What Are Bayesian Neural Network Posteriors Really Like?

Pavel Izmailov New York University
Sharad Vikram Google Research
Matthew D. Hoffman Google Research
Andrew Gordon Wilson New York University

Abstract

The posterior over Bayesian neural network (BNN) parameters is extremely high-dimensional and non-convex. For computational reasons, researchers approximate this posterior using inexpensive mini-batch methods such as mean-field variational inference or stochastic-gradient Markov chain Monte Carlo (SGMCMC). To investigate foundational questions in Bayesian deep learning, we instead use full-batch Hamiltonian Monte Carlo (HMC) on modern architectures. We show that (1) BNNs can achieve significant performance gains over standard training; (2) a single long HMC chain can provide a comparable representation of the posterior to multiple shorter chains; (3) in contrast to recent studies, we find posterior tempering is not needed for near-optimal performance, with little evidence for a "cold posterior" effect; (4) BMA performance is robust to the choice of prior variance scale; (5) Bayesian neural networks show surprisingly poor generalization under domain shift; (6) while cheaper alternatives such as deep ensembles and SGMCMC methods can provide strong performance, they fail to match the predictive distributions attained by HMC.

1. Introduction

Over the last 25 years, there have been several strong arguments favouring a Bayesian approach to deep learning (e.g., MacKay, 1995; Neal, 1996; Blundell et al., 2015; Gal, 2016; Wilson & Izmailov, 2020). Bayesian inference for neural networks promises improved predictions, reliable uncertainty estimates, and principled model comparison, naturally supporting active learning, continual learning, and decision-making under uncertainty. The Bayesian deep learning community has designed multiple successful practical methods inspired by the Bayesian approach (Blundell et al., 2015; Gal & Ghahramani, 2016; Welling & Teh, 2011; Kirkpatrick et al., 2017; Maddox et al., 2019; Izmailov et al., 2019; Daxberger et al., 2020) with applications ranging from astrophysics (Cranmer et al., 2021) to automatic diagnosis of Diabetic Retinopathy (Filos et al., 2019), click-through rate prediction in advertising (Liu et al., 2017) and modeling of fluid dynamics (Geneva & Zabaras, 2020).

However, inference with modern neural networks is distinctly challenging. We wish to compute a Bayesian model average corresponding to an integral over a multi-million dimensional multi-modal posterior, with unusual topological properties like mode-connectivity (Garipov et al., 2018), under severe computational constraints.

There are therefore many unresolved questions about Bayesian deep learning practice. Variational procedures typically provide unimodal Gaussian approximations to the multimodal posterior. Practically successful methods such as deep ensembles (Lakshminarayanan et al., 2017) have a natural Bayesian interpretation (Wilson & Izmailov, 2020), but only represent modes of the posterior. While Stochastic MCMC (Welling & Teh, 2011; Chen et al., 2014; Zhang et al., 2020b) is computationally convenient, it could be providing heavily biased estimates of posterior expectations. Moreover, Wenzel et al. (2020) question the quality of standard Bayes posteriors, citing results where "cold posteriors", raised to a power $1/T$ with $T < 1$, improve performance.

Additionally, Bayesian deep learning methods are typically evaluated on their ability to generate useful, well-calibrated predictions on held-out or out-of-distribution data. However, strong performance on benchmark problems does not imply that the algorithm accurately approximates the true Bayesian model average (BMA).

In this paper, we investigate fundamental open questions in Bayesian deep learning, using multi-chain full-batch Hamiltonian Monte Carlo (HMC, Neal et al., 2011). HMC is a highly-efficient and well-studied Markov Chain Monte Carlo (MCMC) method that is guaranteed to asymptotically produce samples from the true posterior. However it is enormously challenging to apply HMC to modern neural networks due to its extreme computational requirements: HMC can take *tens of thousands of training epochs* to produce a single sample from the posterior. To address this computational challenge, we parallelize the computation over hundreds of Tensor processing unit (TPU) devices.

We argue that full-batch HMC provides the most precise

tool for studying the Bayesian posterior to date. Using our implementation of HMC, we explore questions about the posterior geometry, performance of BNNs, effect of priors and posterior temperature.

In particular, we show: (1) BNNs can achieve significant performance gains over standard training on a range of regression, image, and text classification tasks; (2) a single long HMC chain can provide a comparable representation of the posterior to multiple shorter chains; (3) in contrast to recent studies, we find posterior tempering is not needed for near-optimal performance, with little evidence for a "cold posterior" effect; (4) BMA performance is robust to the choice of prior variance scale; (5) Bayesian neural networks generalize surprisingly poorly under data distribution shift, such as noise and corruptions in the data; (6) we study the predictive distributions of the cheaper alternatives to HMC such as deep ensembles and stochastic gradient MCMC methods and find that while these methods can perform well, they make very different predictions compared to HMC.

We additionally show how to effectively deploy full batch HMC on modern neural networks, including insights about how to tune crucial hyperparameters for good performance, and parallelize sampling over hundreds of TPUs. We will make our HMC samples publicly available, to make it easier for other researchers to explore fundamental questions about inference in Bayesian deep learning.

2. Background

In this section we provide background material on Bayesian deep learning and MCMC methods.

Bayesian neural networks In classical machine learning, the goal of learning is typically to find a single best setting of the parameters for the model (usually through maximum-likelihood optimization). In the Bayesian framework, the learner instead comes up with a *posterior* distribution $p(w|\mathcal{D})$ over the parameters w of the model after observing the data $\mathcal{D}$. The posterior distribution is given by Bayes' rule: $p(w|\mathcal{D}) \propto p(\mathcal{D}|w)p(w)$, where $p(\mathcal{D}|w)$ is the likelihood of $\mathcal{D}$ given by the model with parameters w, and $p(w)$ is the prior distribution over the parameters. The predictions of the model on a new test example x are then given by the *Bayesian model average* (BMA)

$$p(y|x, \mathcal{D}) = \int_w p(y|x, w)p(w|\mathcal{D})dw, \qquad (1)$$

where $p(y|x, w)$ is the predictive distribution for a given value of the parameters w. In Bayesian deep learning, the distribution $p(y|x, w)$ is the predictive distribution of a neural network with parameters w on an input x; in this setting we aim to compute a posterior distribution over the parameters of the network, and average the predictions over this distribution according to Eq. (1). Unfortunately, the posterior distribution $p(w|\mathcal{D})$ is intractable for neural networks due to the complex functional form of $p(y|x, w)$. Hence, approximate inference methods are required to estimate the integral in Eq. (1). For a detailed discussion of Bayesian deep learning, see e.g. Wilson & Izmailov (2020).

Markov Chain Monte Carlo The integral in Eq. (1) can be approximated by sampling: $p(y|x, \mathcal{D}) \approx \frac{1}{M}\sum_{i=1}^{M} p(y|x, w_i)$, where $w_i \sim p(w|\mathcal{D})$ are samples drawn from the posterior. MCMC methods construct a Markov chain that, if simulated for long enough, generates approximate samples from the posterior. In this work, we focus on Hamiltonian Monte Carlo (Neal et al., 2011), a method that produces asymptotically exact samples assuming access to the unnormalized posterior density $p(\mathcal{D}|w)p(w)$ and its gradient.

3. Related work

The bulk of work on Bayesian deep learning has focused on scalable approximate inference methods. Researchers proposed multiple successful methods based on stochastic variational inference (Hoffman et al., 2013; Graves, 2011; Blundell et al., 2015; Hernández-Lobato & Adams, 2015; Kingma et al., 2015; Molchanov et al., 2017; Louizos & Welling, 2017; Khan et al., 2018; Zhang et al., 2018; Wu et al., 2018; Dusenberry et al., 2020), dropout (Srivastava et al., 2014; Gal & Ghahramani, 2016; Kendall & Gal, 2017; Gal et al., 2017), Laplace approximation (MacKay, 1992; Kirkpatrick et al., 2017; Ritter et al., 2018; Li, 2000) and leveraging the stochastic gradient descent (SGD) trajectory (Mandt et al., 2017; Maddox et al., 2019; Izmailov et al., 2018; Wilson & Izmailov, 2020). While these (and many other) methods often provide improved predictions or uncertainty estimates, to the best of our knowledge *none* of these methods have been directly evaluated on their ability to match the *true* posterior distribution. Moreover, these methods are often designed with train-time constraints in mind, to take roughly the same amount of compute as regular SGD training. In this work, we attempt to construct a posterior approximation of the highest possible quality, ignoring the practicality of the method.

The Monte Carlo literature for Bayesian neural networks has mainly focused on stochastic gradient-based methods (Welling & Teh, 2011; Ahn et al., 2014; Chen et al., 2014; Ma et al., 2015; Ahn et al., 2012; Ding et al., 2014; Zhang et al., 2020b) for computational efficiency reasons. These methods are fundamentally biased: (1) they omit the Metropolis-Hastings correction step, and (2) the noise from subsampling the data perturbs their stationary distribution. In particular, Betancourt (2015) argues that HMC is incompatible with data subsampling. Notably, Zhang et al. (2020a) recently proposed a second-order stochastic gradient MCMC method that is asymptotically exact.

Since the classic work of Neal (1996), there have been a few recent attempts at using full-batch HMC in BNNs (e.g.; Cobb & Jalaian, 2020; Wenzel et al., 2020). These studies tend to use relatively short trajectory lengths (generally not considering a number of leapfrog steps greater than 100), and tend to focus on relatively small datasets and/or networks. We on the other hand experiment with practical architectures and datasets and use up to 10^5 leapfrog steps per iteration to ensure good mixing.

Our work is aimed at *understanding* Bayesian deep learning, we attempt to study the properties of true Bayesian neural networks. In a similar direction, Wenzel et al. (2020) have recently explored the effect of the posterior temperature in Bayesian neural networks. We discuss their results in detail in Section 7, and provide our own exploration of the posterior temperature with a different result: we find that BNNs achieve strong performance at temperature 1 and do not require posterior tempering.

4. HMC for Deep Neural Networks

In this work, we use full-batch Hamiltonian Monte Carlo to sample the parameters of Bayesian neural networks. We summarize HMC in Appendix Algorithm 1 and Algorithm 2. Intuitively, HMC is simulating the dynamics of a particle sliding on the plot of the density function that we are trying to sample from.[1] In this section we discuss the effect of the hyper-parameters of HMC and how to set them for BNNs.

Implementation To scale HMC to modern neural network architectures and for datasets like CIFAR-10 and IMDB, we parallelize the computation over 512 TPUv3 devices[2] (Jouppi et al., 2020). We execute HMC in a single-program multiple-data (SPMD) configuration, wherein a dataset is sharded evenly over each of the devices and an identical HMC implementation is run on each device. Each device maintains a synchronized copy of the Markov chain state, where the full-batch gradients needed for leapfrog integration are computed using cross-device collectives. All experiments were implemented using JAX (Bradbury et al., 2018).

Neural Network Architectures In our evaluation, following Wenzel et al. (2020), we primarily focus on two architectures: ResNet-20-FRN and CNN-LSTM. ResNet-20-FRN is a residual architecture (He et al., 2016) of depth 20 with batch normalization layers (Ioffe & Szegedy, 2015) replaced with filter response normalization (FRN; Singh & Krishnan, 2020). Batch normalization makes the likelihood harder to interpret by creating dependencies between training examples, whereas the outputs of FRN layers are independent across inputs. We use Swish (SiLU) activations (Hendrycks & Gimpel, 2016; Elfwing et al., 2018; Ramachandran et al., 2017) instead of ReLUs to ensure smoothness of the posterior density surface; we found it to improve acceptance rates of HMC proposals without hurting the overall performance. The CNN-LSTM is a long-short term memory network (Hochreiter & Schmidhuber, 1997) adapted from Wenzel et al. (2020) without modifications.

Datasets and Data Augmentation In our main evaluations we use the CIFAR image classification datasets (Krizhevsky et al., 2014) and the IMDB dataset (Maas et al., 2011) for sentiment analysis. We do not use any data augmentation, both because the random augmentations introduce stochasticity into the evaluation of the posterior log-density and its gradient, and because the expected randomly perturbed log-likelihood does not have a clean interpretation as a valid likelihood function (Wenzel et al., 2020).

Ablation details For the ablations in this section we use the ResNet-20-FRN architecture on CIFAR-10 and CNN-LSTM on IMDB. We report details on hyper-parameters in **??**.

4.1. Trajectory length τ

The trajectory length parameter τ determines the length of the dynamics simulation on each HMC iteration. Effectively, it determines the correlation of subsequent samples produced by HMC. To suppress random-walk behavior and speed up mixing, we want the length of the trajectory to be relatively high. But increasing the length of the trajectory also leads to an increased computational cost: the number of evaluations of the gradient of the target density (evaluations of the gradient of the loss on the full dataset) is equal to the ratio τ/α of the trajectory length to the step size.

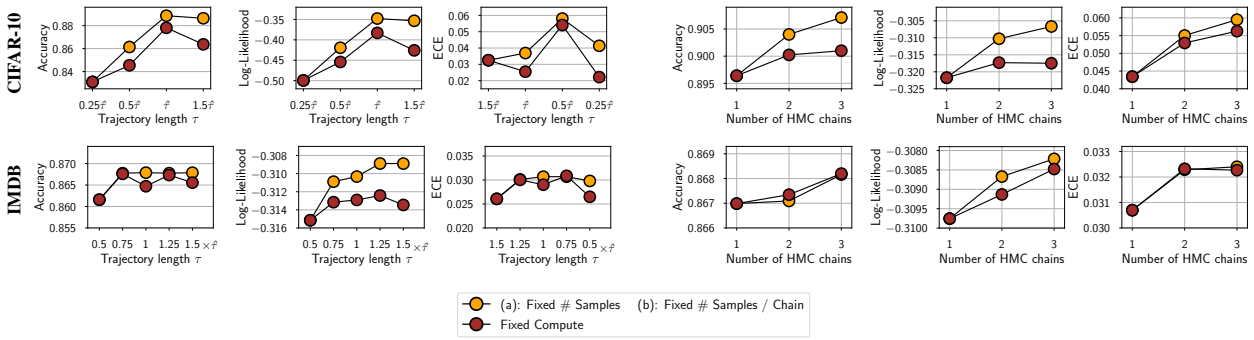
We suggest the following value of the trajectory length τ: $\hat{\tau} = \frac{\pi\sigma_{\text{prior}}}{2}$, where σ_{prior} is the standard deviation of the prior distribution over the parameters. If applied to a spherical Gaussian distribution, HMC with a small step size and this trajectory length will generate exact samples[3]. While we are interested in sampling from the posterior rather than from the spherical Gaussian prior, we argue that in large BNNs the prior tends to determine the scale of the posterior. We provide more detail and confirm this intuition empirically in the Appendix E.

In order to test the validity of our recommended trajectory length, we perform an ablation and report the results in Figure 1. As expected, longer trajectory lengths provide

[1] For a detailed introduction to HMC please see Neal et al. (2011). See also interactive visualization here.

[2] We use other hardware configurations in several experiments. We explicitly state the hardware that was used in the corresponding sections.

[3] Since the Hamiltonian defines a set of independent harmonic oscillators with period $2\pi\sigma$, $\tau = \pi\sigma/2$ applies a quarter-turn in phase space, swapping the positions and momenta.



(a) Trajectory length ablation

(b) Number of chains ablation

Figure 1. **Ablations of HMC hyper-parameters.** BMA accuracy, log-likelihood and expected calibration error (ECE) as a function of **(a)**: the trajectory length τ and **(b)**: number of HMC chains. The orange curve shows the results for a fixed number of samples in (a) and for a fixed number of samples per chain in (b); the brown curve shows the results for a fixed amount of compute. All experiments are done on CIFAR-10 using the ResNet-20-FRN architecture ond on IMDB using CNN-LSTM. Longer trajectory lengths decrease correlation between subsequent samples improving accuracy and log-likelihood. For a given amount of computation, increasing the number of chains from one to two modestly improves the accuracy and log-likelihood. For details of the experimental set-up, please see the Appendix.

better performance in terms of accuracy and log-likelihood. Expected calibration error is generally low across the board. The trajectory length $\hat{\tau}$ provides good performance in all three metrics. This result confirms that, despite the expense, when applying HMC to BNNs it is actually helpful to use tens of thousands of gradient evaluations per iteration.

4.2. Step size α

The step size parameter α determines the discretization step size of the Hamiltonian dynamics and consequently the number of leapfrog integrator steps. Lower step sizes lead to a better approximation of the dynamics and higher rates of proposal acceptance at the Metropolis-Hastings correction step. However, lower step sizes require more gradient evaluations per iteration to hold the trajectory length τ constant.

We run HMC for 50 iterations with step sizes of $1 \cdot 10^{-5}$, $5 \cdot 10^{-5}$, $1 \cdot 10^{-4}$, and $5 \cdot 10^{-4}$ respectively, ignoring the Metropolis-Hastings correction. We find the chains achieve average accept probabilities of 72.2%, 46.3%, 22.2%, and 12.5%, reflecting large drops in accept probability as step size is increased. We also observe BMA log-likelihoods of -0.331, -0.3406, -0.3407, and -0.895, indicating that higher accept rates result in higher likelihoods.

4.3. Number of HMC chains

We can improve the coverage of the posterior distribution by running multiple independent chains of HMC. Effectively, each chain is an independent run of the procedure using a different random initialization. Then, we combine the samples from the different chains. The computational requirements of running multiple chains are hence proportional to the number of chains.

We report the Bayesian model average performance as a function of the number of chains in Section 3. Holding compute budget fixed, using two chains is only slightly better than using one chain, and adding a third chain provides virtually no benefit. This result suggests that a single chain can produce a diverse set of samples without getting stuck in a single mode of the posterior — Section 5.1 provides further evidence for this hypothesis.

5. How well does HMC mix?

The primary goal of our paper is to construct accurate samples from the posterior, and use them to understand the properties of Bayesian neural networks better. In this section we consider several diagnostics to evaluate whether our HMC sampler has converged, and discuss their implications to the posterior geometry.

Summary: HMC is able to mix surprisingly well in function space, but not in parameter space. Geometrically, HMC is able to explore connected basins of the posterior with high functional diversity.

5.1. $\hat{R}$ diagnostics

In this section, we apply the classic Gelman et al. (1992) "$\hat{R}$" potential-scale-reduction diagnostic to our HMC runs. Given two or more chains, $\hat{R}$ estimates the ratio between the between-chain variance (i.e., the variance estimated by pooling samples from all chains) and the average within-chain variance (i.e., the variances estimated from each chain independently). The intuition is that, if the chains are stuck in isolated regions, then combining samples from multiple

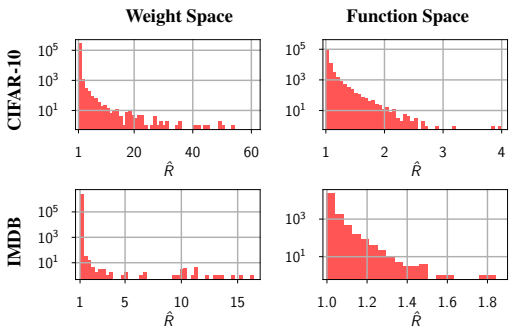


Figure 2. **Log-scale histograms of $\hat{R}$ convergence diagnostics.** Function-space $\hat{R}$s are computed on the test-set softmax predictions of the classifiers, weight-space $\hat{R}$s are computed on the raw weights. About 91% of CIFAR-10 and 98% of IMDB posterior-predictive probabilities get an $\hat{R}$ less than 1.1. Most weight-space $\hat{R}$ values are quite small, but enough parameters have very large $\hat{R}$s to make it clear that the chains are sampling from quite different metastable distributions in weight space.

chains will yield greater diversity than taking samples from a single chain. For the precise mathematical definition of $\hat{R}$, please see the Appendix D.

We compute $\hat{R}$ using TensorFlow Probability's implementation[4] (Lao et al., 2020) for both the weights and the test-set softmax predictions on the CIFAR-10 with ResNet-20-FRN and on IMDB with CNN-LSTM. We report the results in Figure 2. We observe that on both IMDB and CIFAR, the bulk of the function-space $\hat{R}$ values is concentrated near 1, meaning intuitively that a single chain can capture the diversity of predictions on most of the test data points nearly as well as multiple chains. The mixing is especially good on the IMDB dataset, where only 2% of inputs correspond to $\hat{R}$ larger than 1.1.

In weight space, although most parameters show no evidence of poor mixing, some have very large $\hat{R}$s, indicating that there are directions in which the chains fail to mix.

Implications for the Posterior Geometry The fact that a single HMC chain is able to mix well in prediction space suggests that the posterior contains connected regions which correspond to high functional diversity. Moreover, HMC is able to navigate these regions effeciently. Prior work on *mode connectivity* (Garipov et al., 2018; Draxler et al., 2018) has shown that there exist paths of high density connecting different modes of the posterior. Our observations suggest a stronger version of mode connectivity: not only do mode-connecting paths exist between functionally diverse modes, but also at least some of them can be leveraged by Monte Carlo methods to efficiently explore the posterior.

[4] tfp.mcmc.potential_scale_reduction

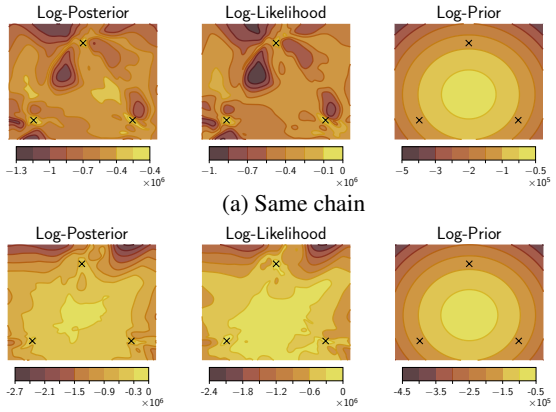


Figure 3. **Posterior density visualization.** Visualizations of posterior log-density, log-likelihood and log-prior in the two-dimensional subspace of the parameter space spanned by three HMC samples from **(a)** the same chain and **(b)** three independent chains. Each HMC chain explores a region of high posterior density of a complex non-convex shape, that appears multi-modal in the presented cross-sections.

5.2. Posterior density visualizations

To provide another angle on how HMC is able to explore the posterior over the weights, we visualize a cross-section of the posterior density in subspaces of the parameter space containing the samples. Following Garipov et al. (2018), we study two-dimensional subspaces of the parameter space of the form

$$\mathcal{S} = \{w | w = w_1 \cdot a + w_2 \cdot b + w_3 \cdot (1 - a - b)\}. \quad (2)$$

$\mathcal{S}$ is the unique two-dimensional affine subspace (plane) of the parameter space that includes parameter vectors w_1, w_2 and w_3.

In Figure 3 (a) we visualize the posterior log-density, log-likelihood and log-prior density of a ResNet-20-FRN on CIFAR-10. For the visualization, we use the subspace $\mathcal{S}$ defined by the parameter vectors w_1, w_{51} and w_{101}, the samples produced by HMC at iterations $1, 51$ and 101 after burn-in respectively. We observe that HMC is able to navigate complex geometry: the samples fall in three seemingly isolated modes in our two-dimensional cross-section of the posterior. In other words, HMC samples from a single chain are not restricted to any specific convex Gaussian-like mode, and instead explore a region of high posterior density of a complex shape in the parameter space.

Comparing the visualizations for samples from the same chain and samples from independent chains in Figure 3, we see that the shapes of the posterior surfaces are different, with the latter appearing more regular and symmetric. See Appendix C for additional visualizations and further discussion.

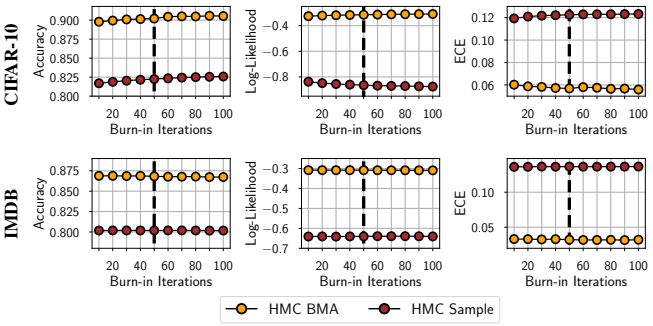


Figure 4. **HMC convergence.** The performance of an individual HMC sample and a BMA ensemble of 100 samples from each one of 3 HMC chains after the burn-in as a function of burn-in length. We use ResNet-20-FRN on CIFAR-10.

5.3. Convegence of the HMC chains

As another diagnostic, we look at the convergence of the performance of HMC BMA estimates and individual samples as a function of the length of the burn-in. We use the samples from 3 HMC chains, and evaluate performance of the ensemble of the first 100 HMC samples in each chain after discarding the first n_{bi} samples, where n_{bi} is the length of the burn-in. Additionally, we evaluate the performance of the individual HMC samples after n_{bi} iterations in each of the chains. We report the results in Figure 4.

6. Evaluating Bayesian Neural Networks

In this section we evaluate Bayesian neural networks in various problems using our implementation of HMC. Throughout the experiments, we use posterior temperature $T = 1$.

We emphasize that the main goal of our paper and this section in particular is *not* to argue for HMC as a practical method for Bayesian deep learning. Instead, we aim to *understand* the behavior of true Bayesian neural network posteriors using HMC as a precise tool.

Summary: Bayesian neural networks achieve strong performance competitive with large deep ensembles in a range of evaluations. Surprisingly, BNNs are *less* robust to distribution shift than conventionally-trained models.

6.1. Regression on UCI datasets

Bayesian deep learning methods are often evaluated on small-scale regression problems using fully connected networks (e.g., Wu et al., 2018; Izmailov et al., 2019; Maddox et al., 2019). Following these works, we evaluate Bayesian neural networks using HMC on five UCI regression datasets: Concrete, Yacht, Boston, Energy and Naval. For each of these datasets, we construct 20 random 90-to-10 train-test splits and report the mean and standard deviation of performance over the splits. We use a fully connected neural network with a single hidden layer of size 50 and 2 outputs representing the predictive mean and standard deviation. For HMC we used a single chain with 10 burn-in iterations and 90 iterations of sampling. For more details, please see Appendix A.

We report the results in Figure 5. HMC typically outperforms all the baselines, often by a significant margin, both in test RMSE and log-likelihood. On the Boston dataset, HMC achieves a slightly higher average RMSE compared to the subspace inference and SWAG (Izmailov et al., 2019; Maddox et al., 2019) but outperforms both these methods significantly in terms of log-likelihood.

6.2. Image Classification on CIFAR

Next, we evaluate Bayesian Neural Networks using HMC on image classification problems. We use the ResNet-20-FRN architecture on CIFAR-10 and CIFAR-100. For HMC we picked a random subset of 40960 of the 50000 images for each of the datasets to be able to evenly shard the data across the TPU devices; for the baselines, we report the performance on the full datasets. We run 3 HMC chains using step size 10^{-5} and a prior variance of $1/5$, resulting in 70,248 leapfrog steps per sample. In each chain we discard the first 50 samples as burn-in, and then draw 240 samples (720 in total for 3 chains)[5]. For SGLD, we use a single chain with 200 burn-in epochs and 4000 epochs of sampling producing 480 samples; we also report the performance of an ensemble of 5 independent SGLD chains. For more details, see Appendix A.

We report the performance of HMC, SGD, SGLD and a Deep Ensemble of 10 models in Figure 6. Bayesian neural networks based on HMC outperform SGD by a large margin on both CIFAR-10 and CIFAR-100. Perhaps surprisingly, SGLD and Deep Ensembles perform on par with HMC on all three metrics, at a fraction of computational cost.

Out-of-distribution detection Bayesian deep learning methods are often evaluated on out-of-distribution detection, due to their improved uncertainty calibration. In the Table 1 we report the performance of HMC-based Bayesian neural network on out-of-distribution (OOD) detection. To detect OOD data, we use the level of predicted confidence (value of the softmax class probability for the predicted class) from the HMC ensemble, measuring the area under the receiving operator characteristic curve (AUC-ROC). We find that BNNs perform competitively with the specialized ODIN method in the challenging near-OOD detection set-

[5] In total, on CIFAR-10 our HMC run is equivalent to *over 60 million training epochs* in terms of compute.

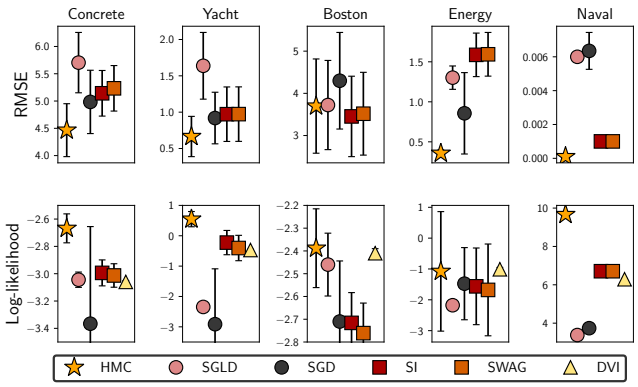


Figure 5. **UCI regression datasets.** Performance of Hamiltonian Monte Carlo (HMC), stochastic gradient Langevin dynamics (SGLD), stochastic gradient descent (SGD), subspace inference (SI) (Izmailov et al., 2019), SWAG (Maddox et al., 2019) and deterministic variational inference (DVI; Wu et al., 2018). The results reported for each method are mean and standard deviation computed over 20 random train-test splits of the dataset. For SI, SWAG and DVI we report the results presented in Izmailov et al. (2019). **Top:** test root-mean-squared error. **Bottom:** test log-likelihood. HMC performs on par with or better than all other baselines in each experiment, often providing a significant improvement.

ting (i.e. when the OOD data distribution is similar to the training data) of CIFAR-100, while underperforming in the easier far-OOD setting on SVHN relative to the baselines (Liang et al., 2017; Lee et al., 2018). For more details, please see Appendix.

Table 1. **Out-of-distribution detection.** We use a ResNet-20 model trained on CIFAR-10 to detect out-of-distribution data coming from SVHN or CIFAR-100. We also report the performance of specialized ODIN (Liang et al., 2017) and Mahalanobis (Lee et al., 2018) methods. HMC is competitive with ODIN on the harder near-OOD task of detecting CIFAR-100 images, but underperforms on the easier far-OOD task of detecting SVHN images.

	AUC-ROC			
OOD Dataset	HMC	DE	ODIN	Mahal.
CIFAR-100	0.857	0.853	0.858	0.882
SVHN	0.8814	0.8529	0.967	0.991

Robustness to distribution shift Bayesian methods are often specifically applied to covariate shift problems (Ovadia et al., 2019; Wilson & Izmailov, 2020; Dusenberry et al., 2020). We evaluate the the performance of HMC and Deep Ensemble-based Bayesian neural networks on the CIFAR-10-C dataset (Hendrycks & Dietterich, 2019), which applies a set of corruptions to CIFAR-10 with varying intensities. Mimicking the setup in Ovadia et al. (2019), we use the same 16 corruptions, evaluating the performance at all intensities.

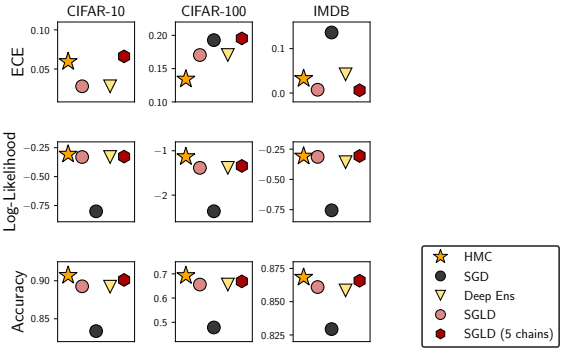


Figure 6. **Image and text classification.** Performance of Hamiltonian Monte Carlo (HMC), stochastic gradient Langevin dynamics (SGLD) with 1 and 5 chains, stochastic gradient descent (SGD), and Deep Ensembles. Bayesian neural networks via HMC consistently outperform SGD-based training and deep ensembles. SGLD is a strong baseline, often performing on par with HMC.

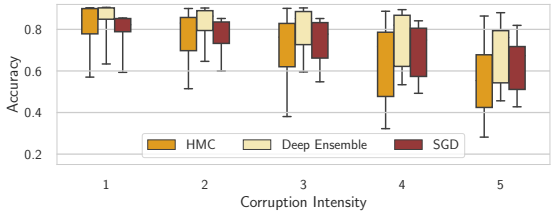


Figure 7. **Evaluation on CIFAR-10-C (Hendrycks & Dietterich, 2019).** Accuracy of HMC and Deep Ensembles on a distribution shift task, where CIFAR-10 is corrupted in 16 ways at various intensities. Boxes capture the quartiles of performance over each corruption, with the whiskers indicating the minimum and maximum. Deep Ensembles are consistently more robust to distribution shift than HMC-trained Bayesian neural networks.

Surprisingly, we find that Deep Ensembles are consistently more robust to distribution shift than HMC-based BNNs.

In Appendix B we provide a detailed study of this effect: it turns out that HMC samples are significantly less robust to certain types of noise than conventionally-trained SGD samples; this effect can be removed by sampling from the BNN posterior at a lower temperature.

6.3. Language Classification on IMDB

We use a CNN-LSTM architecture on the IMDB binary text classification dataset. We use HMC with a step size of 10^{-5} and a prior variance of $1/40$, resulting in 24,836 leapfrog steps per sample. We run 3 chains, burning-in for 50 samples, and drawing 400 samples per chain (1,200 total). We report the results in Figure 6. Analogously to the image classification experiments, HMC outperforms SGD and deep ensembles, while SGLD provides competitive performance at a fraction of the cost.

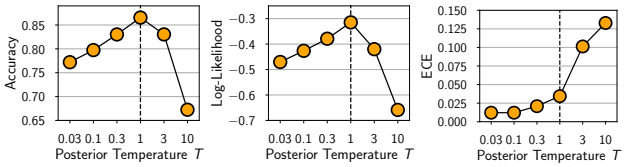


Figure 8. **Effect of posterior temperature.** The effect of posterior temperature T on the negative log-likelihood, accuracy and expected calibration error using the CNN-LSTM model on the IMDB dataset. For both the log-likelihood and accuracy $T = 1$ provides optimal performance, while for the ECE the colder posterior at $T = 3 \cdot 10^{-1}$ provides a slight improvement. For all three metrics, the posterior at $T = 1$ outperforms the SGD baseline as well as a deep ensemble of 10 independently trained models.

7. Do we need cold posteriors?

Multiple works have considered tempering the posterior in Bayesian neural networks (e.g. Wenzel et al., 2020; Wilson & Izmailov, 2020; Zhang et al., 2020b; Ashukha et al., 2020). Specifically, we can consider a distribution $p_T(w|\mathcal{D}) \propto \big(p(\mathcal{D}|w) \cdot p(w)\big)^{1/T}$, where w are the parameters of the network, $\mathcal{D}$ is the training dataset, $p(\mathcal{D}|w)$ is the likelihood of $\mathcal{D}$ for the network with parameters w and T is the *temperature*. Note that at temperature $T = 1$, p_T corresponds to the standard Bayesian posterior over the parameters of the network. Temperatures $T < 1$ correspond to *cold posteriors*, distributions that are sharper than the Bayesian posterior. Similarly, temperatures $T > 1$ correspond to *warm posteriors* which are softer than the Bayesian posterior. See Appendix Figure 12 (d) for a visualization of the log-likelihood density surface at different temperatures.

Wenzel et al. (2020) argue that Bayesian neural networks require a cold posterior, and the performance at temperature $T = 1$ is inferior to even a single model trained with SGD. The authors refer to this phenomenon as *the cold posteriors effect*. However, our results are different:

> **Summary**: We show that with an accurate posterior approximation, cold posteriors are not needed to obtain near-optimal performance with Bayesian neural networks and may even hurt performance.

7.1. Testing the cold posteriors effect

Wenzel et al. (2020) demonstrate the cold posteriors with two main experiments: ResNet-20 on CIFAR-10 and CNN-LSTM on IMDB. In these experiments the authors show poor performance at temperature $T = 1$, with strong benefits from decreasing the temperature. However, for the CIFAR-10 experiment, the authors show (Wenzel et al., 2020, Appendix K, Figure 28) that the results at $T = 1$ are near-optimal for the ResNet on CIFAR-10 if data augmentation is turned off and batch normalization is replaced with filter response normalization, which is in fact necessary to follow the pure Bayesian neural network model.

Furthermore, in Section 6, we show that Bayesian neural networks can achieve performance superior to SGD and even deep ensembles at temperature $T = 1$, in particular using the same ResNet-20-FRN model on CIFAR-10 and CNN-LSTM model on IMDB used by Wenzel et al. (2020).

To further understand the effect of posterior temperature T, we compare the performance of the CNN-LSTM model at different T using our Hamiltonian Monte Carlo sampler. In all runs we used a fixed prior variance $\sigma^2 = \frac{1}{40}$. We report the results in Figure 8.

We find that the performance of the BNN at $T = 1$ is better than the SGD baseline as well as a deep ensemble of 10 independent models. Moreover, the performance at $T = 1$ is better compared to all other temperatures we tested in terms of both test accuracy and log-likelihood.

Our results are in contrast with Wenzel et al. (2020), who argue that cold posteriors are needed for good performance with BNNs. In Appendix G we provide additional discussion of what may have caused the poor performance of BNNs in Wenzel et al. (2020), identifying their use of data augmentation as an important factor.

We also note that while posterior tempering does not seem necessary for good predictive performance with BNNs, it may be helpful under distribution shift. In Appendix B we show that decreasing the temperature can significantly improve the robustness of BNN predictions to noise in the test inputs.

8. Evaluating cheaper alternatives to HMC

> **Summary**: While SGLD and Deep Ensembles can provide similar performance to HMC, their predictive distributions differ significantly.

While HMC shows strong performance in our evaluation in Section 6, in most realistic BNN settings it is an impractical method. In this section we consider SGLD (Welling & Teh, 2011) and deep ensembles (Lakshminarayanan et al., 2017) as cheaper practical alternatives to HMC. We have seen that these methods can provide competitive performance to HMC, and in this section we explore the discrepancies between the predictions attained by these methods and HMC.

SGLD with a non-vanishing step size samples from a perturbed version of the posterior, both because it omits a Metropolis-Hastings accept-reject step and because its updates include minibatch noise. Both of these perturbations

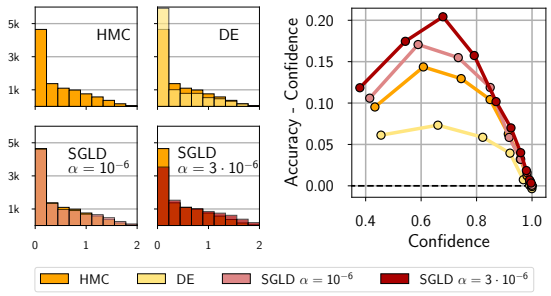


Figure 9. Distribution of predictive entropies (**left**) and calibration curve (**right**) of posterior predictive distributions for HMC, SGLD (with learning rates $\alpha = 10^{-6}$ and $3 \cdot 10^{-6}$), and deep ensemble on ResNet20-FRN on CIFAR-10. On the left, for all methods, except HMC we plot a pair of histograms: for HMC and for the corresdponding method. Deep ensembles provide more confident predictions than HMC, SGLD with high learning rate is undercondident, while SGLD with $\alpha = 10^{-6}$ matches HMC well.

should tend to make the entropy of SGLD's stationary distribution increase with its step size; we might expect this to translate to approximations to the BMA that are overdispersed. Figure 9 suggests that this is the case — while the HMC ensemble often makes predictions with high confidence (low entropy), the SGLD predictions are less likely to be highly confident, especially when using the larger step size of $3 \cdot 10^{-6}$. The deep ensemble makes even more confident predictions. All methods make conservative predictions: their confidences tend to underestimate their accuracies (Figure 9, right).

In Appendix H we additionally explore the calibration of HMC, deep ensembles and SGLD under distribution shift. Interestingly, the behavior of SGLD appears more similar to that of deep ensembles than that of HMC. So, while both SGLD and deep ensembles are very compelling practically, they do not provide the same predictions as HMC. Thus, we should be very careful when making judgements about *true* Bayesian neural networks based on the SGLD performance.

9. Conclusion

To the best of our knowledge, our work provides the first realistic evaluation of Bayesian neural networks with precise and exhaustive posterior sampling. Through using an exact sampler, we are able to established several properties of Bayesian neural networks which have not previously been demonstrated. In particular, we have shown that (1) BNNs can achieve performance competitive with deep ensembles, (2) do not suffer from the cold posterior effect and (3) are robust to the prior scale specification (see Appendix). A surprising negative result that we observed was that BNNs are *less* robust to domain shift than conventionally trained models. We hope that our observations and the tools that we develop will facilitate fundamental progress in understanding the behaviour of true Bayesian neural networks.

References

Ahn, S., Korattikara, A., and Welling, M. Bayesian posterior sampling via stochastic gradient fisher scoring. *arXiv preprint arXiv:1206.6380*, 2012.

Ahn, S., Shahbaba, B., and Welling, M. Distributed stochastic gradient mcmc. In *International conference on machine learning*, pp. 1044–1052. PMLR, 2014.

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

Betancourt, M. The fundamental incompatibility of hamiltonian monte carlo and data subsampling. *arXiv preprint arXiv:1502.01510*, 2015.

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

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Chen, T., Fox, E., and Guestrin, C. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pp. 1683–1691. PMLR, 2014.

Cobb, A. D. and Jalaian, B. Scaling hamiltonian monte carlo inference for bayesian neural networks with symmetric splitting. *arXiv preprint arXiv:2010.06772*, 2020.

Cranmer, M., Tamayo, D., Rein, H., Battaglia, P., Hadden, S., Armitage, P., Ho, S., and Spergel, D. N. A bayesian neural network predicts the dissolution of compact planetary systems. 2021.

Daxberger, E., Nalisnick, E., Allingham, J. U., Antorán, J., and Hernández-Lobato, J. M. Expressive yet tractable bayesian deep learning via subnetwork inference. *arXiv preprint arXiv:2010.14689*, 2020.

Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R., and Neven, H. Bayesian sampling using stochastic gradient thermostats. 2014.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. A. Essentially no barriers in neural network energy landscape. *arXiv preprint arXiv:1803.00885*, 2018.

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

Elfwing, S., Uchibe, E., and Doya, K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

Filos, A., Farquhar, S., Gomez, A. N., Rudner, T. G., Kenton, Z., Smith, L., Alizadeh, M., de Kroon, A., and Gal, Y. A systematic comparison of bayesian deep learning robustness in diabetic retinopathy tasks. *arXiv preprint arXiv:1912.10481*, 2019.

Gal, Y. *Uncertainty in deep learning*. PhD thesis, PhD thesis, University of Cambridge, 2016.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Gal, Y., Hron, J., and Kendall, A. Concrete dropout. *arXiv preprint arXiv:1705.07832*, 2017.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Neural Information Processing Systems*, 2018.

Gelman, A., Rubin, D. B., et al. Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4):457–472, 1992.

Geneva, N. and Zabaras, N. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403: 109056, 2020.

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

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869. PMLR, 2015.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging weights leads to wider optima and better generalization. In *Uncertainty in Artificial Intelligence (UAI)*, 2018.

Izmailov, P., Maddox, W. J., Kirichenko, P., Garipov, T., Vetrov, D., and Wilson, A. G. Subspace inference for Bayesian deep learning. *Uncertainty in Artificial Intelligence*, 2019.

Jouppi, N. P., Yoon, D. H., Kurian, G., Li, S., Patil, N., Laudon, J., Young, C., and Patterson, D. A domain-specific supercomputer for training deep neural networks. *Communications of the ACM*, 63(7):67–78, 2020.

Kendall, A. and Gal, Y. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pp. 5574–5584, 2017.

Khan, M. E., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. Fast and scalable Bayesian deep learning by weight-perturbation in adam. *arXiv preprint arXiv:1806.04854*, 2018.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. *arXiv preprint arXiv:1506.02557*, 2015.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Krizhevsky, A., Nair, V., and Hinton, G. The CIFAR-10 dataset. 2014. http://www.cs.toronto.edu/kriz/cifar.html.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017.

Lao, J., Suter, C., Langmore, I., Chimisov, C., Saxena, A., Sountsov, P., Moore, D., Saurous, R. A., Hoffman, M. D., and Dillon, J. V. tfp. mcmc: Modern markov chain monte carlo tools built for modern hardware. *arXiv preprint arXiv:2002.01184*, 2020.

Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pp. 7167–7177, 2018.

Li, D. X. On default correlation: A copula function approach. *Journal of Fixed Income*, 9(4):43–54, 2000.

Liang, S., Li, Y., and Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

Liu, X., Xue, W., Xiao, L., and Zhang, B. Pbodl: Parallel bayesian online deep learning for click-through rate prediction in tencent advertising system. *arXiv preprint arXiv:1707.00802*, 2017.

Louizos, C. and Welling, M. Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*, pp. 2218–2227. PMLR, 2017.

Ma, Y.-A., Chen, T., and Fox, E. B. A complete recipe for stochastic gradient mcmc. *arXiv preprint arXiv:1506.04696*, 2015.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

MacKay, D. J. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

MacKay, D. J. Probable networks and plausible predictions?a review of practical Bayesian methods for supervised neural networks. *Network: computation in neural systems*, 6(3):469–505, 1995.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. A simple baseline for Bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, 2019.

Mandt, S., Hoffman, M. D., and Blei, D. M. Stochastic gradient descent as approximate Bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907, 2017.

Molchanov, D., Ashukha, A., and Vetrov, D. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pp. 2498–2507. PMLR, 2017.

Neal, R. *Bayesian Learning for Neural Networks*. Springer Verlag, 1996. ISBN 0387947248.

Neal, R. M. et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J. V., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019.

Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Rasmussen, C. E. and Nickisch, H. Gaussian processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research (JMLR)*, 11:3011–3015, Nov 2010.

Ritter, H., Botev, A., and Barber, D. A scalable Laplace approximation for neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Singh, S. and Krishnan, S. Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11237–11246, 2020.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Tran, B.-H., Rossi, S., Milios, D., and Filippone, M. All you need is a good functional prior for bayesian deep learning. *arXiv preprint arXiv:2011.12829*, 2020.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Wenzel, F., Roth, K., Veeling, B. S., Świątkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. How good is the Bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*, 2020.

Wilson, A. G. and Izmailov, P. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.

Wu, A., Nowozin, S., Meeds, E., Turner, R. E., Hernández-Lobato, J. M., and Gaunt, A. L. Deterministic variational inference for robust Bayesian neural networks. *arXiv preprint arXiv:1810.03958*, 2018.

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

Zhang, R., Cooper, A. F., and De Sa, C. Amagold: Amortized metropolis adjustment for efficient stochastic gradient mcmc. In *International Conference on Artificial Intelligence and Statistics*, pp. 2142–2152. PMLR, 2020a.

Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. Cyclical stochastic gradient MCMC for Bayesian deep learning. In *International Conference on Learning Representations*, 2020b.

Method	Dataset Architecture	Experiments CIFAR-10 ResNet-20-FRN	 CIFAR-100 ResNet-20-FRN	 IMDB CNN LSTM
HMC	Prior Variance	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{40}$
	Step Size	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$
	Metropolis-Hastings Burnin	10	10	10
	Num. Burnin	50	50	50
	Num. Samples per Chain	240	40	400
	Num. of Chains	3	3	3
	Total Samples	720	120	1200
	Total Epochs	$5 \cdot 10^7$	$8.5 \cdot 10^6$	$3 \cdot 10^7$
SGLD	Prior Variance	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$
	Step Size	10^{-6}	$3 \cdot 10^{-6}$	$1 \cdot 10^{-5}$
	Step Size Schedule	constant	constant	constant
	Batch Size	80	80	80
	Num. Epochs	10000	10000	10000
	Num. Burnin	1000	1000	1000
	Num. Samples per Chain	900	900	900
	Num. of Chains	5	5	5
	Total Samples	4500	4500	4500
	Total Epochs	$5 \cdot 10^4$	$5 \cdot 10^4$	$5 \cdot 10^4$
SGD	Weight Decay	10	10	3
	Initial Step Size	$3 \cdot 10^{-7}$	$1 \cdot 10^{-6}$	$3 \cdot 10^{-7}$
	Step Size Schedule	cosine	cosine	cosine
	Batch Size	80	80	80
	Num. Epochs	500	500	500
	Momentum	0.9	0.9	0.9
	Total Epochs	$5 \cdot 10^2$	$5 \cdot 10^2$	$5 \cdot 10^2$
Deep Ensembles	Num. Models	50	50	50
	Total Epochs	$2.5 \cdot 10^4$	$2.5 \cdot 10^4$	$2.5 \cdot 10^4$

Table 2. **Hyper-parameters for CIFAR and IMDB.** We report the hyper-parameters for each method our main evaluations in Section 6. For each method we report the total number of training epochs equivalent to the amount of compute spent. We run HMC on a cluster of 512 TPUs, and the baselines on a cluster of 8 TPUs.

Appendix Outline

This appendix is organized as follows. We present the Hamiltonian Monte Carlo algorithm that we implement in the paper in Algorithm 1, Algorithm 2. In Appendix A we provide the hyper-parameters used in our experiments. In Appendix B we show that BNNs are not robust to distribution shift and discuss the reasons for this behavior. In Appendix C we provide additional posterior density surface visualizations. In Appendix D we provide a description of the $\hat{R}$ statistic used in Section 5.1. In Appendix E we study the marginals of the posterior over parameters estimated by HMC. In Appendix F we explore the effect of the prior in Bayesian neural networks. In Appendix G we provide a further discussion of the effect of posterior temperature. Finally, in Appendix H we discuss the differences between the predictions of HMC and cheaper alternatives.

A. Hyper-Parameters and Details

CIFAR and IMDB. In Table 2 we report the hyperparameters used by each of the methods in our main evaluation in Section 6. HMC was run on a cluster of 512 TPUs and the other baselines were run on a cluster of 8 TPUs. We tuned the hyper-parameters for all methods via cross-validation. For most HMC hyper-parameters, we provide ablations illustrating their effect in Section 4. Producing a single sample on CIFAR datasets takes roughly one hour on our hardware, and on IMDB it takes 105 seconds; we can run up to three chains in parallel.

Temperature scaling on IMDB. For the experiments in Section 7 we run a single HMC chain producing 40 samples after 10 burn-in epochs for each temperature. We used stepsizes $5 \cdot 10^{-5}$, $3 \cdot 10^{-5}$, 10^{-5}, $3 \cdot 10^{-6}$, 10^{-6} and $3 \cdot 10^{-7}$ for temperatures 10, 3, 1, 0.3, 0.1 and 0.03 respectively, ensuring that the accept rates were close to 100%. We used a prior variance of $1/50$ in all experiments. We ran the experiments on 8 NVIDIA Tesla V-100 GPUs, as we found that sampling at low temperatures requires `float64` precision which is not supported on TPUs.

UCI Datasets. For HMC we run a single chain for 100 steps, discarding the first 10 samples as burn-in. On Concrete, Yacht, Energy and Boston we use a prior scale of $1/10$ and a step-size of 10^{-5}; on Naval we use a prior scale of $1/40$ and a step-size of $5 \cdot 10^{-7}$. We run each experiment on a single NVIDIA Tesla V-100 GPU. For the SGD and SGLD baselines, we tuned the weight decay and step size and number of training epochs individually for each dataset.

Algorithm 1 Hamiltonian Monte Carlo

Input: Trajectory length τ, number of burn-in interations N_{burnin}, initial parameters w_{init}, step size α, unnormalized posterior log-density function $f(w) = \log p(w|D) + \log Z$.
Output: Set S of samples w of the parameters.

$w \leftarrow w_{\text{init}}$; $\quad N_{\text{leapfrog}} \leftarrow \frac{\tau}{\alpha}$;
# Burn-in stage
for $i \leftarrow 1 \dots N_{\text{burnin}}$ **do**
  $m \sim \mathcal{N}(0, I)$;
  $(w, m) \leftarrow \text{Leapfrog}(w, m, \alpha, N_{\text{leapfrog}}, f)$;
end for
# Sampling
for $i \leftarrow 1 \dots K$ **do**
  $m \sim \mathcal{N}(0, I)$;
  $(w', m') \leftarrow \text{Leapfrog}(w, m, \alpha, N_{\text{leapfrog}}, f)$;

  # Metropolis-Hastings correction
  $p_{\text{accept}} \leftarrow \min\left\{1, \frac{f(w')}{f(w)} \cdot \exp\left(\frac{1}{2}\|m\|^2 - \|m'\|^2\right)\right\}$;
  $u \sim \text{Uniform}[0, 1]$;
  if $u \leq p_{\text{accept}}$ **then**
    $w \leftarrow w'$;
  end if
  $S \leftarrow S \cup \{w\}$;
end for

Algorithm 2 Leapfrog integration

Input: Parameters w_0, initital momentum m_0, posterior log-density function $f(w) = \log p(w|D)$, number of leapfrog steps N_{leapfrog}, step size α.
Output: New parameters w; new momentum m.

$w \leftarrow w_0$; $\quad m \leftarrow m_0$;
for $i \leftarrow 1 \dots N_{\text{leapfrog}}$ **do**
  $m \leftarrow m + \frac{\alpha}{2} \cdot \nabla f(w)$;
  $w \leftarrow w + \alpha \cdot m$;
  $m \leftarrow m + \frac{\alpha}{2} \cdot \nabla f(w)$;
end for
$\text{Leapfrog}(w_0, m_0, \alpha, N_{\text{leapfrog}}, f) \leftarrow (w, m)$

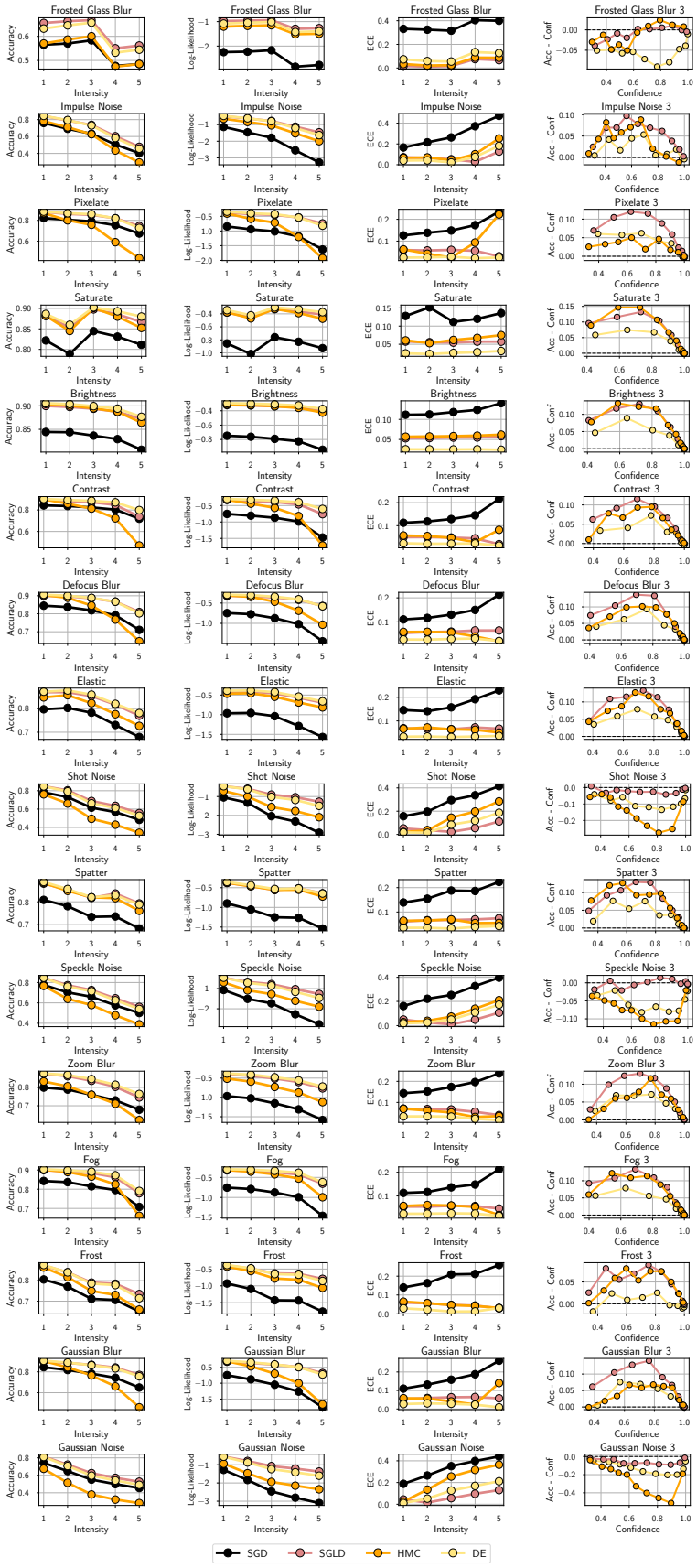


Figure 10. **Performance under corruption.** Columns 1-3 show accuracy, log-likelihood and ECE of SGD, SGLD with learning rate $\alpha = 10^{-6}$, HMC and Deep Ensembles for all 16 CIFAR-10-C corruptions as a function of corruption intensity. The last column shows the calibration diagram for each corruption at intensity 3 out of 5; we omit the calibration diagram for SGD which is typically extremely overconfident for readability. HMC shows poor accuracy on most of the corruptions with a few exceptions. The calibration of the uncertainties provided by HMC is generally reasonable, although typically inferior to deep ensembles.

B. BNNs are not Robust to Domain Shift

In Section 6.2 we have seen (Figure 7) that surprisingly BNNs via HMC underperform significantly on corrupted data from CIFAR-10-C compared to deep ensembles and even SGD. We provide detailed results in Figure 10. HMC shows surprisingly poor robustness in terms of accuracy across the corruptions. In most cases, the HMC ensemble of 720 models underperforms to a single SGD solution! The uncertainty estimates provided by HMC are more reasonable: HMC outperforms SGD on the log-likelihoods and calibration in most cases; deep ensembles and SGLD provide better likelihoods and calibration.

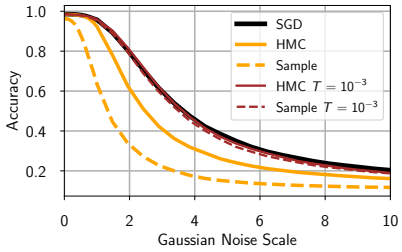
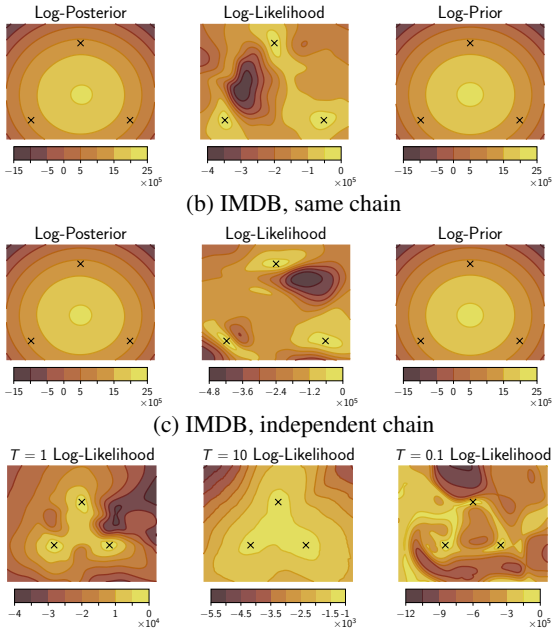


Figure 11. **Robustness on MNIST.** Performance of SGD, BMA ensembles and individual samples constructed by HMC at temperatures $T = 1$ and $T = 10^{-3}$ on the MNIST test set corrupted by Gaussian noise. Temperature 1 HMC shows very poor robustness, while lowering the temperature allows us to close the gap to SGD.

The poor performance of HMC on OOD data is surprising. Bayesian methods average the predictions over multiple models for the data, and faithfully represent uncertainty. Hence, Bayesian deep learning methods are expected to be robust to noise in the data, and often explicitly evaluated on CIFAR-10-C (e.g. Wilson & Izmailov, 2020; Dusenberry et al., 2020).

To further understand the robustness results, we reproduce the same effect on a small fully-connected network with two hidden layers of width 256 on MNIST. We run HMC at temperatures $T = 1$ and $T = 10^{-3}$ and SGD and report the results for both the BMA ensembles and individual samples in Figure 11. For all methods, we train the models on the original MNIST training set, and evaluate on the test set with random Gaussian noise $\mathcal{N}(0, \sigma^2 I)$ of varying scale σ. We report the test accuracy as a function of σ. We find that while the performance on the original test set is very close for all methods, the accuracy of HMC at $T = 1$ drops much quicker compared to that of SGD as we increase the noise scale.

Notably, the individual sample performance of $T = 1$ HMC is especially poor compared to SGD. For example, at noise scale $\sigma = 3$ the SGD accuracy is near 60% while the HMC sample only achieves around 20% accuracy!



(b) IMDB, same chain

(c) IMDB, independent chain

(d) IMDB, Log-Likelihood at different T

Figure 12. **Additional posterior density visualizations.** Visualizations of posterior log-density, log-likelihood and log-prior in two-dimensional subspaces of the parameter space spanned by three HMC samples in different settings. **(a):** samples from independent chains on CIFAR-10 (see Figure 3 for samples from the same chain); **(b):** samples from the same chain and **(c)**: independent chains on IMDB; **(d):** Log-likelihood surfaces for samples from the same chain at posterior temperatures $T = 1$, 10 and 0.1.

HMC can be though of as sampling points at a certain suboptimal level of the training loss, significantly lower than that of SGD solutions. As a result, HMC samples are individually inferior to SGD solutions. On the original test data ensembling the HMC samples leads to strong performance significantly outperforming SGD (see Section 6). However, as we apply noise to the test data, ensembling can no longer close the gap to the SGD solutions. To confirm this explanation, we run evaluate HMC at a very low temperature $T = 10^{-3}$. We find that at this temperature, HMC performs comparably with SGD, closing the gap in robustness[6].

We expect that using a lower temperature with HMC would also significantly improve robustness on CIFAR-10-C. Verifying this hypothesis, and generally understanding the robustness of BNNs further is an exciting direction of future work.

[6]We have also experimented with varying the prior scale but were unable to close the gap in robustness at temperature $T = 1$.

C. Additional Posterior Visualizations

In Section 5.2 we study two-dimensional cross-sections of posterior log-density, log-likelihood and log-prior surfaces. We provide additional visualizations in Figure 12.

On IMDB, the posterior log-density is dominated by the prior, and the corresponding panels are virtually indistinguishable in Figure 12 (b), (c). For the CNN-LSTM on IMDB the number of parameters is much larger than the number of data points, and hence the scale of the prior density values is much larger than the scale of the likelihood. Note that the likelihood still affects the posterior typical set, and the HMC samples land in the modes of the likelihood in the visualization. In contrast, on ResNet-20, the number of parameters is smaller and the number of data points is larger, so the posterior is dominated by the likelihood in Figure 12 (a). The log-likelihood panels for both datasets show that HMC is able to navigate complex geometry: the samples fall in three isolated modes in our two-dimensional cross-sections. On IMDB, the visualizations for samples from a single chain and for samples from three independent chains are qualitatively quite similar, hinting at better parameter-space mixing compared to CIFAR-10 (see Section 5.1).

In Figure 12 (d), we visualize the likelihood cross-sections using our runs with varying posterior temperature on IMDB. The visualizations show that, as expected, low temperature leads to a sharp likelihood, while the high-temperature likelihood appear soft. In particular, the scale of the lowest likelihood values at $T = 10$ is only 10^3 while the scale at $T = 0.1$ is 10^6.

D. Description of $\hat{R}$ Statistics

$\hat{R}$ (Gelman et al., 1992) is a popular MCMC convergence diagnostic. It is defined in terms of some scalar function $\psi(\theta)$ of the Markov chain iterates $\{\theta_{mn} | m \in \{1, \ldots, M\}, n \in \{1, \ldots, N\}\}$, where θ_{mn} denotes the state of the mth of M chains at iteration n of N. Letting $\psi_{mn} \triangleq \psi(\theta_{mn})$, $\hat{R}$ is defined as follows:

$$\bar{\psi}_{m\cdot} \triangleq \frac{1}{N}\sum_{n}\psi_{mn}; \quad \bar{\psi}_{\cdot\cdot} \triangleq \frac{1}{MN}\sum_{m,n}\psi_{mn}; \tag{3}$$

$$\frac{B}{N} \triangleq \frac{1}{M-1}\sum_{m}(\bar{\psi}_{m\cdot} - \bar{\psi}_{\cdot\cdot})^2; \tag{4}$$

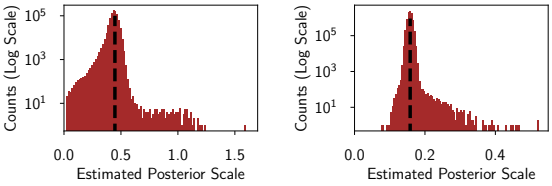
$$W \triangleq \frac{1}{M(N-1)}\sum_{m,n}(\psi_{mn} - \bar{\psi}_{m\cdot})^2; \tag{5}$$

$$\hat{\sigma}_+^2 \triangleq \frac{N-1}{N}W + \frac{B}{N}; \tag{6}$$

$$\hat{R} \triangleq \frac{M+1}{M}\frac{\hat{\sigma}_+^2}{W} - \frac{N-1}{MN}. \tag{7}$$

If the chains were initialized from their stationary distribution, then $\hat{\sigma}_+^2$ would be an unbiased estimate of the stationary distribution's variance. W is an estimate of the average within-chain variance; if the chains are stuck in isolated regions, then W should be smaller than $\hat{\sigma}_+^2$, and $\hat{R}$ will be clearly larger than 1. The $\frac{M+1}{M}$ and $\frac{N-1}{MN}$ terms are there to account for sampling variability—they vanish as N gets large if W approaches $\hat{\sigma}_+^2$.

Since $\hat{R}$ is defined in terms of a function of interest ψ, we can compute it for many such functions. In the main text we evaluated it for each weight and each predicted softmax probability in the test set.



(a) ResNet-20-FRN (b) CNN-LSTM

Figure 13. **Log-scale histograms of estimated marginal posterior standard deviations for ResNet20-FRN on CIFAR-10 and CNN-LSTM on IMDB.** These histograms show how many parameters have empirical standard deviations that fall within a given bin.

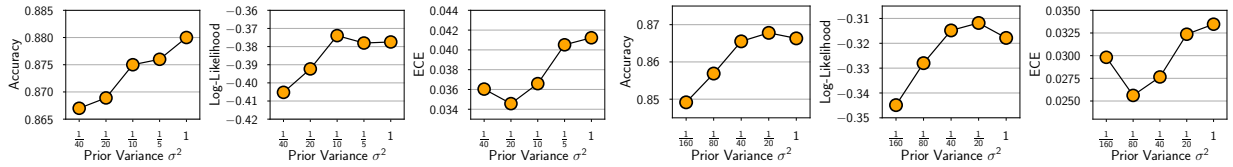
E. Marginal distributions of the weights

In Section 4.1, we argued for using a trajectory length $\tau = \frac{\pi\sigma_{\text{prior}}}{2}$ based on the intuition that the posterior scale is determined primarily by the prior scale. Figure 13 examines this intuition. For each chain and parameter, we estimate the marginal standard deviation of that parameter under the distribution sampled from by that chain. Most of these marginal scales are close to the prior scale, and only a few are significantly larger (note logarithmic scale on y-axis), confirming that the posterior's scale is determined by the prior.

F. What is the effect of priors in Bayesian Neural Networks?

> High-variance Gaussian priors over parameters of Bayesian neural networks lead to strong performance. The results are robust with respect to the prior scale.

Bayesian deep learning is often criticized for the lack of intuitive priors over the parameters. For example, Wenzel et al. (2020) hypothesize that the popular Gaussian priors of the form $\mathcal{N}(0, \sigma^2 I)$ are inadequate and lead to poor performance. Tran et al. (2020) propose a new prior for Bayesian



(a) ResNet20-FRN on CIFAR-10 (b) CNN-LSTM on IMDB

Figure 14. **Ablation of prior variance.** The effect of prior variance on BNN performance. While low prior variance may lead to over-regularization and hurt performance, all the considered prior scales lead to better results than the performance of an SGD-trained neural net of the same architecture (see Figure 6).

PRIOR	LOGISTIC	GAUSSIAN	MIXTURE OF GAUSSIANS
ACCURACY	0.869	0.866	0.863
ECE	0.024	0.029	0.025
LOG LIKELIHOOD	-0.304	-0.311	-0.317

Table 3. **HMC for CNN LSTM with Gaussian and non-Gaussian priors.** We report accuracy, ECE, and log-likelihood numbers for a BMA of 80 examples from a single HMC chain. We evaluate three different priors (logistic, Gaussian, and mixture of Gaussians), and find that the logistic prior achieves better accuracy, log-likelihood, and ECE than the others.

neural networks inspired by Gaussian processes (Rasmussen & Nickisch, 2010) based on this hypothesis. Wilson & Izmailov (2020) on the other hand, argue that vague Gaussian priors in the parameter space induce useful function-space priors.

In Section 6 we have shown that Bayesian neural networks can achieve strong performance with vague Gaussian priors. In this section, we explore the sensitivity of BNNs to the choice of the prior scale, as a step towards better understanding the role of the prior in BNNs.

We use priors of the form $\mathcal{N}(0, \sigma^2 I)$ and vary the prior variance σ^2. For all cases, we use a single HMC chain producing 40 samples.

We report the results for the CIFAR-10 and IMDB datasets in Figure 14. When the prior variance is too small, the regularization is too strong, hindering the performance. Setting the prior variance too large does not seem to hurt the performance as much. On both problems, the performance is fairly robust: a wide window of prior variances lead to strong performance (in particular, for all considered prior scales, the results were better than those of SGD training in Figure 6).

One possible explanation for the relatively flat curves in Figure 14 is that large prior variances imply a strong prior belief that the "true" classifier (i.e., the model that would be learned given infinite data) should make high-confidence predictions. Since the model is powerful enough to achieve any desired training accuracy, the likelihood does not overrule this prior belief, and so the posterior assigns most of its mass to very confident classifiers. Past a certain point, increasing the prior variance on the weights may have no effect on the classifiers' already saturated probabilities.

This means that nearly every member of the BMA may be highly overconfident. But this does not imply that the *ensemble* is overconfident—a mixture of overconfident experts can still make well-calibrated predictions. Figure 15 provides some qualitative evidence for this explanation; for some CIFAR-10 test-set images, the Markov chain oscillates between assigning the true label probabilities near 1 and probabilities near 0.

F.1. Non-gaussian priors

We also investigate non-Gaussian priors and their effect on BNN performance. In Table 3, we report BMA accuracy, ECE, and log-likelihood numbers for two non-Gaussian priors on the IMDB dataset with a CNN LSTM architecture. We pick prior scales to match a prior variance of $\frac{1}{40}$. The first is the logistic distribution, which has heavier tails than a Normal. The mixture of Gaussians is a equal mixture of two zero-centered normals, one with prior variance $\frac{1}{40}$ and the other with prior variance $\frac{1}{160}$. Each prior was evaluated with an BMA of 80 samples from a single HMC chain. We find that the heavier-tailed logistic prior performs better than the others, hinting that despite vague Gaussian priors performing well overall, there is room for improvement, especially considering the observed marginal distributions over weights (Figure 13) having values with large standard deviations.

Table 4. **Role of data augmentation in the cold posterior effect.** Results of a single chain ensemble constructed with the SGLD sampler of Wenzel et al. (2020) at temperatures $T = 1$ and $T = 0.1$ for different combinations of batch normalization (BN) or filter response normalization (FRN) and data augmentation (Aug). Regardless of the normalization technique, the cold posteriors effect is present when data augmentation is used, and not present otherwise.

	ACC, $T = 1$	ACC, $T = 0.1$	CE, $T = 1$	CE, $T = 0.1$
BN + AUG	87.46	91.12	0.376	0.2818
FRN + AUG	85.47	89.63	0.4337	0.317
BN + NO AUG	86.93	85.20	0.4006	0.4793
FRN + NO AUG	84.27	80.84	0.4708	0.5739

G. Further discussion of cold posteriors

In Section 7 we have seen that the cold posteriors are not needed to achieve strong performance with BNNs. We have even shown that cold (as well as warm) posteriors may hurt the performance. On the other hand, in Appendix B we have shown that lowering the temperature can improve robustness under the distribution shift, at least for a small MLP on MNIST. Here, we discuss the potential reasons for the cold posteriors effect in Wenzel et al. (2020).

G.1. What causes the difference with Wenzel et al. (2020)?

There are several key differences between the experiments in our study and Wenzel et al. (2020).

First of all, as discussed in Section 8, the predictive distributions of SGLD (a version of which was used in Wenzel et al. (2020)) are highly dependent on the hyper-parameters such as the batch size and learning rate, and are inherently biased. Furthermore, Wenzel et al. (2020) show in Figure 6 that with a high batch size they achieve good performance at $T = 1$ for the CNN-LSTM. Using the code provided by the autors[7] we achieved good performance at $T = 1$ for the CNN-LSTM (accuracy of 0.855 and cross-entropy of 0.35, compared to 0.81 and 0.45 reported in Figure 1 of Wenzel et al. (2020)); we were, however, able to reproduce the cold posteriors effect on CIFAR-10 using the same code.

On CIFAR-10, the main difference between our set-up and the set-up of Wenzel et al. (2020) is the use of batch normalization and data augmentation. In the appendix K and Figure 28 of Wenzel et al. (2020), the authors show that if both the data augmentation and batch normalization are turned off, we no longer observe the cold posteriors effect. In Table 4 we confirm using the code provided by the authors that in fact it is sufficient to turn off just the data augmentation to remove the cold posteriors effect.

Data augmentation and Bayesian neural networks. It is thus likely that the results in Wenzel et al. (2020) are at least partly affected by the use of data augmentation. Data augmentation can not be naively incorporated in the Bayesian neural network model (see the discussion in appendix K of Wenzel et al. (2020)), and arguably it may be reasonable to decrease the temperature when using data augmentation: in a way, data augmentation increases the amount of data observed by the model, and should lead to higher posterior contraction. We leave incorporating data augmentation in our HMC evaluation framework as an exciting direction of future work.

[7] https://github.com/google-research/google-research/tree/master/cold_posterior_bnn

H. Further Discussion of Cheaper Alternatives to HMC

In Section 8 we studied SGLD and deep ensembles as cheaper alternatives to HMC. We found that SGLD, for some values of hyper-parameters, can provide similar uncertainty distribution on the test data to HMC. However, in Figure 10 we can see that under distribution shift, the behavior of HMC and SGLD is quite different. In fact, SGLD performs *better* than HMC under distribution shift.

So, in terms of predictive performance, both SGLD and deep ensembles can achieve competitive results to HMC at a fraction of the cost. However, the predictions made by these methods can be quite different from those of HMC. Thus, we should be very careful in generalizing the observations about the SGLD and other efficient approximate Bayesian deep learning methods to *true* Bayesian neural networks.

Combining deep ensembles and BNNs. To demonstrate the qualitative difference between deep ensembles and BNNs, we ensemble the predictions of these two methods on CIFAR-10. We find that this ensemble achieves a *significant improvement* over both methods: it achieves an accuracy of 0.9141 and Log-Likelihood of -0.281 compared to 0.907 and 0.306 for HMC and 0.905 and 0.293 for deep ensembles. This result is especially interesting given that both our HMC and deep ensemble are near-saturated: HMC contains 720 samples, and the deep ensemble contains 50 independently trained models. We leave a further exploration of this effect for future work.

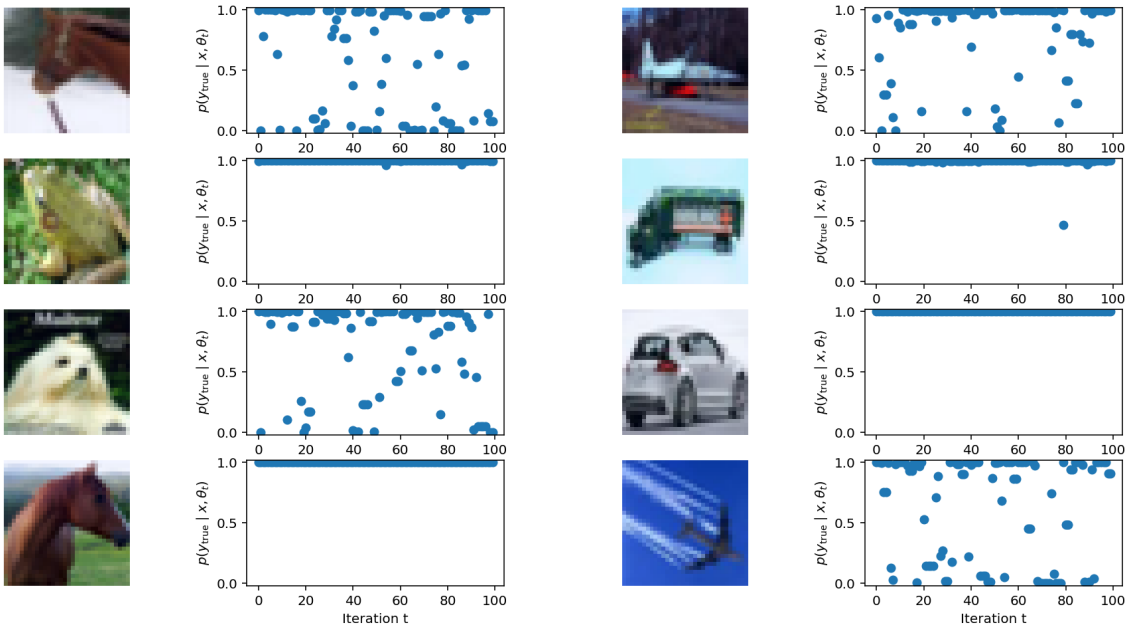


Figure 15. **HMC samples are (over)confident classifiers.** Plots show the probability assigned by a series of HMC samples to the true label of a held-out CIFAR-10 image. In many cases these probabilities are overconfident (i.e., assign the right answer probability near 0), but there are always *some* samples that assign the true label high probability, so the Bayesian model average is both accurate and well calibrated. These samples were generated with a spherical Gaussian prior with variance $\frac{1}{5}$.