Bayesian Neural Ordinary Differential Equations

Raj Dandekar
Massachusetts Institute of Technology
rajd@mit.edu

Karen Chung
Massachusetts Institute of Technology

Vaibhav Dixit
Julia Computing Inc.,
Pumas AI

Mohamed Tarek
University of New South Wales at Canberra, Australia,
Pumas AI

Aslan García-Valadez
National Autonomous University of Mexico.

Krishna Vishal Vemula
Pramati Technologies

Chris Rackauckas
Massachusetts Institute of Technology,
University of Maryland Baltimore,
Pumas AI
crackauc@mit.edu

Abstract

Recently, Neural Ordinary Differential Equations has emerged as a powerful framework for modeling physical simulations without explicitly defining the ODEs governing the system, but instead learning them via machine learning. However, the question: "Can Bayesian learning frameworks be integrated with Neural ODE's to robustly quantify the uncertainty in the weights of a Neural ODE?" remains unanswered. In an effort to address this question, we primarily evaluate the following categories of inference methods: (a) The No-U-Turn MCMC sampler (NUTS), (b) Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) and (c) Stochastic Langevin Gradient Descent (SGLD). We demonstrate the successful integration of Neural ODEs with the above Bayesian inference frameworks on classical physical systems, as well as on standard machine learning datasets like MNIST, using GPU acceleration. On the MNIST dataset, we achieve a posterior sample accuracy of 98.5% on the test ensemble of 10,000 images. Subsequently, for the first time, we demonstrate the successful integration of variational inference with normalizing flows and Neural ODEs, leading to a powerful Bayesian Neural ODE object. Finally, considering a predator-prey model and an epidemiological system, we demonstrate the probabilistic identification of model specification in partially-described dynamical systems using universal ordinary differential equations. Together, this gives a scientific machine learning tool for probabilistic estimation of epistemic uncertainties.

1 Introduction

The underlying scientific laws describing the physical world around us are often prescribed in terms of ordinary differential equations (ODEs). Recently, Neural Ordinary Differential Equations [5] has emerged as a powerful framework for modeling physical simulations without explicitly defining the ODEs governing the system, but learning them via machine learning. By noticing that in the limit of infinite layers, a ResNet module [12] behaves as a continuous time ODE, Neural ODEs allow the

coupling of neural networks as expressive function transformations, and powerful purpose built ODE solvers. While [5] explored a number of applications of the Neural ODE framework, their success in a Bayesian inference framework remains unexplored.

Simultaneously, there has been an emergence of efficient Bayesian inference methods suited for high-dimensional parameter systems, such as the No-U-Turn MCMC sampler (NUTS) [14] which is an extension of the Hamiltonian Monte Carlo Algorithm, and Stochastic Gradient Markov Chain Monte Carlo (SGMCMC) methods like Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) [6] and Stochastic Gradient Langevin Descent (SGLD) [34].

A number of works in literature explored the use of Bayesian methods to infer parameters of systems defined by ODEs [21, 32, 10, 15] and others used Bayesian methods to infer parameters of neural network models, e.g. [18, 22, 16]. Bayesian neural networks in particular has been an active area of research for a while. The readers are referred to the excellent recent tutorial by [18] for an overview of recent advances in the field. However, this prompts the question: "Can Bayesian learning frameworks be integrated with Neural ODE's to robustly quantify the uncertainty in the weights of a Neural ODE?"

In an effort to address this question, we demonstrate and compare the integration of Neural ODEs with the following methods of Bayesian Inference: (a) The No-U-Turn MCMC sampler (NUTS), (b) Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) and (c) Stochastic Langevin Gradient Descent (SGLD). We present successful results on classical physical systems and on standard machine learning datasets (using GPU acceleration); especially on the standard MNIST dataset, we achieve a test ensemble accuracy of 98.5% on 10000 images. This is a performance competitive with current state-of-the-art image classification methods, which meanwhile lack our method's ability to quantify the confidence in its predictions. Subsequently, we premiere the integration of Bayesian Neural ODEs with variational inference, the predictive power of which improves with the introduction of normalizing flow.

Finally, advancing from learning a physical system's differential equations via Bayesian Neural ODEs, we consider the problem of recovering missing terms from a dynamical system using universal differential equations (UDEs) [25]. Using the Preconditioned SGLD variation of SGLD, we demonstrate the predictive success of Bayesian UDEs on (a) a predator-prey model and (b) the epidemiological model of COVID-19 spread. Through this, we present a viable method for the probabilistic quantification of epistemic uncertainties via a hybrid machine-learning and mechanistic-model-based technique.

Our approach differs from that of [1] who mainly looked at integration of Bayesian methods with Neural SDE's, and not Neural ODEs describing physical systems or large scale deep learning datasets like the MNIST dataset, which we consider here.

In this study, we used the Julia differentiable programming stack [24] to compose the Julia differential equation solvers [26] with the Turing probabilistic programming language [8, 35]. The study was performed without modifications to the underlying libraries due to the composability afforded by the differentiable programming stack.

2 Results

We illustrate the robustness of the Bayesian Neural ODE framework through the following case studies:

Case study 1: Spiral ODE

The Spiral ODE model is prescribed by the following system of equations:

$$\frac{du_1}{dt} = -\alpha u_1^3 + \beta u_2^3 \tag{1}$$

$$\frac{du_2}{dt} = -\beta u_1^3 - \alpha u_2^3 \tag{2}$$

Case study 2: Lotka-Volterra ODE

The Lotka-Volterra predator-prey model is prescribed by the following system of equations:

$$\frac{du_1}{dt} = -\alpha u_1 - \beta u_1 u_2 \tag{3}$$

$$\frac{du_2}{dt} = -\delta u_2 + \gamma u_1 u_2 \tag{4}$$

2.1 Bayesian Neural ODE: NUTS Sampler

The No-U-Turn-Sampler (NUTS) is an extension of the Hamiltonian Monte Carlo (HMC) algorithm. Through a recursive algorithm, NUTS automatically determines when the sampler should stop an iteration, and thus prevents the need to specify user defined parameters, like the number of steps L. In addition, through a dual averaging algorithm, NUTS adapts the step size ϵ throughout the sampling process.
We define the parameters of the d dimensional Neural ODE by θ. The action of the Neural ODE on an input value u_0 generates an output $\tilde{Y} = \text{NNODE}_\theta(u_0)$. The input data is denoted by $\hat{Y}$. The loss function, L is defined as

$$L(\theta) = \sum_{i=1}^{d} ||\hat{Y}_i - \tilde{Y}_i||^2 \tag{5}$$

The model variables θ and the momentum variables r are drawn from the joint distribution

$$p(\theta, r) \propto \exp[\mathcal{L}(\theta) - \frac{1}{2} r.r] \tag{6}$$

where $\mathcal{L}$ is the logarithm of the joint density of θ. In terms of a physical analogy, if θ denotes a particle's position, then $\mathcal{L}$ can also be viewed as the negative of the potential energy function and $\frac{1}{2} r.r$ denotes the kinetic energy of the particle.

In the Bayesian Neural ODE framework, we define $\mathcal{L}(\theta)$ as

$$\mathcal{L}(\theta) = -\sum_{i=1}^{d} ||\hat{Y}_i - \tilde{Y}_i||^2 - \theta.\theta \tag{7}$$

The $\theta.\theta$ term indicates the use of Gaussian priors. We adapt the step size of the leapfrog integrator using Nesterov's dual averaging algorithm [14] with δ as the target acceptance rate. Finally, we define the number of warmup samples as n_w and the number of posterior samples collected as n_p.

We apply the Bayesian Neural ODE framework outlined above to case studies 1 and 2 given in Equations 1-4. For case study 1, we use $\alpha = 0.1, \beta = 2$. In the NUTS algorithm, we use $\delta = 0.45, n_w = 1000, n_s = 500$. For case study 2, we use $\alpha = 1.5, \beta = 1, \gamma = 3, \delta = 1$; with $\delta = 0.45, n_w = 500, n_s = 1000$. 2 layers with 50 units in each layer and tanh activation function was used as the neural ODE architecture for both examples.

From figure 1, we can see that the Bayesian Neural ODE: NUTS prediction and forecasting for both case studies outlined in Equations 1-4 are consistent with the ground truth data.

Figure 2a showing the posterior density and trace plots for the first 5 parameters of the Spiral ODE example, shows that the samples are well mixed. The quick decay seen in the auto-corelation plot shown in figure 2b also indicates a fast mixing Markov chain. This is also confirmed by the effective sample size extracted for the posterior chain of 500 samples, which shows values of $362, 470, 134, 509, 661$ for the first 5 parameters. Similar well mixed plots are seen for all parameters, but are not shown here for the sake of brevity.

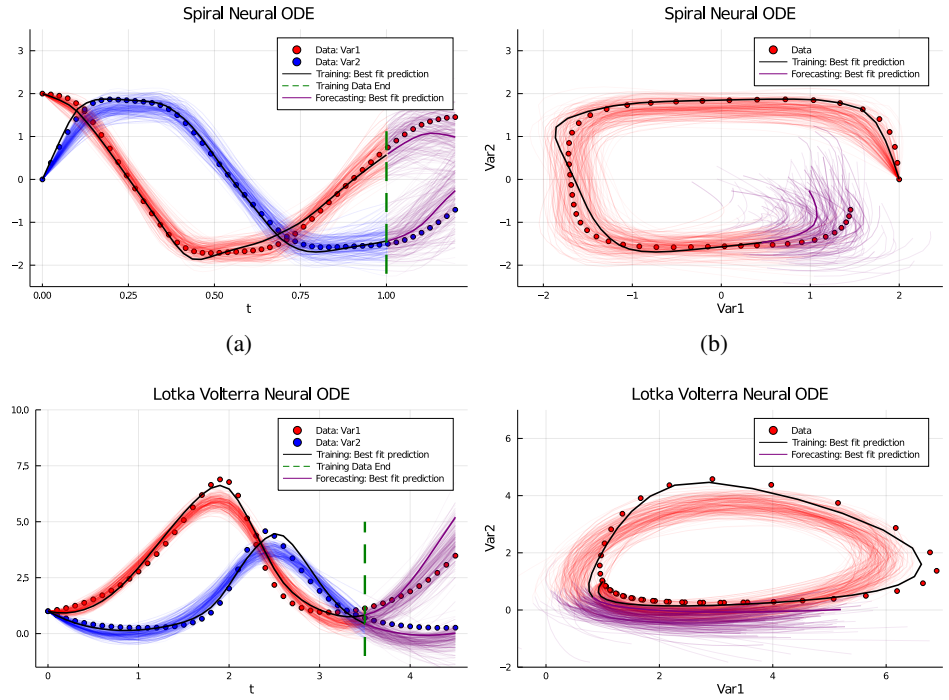


Figure 1: Comparison of the Bayesian Neural ODE: NUTS prediction and estimation compared with data for (a,b) Case study 1 and (c,d) Case study 2.

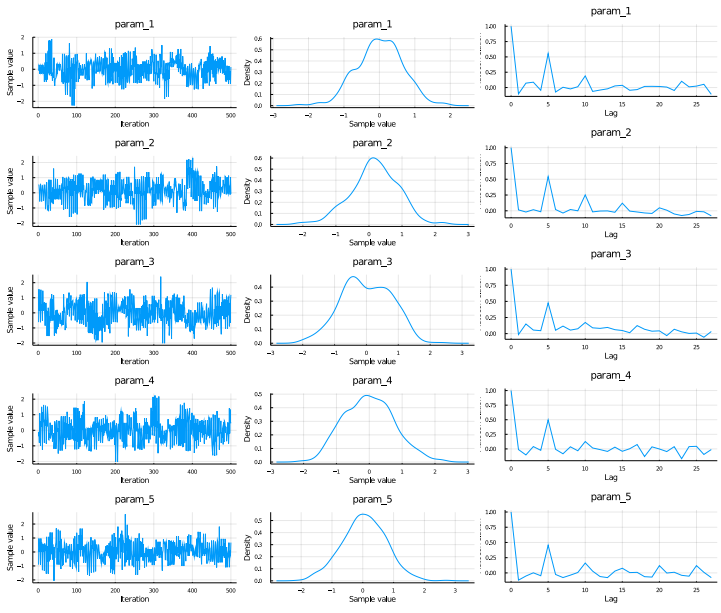


Figure 2: For the Spiral ODE example (Equations 1-2), using the NUTS framework, figure shows: (a) Trace plots and Density plots of the posterior and (b) auto-corelation plot for the first 5 parameters.

Table 1: Spiral ODE: Effect of NUTS acceptance ratio and Neural ODE architecture. Number of warmup samples, $n_w = 500$ and number of posterior samples, $n_s = 500$ for all cases shown. The minimum loss value obtain is similar in all cases shown.

δ	Units	Layers	Time (s)
0.45	5	2	480
0.45	10	2	900
0.45	50	2	2100
0.45	100	2	3900
0.45	10	3	3300
0.45	10	4	10200
0.65	50	2	3400
0.85	50	2	7900
0.95	50	2	8600

Table 1 shows the effect of NUTS acceptance ratio and Neural ODE architecture on the Bayesian Neural ODE performance for the Spiral ODE example. The minimum loss value obtain is similar in all cases shown. Thus, we see that even the smallest neural architecture with 2 layers and 5 units in each layer gives the optimal loss performance, and with a considerably better timing performance. Among different NUTS acceptance ratios (δ) tested, the best timing performance is given by the lowest acceptance ratio, $\delta = 0.45$.

2.2 Bayesian Neural ODE: SGHMC

The Stochastic Gradient Hamiltonian Monte Carlo Sampler (SGHMC) is a method that combines HMC's effective state space exploration with stochastic gradient methods' computational efficiencies [6]. SGHMC injects friction to the "momentum" auxiliary variables that parameterize the target distribution's Hamiltonian dynamics.

The Hamiltonian function $H(\theta, r) = U(\theta) + \frac{1}{2} r^T M^{-1} r$ measures the total "energy" of a system with position variables θ and momentum variables r. The potential energy function is given by $U = -\sum_{x \in \mathcal{D}} \log p(x|\theta) - \log p(\theta)$; mass matrix M and r define the kinetic energy term.

To sample from the posterior distribution $p(\theta|\mathcal{D})$, HMC considers generating samples from the joint distribution $\pi(\theta, r) \propto \exp(H(\theta, r))$, proposing samples according to the Hamiltonian dynamics:

$$d\theta = M^{-1} r dt \tag{8}$$

$$dr = -\nabla U(\theta) dt \tag{9}$$

Here, SGHMC adds a "friction" term to the momentum update. In practice, we consider $\nabla \tilde{U}$, a noisy estimate of ∇U. Similarly, $\hat{B}$ is defined as an estimate of $B(\theta) = \frac{1}{2}\epsilon V(\theta)$, the diffusion matrix contributed by the covariance of the stochastic gradient noise $V(\theta)$. SGHMC additionally introduces a user-specified friction term $C \succeq \hat{B}$. In total, by defining $v = \epsilon M^{-1} r$, $\eta = \epsilon^2 M^{-1}$, $\alpha = \epsilon M^{-1} C$, and $\hat{\beta} = \epsilon M^{-1} \hat{B}$ for stepsize ϵ, SGHMC iteratively updates the variables for sampling according to:

$$\theta_t := \theta_t + \Delta\theta_t \tag{10}$$

$$v_t := v_t + \Delta v_t \tag{11}$$

$$\Delta\theta_t := v \tag{12}$$

$$\Delta v_t := -\eta \nabla \tilde{U}(x) - \alpha v + \mathcal{N}(0, 2(\alpha - \hat{\beta})\eta) \tag{13}$$

Naturally, η corresponds to the learning rate and α the momentum decay.

We apply the Bayesian Neural ODE framework outlined above to case studies 1 and 2 given in Equations 1-4. For case study 1, we use α = 1, β = 1. In the SGHMC algorithm, we use $\eta = 1.5^{-6}$ and α = 0.07 and draw 2500 posterior samples. We define a prior distribution centered at the MAP point with a standard deviation of 0.4. For case study 2, we use α = 1.5, β = 1.0, γ = 3.0, and δ = 1.0; we draw 350 samples with SGHMC using hyperparameters $\eta = 7.0^{-6}$ and $\alpha = 0.07$. We define a prior distribution centered at the MAP point with a standard deviation of 1.0. For both case studies,

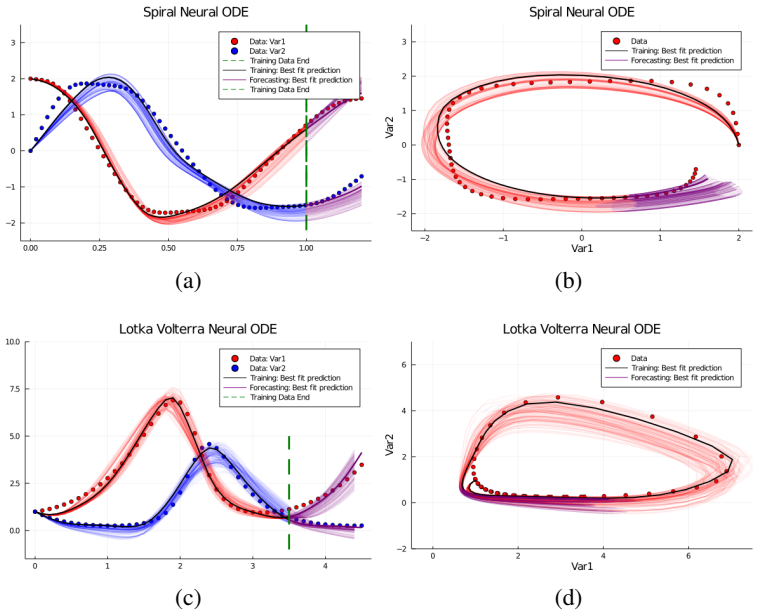
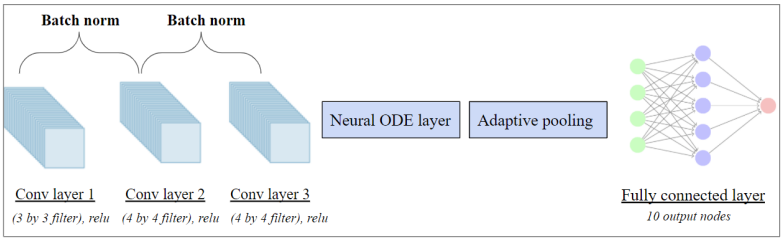


Figure 3: Comparison of Bayesian Neural ODE SGHMC's prediction and forecasting against ground truth data for (a,b) case study 1 and (c,d) case study 2.

the neural ODE architecture consists of 2 layers with tanh activation function; there are 50 units in each layer for case study 1, and 10 units for case study 2.

Figure 3 illustrates the consistency of Bayesian Neural ODE SGHMC's prediction and forecasting with ground truth data, for both case studies (Equations 1-4).

2.2.1 SGHMC on the MNIST dataset



(a)

Figure 4: Neural network architecture used for the image classification task on MNIST. The Neural ODE contains two convolutional layers. The network has 208010 parameters in total. This architecture was combined with the SGHMC method to lead to a Bayesian Neural ODE object which can be used for image classification.

We now apply Bayesian Neural ODE with SGHMC to a image classification task on the MNIST dataset. A ResNet [11] layer behaves as a continuous time ODE at the limit of infinite layers. Given this natural analogy, we here implement ODE layers in place of the residual layers used in classic image recognition architectures.

Specifically, we design three convolutional layers interspersed with Batch Normalization layers: the initial layer has a 3×3 filter and the next two convolutional layers each have a 4×4 filter (stride 2×2, padding 1×1, ReLU activations). A Neural ODE layer with two convolutional layers with 3×3 filter (padding 0, ReLU activations) is appended to act as a residual layer. Finally, we add an

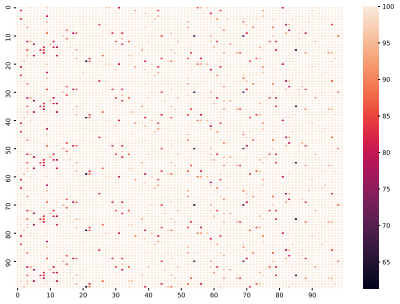
Table 2: Performance on MNIST using the Bayesian Neural ODE: SGHMC approach is outlined in the present study. Here, 310 posterior samples for each image in a test set of 10,000 images is considered. The best fit test error represents the mean of the number of erroneous predictions in all the posterior samples (310) for all images (10000).

	Error estimates?	Neural ODE?	Best fit test error	Reference test error
RK-Net	✗	✓	0.47 %	Chen (2018)
ODE-Net	✗	✓	0.42 %	Chen (2018)
Bayesian Alex-Net	✓	✗	1 %	Shridhar (2019)
Bayesian LeNet-5	✓	✗	2 %	Shridhar (2019)
Bayesian Neural ODE (ensemble)	✓	✓	0.78 %	Our study

adapative average pooling layer and a fully connected layer consisting of 10 neurons (one per class). The full architecture is visualized in figure 4.

Given the Neural ODE architecture, we initialize SGHMC with decay schedule of $\epsilon_t = \eta.t^{-\gamma}$ and parameters $\gamma = 0.01$, $\eta = 0.5$ and $\alpha = 0.1$, execute for 2530 iterations, and sample the last 310 parameter updates. We find that tempering—scaling the standard deviation of the added noise in SGHMC by a constant [20]—significantly reduces time for convergence. Our experiment uses 10^4 as the tempering constant. SGHMC is more computationally expensive, requiring ten epochs through the training dataset, than a simple MAP estimation; a trial MAP optimization with ADAM yields a 98.7% test accuracy after a single-epoch run.

The test set consists of all 10,000 images in the MNIST dataset. Each cell in the heatmap of figure 5 represents the percentage of correct predictions out of 310 posterior samples on a single image. 91.8% of these cells have more than 99% confidence.

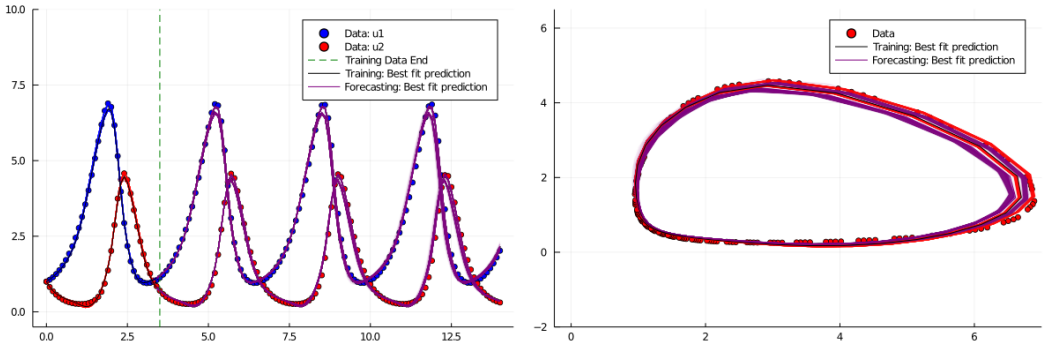


(a)

Figure 5: Bayesian Neural ODE with SGHMC is applied to the MNIST dataset. Each cell in this figure represents the percentage of correct predictions out of 310 posterior samples on a single image. Results for the entire test set of $10,000$ images is visualized here as a 100×100 heatmap

Table 2 shows the performance of our Bayesian approach on the MNIST data. Out of the 310 posterior samples for each image in the test set of 10,000 images considered, the best fit test error represents the mean of the number of erroneous predictions in all samples for all images. In our study, we have obtained a test ensemble accuracy of 99.22 %, which is performance competitive with current state-of-the-art image classification methods.

From table 2, we note that previous architectures for MNIST analysis either have a Neural ODE architecture without error estimates [5] or do not incorporate a Neural ODE for error estimation [30]. Incorporated Neural ODEs in our approach, we not only demonstrate a classification performance competitive with current state-of-the-art image classification methods; but also quantify the confidence of our prediction.



(a) (b)

Figure 6: Comparison of the Bayesian Neural ODE: SGLD estimation and data for the Lotka Volterra ODE case study shown as (a) time series plots and (b) contour plots.

2.3 Bayesian Neural ODE: SGLD

Stochastic Gradient Langevin Dynamics (SGLD) is an adaptation, designed to sample from the posterior as the iterations increase, of the usual stochastic gradient descent algorithm. In each iteration, we update our vector θ of parameters according to the rule

$$\theta_t := \theta_t - \Delta\theta_t \tag{14}$$

$$\Delta\theta_t := \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{1}{n}\sum_{i=1}^{n} \nabla \log p(\mathcal{D}_n|\theta)\right) + \eta_t \tag{15}$$

$$\eta_t \sim \mathcal{N}(0, \epsilon_t) \tag{16}$$

where $\mathcal{D}_n$ are the minibatches the training dataset $\mathcal{D}$ has been split into. $p(\mathcal{D}_n|\theta)$ is the likelihood, whose logarithm is equivalent to the loss function, and $p(\theta_t)$ is any priors, also known as regularisation terms, imposed onto the parameters θ. The stepsizes ϵ_t must follow a decaying scheme which satisfies the conditions [34]:

$$\sum_{t=1}^{\infty} \epsilon_t = \infty \quad \sum_{t=1}^{\infty} \epsilon_t^2 < \infty \tag{17}$$

in this article we have chose a polynomial decaying scheme $\epsilon_t = a(b+t)^{-\gamma}$ with a, b, and γ as tuneable hyperparameters.

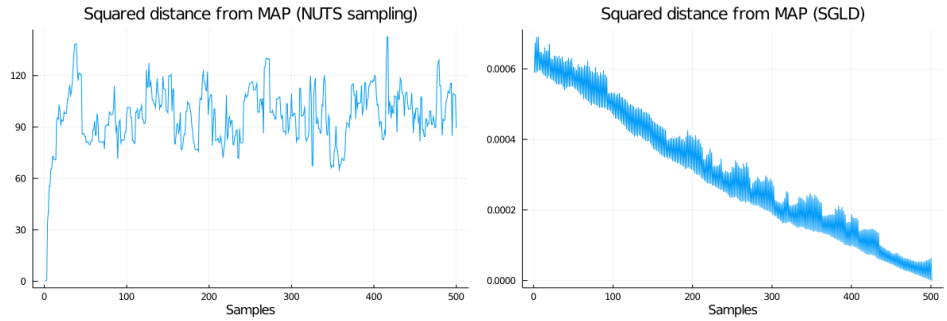
The update scheme for θ is composed of two stages. In the first stage, where the approximate gradient dominates, we approach the regions with higher mass probability. During the second phase, instead of allowing θ to converge to a single value, it walks randomly with a predominantly Gaussian noise since the gradient is $\mathcal{O}(\epsilon_t)$ and the Gaussian noise is $\mathcal{O}(\sqrt{\epsilon_t})$. It is on this second stage where it is theoretically guaranteed to converge to the posterior, and hence, we may use this stage to sample parameters from the posterior.

We now apply SGLD on the Lotka Volterra system in Equations 3-4. The Lotka Volterra system used in this case has the same parameters as the one used for NUTS. Again, we apply SGLD with 45000 iterations and sampled the last 2000 updates. The hyperparameters used were $a = 0.0025, b = 0.05, \gamma = 0.35$. The neural ODE architecture was again 2 layers with 50 neurons and tanh activation. The algorithm took approximately 679 seconds to run.

From figure 6 we notice a good fit on the training dataset. The Bayesian Neural ODE trained using the SGLD approach has accurately captured the periodicity of the system; and is seen to generalize for a much longer duration than the NUTS sampler.

2.3.1 Comparison between SGLD and NUTS

Through figures 1 and 6, we note that SGLD generally has a better mean prediction accuracy than NUTS. This can be attributed to the non-convexity/multi-modality of the likelihood function where



(a) (b)

Figure 7: (a) Figure shows that for the NUTS sample initialized at the MAP point, the sampler quickly jumps away from the MAP point and never returns back. (b) Figure shows that for the SGLD sampler initialized at the MAP, all posteriors samplers are close to the MAP point.

the MAP point is likely to be in a region with low probability mass, leading the NUTS sampler which uses non-stochastic gradients to not "find" the region surrounding the MAP point in the time of the sampling. The use of stochastic gradients in SGLD seems to have led to a sample much closer to the MAP point and with a much lower mean prediction error. This difference between NUTS and SGLD is illustrated in figure 7, where the distance of the posterior samples from the MAP point is shown. Figure 7a shows that for the NUTS sample initialized at the MAP point, the sampler quickly jumps away from the MAP point and never returns back. Figure 7b shows that for the SGLD sampler, all posteriors samplers are much closer to the MAP point, than the NUTS sampler.

2.4 Bayesian Neural ODE: Variational Inference

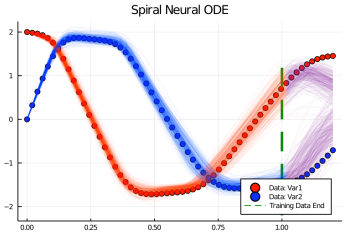
Since the last decade, there has been explosive interest in applying variational inference (VI) to text analysis, generative image modeling, and physical/chemical systems analysis [28]. VI, an optimization-based approach, generally approximates the posterior much faster than traditional MCMC methods. Variational inference was traditionally used for learning in graphical models (e.g. the sigmoid belief network) [17, 29]. However, there have been significant expansions in the methodology and application domains of VI; examples include stochastic variational inference [13] on large-scale data analysis and black-box variational inference's [27] usage beyond conditionally conjugate models. Through the use of normalizing flows [28], implicit distributions [23, 31] and importance-weighted variational autoencoders [7], the posterior-approximating variational inference family was made more expressive and thus powerful.

Here, we aim to address whether variational inference methods can be integrated into, and thus further expand, our Bayesian Neural ODEs framework. We use the Turing.jl interface in Julia [9]. Initially, we define a multivariate normal distribution as the posterior family approximating the true posterior with a mean-field approximation. For maximizing the expected lower bound, we use ADVI (Automatic Differentiation Variational Inference) [19], with 10 samples per step and 5000 as the upper bound on the number of gradient steps. Figure 8a shows the poor forecasting performance of the Bayesian Neural ODE: Variational Inference framework applied to the Spiral ODE example, compared to HMC methods explored above (figures 1, 3).

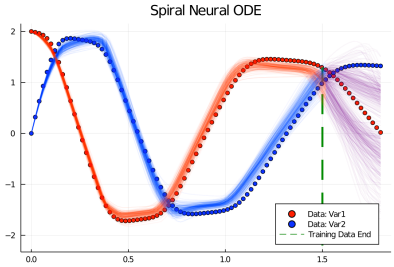
2.4.1 Integration of Normalizing Flows

One possible reason for the poor forecasting performance of the VI + Neural ODE framework could be due to the posterior family not being expressive enough. To further explore the effects of a powerful posterior family, we look at normalizing flows.

Through a chain of invertible mappings, normalizing flows transform an initially simple posterior family distribution (such as the multivariate normal in the above example) into an arbitrary complex distribution [28]. Considering an initial random variable z with a distribution prescribed by $q(z)$.



(a)



(b)

Figure 8: For the Spiral ODE example (Equations 1-2), figure shows the retrodiction plots for (a) Variational Inference framework used in the present study and (b) Variational Inference integrated with Normalizing Flow. We can see that integration with normalizing flows used shows marginal improvement over plain Variational Inference with mean field approximation.

A bijective mapping function f with inverse g, when applied to z results in a new variable z' with distribution given by

$$q'(z') = q(z)|\det\frac{\partial g}{\partial z'}| = q(z)|\det\frac{\partial f}{\partial z}|^{-1} \tag{18}$$

In their original work, [28] demonstrated two types of normalizing flows: planar layer and radial layer. In this study, we have employed the planar layer. The planar layer, parametrized by u, w, b is given by the invertible function

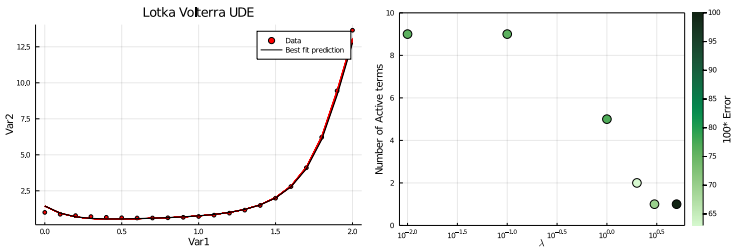
$$f(z) = z + uh(w^T z + b) \tag{19}$$

where h is a differentiable element-wise non-linearity. Thus, applying a sequence of maps f_k to an initial density leads to the transformed variable and corresponding density as

$$\begin{aligned} z_k &= f_k \circ f_{k-1} \circ \ldots \circ f_1(z) \\ \ln q_k(z_k) &= \ln q_0(z) - \sum_{k=1}^{K} \ln|1 + u_k^T \psi_k z_{k-1}| \end{aligned} \tag{20}$$

where $\psi(z) = h'(w^T z + b)w$. Application of a planar flow to a standard Gaussian distribution leads to flexible expansions/contractions along hyperplanes and thus makes the base distribution a lot more expressive.

In the present study, the base distribution was initialized as a multivariate normal distribution with mean as a randomly initialized vector with length governed by the number of parameters of the neural



(a) (b)

Figure 9: Bayesian Neural UDE estimation is demonstrated for the Lotka Volterra example with a missing term as shown in (21); using the PSGLD approach. (a) Comparison of the recovered missing term and the actual term and (b) Sparsity plot using the STRRidge algorithm. The highlighted box shows the optimal point which gives the sparsest solution (1 term) with a low error. This plot is seen to be the same for 100 trajectories considered in the sampling phase.

ODE. We transformed this base distribution using a composition of 2 such planar layers described in Equation (19); with z being the parameters of the neural ODE.

Figure 8b shows that through the inclusion of normalizing flows, the estimation performance of the Bayesian Neural ODE: Variational Inference object is marginally better compared to figure 8a. For the Variational Inference experiment we used During experiments with different configurations such as Neural Network size, time span of solution and initial weights for the Neural Network used as well for the Normalizing Flow layers had considerable effect on the training performance and forecasting capability of the model. There are quite a lot of open questions that need to be addressed to ensure suitability of Variational Inference with Neural ODEs and will be part of future work.

2.5 Bayesian Neural UDE: SGLD and PSGLD

In this section, we aim to demonstrate a viable method for the probabilistic quantification of epistemic uncertainties via a hybrid machine-learning and mechanistic-model-based technique.

More efficient training of deep neural networks is achieved by using a preconditioned matrix $G(\theta)$ in the gradient update step of Equation 15 of the SGLD method [20]. Combining this with an adaptive step size method like RMSprop leads to much faster sampling than the standard SGLD approach, as outlined in the Preconditioned SGLD with RMSprop algorithm by [20]. We demonstrate the use of this algorithm in this section, where standard SGLD failed to converge to the true posterior.

2.5.1 Application to a predator-prey model

As outlined by [25], universal differential equations (UDE's) can be used to recover missing terms of governing equations describing dynamical systems. As an example, we look at the Lotka Volterra system with a missing term denoted by $M(u_1, u_2)$ in the first variable derivative as

$$\frac{du_1}{dt} = -\alpha u_1 - M(u_1, u_2) \tag{21}$$

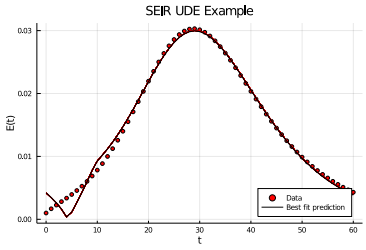
$$\frac{du_2}{dt} = -\delta u_2 + \gamma u_1 u_2 \tag{22}$$

$$\tag{23}$$

Using the PSGLD method outlined in [25], we trained $M_\theta(u_1, u_2)$ as a neural network to optimize the weights θ; and recover the missing term time series. We sampled from the last 100 updates of the converged sampler. Figure 9a shows that the recovered time series $M_\theta(u_1, u_2)$ from 100 trajectories matches very well with the actual term $M_\theta(u_1, u_2) = u_1 u_2$. These optimized parameter space for all 100 trajectories lies very close to each other, indicating that the PSGLD method indeed converges and then subsequently samples closer to the true posterior.

Table 3: Bayesian Neural UDE: Recovery of dominant terms for the Lotka Volterra example, as the sparsity parameter λ is varied. Highlighted row shows the sparsity parameter with the lowest positive AIC score.

λ	Number of Active terms	Dominant terms	Error	Mean AIC score	% sampled
0.01	9	u_1^2, u_2^2, u_1u_2 $u_1^2u_2^2, u_1^2u_2, u_2^2u_1$ u_1u_2, const	0.765	40.4	100
0.1	9	u_1^2, u_2^2, u_1u_2 $u_1^2u_2^2, u_1^2u_2, u_2^2u_1$ u_1u_2, const	0.764	35	100
1	5	u_1^2, u_2^2, u_2 $u_1^2u_2, u_1u_2$	0.764	21.6	100
2	2	$u_1^2u_2, u_1u_2$	0.634	7.2	100
3	**1**	u_1u_2	**0.7**	**4.1**	**100**
5	1	$u_1^2u_2$	2.49	-1	100



(a)

Figure 10: Bayesian Neural UDE estimation is demonstrated for the SEIR example with a missing term as shown in (29); using the PSGLD approach. (a) Comparison of the recovered missing term and the actual term shown for 100 trajectories considered in the sampling phase.

Subsequently, we used a sparse regression technique called Sequential Thresholded Ridge Regression (STRRidge) algorithm [4] on the neural network output to reconstruct the missing dynamical equations for 100 trajectories of the sampled parameter space. The STRRidge algorithm has a tunable sparsity parameter λ to control the sparsity of the obtained dominant terms. Optimally, we would want the sparsest solution with the least possible error.

Figure 9b shows the variation of the number of terms recovered by the STRRidge algorithm with the sparsity parameter, λ; which shows a decreasing trend as expected. The colorbar indicates the error between the sparse recovered solution and the neural network output $M_\theta(u_1, u_2)$. It can be seen that the highlighted box indicates the optimal point which has the sparsest solution (1 term) with a very low error. This point also corresponds to the lowest positive AIC score (Akaike Information Criteria) [3, 33] which strives to minimize the model error as well as its complexity (shown in Table 3). Along with showing the AIC score as function of the sparsity parameter λ, table 3 also shows the dominant terms as the sparsity parameter λ is varied.

This optimal solution has the quadratic form $\sim u_1u_2$, for all 100 trajectories indicating that Bayesian Neural UDE approach recovers the correct solution for all sampled trajectories. Out of the 100 models sampled for the optimal sparsity parameter, the model with the lowest AIC score was found to be $M(u_1, u_2) = 0.96u_1u_2$, which is very close to the true solution $M(u_1, u_2) = u_1u_2$.

2.5.2 Application to an epidemiology model

As another example to test our approach, we consider a SEIR epidemiological model. The SEIR is a compartment based model which models transfer of population between four compartments: Susceptible, Exposed, Infected and Recovered using the following set of equations:

$$\frac{dS}{dt} = -\beta SI \tag{24}$$
$$\frac{dE}{dt} = \beta SI - \sigma E \tag{25}$$
$$\frac{dI}{dt} = \sigma E - \gamma I \tag{26}$$
$$\frac{dR}{dt} = \gamma I \tag{27}$$
$$\tag{28}$$

In Equation 24, we will try to infer the exposure term using Bayesian Neural UDE: PSGLD approach. Thus, we will try to learn $M(E, I) = \sigma E$ in the following system of equations

$$\frac{dS}{dt} = -\beta SI \tag{29}$$
$$\frac{dE}{dt} = \beta SI - M(E, I) \tag{30}$$
$$\frac{dI}{dt} = \sigma E - \gamma I \tag{31}$$
$$\frac{dR}{dt} = \gamma I \tag{32}$$
$$\tag{33}$$

Using the PSGLD method outlined in [25], we trained $M_\theta(E, I)$ as a neural network to optimize the weights θ; and recover the missing term time series. We sampled from the last 100 updates of the converged sampler. Figure 10a shows that the recovered time series $M_\theta(E, I)$ from 100 trajectories matches very well with the actual term $M_\theta(E, I) = \sigma E$.

Subsequently, we applied the STRRidge algorithm to recover the symbolic equations for the missing terms, with the sparsity parameter λ ranging from $0.005 - 0.5$. The model for which the lowest AIC score was obtained, was found to contain just one dominant term (E) and with the symbolic form $M(E, I) = 0.099E$ compared to the ground truth data of $M(E, I) = 0.1E$, for all 100 trajectories sampled.

Thus, the model discovery for both the predator-prey example and the epidemiological model is robust to uncertainty. The Bayesian Neural UDE framework is thus an added arsenal to the recently demonstrated methods on bayesian system identification [2, 36]. Future work will further identify the relationship between model uncertainties and probabilistic automated discovery.

3 Conclusion and Future Work

We have shown that Bayesian learning frameworks can be integrated with Neural ODE's to quantify the uncertainty in the weights of a Neural ODE, using three sampling methods: NUTS, SGHMC and SGLD. Bayesian Neural ODEs with SGLD sampling is seen to provide better prediction accuracy and less bias from the MAP than NUTS sampling, possibly due to non-convexity/multi-modality of the likelihood function where the MAP point is likely to be in a region with low probability mass. However, a better understanding of why the two algorithms differ and how they compare to other algorithms is needed.

In addition, using a novel architecture which integrates convolution layers and Neural ODEs with the SGHMC framework; we demonstrate a test ensemble accuracy of 99.22% which is comparable with

the accuracy of state-of-the-art image classification methods.

Subsequently, for the first time, we demonstrate the integration of Neural ODEs with Variational Inference. We show that when variational inference is combined with normalizing flows, it leads to a good prediction and estimation performance on physical systems; thus leading to a potentially powerful Bayesian Neural ODE object.

Finally, considering the problem of recovering missing terms from a dynamical system using universal differential equations (UDEs); we demonstrate the Bayesian recovery of missing terms from dynamical systems for (a) a predator-prey model and (b) an epidemiological model.

Currently, it is observed that Bayesian learning of Neural ODEs is computationally expensive for large datasets. Therefore, more work is needed to evaluate, understand and improve the convergence of various approximate Bayesian inference and MCMC algorithms in the context of Neural ODEs. Another research direction we plan to delve into further is Bayesian Neural SDE's and their applicability to physical systems and large scale machine learning datasets.

4 Code Availability

All codes for SGHMC, Bayesian Neural UDE are publicly available at `https://github.com/RajDandekar/MSML21_BayesianNODE`
and codes for the Variational Inference Neural ODE object is available at `https://github.com/mohamed82008/BayesNeuralODE.jl`

References

[1] Look Andreas and Melih Kandemir. Differential bayesian neural nets. *arXiv preprint arXiv:1912.00796*, 2019.

[2] Steven Atkinson. Bayesian hidden physics models: Uncertainty quantification for discovery of nonlinear partial differential operators from data. *arXiv preprint arXiv:2006.04228*, 2020.

[3] Hamparsum Bozdogan. Model selection and akaike's information criterion (aic): The general theory and its analytical extensions. *Psychometrika*, 52(3):345–370, 1987.

[4] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[5] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

[6] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691. PMLR, 2014.

[7] Chris Cremer, Quaid Morris, and David Duvenaud. Reinterpreting importance-weighted autoencoders. *arXiv preprint arXiv:1704.02916*, 2017.

[8] Hong Ge, Kai Xu, and Zoubin Ghahramani. Turing: a language for flexible probabilistic inference. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, pages 1682–1690, 2018.

[9] Hong Ge, Kai Xu, and Zoubin Ghahramani. Turing: a language for flexible probabilistic inference. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, pages 1682–1690, 2018.

[10] Mark Girolami. Bayesian inference for differential equations. *Theoretical Computer Science*, 408(1):4 – 16, 2008. Computational Methods in Systems Biology.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.

[14] Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

[15] Hanwen Huang, Andreas Handel, and Xiao Song. A bayesian approach to estimate parameters of ordinary differential equation. *Computational Statistics*, 35(3):1481–1499, Sep 2020.

[16] Pavel Izmailov, Wesley J. Maddox, Polina Kirichenko, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Subspace inference for bayesian deep learning. *CoRR*, abs/1907.07504, 2019.

[17] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[18] Laurent Valentin Jospin, Wray L. Buntine, F. Boussaid, Hamid Laga, and M. Bennamoun. Hands-on bayesian neural networks - a tutorial for deep learning users. *ArXiv*, abs/2007.06823, 2020.

[19] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *The Journal of Machine Learning Research*, 18(1):430–474, 2017.

[20] Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. *arXiv preprint arXiv:1512.07666*, 2015.

[21] David J. Lunn, Nicky Best, Andrew Thomas, Jon Wakefield, and David Spiegelhalter. Bayesian analysis of population pk/pd models: General concepts and software. *Journal of Pharmacokinetics and Pharmacodynamics*, 29(3):271–307, Jun 2002.

[22] Wesley Maddox, Timur Garipov, Pavel Izmailov, Dmitry P. Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. *CoRR*, abs/1902.02476, 2019.

[23] Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

[24] Christopher Rackauckas, Alan Edelman, Keno Fischer, Mike Innes, Elliot Saba, Viral B Shah, and Will Tebbutt. Generalized physics-informed learning through language-wide differentiable programming. In *AAAI Spring Symposium: MLPS*, 2020.

[25] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Ramadhan. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

[26] Christopher Rackauckas and Qing Nie. Differentialequations. jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1), 2017.

[27] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial intelligence and statistics*, pages 814–822. PMLR, 2014.

[28] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[29] Lawrence K Saul, Tommi Jaakkola, and Michael I Jordan. Mean field theory for sigmoid belief networks. *Journal of artificial intelligence research*, 4:61–76, 1996.

[30] Kumar Shridhar, Felix Laumann, and Marcus Liwicki. A comprehensive guide to bayesian convolutional neural network with variational inference. *arXiv preprint arXiv:1901.02731*, 2019.

[31] Michalis K Titsias. Learning model reparametrizations: Implicit variational inference by fitting mcmc distributions. *arXiv preprint arXiv:1708.01529*, 2017.

[32] Udo von Toussaint. Bayesian inference in physics. *Rev. Mod. Phys.*, 83:943–999, Sep 2011.

[33] Eric-Jan Wagenmakers and Simon Farrell. Aic model selection using akaike weights. *Psychonomic bulletin & review*, 11(1):192–196, 2004.

[34] 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.

[35] Kai Xu, Hong Ge, Will Tebbutt, Mohamed Tarek, Martin Trapp, and Zoubin Ghahramani. Advancedhmc. jl: A robust, modular and efficient implementation of advanced hmc algorithms. 2019.

[36] Yibo Yang, Mohamed Aziz Bhouri, and Paris Perdikaris. Bayesian differential programming for robust systems identification under uncertainty. *Proceedings of the Royal Society A*, 476(2243):20200290, 2020.