HMC in Section 4. In Section 5, we compare our new scheme to previous splitting approaches, where we show how our new method scales more efficiently to large data. In Section 6, we compare symmetric split HMC with stochastic gradient approaches, demonstrating its superiority in uncertainty quantification. We then discuss the implications of our results in Section 7 and conclude is Section 8.

2 RELATED WORK

Augmenting the Hamiltonian to increase the feasibility of implementing full HMC is a well-known approach, yet it has been relatively untouched by the machine learning community in recent years, with the majority of effort focusing on stochastic gradient approaches (e.g. Chen et al. [2014], Ding et al. [2014], Zhang et al. [2020]). However, if we go back to the original work of Neal [1995], we see the introduction of splitting according to data subsets. Neal's motivation for splitting was to improve exploration by taking advantage of data sets that are redundant, such that one can achieve a bigger effective step size. This splitting approach, which we will refer to as *randomised splitting* due to its formulation, was not symmetrical and in subsequent work, Neal [2011] wrote that "some symmetrical variation ... might produce better results." The other appearance of split HMC in the literature comes from Shahbaba et al. [2014], where the Hamiltonian was split into two parts, such that one part was solved for analytically. This splitting approach facilitated larger step sizes to be taken and improved the exploration of the sampler. Shahbaba et al. [2014] also introduced the idea of splitting the data into two subsets, one for data lying near the decision boundary (first inferred by a MAP approximation) and the other for data far away from the boundary. This data splitting approach relies on both the symmetry of the log likelihood in logistic regression and on the ability to quickly perform a MAP approximation. Since these works, we are not aware of any further advances in split HMC that make it feasible to implement a full Hamiltonian on a single GPU. In Section 5, we will show that our symmetrical version of split HMC does better than randomised splitting, as had been predicted by Neal [2011].

A further challenge that arises in HMC is when the target distribution's geometry prevents easy mixing. One can look to work that aims to alleviate these issues such as the framework introduced by Girolami and Calderhead [2011] of Riemannian manifold HMC (RMHMC) or a more scalable approach to Bayesian hierarchical models by using the variant referred to as semi-separable HMC [Zhang and Sutton, 2014]. Another related line of work involves sampling in a transformed parameter space and then using the inverse transform to go back to the original space [Marzouk et al., 2016]. Recent work by Hoffman et al. [2019] actually demonstrates that this transformation can result in an equivalence to RMHMC, where the authors utilise normalising flows as their invertible transformation. Instead of improving exploration by alleviating the detrimental effects of bad geometry, another approach is to improve HMC's performance by constructing samplers that automatically tune their hyperparameters [Hoffman and Gelman, 2014, Betancourt, 2013, Wang et al., 2013]. These adaptations to HMC all follow a different direction to our work, since they do not directly address the challenge of scaling to large data with BNNs. However, these improvements are also complementary as they have the potential to be combined with our work in the future.

Stochastic gradient approaches to MCMC have become the main way to perform MCMC in BNNs since their introduction by Welling and Teh [2011]. While SGLD naturally arises by combining a Robbins-Monro-type algorithm [Robbins and Monro, 1951] with Langevin dynamics, the equivalent formulation for HMC (i.e. SGHMC) is more challenging and requires a limiting Gaussian assumption and the introduction of a friction term [Chen et al., 2014]. Although SGHMC has seen wide use in the machine learning community since its introduction (e.g Springenberg et al. [2016], Gustafsson et al. [2020]), there are various works that criticise the approach. Bardenet et al. [2014] demonstrate that relying on a Gaussian noise assumption can result in poor performance. Betancourt [2015] further criticises the use of stochastic gradients in HMC, reporting that the only way to reduce bias with data subsampling is to "subsample twice in a symmetric composition," where Betancourt [2015] directly refers to Neal [1995] and Shahbaba et al. [2014]. However, this proposition did not come with an applicable solution, but instead it came with a call to devise new ways of avoiding stochastic approximations. Despite some of the potential limitations of stochastic gradient MCMC approaches, there are now multiple implementations that enable successful inference in BNNs. For example, a popular direction of research has been to propose approaches that aim to cover multiple modes in the posterior, which has been motivated by the success of using deep ensembles [Lakshminarayanan et al., 2017, Ashukha et al., 2020]. Therefore, we now see schemes employing cyclic learning rates [Zhang et al., 2020] and utilising thermostats [Ding et al., 2014, Leimkuhler and Shang, 2016] to attain improved performance. As a final note, these approaches avoid relying on the Metropolis-Hastings step and instead decrease their step sizes to zero to ensure that they converge to the target distribution.

Despite the existence of a few examples that have employed data subsampling with full HMC, there is still no widely-used scheme that can compete with stochastic gradient MCMC on a large data set. Furthermore, some works in the field have criticised stochastic gradient approaches and hinted at symmetric splitting approaches as a possible way forward. In this paper, we will show that symmetric splitting does offer a scalable and robust approach for inference in BNNs.

3 SPLIT HAMILTONIAN MONTE CARLO

In this section we first provide a brief overview of HMC. We then describe the work by Neal [2011] and Shahbaba et al. [2014] and conclude by introducing our new variation of split HMC in Section 4.

3.1 HAMILTONIAN MONTE CARLO

HMC is a gradient-based MCMC sampler that employs Hamiltonian dynamics to traverse the parameter space of models. We can use HMC to overcome the challenge of performing inference in highly complex Bayesian models by materialising samples from the unnormalised log posterior via the proportionality,

$$p(\boldsymbol{\omega}|\mathbf{Y},\mathbf{X}) \propto p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega})\,p(\boldsymbol{\omega}),$$

which is derived from Bayes' rule. The model is a function of the parameters, $\boldsymbol{\omega} \in \mathbb{R}^D$, and is defined by the likelihood $p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega})$ and the prior $p(\boldsymbol{\omega})$, where $\{\mathbf{X},\mathbf{Y}\}$ are the input-output data pairs. The prior encodes assumptions over the model parameters before observing any data. To take advantage of Hamiltonian dynamics in our Bayesian model, we can augment our system by introducing a momentum variable $\mathbf{p} \in \mathbb{R}^D$, such that we now have a log joint distribution, $\log[p(\boldsymbol{\omega},\mathbf{p})] = \log[p(\boldsymbol{\omega}|\mathbf{Y},\mathbf{X})\,p(\mathbf{p})]$, that is proportional to the Hamiltonian, $H(\boldsymbol{\omega},\mathbf{p})$. If we let $p(\mathbf{p}) = \mathcal{N}(\mathbf{p}|\mathbf{0},\mathbf{M})$, where the covariance $\mathbf{M}$ denotes the mass matrix, our Hamiltonian can then be written as:[2]

$$H(\boldsymbol{\omega},\mathbf{p}) = \underbrace{-\log[p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega})\,p(\boldsymbol{\omega})]}_{\substack{\text{Potential Energy}\\ U(\boldsymbol{\omega})}} + \underbrace{{}^1\!/_2\,\mathbf{p}^\top\mathbf{M}^{-1}\mathbf{p}.}_{\substack{\text{Quadratic Kinetic Energy}\\ K(\mathbf{p})}} \tag{1}$$

This form consists of a quadratic kinetic energy term derived from the log probability distribution of a Gaussian and a potential energy term, which is our original Bayesian model. We can then use Hamiltonian dynamics to collect samples from our posterior distribution, which we know up to a normalising constant. These equations of motion,

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}t} &= \frac{\partial H}{\partial \mathbf{p}} = \mathbf{M}^{-1}\mathbf{p};\\ \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} &= -\frac{\partial H}{\partial \boldsymbol{\omega}} = \nabla_{\boldsymbol{\omega}} \log[p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega})\,p(\boldsymbol{\omega})],\end{aligned} \tag{2}$$

determine how trajectories on the parameter space propagate. However, solving these equations in practice requires simulation via discrete steps. The Stormer–Verlet or leapfrog integrator is an integration scheme that ensures reversibility by being symmetric in its sequencing, as well as being symplectic (which implies volume preservation as is required for Hamiltonian systems). Therefore we can introduce the leapfrog integrator by following the series of transformations:

$$\begin{aligned}&\mathbf{p}_{t+\epsilon/2} = \mathbf{p}_t + \frac{\epsilon}{2}\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t}(\boldsymbol{\omega}_t), \quad \boldsymbol{\omega}_{t+\epsilon} = \boldsymbol{\omega}_t + \epsilon\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}t}(\mathbf{p}_{t+\epsilon/2}),\\ &\mathbf{p}_{t+\epsilon} = \mathbf{p}_{t+\epsilon/2} + \frac{\epsilon}{2}\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t}(\boldsymbol{\omega}_{t+\epsilon}),\end{aligned} \tag{3}$$

where t is the leapfrog step iteration and ϵ is the step size. We can then use this scheme to simulate L steps that closely approximate the dynamics of the Hamiltonian system. Furthermore, for ease of notation, we can rewrite these transformations as a series of function compositions:

$$\phi^U_\epsilon : (\boldsymbol{\omega}_t,\mathbf{p}_t) \to (\boldsymbol{\omega}_t,\mathbf{p}_{t+\epsilon}),\ \phi^K_\epsilon : (\boldsymbol{\omega}_t,\mathbf{p}_t) \to (\boldsymbol{\omega}_{t+\epsilon},\mathbf{p}_t), \tag{4}$$

such that the overall symmetric mapping of Equation (3) can be denoted as $\phi^U_{\epsilon/2} \circ \phi^K_\epsilon \circ \phi^U_{\epsilon/2}$ [Strang, 1968].

Finally, HMC is performed by sampling $\mathbf{p}_t \sim p(\mathbf{p})$ and then using Hamiltonian dynamics, starting from $\{\mathbf{p},\boldsymbol{\omega}\}_t$, to propose a new pair of parameters $\{\mathbf{p},\boldsymbol{\omega}\}_{t+L}$. We then require a Metropolis-Hastings step to either accept or reject the proposed parameters to correct for any possible error due to approximating the dynamics with discrete steps. For further details of HMC, please refer to Neal [2011].

3.2 SPLIT HAMILTONIAN MONTE CARLO

The splitting of a Hamiltonian into a sum of its constituent parts has been previously described by both Leimkuhler

[2]We are ignoring the constants.

and Reich [2004] and Sexton and Weingarten [1992]. Its appearance in HMC in the context of data subsets first came in Neal [1995], who introduced the randomised splitting approach.[3] The general idea is to split the Hamiltonian into a sum of Q terms such that

$$H(\boldsymbol{\omega}, \mathbf{p}) = H_1(\boldsymbol{\omega}, \mathbf{p}) + H_2(\boldsymbol{\omega}, \mathbf{p}) + \cdots + H_Q(\boldsymbol{\omega}, \mathbf{p}). \quad (5)$$

This splitting is especially suited to the scenario, where the log-likelihood can be written as a sum over the data (i.e. data is independent), which is almost always the assumption for BNNs. Therefore Neal [1995] introduced the following split into M data subsets:

$$H(\boldsymbol{\omega}, \mathbf{p}) = \sum_{m=1}^{M} \left[U_m(\boldsymbol{\omega})/2 + K(\mathbf{p})/M + U_m(\boldsymbol{\omega})/2\right], \quad (6)$$

where $U_m(\boldsymbol{\omega}) = -\log(p(\boldsymbol{\omega}))/M - \ell_m(\boldsymbol{\omega})$ and $\ell_m(\boldsymbol{\omega}) = \log p(\mathbf{Y}_m|\mathbf{X}_m, \boldsymbol{\omega})$ is the log-likelihood over the data subset $\{\mathbf{X}_m, \mathbf{Y}_m\}$. Although the original purpose of this splitting was not with the intention of scaling to large data sets, its formulation nicely fits this scenario.

The order of the splitting is important because the sequence of mappings corresponding the Hamiltonian dynamics of each H_i must be symmetrical if we are to ensure the overall transition is reversible i.e. $H_i = H_{Q-i+1}$. Unfortunately, the above splitting follows the sequence $H_{3m-2}(\boldsymbol{\omega}, \mathbf{p}) = H_{3m}(\boldsymbol{\omega}, \mathbf{p}) = U_m(\boldsymbol{\omega})/2$ and $H_{3m-1}(\boldsymbol{\omega}, \mathbf{p}) = K(\mathbf{p})/M$, where $Q = 3M$, such that the flow follows

$$\phi_\epsilon^H = \phi_{\epsilon/2}^{U_1} \circ \phi_\epsilon^{K/M} \circ \phi_{\epsilon/2}^{U_1} \circ \cdots \circ \phi_{\epsilon/2}^{U_M} \circ \phi_\epsilon^{K/M} \circ \phi_{\epsilon/2}^{U_M}. \quad (7)$$

This splitting is no longer symmetrical and therefore requires an extra step whereby the ordering of the M subsets for each iteration is randomised. This randomisation ensures that the reverse trajectory and the forward trajectory have the same probability.

Other than randomised splitting, Shahbaba et al. [2014] introduced the "nested leapfrog", which followed a symmetrical formulation. The purpose of their "nested leapfrog" was to enable parts of the Hamiltonian to be solved either analytically or more cheaply. For their data splitting approach, they rely on a MAP approximation that must be computed in advance. This is then followed by an analysis of which data lies along the decision boundary. Their dependence on the quality of the MAP approximation as well as prior analysis of the data makes their approach less feasible when looking to scale to large data with BNNs. However, we offer our own data splitting baseline, which we refer to as naive splitting that is simply a nested leapfrog. This is the simplest way of building an integration scheme that both mimics full HMC and is symmetrical, i.e.

$$\phi_\epsilon^H = \phi_{\epsilon/2}^{U_1} \circ \phi_{\epsilon/2}^{U_2} \circ \cdots \circ \phi_\epsilon^{K} \circ \cdots \circ \phi_{\epsilon/2}^{U_2} \circ \phi_{\epsilon/2}^{U_1}. \quad (8)$$

[3]This is explicitly described by Neal [2011, Sec 5.1].

This splitting is equivalent to implementing the original leapfrog in (3), where we simply evaluate parts of the likelihood in chunks and then sum them.

We have now introduced two baselines that split the Hamiltonian according to data subsets. In the next section we will introduce our new symmetrical alternative that results in a better-behaved sampling scheme.

4 SYMMETRIC SPLIT HAMILTONIAN MONTE CARLO

Instead of following previous splitting approaches, we offer a symmetrical alternative that we will show to produce improved behaviour. We split our Hamiltonian into the same M data subsets as for randomised splitting, however we now change the ordering and rescale the kinetic energy term by a value depending on the number of splits. Our symmetrical splitting is structured such that $H_{2m-1}(\boldsymbol{\omega}, \mathbf{p}) = H_{2(2M-m)}(\boldsymbol{\omega}, \mathbf{p}) = U_m(\boldsymbol{\omega})/2$ and $H_{2j}(\boldsymbol{\omega}, \mathbf{p}) = H_{2(2M-j)-1}(\boldsymbol{\omega}, \mathbf{p}) = K(\mathbf{p})/D$, where $D = (M-1) \times 2$, $m = 1, \ldots, M$, and $j = 1, \ldots, M-1$. As an example the overall transformation for $M = 2$ would be written as

$$\phi_\epsilon^H = \phi_{\epsilon/2}^{U_1} \circ \phi_\epsilon^{K/2} \circ \phi_{\epsilon/2}^{U_2} \circ \phi_{\epsilon/2}^{U_2} \circ \phi_\epsilon^{K/2} \circ \phi_{\epsilon/2}^{U_1}, \quad (9)$$

where $D = 2$, and as a further example for $M = 3$:

$$\begin{aligned}\phi_\epsilon^H = \phi_{\epsilon/2}^{U_1} \circ \phi_\epsilon^{K/4} \circ \phi_{\epsilon/2}^{U_2} \circ \phi_\epsilon^{K/4} \circ \phi_{\epsilon/2}^{U_3} \\ \circ \phi_{\epsilon/2}^{U_3} \circ \phi_\epsilon^{K/4} \circ \phi_{\epsilon/2}^{U_2} \circ \phi_\epsilon^{K/4} \circ \phi_{\epsilon/2}^{U_1}, \quad (10)\end{aligned}$$

where $D = 4$. More generally, Algorithm 1 describes the symmetric split leapfrog scheme.

Algorithm 1 Symmetric Split Leapfrog Scheme

```
    Inputs: $\mathbf{p}_0, \boldsymbol{\omega}_0, \epsilon, L, M$
 1: $D = 2 \times (M-1)$        ▷ Set the scaling factor for the parameter update step.
 2: for $l$ in $1, \ldots, L$ do
 3:     for $m$ in $1, \ldots, M$ do
 4:         $\mathbf{p} = \mathbf{p} + \frac{\epsilon}{2}\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t}_{(m)}(\boldsymbol{\omega})$
 5:         if $m < M$ then
 6:             $\boldsymbol{\omega} = \boldsymbol{\omega} + \frac{\epsilon}{D}\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}t}(\mathbf{p})$
 7:         end if
 8:     end for
 9:     for $m$ in $M, \ldots, 1$ do        ▷ Note the reversal of the loop indexing.
10:         $\mathbf{p} = \mathbf{p} + \frac{\epsilon}{2}\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t}_{(m)}(\boldsymbol{\omega})$
11:         if $m > 1$ then
12:             $\boldsymbol{\omega} = \boldsymbol{\omega} + \frac{\epsilon}{D}\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}t}(\mathbf{p})$
13:         end if
14:     end for
15: end for
```

Unlike randomised splitting, our integrator is symmetrical and leads to a discretisation that is now reversible such that setting $\mathbf{p} = -\mathbf{p}$ results in the original $\boldsymbol{\omega}$. This property of reversibility is sufficient for ensuring the Markov chain converges to the target distribution [Robert and Casella, 2013, Page 244].

We can then implement symmetric split HMC by replacing HMC's original leapfrog integrator with Algorithm 1. This replacement provides the ability to operate sequentially on smaller individual batches of the data set, rather than requiring operations over the entire data. A key motivation for operating over batches arises from hardware constraints. For example, one may want to perform HMC on a GPU with limited memory (as is the case for BNNs). To complete the algorithm, the only additional requirement is that the Metropolis-Hastings step must also be calculated in batches to ensure its memory footprint is no more than that of the new integration scheme.

5 COMPARISON TO OTHER SPLITTING APPROACHES

We now compare our new approach to both randomised splitting and naive splitting. We also include full HMC as a comparison for when memory-constraints allow for operations over the entire data set.

5.1 LOGISTIC REGRESSION EXAMPLE

We start with a simple logistic regression example where the posterior is log-concave. For this example, we compare the three splitting approaches of naive split HMC, randomised split HMC, and symmetric split HMC. We use the MNIST data set [LeCun et al., 1998] and run 5 HMC chains for each inference scheme. Each chain is run for 3000 iterations, with a trajectory length of $L = 20$ and a step size of $\epsilon = 6 \times 10^{-4}$. For the splitting approaches, the data is split into 10 subsets, each of 4,800 digits.

Table 1 displays the results, where all approaches use the same hyperparameters. Our symmetric split HMC achieves both the highest acceptance rate and the highest mean effective sample size (Mean ESS). The table also includes full HMC and displays the Mean ESS normalised by the time taken. The main result is that our new splitting approach does better at exploring the target distribution compared to the alternative splitting approaches by materialising fewer correlated samples.

Table 1: Logistic regression example calculated over 5 HMC chains. The Mean ESS was calculated using Pyro's in-built function [Bingham et al., 2019] and averaging over the parameters. The Mean ESS / sec comes from dividing by the wall-clock time. The acceptance rate is reported with its standard deviations. A higher Mean ESS and a higher acceptance rate, demonstrate the better mixing performance from symmetric split HMC.

Inference Scheme	Acc. Rate	Mean ESS	Mean ESS / sec
Full HMC	0.87 ± 0.01	68.7	$\mathbf{0.211}$
Naive Split HMC	0.87 ± 0.01	68.7	0.018
Rand. Split HMC	0.45 ± 0.01	38.4	0.010
Sym. Split HMC	$\mathbf{0.96 \pm 0.01}$	$\mathbf{75.0}$	0.021

5.2 BNN REGRESSION EXAMPLE

We now illustrate the regression performance for a Bayesian neural network model, where we use the simple 1D data set from Izmailov et al. [2019] and set the architecture to a fully connected NN with 3 hidden layers of 100 units. Our model uses a Gaussian likelihood $p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) = \mathcal{N}(\mathbf{f}(\mathbf{X}; \boldsymbol{\omega}), \tau^{-1}\mathbf{I})$, where the output precision, τ, must be fitted to characterise the inherent noise (aleatoric uncertainty) in the data. We implement a Gaussian process model with a Matérn ${}^{3}/_{2}$ kernel to learn this output precision with GPyTorch [Gardner et al., 2018].[4] For the splitting approaches we section the data into four subsets of 100 training points each. All other hyperparameters are kept constant across the approaches to enable a fair comparison ($L = 30$, $\epsilon = 5e^{-4}$, $\mathbf{M} = \mathbf{I}$, and $p(\boldsymbol{\omega}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$).[5]

All inference schemes achieve comparable test log-likelihood scores (squared errors) and plateau after $200/1000$ samples are collected. However the acceptance rates across the schemes vary considerably, which can be seen from the results of Table 2. These results are calculated for ten randomly initialised HMC chains and show the mean and standard deviation for the acceptance rate, as well as the mean ESS.

Our symmetric splitting scheme achieves a higher acceptance rate than both randomised split HMC and naive split HMC for the same step size and trajectory length. These results further demonstrate that our new splitting approach provides a better effective sample size than the previous splitting approaches whilst also keeping a higher acceptance rate. In addition, we also provide the performance of full HMC and the mean ESS normalised by the time taken. Unsurprisingly, full HMC is more efficient in terms of wall-clock time, however in scenarios where splitting is necessary our new approach provides the better performance. An example of symmetric split HMC can be seen in Figure 1, where the credible intervals appropriately widen outside the range of the data.

[4]In practice, τ can be learnt using cross validation as is the case for higher-dimensional problems.

[5]These hyperparameters achieve a well-calibrated performance. For full HMC 30.5 % of the data lies outside the 1σ credible interval and 3.5 % for the 2σ interval.

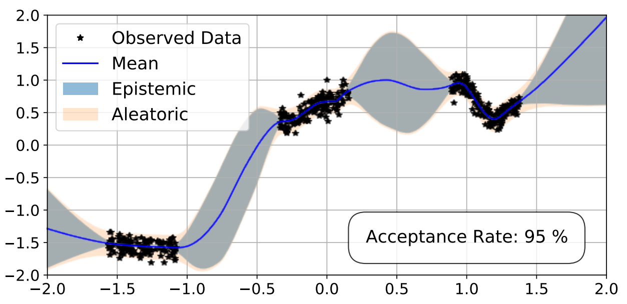


Figure 1: BNN regression example demonstrating the performance of symmetric HMC. A higher acceptance rate leads to better exploration and an increased epistemic uncertainty outside the range of the data.

Table 2: The BNN regression example calculated over 10 HMC chains. The Mean ESS was calculated by taking an average over the network's parameters ($\boldsymbol{\omega} \in \mathbb{R}^{10401}$). The acceptance rate is reported with its standard deviations. A higher mean ESS and a higher acceptance rate, demonstrate the better mixing performance from symmetric split HMC.

Inference Scheme	Acc. Rate	Mean ESS	Mean ESS / sec
Full HMC	0.71 ± 0.07	9.27	**0.0584**
Naive Split HMC	0.68 ± 0.06	9.13	0.0082
Rand. Split HMC	0.75 ± 0.06	9.78	0.0087
Sym. Split HMC	$\mathbf{0.89 \pm 0.05}$	**10.01**	0.0089

5.3 BNN CLASSIFICATION EXAMPLE

We offer a further example to compare all four approaches where the difficulty of the task requires a larger model with two convolutional layers followed by two fully connected layers. This model has 38,390 parameters. Our classification example uses the Fashion MNIST (FMNIST) data set [Xiao et al., 2017], which we divide into a training set of 48,000 images and a validation set of 12,000 images. For the split HMC approaches, the training set is further split into three subsets of 16,000. As for both the regression example and the logistic regression example, all hyperparameters are set to the same values ($L = 30$, $\epsilon = 2e^{-5}$, $\mathbf{M} = 0.01\mathbf{I}$, and $p(\boldsymbol{\omega}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$).

The results of this experiment can be seen in Table 3, where symmetric split HMC achieves both a higher acceptance rate and higher mean effective sample size. This result is consistent with the previous examples. In addition, we also display the negative log-likelihood (NLL) and accuracy for each solution. Our splitting approach achieves improved accuracy and NLL compared to all other inference schemes. We also display the shortest wall-clock time of the all the chains for each method to indicate the cost in time.

In this example, we see the advantage of using a splitting approach for tackling larger data tasks. For our specific hardware configuration (CPU: Intel i7-9750H; GPU: GeForce RTX 2080 with Max-Q), the maximum GPU memory usage with full HMC (using 48,000 training images) is 7,928 MB out of the available 7,982 MB. As a result, by splitting the data into three subsets, it would be possible to extend the current training set to 144,000 training images without requiring a change in hardware. Therefore splitting makes it possible to perform HMC over larger data sets, without the need for relying on stochastic subsampling.

6 SCALING HMC TO VEHICLE CLASSIFICATION FROM ACOUSTIC SENSORS

We will now show that our novel symmetric splitting approach facilitates applications to real-world scenarios, where the size of the data prevents the use of classical HMC. In our real-world example, the objective of the task is to detect and classify vehicles from their acoustic microphone recordings.

6.1 THE DATA SET

The data consists of 223 audio recordings from the Acoustic-seismic Classification Identification Data Set (ACIDS). ACIDS was originally used by Hurd and Pham [2002] for harmonic feature extraction of ground vehicles for acoustic classification, identification, direction of arrival estimation and beamforming, but in this work we focus on acoustic classification. There are nine classes of vehicles, where each vehicle is recorded via a triangular array of three microphones.[6]

In order to take advantage of the data structure from the three microphone sources, we transform each full time-series recording into the frequency domain using a short time Fourier transform (STFT), using the Scikit-learn default settings of `scipy.signal.spectrogram` [Pedregosa et al., 2011]. We randomly shuffle the recordings into eight cross-validation splits, where one is kept for hyperparameter optimisation. Once the audio recordings are divided, they are split into smaller ($\approx$ 10 s) chunks. We then work with the log power spectral density and build our training data by concatenating corresponding time chunks from all three microphones together into one spectrogram (e.g. see Figure 1 in Appendix A.1). Finally, the data is normalised using the mean and standard deviation of the log amplitude across the entire training data for each cross-validation split.

6.2 BASELINES

We compare symmetric split HMC with Stochastic Gradient Descent (SGD), SGLD and SGHMC. All inference scheme

[6]Audio was recorded at a sampling rate of 1025.641 Hz.

Table 3: The BNN classification example calculated over 10 HMC chains. The ESS was calculated using Pyro's in-built function [Bingham et al., 2019], followed by taking an average over the network's parameters ($\boldsymbol{\omega} \in \mathbb{R}^{38390}$). The acceptance rate, NLL, and accuracy are reported with their standard deviations. The minimum time taken for a chain is also reported. A higher mean ESS and a higher acceptance rate, demonstrate the better mixing performance of symmetric split HMC. We also see that our symmetric splitting approach results in improved NLL and accuracy.

Inference Scheme	Acc. Rate	Mean ESS	Accuracy	NLL	Min. Wall-Clock Time
Full HMC	0.76 ± 0.06	6.26	89.82 ± 0.22	0.282 ± 0.007	**1 hour, 18 minutes**
Naive Split HMC	0.72 ± 0.11	6.21	89.76 ± 0.15	0.283 ± 0.005	3 hours, 13 minutes
Randomised Split HMC	0.66 ± 0.06	6.24	89.83 ± 0.39	0.282 ± 0.009	3 hours, 12 minutes
Symmetric Split HMC	$\mathbf{0.89 \pm 0.02}$	$\mathbf{6.37}$	$\mathbf{89.97 \pm 0.27}$	$\mathbf{0.276 \pm 0.009}$	2 hours 58 minutes

Table 4: Vehicle classification results from acoustic data. Our symmetric split inference scheme outperforms in accuracy, NLL, and Brier score. The standard deviations are over seven randomised train-test splits.

Method	Accuracy	NLL	Brier Score
SGD	80.3 ± 3.1	0.72 ± 0.15	0.297 ± 0.052
SGLD	78.6 ± 3.3	0.69 ± 0.10	0.307 ± 0.043
SGHMC	82.6 ± 3.1	0.59 ± 0.11	0.252 ± 0.042
NSS HMC	$\mathbf{84.4 \pm 2.1}$	$\mathbf{0.51 \pm 0.05}$	$\mathbf{0.228 \pm 0.027}$

hyperparameters are optimised via Bayesian optimisation using BoTorch [Balandat et al., 2019], adapted from the URSABench tool [Vadera et al., 2020].

We use a neural network model that consists of four convolutional layers with max-pooling, followed by a fully-connected last layer. Importantly, we use Scaled Exponential Linear Units (SELUs) as the activation function [Klambauer et al., 2017], which we find yields an improvement over commonly-used alternatives such as rectified linear units. This is also seen by Heek and Kalchbrenner [2019] for their stochastic gradient MCMC approach.

6.3 CLASSIFICATION RESULTS

Table 4 displays the results of the experiment. We compare the four inference approaches and report their accuracy, Negative Log-Likelihood (NLL), and Brier score [Brier, 1950], the last of which can be used to measure calibration performance. In our experimental set-up, we randomly allocate the data into seven train-test splits and provide mean and standard deviations in Table 4. We note that a different initial split was used for hyperparameter optimisation. The result is that symmetric split HMC achieves an overall better performance compared to the stochastic gradient approaches. This demonstrates that one can perform HMC without using a stochastic gradient approximation on a single GPU and still achieve better accuracy and calibration.[7]

[7]We note that for our hardware, it was only possible to run full HMC on our GPU with 53 % of the training data.

6.4 UNCERTAINTY QUANTIFICATION

In addition to reporting the results in Table 4, we analyse the behaviour of the uncertainty across all cross-validation splits. We will focus on two ways to analyse the quality of these results. First, we will focus on the predictive entropy as the proxy for uncertainty because this is directly related to the softmax outputs and is therefore the most likely to be used in practice. The posterior predictive entropy for each test datum $\mathbf{x}^*$ is given by the entropy of the expectation over the predictive distribution with respect to the posterior, $\mathcal{H}[\mathbb{E}_{\boldsymbol{\omega}}[p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})]]$, which we will refer to via $\tilde{\mathcal{H}}$.

We can then plot the empirical cumulative distribution function (CDF) of all erroneous predictions across all cross-validation splits, as shown in Figure 2. It is desirable for a model to make predictions with high $\tilde{\mathcal{H}}$, when the predictions are wrong, which is the case for the misclassified data in Figure 2. Curves that follow this desirable behaviour remain close to the bottom right corner of the graph. Our new approach of symmetric split HMC behaves closer to the ideal behaviour in comparison to the baselines. This improved behaviour can be seen from the purple curve, which falls closer to the x-axis than the other curves.

The second way that we will assess uncertainty is by relying on the mutual information (MI) between the predictions and model posterior. The MI can help distinguish between data uncertainty and model uncertainty, whereby our interest lies in the model uncertainty. Data points with high MI indicate that the model is uncertain due to the disagreement between the samples (this is in comparison to a model that is confident in its uncertainty, which would result in low MI). In the literature, the use of MI for uncertainty quantification can be seen in works by Houlsby et al. [2011] and Gal et al. [2017] using Bayesian Active Learning by Disagreement (BALD) and via knowledge uncertainty in works by Depeweg et al. [2017] and Malinin et al. [2020].

To analyse MI, in Figure 3, we display "confusion-style" matrices for the top performing inference schemes according to Table 4, SGHMC and symmetric split HMC. Each square in the matrix contains the average MI over all the data corresponding to that square across all the cross-validation

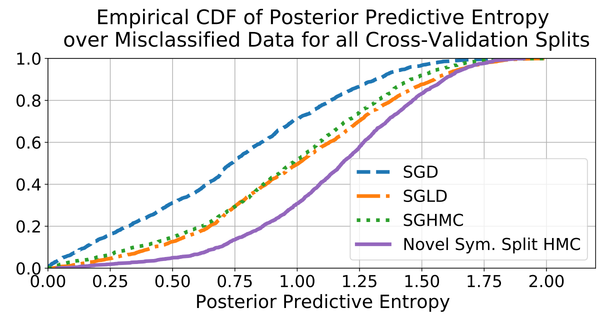
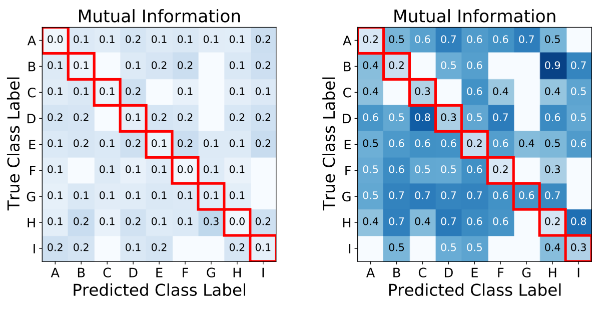


Figure 2: Cumulative posterior predictive entropy of misclassified data points. This plot shows that symmetric split HMC makes fewer high confidence errors than the other competing approaches. This is shown by the purple curve falling closer to the x-axis than the other curves. Curves which fall close to the x-axis mean that the probability of a low-entropy prediction is lower.



(a) SGHMC (b) Sym. split HMC

Figure 3: "Confusion-style" matrix showing average mutual information per category. Each square corresponds to the MI averaged over the number of test data corresponding to that box (boxes containing no data are blank). The diagonals (highlighted in red) indicate average MI over correct classifications, where low values are desirable. The off-diagonals indicate the average MI for erroneous predictions, where high values are desirable. (a) The matrix for SGHMC shows low MI everywhere, which is especially noticeable over the misclassifications. (b) Symmetric split HMC is more uncertain over its erroneous predictions and the difference between diagonals and off-diagonals is more obvious.

splits. Low values along the diagonal are desirable because they correspond to confident predictions for correct classifications. However, low values on the off-diagonals are especially undesirable as they correspond to errors that were predicted with high confidence. When we compare SGHMC of Figure 3a to symmetric split HMC of Figure 3b, we see the advantages of our approach. The values on the off-diagonals for SGHMC indicate that the model is overly confident when making errors. Furthermore, there is a large level of overlap between the diagonal and off-diagonal values for the MI. This overlap makes it hard to use model uncertainty to distinguish the errors from the correct predictions. In comparison, our symmetric split approach shows little overlap between the off-diagonal values and the diagonals. This near-separability can help to distinguish erroneous predictions by their high uncertainty.

7 DISCUSSION

There are many challenges associated with performing HMC over large hierarchical models such as BNNs. Our work makes strides in the right direction but there are further areas to explore. As alluded to in Section 2, there are techniques that can be employed to improve hyperparameter optimisation. For example, in this paper we have assumed the mass matrix to be diagonal with one scaling factor, which may not be an optimal choice. Future work to design mass matrices, such as metrics derived by Girolami and Calderhead [2011] or Hoffman et al. [2019], may further improve the current method. Another challenge with MCMC approaches is knowing when enough samples have been collected such that the samples provide a good representation of the target distribution. In high-dimensional models like neural networks, chains may take a long time to converge and it is important to build reliable metrics for convergence such as observing the effective sample size, plotting the log-posterior density of the samples, and plotting the cumulative accuracy (e.g. see Appendix A.4).

8 CONCLUSION

In this work we have shown the advantage of preserving the entire Hamiltonian for performing inference in Bayesian neural networks. In Section 5 we provided two classification tasks and one regression task. We showed symmetric split HMC is better suited to inference in BNNs compared to previous splitting approaches. These previous approaches did not have the same efficiencies as our novel symmetric split integration scheme. We then provided a real-world application in Section 6, where we compared symmetric split HMC with two stochastic gradient MCMC approaches. For this acoustic classification example, we were able to show that our new method outperformed stochastic gradient MCMC, both in classification accuracy and in uncertainty quantification. In particular, the analysis of the uncertainty quantification showed symmetric split HMC achieved a lower confidence for its misclassified labels, whilst also achieving a better overall accuracy. In conclusion, we have introduced a new splitting approach that is easy to implement on a single GPU. Our approach is better than previous splitting schemes and we have shown it is capable of outperforming stochastic gradient MCMC techniques.

Acknowledgements

We would like to thank Tien Pham for making the data available and Ivan Kiskin for his great feedback. ACIDS (Acoustic-seismic Classification Identification Data Set) is an ideal data set for developing and training acoustic classification and identification algorithms. ACIDS along with other data sets can be obtained online through the Automated Online Data Repository (AODR) [Bennett et al., 2018]. Research reported in this paper was sponsored in part by the CCDC Army Research Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. *arXiv preprint arXiv:2002.06470*, 2020.

Maximilian Balandat, Brian Karrer, Daniel R Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: Programmable Bayesian Optimization in Pytorch. *arXiv preprint arXiv:1910.06403*, 2019.

Rémi Bardenet, Arnaud Doucet, and Chris Holmes. Towards scaling up markov chain monte carlo: an adaptive subsampling approach. 2014.

Kelly W Bennett, Dennis W Ward, and James Robertson. Cloud-based security architecture supporting army research laboratory's collaborative research environments. In *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IX*, volume 10635, page 106350G. International Society for Optics and Photonics, 2018.

Michael Betancourt. The fundamental incompatibility of scalable Hamiltonian Monte Carlo and naive data subsampling. In *International Conference on Machine Learning*, pages 533–540, 2015.

Michael J Betancourt. Generalizing the no-U-turn sampler to Riemannian manifolds. *arXiv preprint arXiv:1304.1920*, 2013.

Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep universal probabilistic programming. *J. Mach. Learn. Res.*, 20:28:1–28:6, 2019. URL `http://jmlr.org/papers/v20/18-403.html`.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 1613–1622. JMLR. org, 2015.

Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*, pages 1683–1691, 2014.

Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty for active learning and reliable reinforcement learning in stochastic systems. *ArXiv*, abs/1710.07283, 2017.

Nan Ding, Youhan Fang, Ryan Babbush, Changyou Chen, Robert D Skeel, and Hartmut Neven. Bayesian sampling using stochastic gradient thermostats. In *Advances in neural information processing systems*, pages 3203–3211, 2014.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

Angelos Filos, Panagiotis Tigas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? *arXiv preprint arXiv:2006.14911*, 2020.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192, 2017.

Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. GPytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, pages 7576–7586, 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(2):123–214, 2011.

Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schon. Evaluating scalable Bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 318–319, 2020.

Jonathan Heek and Nal Kalchbrenner. Bayesian inference for large scale image classification. *arXiv preprint arXiv:1908.03491*, 2019.

Matthew Hoffman, Pavel Sountsov, Joshua V Dillon, Ian Langmore, Dustin Tran, and Srinivas Vasudevan. Neutralizing bad geometry in Hamiltonian Monte Carlo using neural transport. *arXiv preprint arXiv:1903.03704*, 2019.

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 (1):1593–1623, 2014.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Harry Hurd and Tien Pham. Target association using harmonic frequency tracks. In *Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002.(IEEE Cat. No. 02EX5997)*, volume 2, pages 860–864. IEEE, 2002.

Pavel Izmailov, Wesley J. Maddox, Polina Kirichenko, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Subspace inference for bayesian deep learning. In *UAI*, 2019.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports*, 7(1):1–14, 2017.

Benedict Leimkuhler and Sebastian Reich. Simulating hamiltonian dynamics. 14, 2004.

Benedict Leimkuhler and Xiaocheng Shang. Adaptive Thermostats for Noisy Gradient Systems. *arXiv preprint arXiv: 1505.06889v2*, 2016.

Chris Xiaoxuan Lu, Stefano Rosa, Peijun Zhao, Bing Wang, Changhao Chen, John A Stankovic, Niki Trigoni, and Andrew Markham. See through smoke: robust indoor mapping with low-cost mmwave radar. In *MobiSys*, pages 14–27, 2020.

Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. Ensemble distribution distillation. In *ICLR*, 2020.

Youssef Marzouk, Tarek Moselhy, Matthew Parno, and Alessio Spantini. An introduction to sampling via measure transport. *arXiv preprint arXiv:1602.05023*, 2016.

Radford M Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

Radford M Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov chain Monte Carlo*, 2(11):2, 2011.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12 (85):2825–2830, 2011. URL `http://jmlr.org/papers/v12/pedregosa11a.html`.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

JC Sexton and DH Weingarten. Hamiltonian evolution for the hybrid monte carlo algorithm. *Nuclear Physics B*, 380(3):665–677, 1992.

Babak Shahbaba, Shiwei Lan, Wesley O Johnson, and Radford M Neal. Split Hamiltonian Monte Carlo. *Statistics and Computing*, 24(3):339–349, 2014.

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In *Advances in neural information processing systems*, pages 4134–4142, 2016.

Gilbert Strang. On the construction and comparison of difference schemes. *SIAM journal on numerical analysis*, 5(3):506–517, 1968.

Meet P Vadera, Adam D Cobb, Brian Jalaian, and Benjamin M Marlin. Ursabench: Comprehensive benchmarking of approximate bayesian inference methods for deep neural networks. *arXiv preprint arXiv:2007.04466*, 2020.

Ziyu Wang, Shakir Mohamed, and Nando Freitas. Adaptive Hamiltonian and Riemann Manifold Monte Carlo. In *International Conference on Machine Learning*, pages 1462–1470, 2013.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *ArXiv*, abs/1708.07747, 2017.

Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for bayesian deep learning. *International Conference on Learning Representations*, 2020.

Yichuan Zhang and Charles Sutton. Semi-separable Hamiltonian Monte Carlo for inference in Bayesian hierarchical models. In *Advances in Neural Information Processing Systems*, pages 10–18, 2014.