Infinitely Deep Bayesian Neural Networks with Stochastic Differential Equations

Winnie Xu [1] **Ricky T.Q. Chen** [1] **Xuechen Li** [2] **David Duvenaud** [1 3]

Abstract

We perform scalable approximate inference in a recently-proposed family of continuous-depth Bayesian neural networks. In this model class, uncertainty about separate weights in each layer produces dynamics that follow a stochastic differential equation (SDE). We demonstrate gradient-based stochastic variational inference in this infinite-parameter setting, producing arbitrarily-flexible approximate posteriors. We also derive a novel gradient estimator that approaches zero variance as the approximate posterior approaches the true posterior. This approach further inherits the memory-efficient training and tunable precision of neural ODEs.

1. Introduction

Taking the limit of neural networks to be the composition of infinitely many residual layers provides a way to implicitly define its output as the solution to an ODE (Haber and Ruthotto, 2017; E, 2017). This continuous-depth parameterization decouples the specification of the model from its computation. While the paradigm adds complexity, it has several benefits: (1) Computational cost can be traded for precision in a fine-grained manner by specifying error tolerances for adaptive computation, and (2) memory costs for training can be significantly reduced by running the dynamics backwards in time to reconstruct activations of intermediate states needed for backpropagation.

On the other hand, the Bayesian treatment for neural networks modifies the typical training pipeline such that instead of performing point estimates, a distribution over parameters is learned. Although this approach adds complexity, it gives an automatic accounting of model uncertainty that helps to combat overfitting and improves calibration, especially on out-of-distribution data (Zhang et al., 2018; Osawa et al., 2019).

[1]University of Toronto [2]Stanford University [3]Google Research. Correspondence to: Winnie Xu <winnie.xu@mail.utoronto.ca>, David Duvenaud <duvenaud@cs.toronto.edu>.

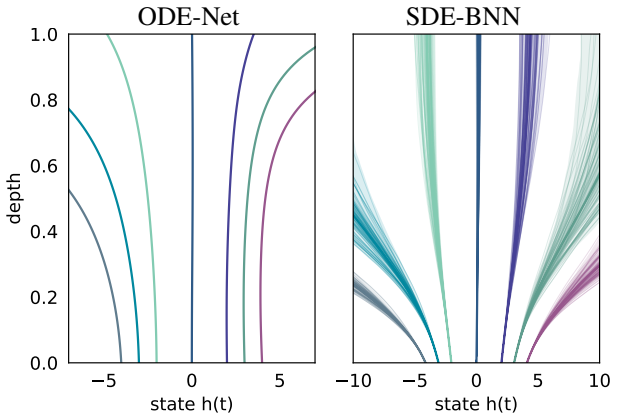


Figure 1. Hidden unit trajectories in an ODE-Net and an SDE-BNN. *Left*: A continuous-depth residual network has deterministic transformations of its hidden units from depths $t = 0$ to $t = 1$. *Right*: Uncertainty in the weights of a Bayesian continuous-depth residual network implies uncertainty in its hidden unit activation trajectories. Shaded regions show densities over samples from the learned posterior dynamics. *Both*: Each distinct color corresponds to a different initial state corresponding to different data inputs.

How can we combine the benefits of continuous-depth models with those of Bayesian neural networks? The simplest approach is a "Bayesian neural ODE" (Yıldız et al., 2019; Dandekar et al., 2020), which integrates out the finitely-many parameters of a standard neural ODE. This approach is straightforward to implement, and inherits the advantages of both Bayesian and continuous-depth neural networks. However, approximate inference for Bayesian neural networks, while well-studied, is still a challenging problem.

In this paper, we show that an alternative construction of Bayesian continuous-depth networks has additional practical benefits. Specifically, we consider the limit of infinite-depth Bayesian neural networks with separate unknown weights at each layer which we refer to as SDE-BNNs. This construction, explored by Peluchetti and Favaro (2020a;b), corresponds to defining an SDE over the weights. A Bayesian treatment of this model class and its variational objective were characterized theoretically by Tzen and Raginsky (2019a).

In this paper, we show that approximate inference in this model class can be done effectively using the scalable gradient-based variational inference scheme described in (Li et al., 2020; Tzen and Raginsky, 2019a). In this approach, the state of the output layer is computed by a black-box adaptive SDE solver and trained to maximize a variational lower bound. Figure 1 contrasts this neural SDE parameterization with the standard neural ODE approach.

This approach maintains the adaptive computation and constant-memory cost of training Bayesian neural ODEs. In addition, it has two unique advantages:

- The variational posterior can be made arbitrarily expressive by simply enlarging the neural network that parameterizes the dynamics of the approximate posterior. Under mild conditions, this family of approximate posteriors contains the true posterior.
- The variational objective admits a variance-reduced gradient estimator that is a natural extension of the "sticking the landing" trick (Roeder et al., 2017). Combined with arbitrarily expressive approximate posteriors, this estimator is consistent and has a vanishing variance as the approximate posterior approaches the true posterior.

The second contribution can also be applied to inference in SDEs more generally, such as in time-series modeling, but such applications are beyond the scope of this paper.

2. Background

2.1. Bayesian Neural Networks

Given a dataset, there are usually many functions that fit the data reasonably well, which a given neural network can express with different settings of its parameters. Instead of making a single point estimate of the parameters, the Bayesian paradigm frames learning as posterior inference, and integrates over many possible parameter settings. Formally, given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, we approximate the posterior over model weights $p(w|\mathcal{D}) \propto p(\mathcal{D}|w)p(w)$ by optimizing the Evidence Lower Bound (ELBO) with respect to a variational distribution $q(w)$:

$$\mathcal{L}_{\text{ELBO}}(\phi) = \mathbb{E}_{q(w)}\left[\log p(\mathcal{D}|w)\right] - D_{\text{KL}}\left(q(w)||p(w)\right). \tag{1}$$

However, exact inference is intractable, and the true posterior must be approximated. Stochastic variational inference (SVI) (Hoffman et al., 2013; Rezende et al., 2014)) turns posterior inference into a stochastic optimization problem. One of the main technical challenges of SVI is choosing a parametric family of approximate posteriors that is tractable to sample from and evaluate, while being flexible enough to approximate the true posterior well. Most scalable inference techniques use Gaussian approximate posteriors with restricted covariance structure between the weights in the network (Graves, 2011; Blundell et al., 2015; Zhang et al., 2018; Mishkin et al., 2018). Others construct complex approximate posteriors using normalizing flows (Krueger et al., 2018; Louizos and Welling, 2017) or through distillation (Balan et al., 2015; Wang et al., 2018).

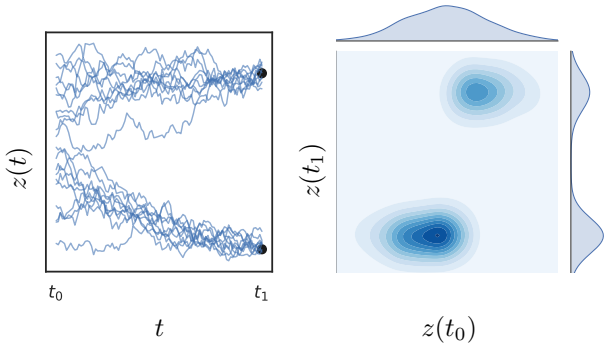


Figure 2. Neural SDEs can learn arbitrarily expressive approximate posteriors. *Left:* Samples from an approximate posterior, defined by an OU prior and conditioned on two observations with Cauchy likelihoods. *Right:* Joint distribution and marginals of the approximate posterior process z at times t_0 and t_1.

2.2. Neural Ordinary Differential Equations

Neural ordinary differential equations (Chen et al., 2018) define ODEs using neural networks:

$$\mathrm{d}h_t = f(h_t, t)\ \mathrm{d}t, \quad t \geq 0, \quad h_0 \in \mathbb{R}^d, \tag{2}$$

where $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ is a Lipschitz function defined by a neural network, and t indexes the layers of the network.

Starting at an initial value given by a data example and integrating these dynamics forward for a finite time can be seen as passing the input through an infinitely-deep residual network. For approximating scalar-valued functions, if extra dimensions are added to h, and the network is capped with a linear layer at the end, then these networks have similar universal approximation properties as standard neural networks (Dupont et al., 2019; Zhang et al., 2019b), and can be trained by standard stochastic gradient descent methods. Using adaptive ODE solvers allows trading speed for precision, and reconstructing the trajectory of the hidden units h by running the dynamics backwards can save memory during training.

2.3. Latent Stochastic Differential Equations

Informally, an SDE can be thought of as an ODE with infinitesimal noise added throughout time. They take the form of

$$\mathrm{d}w_t = f(w_t, t)\ \mathrm{d}t + g(w_t, t)\ \mathrm{d}B_t, \quad t \geq 0,\ w_0 \in \mathbb{R}^d, \tag{3}$$

where $\{w_t\}$ is the process of interest, $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and $g : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^{d\times m}$ are functions Lipschitz in both arguments, dubbed the *drift* and *diffusion*, respectively, and $\{B_t\}$ is an m-dimensional Brownian motion.

Many works have considered training SDEs with dynamics parameterized by neural networks (Li et al., 2020; Tzen and Raginsky, 2019a; Peluchetti and Favaro, 2020b; Innes et al., 2019; Kong et al., 2020; Liu et al., 2019). However, directly optimizing for the best drift and diffusion functions to maximize the log-likelihood of an observation $\log p(y_t|w_t)$ would result in the diffusion approaching 0. A better approach is to introduce a stochastic process over the states $\{w_t\}$ defined by an SDE with the form:

$$\mathrm{d}w_t = f_\theta(w_t, t)\ \mathrm{d}t + g(w_t, t)\ \mathrm{d}B_t, \tag{4}$$

and to perform posterior inference for the state process using another process that induces the approximate posterior on path space. We let this process be defined by

$$\mathrm{d}w_t = f_\phi(w_t, t)\ \mathrm{d}t + g(w_t, t)\ \mathrm{d}B_t. \tag{5}$$

When the dynamics function f_ϕ is parameterized by a neural network, this family of approximate posteriors is extremely expressive. As a simple example, Figure 2 shows that such a variational family can easily approximate non-Gaussian and multi-modal posteriors on path space.

It can be shown that if both the SDE defined on equation 4 and equation 5 share the same diffusion function, then the KL between the two induced measures on path space has the following form (Li et al., 2020; Tzen and Raginsky, 2019a):

$$D_{\mathrm{KL}}\left(\mu_q||\mu_p\right) = \mathbb{E}\left[\int_0^1 \tfrac{1}{2}\left\|u(t,\phi)\right\|_2^2\ \mathrm{d}t + \int_0^1 u(t,\phi)\ \mathrm{d}B_t\right], \tag{6}$$

where μ_q and μ_p are path space probability measures induced respectively by equation 5 and equation 4, $u(t,\phi) = g(w_t,t)^{-1}\left[f_\theta(w_t,t) - f_\phi(w_t,t)\right]$, and the expectation is taken under the approximate posterior process. This KL divergence can be estimated conveniently using simple Monte Carlo by running an SDE solver using the dynamics given by the approximate posterior. The term being integrated over time is analogous to the KL between marginals of the prior and posterior processes, and is simply the difference in drifts scaled by the inverse diffusion. (see (Li et al., 2020, Appendix 9.5) for a derivation).

Why share the same diffusion? To ensure that the KL divergence between the prior and approximate posterior on path space is finite, it is required to set the diffusion function for the approximate posterior to be equal to that of the prior. Surprisingly, this does not limit the expressivity of the approximate posterior. Specifically, a result by Boué et al. (1998) shows that there is a one-to-one correspondence between the space of path measures and the space of functions that can be taken by the the approximate posterior's drift which results in the same path space KL divergence. This implies that any path space measure close to the true posterior can be instantiated by SDEs with appropriate drifts.

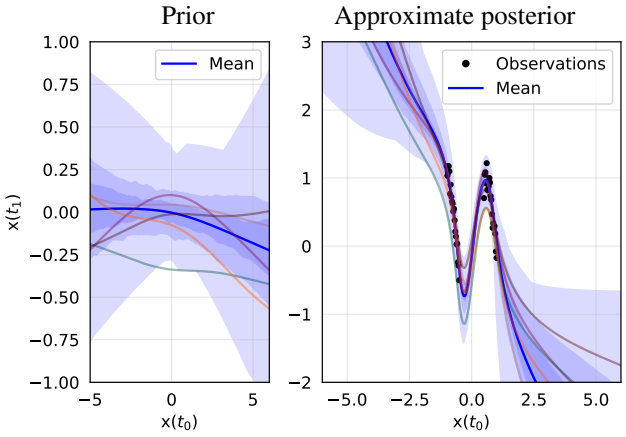


Figure 3. Predictive prior and posterior of the SDE-BNN on a non-monotonic toy dataset. Blue areas indicate density percentiles, and distinct colored lines show model samples.

3. Infinitely Deep Bayesian Neural Nets

Standard discrete-depth residual networks can be defined as a composition of layers of the form:

$$h_{t+\epsilon} = h_t + \epsilon f(h_t, w_t), \quad t = 1 \ldots T, \tag{7}$$

where t is the layer index, $h_t \in \mathbb{R}^{D_h}$ denotes a vector of hidden unit activations at layer t, the input $h_0 = x$, and $w_t \in \mathbb{R}^{D_w}$ represents the weight matrices and biases for layer t. In the discrete setting, $\epsilon = 1$.

We can construct a continuous-depth variant of residual networks by setting $\epsilon = 1/T$ and taking the limit as $T \to \infty$. This yields a differential equation that describes the hidden unit evolution as a function of depth t. Since standard residual networks are parameterized with different "weights" per layer, we denote the weights at layer t by w_t. To specify different weights at each layer with a finite number of parameters, we can introduce a hypernetwork f_w which specifies the change in weights as a function of depth and the current weights (Ha et al., 2016). The evolution of the hidden unit activations and weights can then be combined into a single differential equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} h_t \\ w_t \end{bmatrix} = \begin{bmatrix} f_h(t, h_t, w_t) \\ f_w(t, w_t) \end{bmatrix} \tag{8}$$

with some learned initial weight value w_{t_0}. Using time-varying weights is similar to augmenting the state (Dupont

et al., 2019; Zhang et al., 2019c). See Appendix 9 for details on the effects of augmentation. We then perform Bayesian inference on the weight process w_t, assigning a suitable prior stochastic process and performing variational inference in this infinitesimal limit.

Like all Bayesian neural networks with observation likelihoods, our framework models both aleatoric and epistemic uncertainty: The likelihood $p(y|h_1)$ captures the noise in observations, and the SDE encodes uncertainty about weights.

Prior process on weights. Standard priors for Bayesian neural networks use independent Gaussians across all weights and layers. Taking the limit above would imply a white noise process prior on the weights $w(\cdot)$. However, scaling this noise to result in finite variance is difficult (Peluchetti and Favaro, 2020a;b). Instead, we use the Ornstein–Uhlenbeck (OU) process as the prior. The process is characterized by an SDE with drift and diffusion:

$$f_p(w_t, t) = -w_t, \quad g(w_t, t) = \sigma I_d, \tag{9}$$

where σ is a hyperparameter. We choose this prior due to its simplicity and that its marginal variance approaches a constant in the large time limit. Investigation into alternative priors is beyond the current scope.

Approximate posterior over weights. We parameterize the approximate posterior on weights implicitly using another SDE with the following drift function:

$$f_q(w_t, t, \phi) = f_p(w_t, t) + \text{NN}(w_t, t, \phi). \tag{10}$$

This drift function f_q is parameterized by a small neural network (NN) with parameters ϕ. With this drift, the approximate posterior process may have non-Gaussian marginals, and its expressive capacity may also be increased by increasing that of the neural net.

Evaluating the network. Evaluating our network at a given input requires sampling a weight path $\{w_t\}$ from the posterior process and evaluating the network activations $\{h_t\}$ given the sampled weights and the input. Both steps require solving a differential equation. Luckily, both steps can be done simultaneously by a single SDE solver call with the augmented state SDE:

$$\mathrm{d}\begin{bmatrix} w_t \\ h_t \end{bmatrix} = \begin{bmatrix} f_w(w_t, t, \phi) \\ f_h(h_t, t, w_t) \end{bmatrix} \mathrm{d}t + \begin{bmatrix} g_w(w_t, t) \\ \mathbf{0} \end{bmatrix} \mathrm{d}B_t, \tag{11}$$

where $h_0 = x$, the input. The learnable parameters are the initial weight values at time zero w_0 and those of the drift function ϕ.

Output likelihood. The final state of the hidden units h_1 is used to parameterize the likelihood of the target output y:

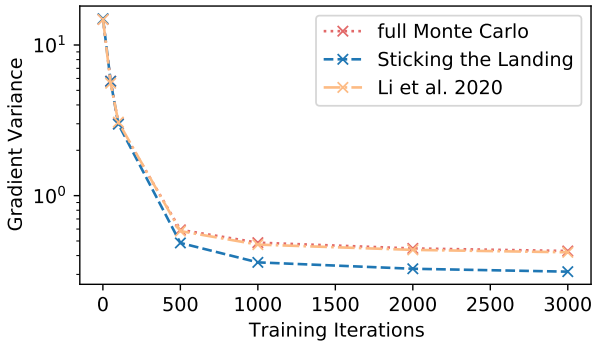


Figure 4. Comparison of the variance in three gradient estimators. Our infinite-dimensional STL gradients converge to zero as the approximate posterior converges to the true posterior.

$\log p(y|x, w) = \log p(y|h_1)$. For instance, $p(y|h_1)$ could be a Cauchy likelihood for regression, or categorical likelihood for classification.

Training objective. With gradient-based optimizers, we maximize the infinite-dimensional ELBO:

$$\begin{aligned} \mathcal{L}_{\text{ELBO}_\infty}(\phi) &= \mathbb{E}\left[\log p(\mathcal{D}|w)\right] - D_{\text{KL}}(\phi) \\ &= \mathbb{E}\left[\log p(\mathcal{D}|w) - \int_0^1 \tfrac{1}{2} \left\| \text{NN}(w_t, t, \phi) g_w(w_t, t)^{-1} \right\|_2^2 \mathrm{d}t\right] \end{aligned} \tag{12}$$

The sampled weights, the hidden activations, and the training objective are all computed simultaneously using a single call to an SDE solver.

3.1. Variance-Reduced Gradients

Roeder et al. (2017) showed that when training with the reparameterization gradient, a gradient estimator with lower variance can be constructed by removing a score function term that has zero expectation. The variance of this gradient estimator approaches zero as the approximate posterior approaches the true posterior. Here, we refer to this general trick as "sticking the landing" (STL). We adapt this idea to the SDE setting by replacing the original estimator of the path space KL with the following estimator:

$$\widehat{\text{KL}}_{\text{STL}} = \int_0^1 \tfrac{1}{2} \left\| u(w_t, t, \phi) \right\|_2^2 \mathrm{d}t + \int_0^1 u(w_t, t, \perp(\phi)) \, \mathrm{d}B_t, \tag{13}$$

where $u := \text{NN}(w_t, t, \phi) g_w(w_t, t)^{-1}$, the path $\{w_t\}_{t \in [0,T]}$ is sampled from the approximate posterior process, and $\perp(\cdot)$ is the stop gradient function that renders the input a constant and with respect to which gradient propagation is stopped.

Because our approximate posterior can be made arbitrarily expressive, we conjecture that our approach can achieve arbitrarily low gradient variance towards the end of training

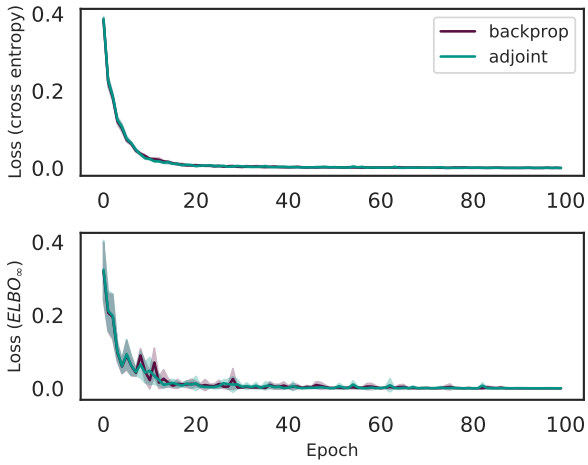


Figure 5. Benchmarking two gradient computation methods: (1) Back-propagation through the SDE solver, and (2) the memory-efficient stochastic adjoint of Li et al. (2020). Both methods have similar optimization dynamics, final performance, and wall-clock time, but the adjoint approach is more memory-efficient. Detailed comparisons of wall-clock time and evaluation step results in Appendix A.3.5.

if the network parameterizing f_w is made expressive enough. See Appendix is A.2 for a heuristic derivation.

We show the variance of different gradient estimators in Figure 4, averaged across the parameters θ, on a 1D regression setting. We compare STL against a fully Monte Carlo estimate of equation 6, as well as a KL estimator that has non-zero gradient variance except at the optimum that was previously used by Li et al. (2020). Figure 4 shows that STL has lower variance than alternatives.

4. Experiments

As a proof of concept for this class of continuous-depth parameterizations, we investigate the effectiveness of our proposed approximate inference method for SDE-BNNs in terms of classification accuracy, calibration, perturbation robustness, and speed-precision trade-offs. Our code is available at `https://github.com/xwinxu/bayesian-sde`.

We consider toy regression tasks and image classification tasks on MNIST and CIFAR-10. We also investigate out-of-distribution generalization of our approach. Notably, our approach does not require *post hoc* recalibration methods such as training with temperature scaling (Guo et al., 2017) or isotonic regression (Zadrozny and Elkan, 2002).

Backpropagation through solvers vs. adjoint. We experimented with fixed- and adaptive-step SDE solvers, as well as the stochastic adjoint of Li et al. (2020). Figure 5 shows similar convergence for both approaches. Appendix A.3 shows that both approaches used similar numbers of dynamics function evaluations, and have similar wall-clock time.

The error estimation overhead in our adaptive solvers was substantial; therefore, for final model evaluation, we trained with fixed-step solvers, where the number of steps is chosen to be large enough to match the convergence speed of our adaptive-step solvers.

Baselines. For a fixed-depth network baseline, we compare to standard residual networks. We then test variational inference on the weights of these residual network architectures.

We also perform ablation studies to compare our approach to more standard variational inference approaches over continuous-depth networks. Specifically, we compare to a "Mean Field ODE-net" where stochastic variational inference is performed over weights that do not vary across depth. This baseline is a standard fully-factorized Gaussian approximate posterior, *i.e.* mean-field approximation, and has been used for Neural ODEs by Look and Kandemir (2019); Dandekar et al. (2020).

We further compare our model to a "Mean Field HyperODEnet", where a learned drift is applied to w, but mean-field inference is instead performed over the parameters of the hypernetwork. Alternatively, one can interpret this as another "Mean Field ODEnet" with a larger state and a more complex drift function but that has similar computational complexity as our SDE-BNN approach. This setting stands in contrast to our approach where Bayesian inference is naturally carried out over the entire continuous-depth network as a stochastic process.

Parameterizing the drift function. We parameterized the drift function of the variational posterior f_w using a simple multilayer perceptron. To make training easier, we added the prior drift function to the output of this multilayer perceptron. This results in a zero initialization of the last layer, which corresponds to an initialization of the approximate posterior SDE to the prior SDE.

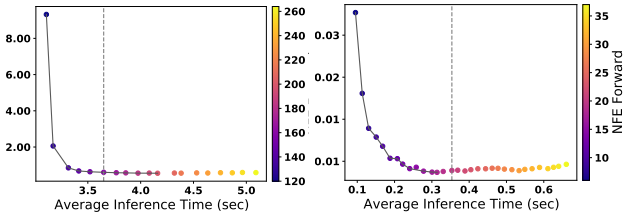
Hyperparameters. We swept across learning rates in the range [1e-4, 1e-3], selecting the optimal value based on the validation set. We train with the default Adam optimizer (Kingma and Ba, 2015). For the image classification experiments, all convolutional layers of the drift network are time-conditional and use the `tanh` non-linearity. The diffusion coefficient σ was selected from validation performance over $\{0.1, 0.2, 0.5\}$.

4.1. 1D Regression

We first verify the capabilities of the SDE-BNN on a 1D regression problem. Conditioned on a sample from the diffusion process, each sample from a one-dimensional SDE-

	MNIST		CIFAR-10	
Model	Accuracy (%)	ECE ($\times 10^{-2}$)	Accuracy (%)	ECE ($\times 10^{-2}$)
ResNet32	99.46 ± 0.00	2.88 ± 0.94	87.35 ± 0.00	8.47 ± 0.39
ODENet	98.90 ± 0.04	1.11 ± 0.10	88.30 ± 0.29	8.71 ± 0.21
HyperODENet	99.04 ± 0.00	1.04 ± 0.09	87.92 ± 0.46	15.86 ± 1.25
Mean Field ResNet32	99.44 ± 0.00	2.76 ± 1.28	86.97 ± 0.00	3.04 ± 0.94
Mean Field ODENet	98.81 ± 0.00	2.63 ± 0.31	81.59 ± 0.01	3.62 ± 0.40
Mean Field HyperODENet	98.77 ± 0.01	2.82 ± 1.34	80.62 ± 0.00	4.29 ± 1.10
SDE BNN	99.30 ± 0.09	0.63 ± 0.10	88.98 ± 0.94	7.60 ± 0.37
SDE BNN (+ STL)	99.10 ± 0.09	0.78 ± 0.12	89.10 ± 0.45	7.97 ± 0.51

Table 1. Classification accuracy and expected calibration error (ECE) on MNIST and CIFAR-10. No batch normalization was used as it is unclear how it can be applied to continuous-depth models. Our approach (SDE BNN and SDE BNN (+STL)) is either competitive or outperforms baselines (ODENet), while obtaining better generalization as seen in lower expected calibration error (ECE).



(a) Negative Log-Likelihood (b) Expected Calibration Error

Figure 6. Adjusting the SDE solver tolerance of an SDE-BNN at test time allows a tradeoff between computational speed and predictive performance. Grey line indicates the solver's tolerance during training. Evaluated on CIFAR10, averaged across 3 seeds.

BNN is a bijective mapping from the inputs to the outputs. This implies that every function sampled from a 1D SDE-BNN is monotonic. To be able to sample non-monotonic functions, we augmented the state with 2 extra dimensions set to zero, as in Dupont et al. (2019). Figure 3 shows that our model learns a reasonably flexible approximate posterior on a synthetic non-monotonic 1D dataset. We emphasize that the samples from our model are smooth because the hidden states h do not possess additive instantaneous noise, and only on the weights w do we apply a diffusion.

4.2. Image Classification

Next, we evaluate the performance of SDE-BNN on standard MNIST and CIFAR-10 classification tasks. Instantaneous changes to the hidden state (f_h) are parameterized using a convolutional neural network, including one strided convolution for downsampling and a transposed convolution layer for upsampling. We then set the weights w to be the filters and biases of all the convolutional layers. The approximate posterior drift dynamics (f_w) is a multilayer perceptron with hidden layer widths of 2, 128, and 2. The small hidden width of the bottleneck layers was chosen to reduce the number of variational parameters and promote linear scaling with respect to the dimension of w. On MNIST, we used one such SDE-BNN block, while on CIFAR-10, we used a multi-scale variant where multiple SDE-BNN blocks were stacked with the invertible downsampling from Dinh et al. (2016) in between.

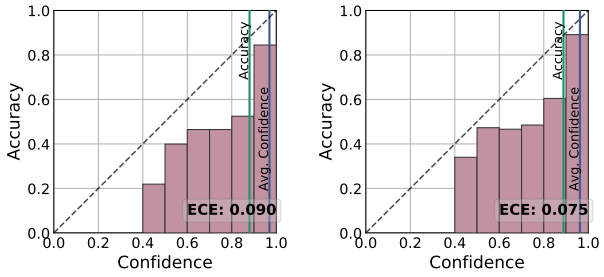
We report classification results in Table 1. The SDE-BNN generally outperforms the baselines, and we notice that while the continuous-depth Neural ODE (ODENet) models can achieve similar classification performance on a standard residual network, it consistently has poorer calibration.

By using instantaneous noise, we recover and outperform the standard residual network on MNIST and CIFAR-10, respectively, all the while obtaining lower expected calibration errors (ECE) on the predictions. From the ablation studies on our approach, we found that it was much harder to achieve similar performance with either of the mean field variants of an ODEnet and that they demonstrated a harsher trade-off between performance and calibration.

Figure 6 demonstrates the ability of SDE-BNNs, like neural ODEs previously, to trade off computation time for precision. Figure 13 in Appendix A.3.3 indicates calibration is insensitive to solver tolerances close to the value used during training.

4.2.1. Calibration

Table 1 quantifies our model's calibration using expected calibration error (ECE; Guo et al. (2017)). The SDE-BNN models, marginally emphasized with the added STL setting, appear better calibrated than the Neural ODE (Chen et al., 2018) and mean field ResNet baselines. Figure 7 shows better calibration than neural ODEs with similar accuracy. Appendix Figure 12 shows the insensitivity of these results



(a) Neural ODE Baseline (b) SDE-BNN

Figure 7. Calibration in a neural ODE versus an SDE-BNN on the test set of CIFAR-10. The SDE-BNN is displays somewhat better calibration and generalization.

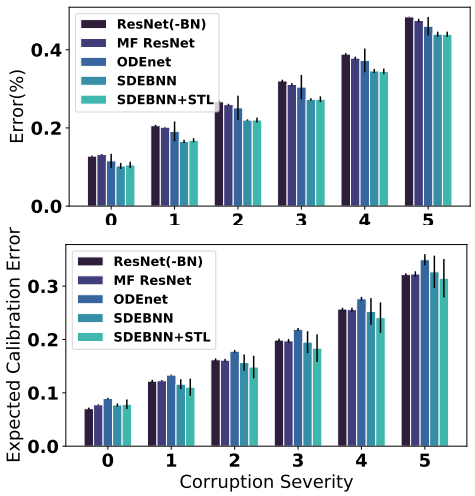


Figure 8. Robustness to input corruption on CIFAR-10. On this task, SDE-based neural nets show better accuracy and calibration than non-Bayesian and mean-field methods. Black bars show standard deviation across 3 random seeds.

to solver step size.

4.2.2. Robustness to Input Corruption

We show the robustness of SDE-BNNs by evaluating on all 19 non-adversarial corruptions across 5 severity levels in the CIFAR10-C (Hendrycks and Dietterich, 2019) benchmark. These corruptions mimic real-world perturbations such as noise, blur, and weather. To evaluate the classification robustness of the SDE-BNN, we compare the mean corruption error (mCE), the average error for each intensity level summed across all 19 perturbations, to the top-1 error rate on the corresponding clean CIFAR-10 dataset.

Figure 8 shows error on the corrupted test set relative to the clean, demonstrating a steady increase in mCE across increasing perturbation severity levels along with the overall error measurement summarized in Table 1. On both CIFAR-10 and CIFAR10-C, the SDE-BNN and SDE-BNN + STL models achieve lower overall test error and better calibration than the baselines.

Compared to the standard baselines ResNet32 and Mean Field (MF) ResNet32, SDE BNN achieves around 4.4% lower absolute corruption error (CE), the total classification error for all corruption tasks across all 5 severity levels (Hendrycks and Dietterich, 2019), in comparison to the clean errors. The effectiveness of learned uncertainty on out-of-domain inputs indicates that the SDE-BNN is more robust to observation perturbations despite not being trained on such diverse forms of corruptions.

5. Scope and Limitations

Computational speed The cost of evaluating our model grows in $\mathcal{O}(DT)$, where D is the number of weights, and T the number of iterations taken by the solver. This may seem advantageous compared to the $\mathcal{O}(D^3)$ cost for non-factorized Gaussian approximate posteriors, but the number of steps required is difficult to characterize. Although our approach allows adjustment of the computational cost at test time, it is harder to control the cost of evaluation during training time, making our method relatively slow to train. Relatedly, Dusenberry et al. (2020) recently demonstrated an $\mathcal{O}(DK)$ cost approximate posterior in standard BNNs.

Batch norm There are two places in our model at which batch normalization could be applied: (1) in the weight hypernetwork, and (2) in the dynamics of the hidden unit activations. However, it is unclear how to incorporate batch-norm into either places while maintaining the consistency properties of Bayesian inference. Zhang et al. (2019a); Chang et al. (2020) proposed initializations that yield the same performance without needing batch normalization.

Low-variance gradients for other domains Our extention of the STL gradient estimator (Roeder et al., 2017) to the infinite-dimensional variational objective could also be used in other settings for faster convergence, such as the time series applications Li et al. (2020) investigated.

6. Related Work

The earliest theoretical characterization of infinitely-deep Bayesian neural networks were made by Neal (1996, Chapter 2), who also characterized infinitely-wider Bayesian neural networks as Gaussian processes. Duvenaud et al. (2014) also investigated the theoretical properties of kernel-based constructions of infinitely-deep Bayesian neural networks.

Theory of Neural SDEs Peluchetti and Favaro (2020b;a) derive correspondences between infinitely deep stochastic neural networks and SDEs, but do not treat them as Bayesian

Method	Posterior over Stochastic Process	Flexible Approximate Posterior	Adaptive Computation	References
Bayes by Backprop	✗	✗	✗	Blundell et al. (2015)
Bayesian Hypernets	✗	✓	✗	Krueger et al. (2018)
BBVI for SDEs	✓	✗	✗	Ryder et al. (2018)
Bayesian Neural ODEs	✗	✗	✓	Yıldız et al. (2019) Dandekar et al. (2020)
SDE-BNN	✓	✓	✓	current work

Table 2. Properties of various Bayesian supervised learning approaches.

neural networks. Specifically, Peluchetti and Favaro (2020a) derived the drift and diffusion functions corresponding to a particular randomly initialized convolutional residual network, and explored the properties of infinite-wide variants of this related model class. In contrast, we use a Bayesian approach, tracking the uncertainty arising from limited data. Tzen and Raginsky (2019b) formulated neural SDEs from a latent variable model, and their connections to deep neural networks was made precise in Tzen and Raginsky (2019a). Specifically, Tzen and Raginsky (2019a) proposed the use of the Radon-Nikodym derivative as an infinite-dimensional analog to the variational evidence lower bound (ELBO) objective. However their approach relied on forward-mode differentiation, so did not scale to large models.

Neural SDEs with heuristic training objectives (Kong et al., 2020) proposed fitting a neural SDE by using a heuristic training objective based on encouraging the diffusion to be large away from the training data, and using a fixed Euler-Maruyama discretization. Innes et al. (2019) trained neural SDEs by backpropagating through the operations of the solver, however their training objective simply matched the first two moments of the training data, implying that it could not consistently estimate diffusion functions. This approach is also relatively memory-intensive. Liu et al. (2019) and Oganesyan et al. (2020) add noise to the solver operations in a neural ODE, although the diffusion must be tuned as a hyperparameter. Hegde et al. (2018) proposed a form of neural SDE using Gaussian processes to parameterize the drift and diffusion functions for a fixed E-M discretization. However, the diffusion functions are based on an *ad-hoc* construction from a Gaussian process posterior conditioned on inducing points. Ryder et al. (2018) used a Gaussian process variational posterior, effectively a continuous-time analog of a mean field approximation which may not always be expressive enough to model the true posterior.

Neural ODEs with finite-dimensional stochasticity Some methods based on building variational autoencoders with a neural ODE share similar training objectives, since the ELBO appears frequently in posterior inference. The Latent ODE model (Rubanova et al., 2019) only performs inference on the distribution at an initial time of a continuous hidden state. De Brouwer et al. (2019) introduced stochastic jumps at data locations, and do not perform continuous-time inference. While performing amortized inference for time series modeling, Yıldız et al. (2019) also infers the weights of an ODE drift function. Dandekar et al. (2020) has a similar setting but for supervised learning.

Approximate posterior defined with neural nets Krueger et al. (2018) and Louizos and Welling (2017) use normalizing flows to construct a non-factorized, non-Gaussian approximate posterior in Bayesian neural networks. However, normalizing flows have poor scaling with dimension and point estimates are often used for most of the weights in the neural network. Table 2 compares qualities of our approach with existing methods for stochastic variational inference in BNNs.

Diffusion Models Song et al. (2020) build a generative model using an encoder-decoder pair of SDEs defined implicitly by the gradients of an energy function. They then use the Fokker-Planck equation to directly update their model without differentiating through sample paths.

7. Conclusion

We developed a practical method for approximate inference in infinitely-deep Bayesian neural networks. In particular, our method allows arbitrarily-expressive, non-factorized approximate posteriors implicitly defined through neural SDEs. We also developed an unbiased gradient estimator whose variance approaches zero as the approximate posterior approaches the true posterior.

The combination of this gradient estimator with arbitrarily flexible approximate posteriors gives this family of Bayesian neural networks a special property, which is that the training gradients' bias and variance can be made arbitrarily small during training. Furthermore, we demonstrated the ability of this continuous-depth model class to use adaptive SDE solvers. This allows a memory-efficient training, and a fine-grained trade-off between precision and speed.

REFERENCES

Balan, A. K., Rathod, V., Murphy, K. P., and Welling, M. (2015). Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pages 3438–3446.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

Boué, M., Dupuis, P., et al. (1998). A variational representation for certain functionals of brownian motion. *The Annals of Probability*, 26(4):1641–1659.

Chang, O., Flokas, L., and Lipson, H. (2020). Principled weight initialization for hypernetworks. In *ICLR*.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural ordinary differential equations. *Advances in Neural Information Processing Systems*.

Dandekar, R., Dixit, V., Tarek, M., Garcia-Valadez, A., and Rackauckas, C. (2020). Bayesian neural ordinary differential equations. *arXiv preprint arXiv:2012.07244*.

De Brouwer, E., Simm, J., Arany, A., and Moreau, Y. (2019). Gru-ode-bayes: Continuous modeling of sporadically-observed time series. *arXiv preprint arXiv:1905.12374*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Dupont, E., Doucet, A., and Teh, Y. W. (2019). Augmented neural odes. In *NeurIPS*.

Dusenberry, M., Jerfel, G., Wen, Y., Ma, Y., Snoek, J., Heller, K., Lakshminarayanan, B., and Tran, D. (2020). Efficient and scalable bayesian neural nets with rank-1 factors. In *International conference on machine learning*, pages 2782–2792. PMLR.

Duvenaud, D., Rippel, O., Adams, R. P., and Ghahramani, Z. (2014). Avoiding pathologies in very deep networks. In *Artificial Intelligence and Statistics*.

E, W. (2017). A Proposal on Machine Learning via Dynamical Systems. *Commun. Math. Stat.*, 5(1):1–11.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org.

Ha, D., Dai, A., and Le, Q. V. (2016). Hypernetworks. *arXiv preprint arXiv:1609.09106*.

Haber, E. and Ruthotto, L. (2017). Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004.

Hegde, P., Heinonen, M., Lähdesmäki, H., and Kaski, S. (2018). Deep learning with differential gaussian process flows. *arXiv preprint arXiv:1810.04066*.

Hendrycks, D. and Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

Innes, M., Edelman, A., Fischer, K., Rackauckus, C., Saba, E., Shah, V. B., and Tebbutt, W. (2019). Zygote: A differentiable programming system to bridge machine learning and scientific computing. *arXiv preprint arXiv:1907.07587*, page 140.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kong, L., Sun, J., and Zhang, C. (2020). Sde-net: Equipping deep neural networks with uncertainty estimates. *arXiv preprint arXiv:2008.10546*.

Krueger, D., Huang, C.-W., Islam, R., Turner, R., Lacoste, A., and Courville, A. (2018). Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*.

Li, X., Wong, T.-K. L., Chen, R. T., and Duvenaud, D. (2020). Scalable gradients for stochastic differential equations. *arXiv preprint arXiv:2001.01328*.

Liu, X., Xiao, T., Si, S., Cao, Q., Kumar, S., and Hsieh, C.-J. (2019). Neural sde: Stabilizing neural ode networks with stochastic noise. *arXiv preprint arXiv:1906.02355*.

Look, A. and Kandemir, M. (2019). Differential bayesian neural nets. *arXiv preprint arXiv:1912.00796*.

Louizos, C. and Welling, M. (2017). Multiplicative normalizing flows for variational bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2218–2227. JMLR. org.

Mishkin, A., Kunstner, F., Nielsen, D., Schmidt, M., and Khan, M. E. (2018). Slang: Fast structured covariance approximations for bayesian deep learning with natural gradient. In *Advances in Neural Information Processing Systems*, pages 6245–6255.

Neal, R. M. (1996). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Oganesyan, V., Volokhova, A., and Vetrov, D. (2020). Stochasticity in neural odes: An empirical study. *arXiv preprint arXiv:2002.09779*.

Osawa, K., Swaroop, S., Jain, A., Eschenhagen, R., Turner, R. E., Yokota, R., and Khan, M. E. (2019). Practical deep learning with bayesian principles. *arXiv preprint arXiv:1906.02506*.

Peluchetti, S. and Favaro, S. (2020a). Doubly infinite residual networks: a diffusion process approach.

Peluchetti, S. and Favaro, S. (2020b). Infinitely deep neural networks as diffusion processes. In *International Conference on Artificial Intelligence and Statistics*, pages 1126–1136. PMLR.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Roeder, G., Wu, Y., and Duvenaud, D. K. (2017). Sticking the landing: Simple, lower-variance gradient estimators for variational inference. In *Advances in Neural Information Processing Systems*, pages 6925–6934.

Rubanova, Y., Chen, R. T. Q., and Duvenaud, D. (2019). Latent odes for irregularly-sampled time series. *arXiv preprint arXiv:1907.03907*.

Ryder, T., Golightly, A., McGough, A. S., and Prangle, D. (2018). Black-box variational inference for stochastic differential equations. In *International Conference on Machine Learning*, pages 4423–4432. PMLR.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020). Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations*.

Tzen, B. and Raginsky, M. (2019a). Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*.

Tzen, B. and Raginsky, M. (2019b). Theoretical guarantees for sampling and inference in generative models with latent diffusions. *arXiv preprint arXiv:1903.01608*.

Wang, K.-C., Vicol, P., Lucas, J., Gu, L., Grosse, R., and Zemel, R. (2018). Adversarial distillation of bayesian neural network posteriors. *arXiv preprint arXiv:1806.10317*.

Yıldız, Ç., Heinonen, M., and Lähdesmäki, H. (2019). Ode^{2}vae: Deep generative second order odes with bayesian neural networks. *arXiv preprint arXiv:1905.10994*.

Zadrozny, B. and Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 694–699.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. (2018). Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pages 5852–5861.

Zhang, H., Dauphin, Y. N., and Ma, T. (2019a). Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*.

Zhang, H., Gao, X., Unterman, J., and Arodz, T. (2019b). Approximation capabilities of neural ordinary differential equations. *arXiv preprint arXiv:1907.12998*.

Zhang, T., Yao, Z., Gholami, A., Keutzer, K., Gonzalez, J., Biros, G., and Mahoney, M. W. (2019c). ANODEV2: A coupled neural ODE evolution framework. *CoRR*, abs/1906.04596.

A. Appendix

Notation. Denote as ϕ the vector of variational parameters, f_q as the approximate posterior on weights, f_p as the prior on weights, f_h as the dynamics of hidden units, and σ as the diffusion function. Denote the Euclidean norm of a vector u by $|u|$. For function f denote its Jacobian as ∇f.

A.1. Derivation of an Alternative Monte Carlo Estimator

The goal of this section is to derive a Monte Carlo estimator of the KL-divergence on path space that is similar to the *fully Monte Carlo* estimator described in (Roeder et al., 2017). This will serve as the basis for the subsequent heuristic derivation of the continuous-time sticking-the-landing trick.

Let w_0 be a fixed initial state. Let $w_1, ..., w_N$ be states at times $\Delta t, 2\Delta t, \ldots, N\Delta t = T$ generated by the Euler discretization:

$$w_{i+1} = w_i + f_q(w_i)\Delta t + \sigma(w_i)(B_{t+\Delta t} - B_t) \tag{14}$$

$$= w_i + f_q(w_i)\Delta t + \sigma(w_i)\Delta t^{1/2}\epsilon_{i+1}, \quad \epsilon_{i+1} \sim \mathcal{N}(0,1). \tag{15}$$

where $\{B_t\}_{t\geq 0}$ is the Brownian motion. This implies that conditional on the previous state, the current state is normally distributed:

$$w_{i+1}|w_i \sim \mathcal{N}(w_i + f_q(w_i)\Delta t, \sigma(w_i)^2\Delta t).$$

Thus, the log-densities can be evaluated as

$$\log q(w_{i+1}|w_i) = -\frac{1}{2}\log(2\pi\sigma(w_i)^2\Delta t) - \frac{1}{2}\frac{\left(w_{i+1} - (w_i + f_q(w_i)\Delta t)\right)^2}{\sigma(w_i)^2\Delta t}, \quad i = 0, \ldots N-1. \tag{16}$$

On the other hand, if at any time, the next state was generated from the current state based on the prior process, we would have the following log-densities:

$$\log p(w_{i+1}|w_i) = -\frac{1}{2}\log(2\pi\sigma(w_i)^2\Delta t) - \frac{1}{2}\frac{\left(w_{i+1} - (w_i + f_p(w_i)\Delta t)\right)^2}{\sigma(w_i)^2\Delta t}, \quad i = 0, \ldots N-1. \tag{17}$$

Now, we substitute the form of w_{i+1} based on equation 14 into equation 16 and equation 17 and obtain

$$\begin{aligned}
\log q(w_{i+1}|w_i) =& -\frac{1}{2}\log(2\pi\sigma(w_i)^2\Delta t) - \frac{1}{2}\epsilon_{i+1}^2, \\
\log p(w_{i+1}|w_i) =& -\frac{1}{2}\log(2\pi\sigma(w_i)^2\Delta t) \\
& -\frac{1}{2}\left(\frac{(f_q(w_i) - f_p(w_i))^2}{\sigma(w_i)^2}\Delta t + \frac{2(f_q(w_i) - f_p(w_i))\epsilon_{i+1}}{\sigma(w_i)}\Delta t^{1/2} + \epsilon_{i+1}^2\right).
\end{aligned}$$

The KL divergence on the path space could then be regarded as a sum of infinitely many KL-divergences between Gaussians:

$$\lim_{N\to\infty}\sum_{i=0}^{N}\mathbb{E}_{w_i}\left[D_{\mathrm{KL}}\left(q(w_{i+1}|w_i)||p(w_{i+1}|w_i)\right)\right] \tag{18}$$

$$= \lim_{N\to\infty}\sum_{i=0}^{N}\mathbb{E}_{w_i}\left[\mathbb{E}_{w_{i+1}\sim q(w_{i+1}|w_i)}\left[\log\frac{q(w_{i+1}|w_i)}{p(w_{i+1}|w_i)}\right]\right] \tag{19}$$

$$= \lim_{N\to\infty}\sum_{i=0}^{N}\mathbb{E}_{w_i}\left[\mathbb{E}_{\epsilon_{i+1}}\left[\frac{(f_q(w_i) - f_p(w_i))^2}{2\sigma(w_i)^2}\Delta t + \frac{(f_q(w_i) - f_p(w_i))}{\sigma(w_i)}\Delta t^{1/2}\epsilon_{i+1}\right]\right] \tag{20}$$

$$= \mathbb{E}\left[\frac{1}{2}\int_0^T |u_t|^2\,\mathrm{d}t + \int_0^T u_t\,\mathrm{d}B_t\right]. \tag{21}$$

A.2. Sticking-the-landing in Continuous Time

For a non-sequential latent variable model, the sticking-the-landing (STL) trick removes from the fully Monte Carlo ELBO estimator a score function term of the form $\partial \log q(w, \phi)/\partial\phi$, where w is sampled using the reparameterization trick and may depend on ϕ. The score function term has 0 expectation, but may affect the variance of the gradient estimator for the inference distribution's parameters.

Here, we exploit this intuition and apply it to each step before taking the limit. More precisely, we apply the STL trick to estimate the gradient of $D_{\mathrm{KL}}(q(w_{i+1}|w_i)||p(w_{i+1}|w_i))$ for $i = 1, 2, \ldots, N$, and thereafter take the limit as the mesh size of the discretization goes to 0. For each individual term, the score function term to be removed is

$$\begin{aligned}\frac{\partial}{\partial\phi}\log q(w_{i+1}|w_i,\phi) =& -\frac{1}{2\sigma^2(w_i)\Delta t}\frac{\partial}{\partial\phi}\left[(w_{i+1}-(w_i+f_q(w_i,\phi)\Delta t))^2\right]\\ =&\frac{\partial}{\partial\phi}\left[\frac{f_q(w_i,\phi)}{\sigma(w_i)}\right]\epsilon_{i+1}\Delta t^{1/2}.\end{aligned}$$

Now, we sum up all of these terms and take the limit as $\Delta t \to 0$. This gives us

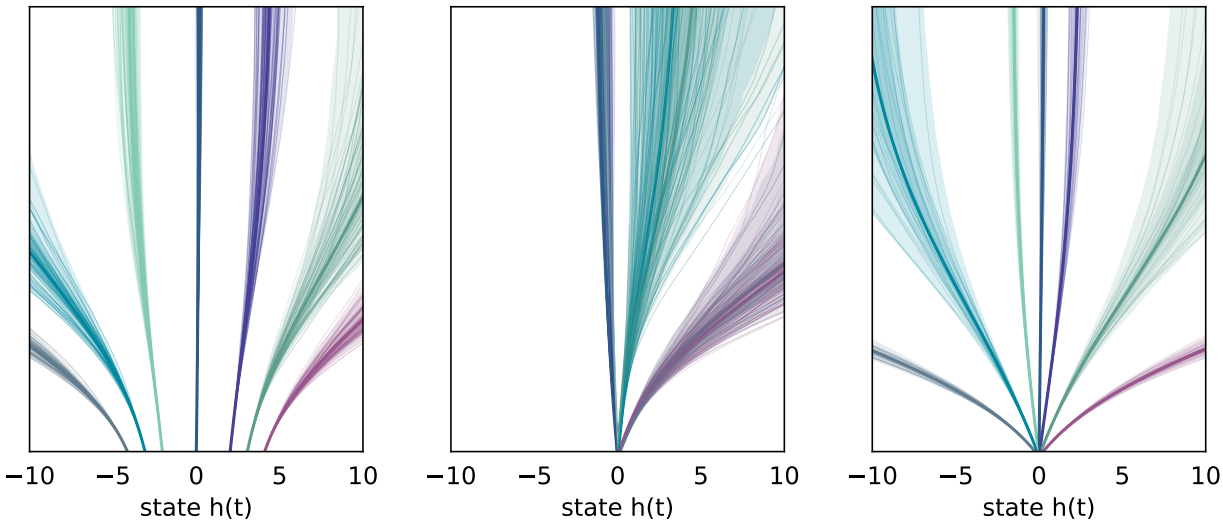
$$\begin{aligned}&\lim_{N\to\infty}\sum_{i=0}^{N}\mathbb{E}_{w_i}\left[\mathbb{E}_{w_{i+1}\sim q(w_{i+1}|w_i)}\left[\frac{\partial}{\partial\phi}\log q(w_{i+1}|w_i)\right]\right]\\ =&\lim_{N\to\infty}\sum_{i=0}^{N}\mathbb{E}_{w_i}\left[\mathbb{E}_{\epsilon_{i+1}}\left[\frac{\partial}{\partial\phi}\left[\frac{f_q(w_i,\phi)}{\sigma(w_i)}\right]\epsilon_{i+1}\Delta t^{1/2}\right]\right]\\ =&\,\mathbb{E}\left[\int_0^T\frac{\partial}{\partial\phi}\left[\frac{f_q(w_t,\phi)}{\sigma(w_t)}\right]\mathrm{d}B_t\right]\\ =&\,\mathbb{E}\left[\int_0^T\frac{\partial}{\partial\phi}\left[u_t\right]\mathrm{d}B_t\right].\end{aligned}$$

Removing this term from the fully Monte Carlo estimator in equation 21 gives rise to the following estimator of a surrogate objective that facilitates implementation:

$$\begin{aligned}\widehat{\mathrm{ELBO}} =& \log p(\mathcal{D}\mid w) - \int_{t_0}^{t_1}\frac{1}{2}\left\|u(w_t,t,\phi)\right\|_2^2\,\mathrm{d}t\\ &-\int_{t_0}^{t_1}u(w_t,t,\texttt{stop\_gradient}(\phi))\,\mathrm{d}B_t,\quad w(\cdot)\sim q_\phi().\end{aligned}$$

A.3. Additional Figures

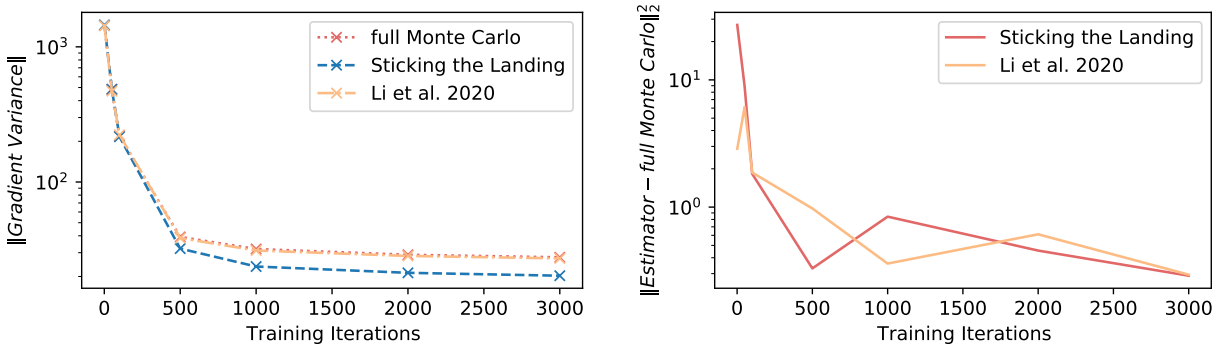
A.3.1. Augmentation in Differential Equation Models



(a) Samples from non-augmented dimension (b) from 2nd augmented dimension (c) from last augmented dimension

Figure 9. Example flows sampled from learned SDE dynamics. All continuous-depth models were trained by augmenting the state by 2 dimensions, refer to Figure 1 for main results. *Left:* The SDE-BNN learns meaningful parameterizations on the non-extraneous dimensions of the input state vector. In the case of a true function being monotonic, the augmented dimensions simply help the main output. *Middle:* The model learns to ignore dimensions that are not necessary to train on, especially on simpler tasks as in the toy setting. Samples in augmented dimensions can overlap for different input values in the given domain $(-5, 5)$. *Right:* Similarly, the last output dimension was also associated with augmentation and was not a well learned representation of the data, ignoring the initial inputs entirely (all values are 0).

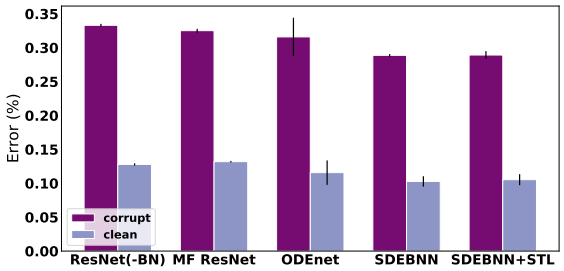
A.3.2. Variance of the STL Estimator



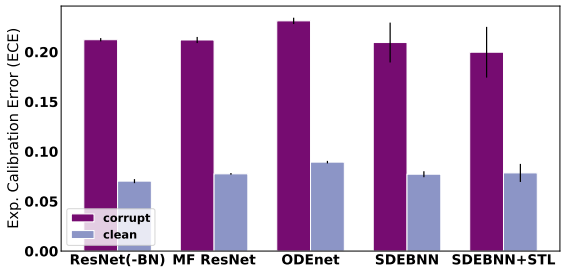
(a) Norm of the standard deviation (b) Relative mean difference

Figure 10. The Sticking the landing (STL) trick drops the expectation of the gradient of the score function, optimizing the variational objective with just the path derivative while remaining an unbiased estimator. The variance of the STL estimator consistently approaches 0 as training converges. The relative difference uses the mean of the full Monte Carlo estimator as reference and computes the norm of the difference between the other two mean variances (STL and Li et al. (2020)) from it. For estimation of these statistics, 5000 different samples were used on a fixed time discretization with step size $dt = 1e-2$. Similar results were confirmed taking 1000 and 3000 samples (not shown).

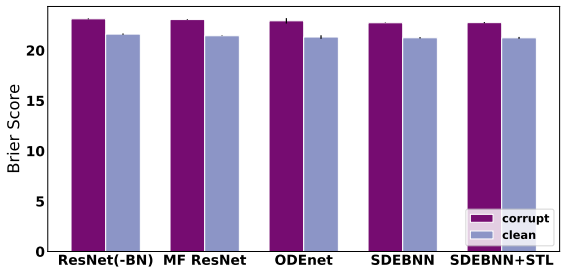
A.3.3. CALIBRATION



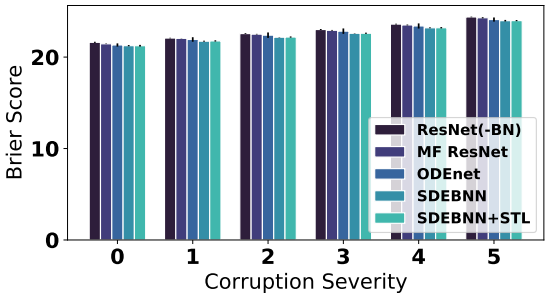
(a) Model vs Classification Error



(b) Model vs Expected Calibration Error



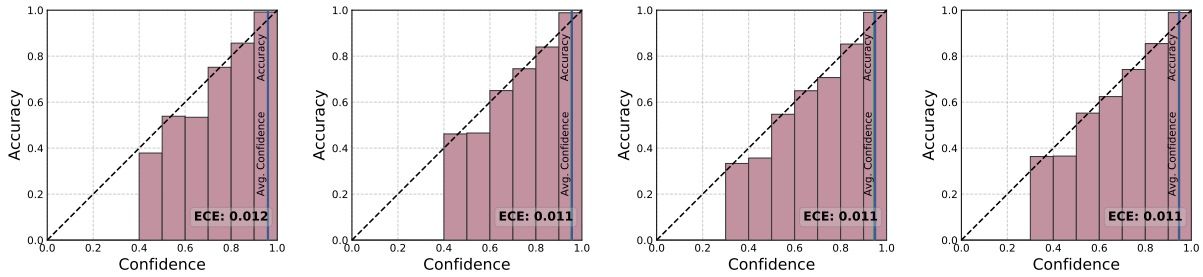
(c) Model vs Brier Score



(d) Corruption severity vs Brier Score

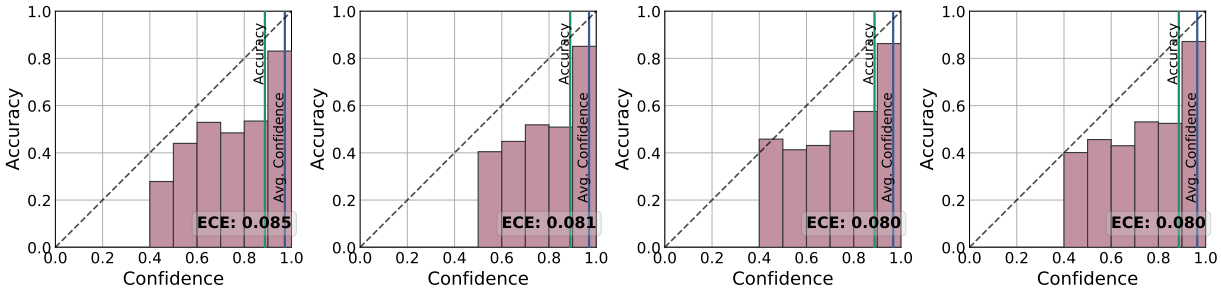
Figure 11. Figures 11a-11c show that the SDE BNN and SDE BNN + STL models outperform their non-continuous depth ResNet counterparts on all three robustness metrics when evaluated on the corrupt CIFAR-10C benchmarks. Figure 11d indicates that the accuracy of predictions is relatively consistent across all severity levels with the SDE-BNN and SDE-BNN + STL models having relatively better calibrated predictions.

A.3.4. Robustness to solver error at test time



(a) 160 steps (b) 176 steps (c) 192 steps (d) 208 steps

Figure 12. CIFAR10 image classification with a SDE-BNN. Better calibration can be obtained by increasing solver step sizes during inference without substantially changing the training error.



(a) 154 steps (b) 160 steps (c) 176 steps (d) 192 steps

Figure 13. CIFAR10 image classification with a SDE-BNN. Generalization improves marginally compared to a trained model during inference in 13b, as tuning solver step size does not yield significant differences in calibration outcomes.

A.3.5. Different SDE solver and adjoint settings

These were run with a SDE-BNN for MNIST image classification, to compare the performance and run-time cost across different solver settings. Comparably, backpropagation through the solver averaged 162.58 sec / epoch while the adjoint method averaged 135.90 sec / epoch in terms of wall clock time.

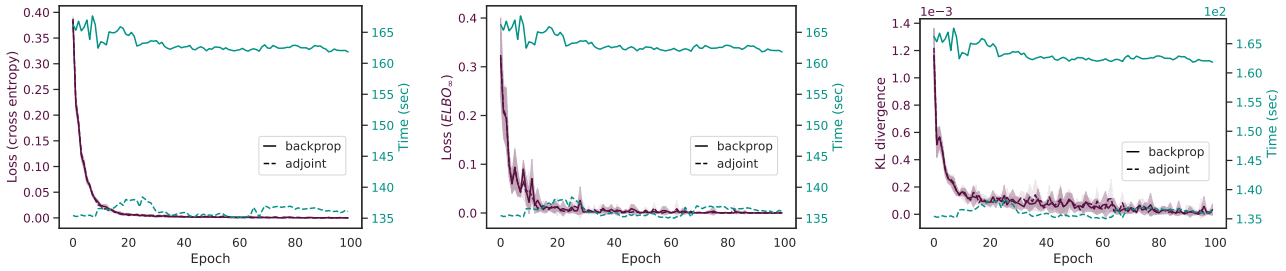


Figure 14. Backpropagation through the SDE solver yields similar optimization dynamics but is less time efficient than the adjoint method.

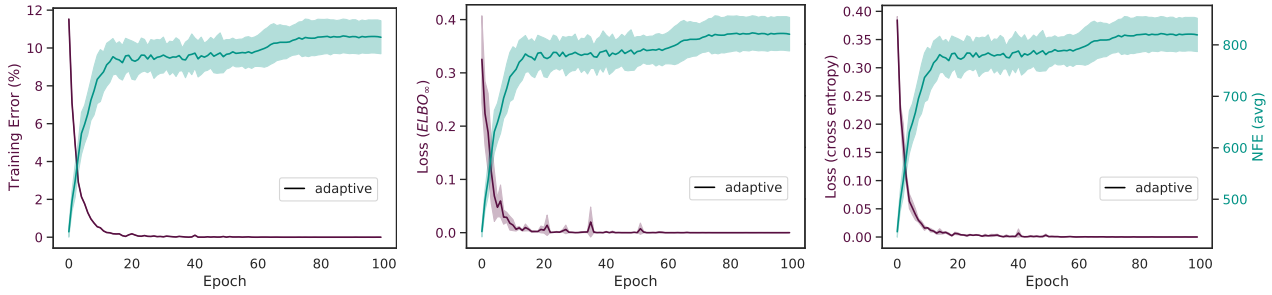


Figure 15. Trade-off between solver speed and convergence during training. Adaptive refers to training with the stochastic adjoint in both forward and reverse modes here.

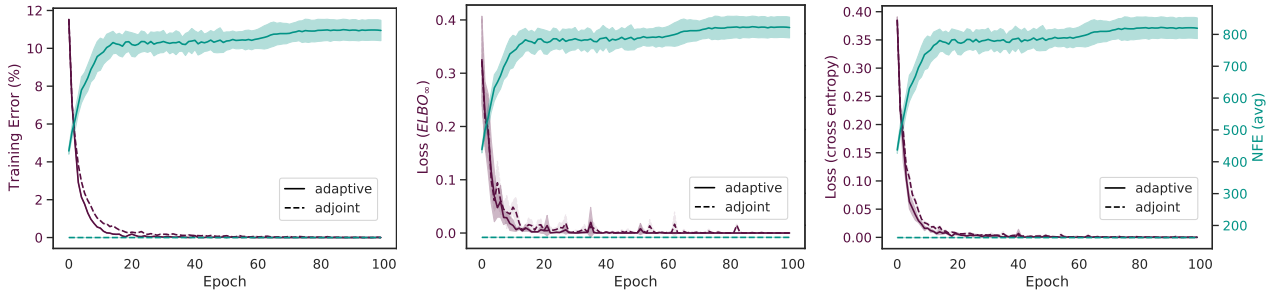


Figure 16. Trade-off between solver speed and precision during training. Adaptive-order optimization trajectories were comparable to fixed-order solvers and were thus not applied to the classification tasks since computational resources were not constrained.