			Experiments		
Method	Hyper-parameter	Was Tuned	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	✓	10^{-5}	10^{-5}	10^{-5}
	Num. Burnin Iterations	✗	50	50	50
	Num. Samples per Chain	✗	240	40	400
	Num. of Chains	✗	3	3	3
	Total Samples	✗	720	120	1200
	Total Epochs		$5 \cdot 10^7$	$8.5 \cdot 10^6$	$3 \cdot 10^7$
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		$2.5 \cdot 10^4$	$2.5 \cdot 10^4$	$2.5 \cdot 10^4$
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 Epochs	✗	1000	1000	1000
	Num. Samples per Chain	✗	900	900	900
	Num. of Chains	✗	5	5	5
	Total Samples	✗	4500	4500	4500
	Total Epochs		$5 \cdot 10^4$	$5 \cdot 10^4$	$5 \cdot 10^4$
MFVI	Prior Variance	✗	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$
	Num. Epochs	✗	300	300	300
	Optimizer	✓	Adam	Adam	Adam
	Initial Step Size	✓	10^{-4}	10^{-4}	10^{-4}
	Step Size Schedule	✗	cosine	cosine	cosine
	Batch Size	✓	80	80	80
	VI mean init	✗	SGD solution	SGD solution	SGD solution
	VI variance init	✓	10^{-2}	10^{-2}	10^{-2}
	Number of samples	✗	50	50	50
	Total Epochs		$8 \cdot 10^2$	$8 \cdot 10^2$	$8 \cdot 10^2$

Table 3. **Hyper-parameters for CIFAR and IMDB.** We report the hyper-parameters for each method our main evaluations on CIFAR and IMDB datasets 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. For each of the hyper-parameters we report whether it was tuned via cross-validation, or whether a value was selected without tuning.

Method	Hyper-parameter	Was Tuned	Experiments: Concrete	Yacht	Energy	Boston	Naval
HMC	Prior Variance	✓	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{40}$
	Step Size	✓	10^{-5}	10^{-5}	10^{-5}	10^{-5}	$5 \cdot 10^{-7}$
	Num. Burnin Iterations	✗	10	10	10	10	10
	Num. Iterations	✗	90	90	90	90	90
	Num. of Chains	✗	1	1	1	1	1
SGD	Weight Decay	✓	10	10^{-1}	10	10^{-1}	1
	Initial Step Size	✓	$3 \cdot 10^{-5}$	$3 \cdot 10^{-6}$	$3 \cdot 10^{-6}$	$3 \cdot 10^{-6}$	10^{-6}
	Step Size Schedule	✗	cosine	cosine	cosine	cosine	cosine
	Batch Size	✗	927	277	691	455	10740
	Num. Epochs	✓	1000	5000	5000	500	1000
	Momentum	✗	0.9	0.9	0.9	0.9	0.9
SGLD	Prior Variance	✓	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	1
	Step Size	✓	$3 \cdot 10^{-5}$	10^{-4}	$3 \cdot 10^{-5}$	$3 \cdot 10^{-5}$	10^{-6}
	Step Size Schedule	✗	constant	constant	constant	constant	constant
	Batch Size	✗	927	277	691	455	10740
	Num. Epochs	✗	10000	10000	10000	10000	10000
	Num. Burnin Epochs	✗	1000	1000	1000	1000	1000
	Num. Samples per Chain	✗	900	900	900	900	900
	Num. of Chains	✗	1	1	1	1	1

Table 4. **Hyper-parameters for UCI.** We report the hyper-parameters for each method our main evaluations on UCI datasets in Section 6. For HMC, the number of iterations is the number of HMC iterations after the burn-in phase; the number of accepted samples is lower. For each of the hyper-parameters we report whether it was tuned via cross-validation, or whether a value was selected without tuning.

Hyper-parameter	Was Tuned	SGLD	SGHMC	SGHMC CLR	SGHMC CLR-Prec
Initial Step size	✓	10^{-6}	$3 \cdot 10^{-7}$	$3 \cdot 10^{-7}$	$3 \cdot 10^{-5}$
Step Size Schedule	✗	constant	constant	cyclical	cyclical
Momentum	✓	0.	0.9	0.95	0.95
Preconditioner	✗	None	None	None	RMSprop
Num. Samples per chain	✗	900	900	180	180
Num. of Chains	✗	3	3	3	3

Table 5. **SGMCMC hyper-parameters on CIFAR-10.** We report the hyper-parameter values used by each of the SGMCMC methods in Section 9. The remaining hyper-parameters are the same as the SGLD hyper-parameters reported in Table 3. For each of the hyper-parameters we report whether it was tuned via cross-validation, or whether a value was selected without tuning.

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 details on hyper-parameters used in our experiments. In Appendix B we provide ablations of HMC hyper-parameters and intuition behind them. In Appendix C we compare the BMA predictions using two independent HMC chains on a synthetic regression problem. In Appendix D we provide a description of the $\hat{R}$ statistic used in Section 5.1. In Appendix E we provide posterior density surface visualizations. In Appendix F we explore whether or not our HMC chains converge. In Appendix G we provide complete results of our experiments on CIFAR and IMDB datasets. In Appendix H we show that BNNs are not robust to distribution shift and discuss the reasons for this behavior. In Appendix I we apply BNNs to OOD detection. In Appendix J we provide a further discussion of the effect of posterior temperature. In Appendix K we study the performance of BNNs with Gaussian priors as a function of prior variance. Finally, in Appendix L we compare the predictive entropies and calibration curves between HMC and scalable approximate inference methods.

A. Hyper-Parameters and Details

CIFAR and IMDB. In Table 3 we report the hyperparameters used by each of the methods in our main evaluation on CIFAR and IMDB datasets in Section 6. HMC was run on a cluster of 512 TPUs and the other baselines were run on a cluster of 8 TPUs. On CIFAR datasets the methods used a subset of 40960 datapoints. All methods were ran at posterior temperature 1. We tuned the hyper-parameters for all methods via cross-validation maximizing the accuracy on a validation set. For the step sizes we considered an exponential grid with a step of $\sqrt{10}$ with 5-7 different values, where the boundaries were selected for each method so it would not diverge. We considered weight decays $1, 5, 10, 20, 40, 100$ and the corresponding prior variances. For batch sizes we considered values 80, 200, 400 and 1000; for all methods lower batch sizes resulted in the best performance. For HMC we set the trajectory length according to the strategy described in Section B.1. For SGLD, we experimented with using a cosine learning rate schedule decaying to a non-zero final step size, but it did not improve upon a constant schedule. For MFVI we experimented with the SGD and Adam optimizers; we initialize the mean of the MFVI distribution with a pre-trained SGD solution, and the per-parameter variance with a value $\sigma^{\text{VI}}_{\text{Init}}$; we tested values $10^{-2}, 10^{-1}, 10^{0}$ for $\sigma^{\text{VI}}_{\text{Init}}$. For all HMC hyper-parameters, we provide ablations illustrating their effect in Section 4. Producing a single sample with HMC 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 step sizes $5 \cdot 10^{-5}, 3 \cdot 10^{-5}, 10^{-5}, 3 \cdot 10^{-6}, 10^{-6}$ 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; the lower prior variance compared to Table 3 was chosen to reduce the number of leapfrog iterations, as we chose the trajectory length according to the strategy described in Section B.1. 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. In Table 4 we report the hyperparameters used by each of the methods in our main evaluation on UCI datasets in Section 6. For each datasets we construct 20 random splits with 90% of the data in the train and 10% of the data in the test split. In the evaluation, we report the mean and standard deviation of the results across the splits. We use another random split for cross-validation to tune the hyper-parameters. For all datasets we use a fully-connected network with a single hidden layer with 50 neurons and 2 outputs representing the predictive mean and variance for the given input. We use a Gaussian likelihood to train each of the methods. For the SGD and SGLD baselines, we did not use mini-batches: the gradients were computed over the entire dataset. We run each experiment on a single NVIDIA Tesla V-100 GPU.

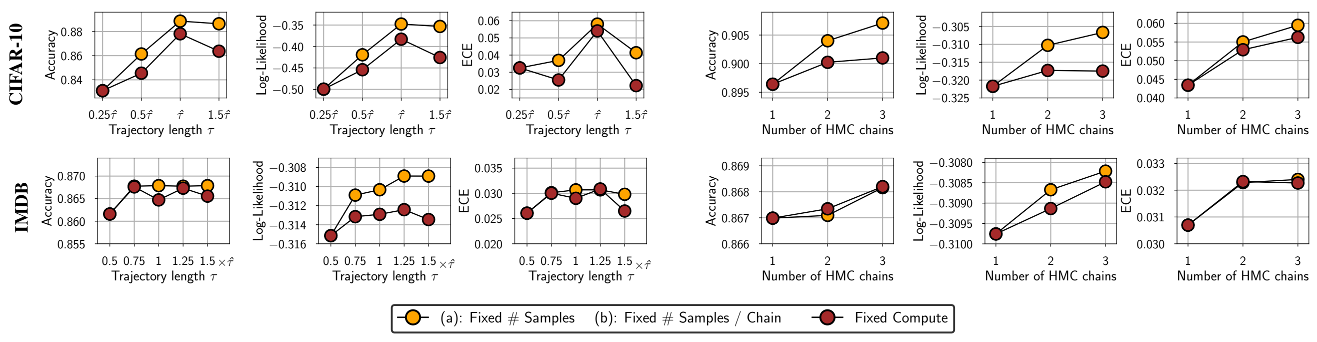
Algorithm 1 Hamiltonian Monte Carlo

```
Input: Trajectory length $\tau$, number of burn-in interations $N_{\text{burnin}}$, initial parameters $w_{\text{init}}$, step size $\Delta$, number of samples $K$, unnormalized posterior log-density function $f(w) = \log p(D|w) + \log p(w)$.
Output: Set $S$ of samples $w$ of the parameters.
$w \leftarrow w_{\text{init}}$;   $N_{\text{leapfrog}} \leftarrow \frac{\tau}{\Delta}$;
# Burn-in stage
for $i \leftarrow 1 \dots N_{\text{burnin}}$ do
    $m \sim \mathcal{N}(0, I)$;
    $(w, m) \leftarrow$ Leapfrog$(w, m, \Delta, N_{\text{leapfrog}}, f)$;
end for
# Sampling
$S \leftarrow \varnothing$;
for $i \leftarrow 1 \dots K$ do
    $m \sim \mathcal{N}(0, I)$;
    $(w', m') \leftarrow$ Leapfrog$(w, m, \Delta, N_{\text{leapfrog}}, f)$;

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

Algorithm 2 Leapfrog integration

```
Input: Parameters $w_0$, initital momentum $m_0$, step size $\Delta$, number of leapfrog steps $N_{\text{leapfrog}}$, posterior log-density function $f(w) = \log p(w|D)$.
Output: New parameters $w$; new momentum $m$.
$w \leftarrow w_0$;   $m \leftarrow m_0$;
for $i \leftarrow 1 \dots N_{\text{leapfrog}}$ do
    $m \leftarrow m + \frac{\Delta}{2} \cdot \nabla f(w)$;
    $w \leftarrow w + \Delta \cdot m$;
    $m \leftarrow m + \frac{\Delta}{2} \cdot \nabla f(w)$;
end for
Leapfrog$(w_0, m_0, \Delta, N_{\text{leapfrog}}, f) \leftarrow (w, m)$
```



(a) Trajectory length ablation (b) Number of chains ablation

Figure 6. **Effect 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 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.

SGMCMC Methods. In Table 5 we report the hyper-parameters of the SGMCMC methods on the CIFAR-10 dataset used in the evaluation in Section 9. We considered momenta in the set of $\{0.9, 0.95, 0.99\}$ and step sizes in $\{10^{-4}, 3 \cdot 10^{-5}, 10^{-5}, 3 \cdot 10^{-6}, 10^{-6}, 3 \cdot 10^{-7}, 10^{-7}\}$. We selected the hyper-parameters with the best accuracy on the validation set. SGLD does not allow a momentum.

B. Effect of HMC Hyper-Parameters

We perform ablations of HMC hyper-parameters using ResNet-20-FRN on CIFAR-10 and CNN-LSTM on IMDB.

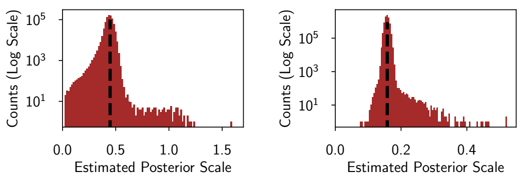
B.1. Trajectory length τ

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/Δ of the trajectory length to the step size.

We suggest the following value of the trajectory length τ:

$$\hat{\tau} = \frac{\pi \alpha_{\text{prior}}}{2}, \tag{2}$$

where α_{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[6]. 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.



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

Figure 7. **Marginal distributions of the weights.** Log-scale histograms of estimated marginal posterior standard deviations for ResNet-20-FRN on CIFAR-10 and CNN-LSTM on IMDB. The histograms show how many parameters have empirical standard deviations that fall within a given bin. For most of the parameters (notice that the plot is logarithmic) the posterior scale is very similar to that of the prior distribution.

In order to test the validity of our recommended trajectory length, we perform an ablation and report the results in Figure 6(a). As expected, longer trajectory lengths provide 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.

In Figure 7 we examine the intuition that the posterior scale is determined by the prior scale. For each parameter, we estimate the marginal standard deviation of that parame-

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

ter under the distribution sampled by HMC. Most of the marginal scales are close to the prior scale, and only a few are significantly larger (note logarithmic scale), confirming that the posterior's scale is determined by the prior.

B.2. Effect of HMC Step Size Δ

The step size parameter Δ 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.

Using ResNet-20-FRN on CIFAR-10, 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.

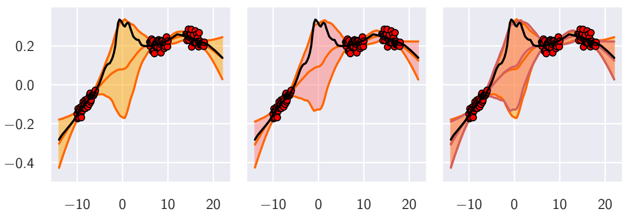
B.3. Number of HMC Chains

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 Figure 6(b). Holding compute budget fixed, using two or three chains is only slightly better than using one chain. This result notably shows that HMC is relatively unobstructed by energy barriers in the posterior surface that would otherwise require multiple chains to overcome. We explore this result further in Section 5.

C. HMC Predictive Distributions in Synthetic Regression

We consider a one-dimensional synthetic regression problem. We follow the general setup of Izmailov et al. (2019) and Wilson & Izmailov (2020). We generate the training inputs as a uniform grid with 40 points in each of the following intervals (120 datapoints in total): $[-10, -6]$, $[6, 10]$ and $[14, 18]$. We construct the ground truth target values using a neural network with 3 hidden layers, each of dimension 100, one output and two inputs: following Izmailov et al. (2019),



(a) Chain 1 (b) Chain 2 (c) Overlaid

Figure 8. **HMC chains on synthetic regression.** We visualize the predictive distributions for two independent HMC chains on a synthetic regression problem with a fully-connected network. The data is shown with red circles, and the true data generating function is shown with a black line. The shaded region shows 3 standard deviations of the predictive distribution, and the predictive mean is shown with a line of the same color. In panels **(a)**, **(b)** we show the predictive distributions for each of the two chains individually, and in panel **(c)** we overlay them on top of each other. The chains provide almost identical predictions, suggesting that HMC mixes well in the prediction space.

for each datapoint x we pass x and x^2 as inputs to the network to enlarge the class of functions that the network can represent. We draw the parameters of the network from a Gaussian distribution with mean 0 and standard deviation 0.1. We show the sample function used to generate the target values as a black line in each of the panels in Figure 8. We then add Gaussian noise with mean 0 and standard deviation 0.02 to each of the target values. The final dataset used in the experiment is shown with red circles in Figure 8.

For inference, we use the same model architecture that was used to generate the data. We sample the initialization parameters of the network from a Gaussian distribution with mean 0 and standard deviation 0.005. We use a Gaussian distribution with mean zero and standard deviation 0.1 as the prior over the parameters, same as the distribution used to sample the parameters of the ground truth solution. We use a Gaussian likelihood with standard deviation 0.02, same as the noise distribution in the data. We run two HMC chains from different random initializations. Each chain uses a step size of 10^{-5} and the trajectory length is set according to the strategy described in Section B.1, resulting in 15708 leapfrog steps per HMC iteration. We run each chain for 100 HMC iterations and collect the predictions corresponding to all the accepted samples, resulting in 89 and 82 samples for the first and second chain respectively. We discard the first samples and only use the last 70 samples from each chain. For each input point we compute the mean and standard deviation of the predictions.

We report the results in Figure 8. In panels (a), (b) we show the predictive distributions for each of the chains, and in panel (c) we show them overlaid on top of each other. Both chains provide high uncertainty away from the data, and low uncertainty near the data as desired (Yao et al., 2019). Moreover, the true data-generating function lies in the 3σ-region of the predictive distribution for each chain. Finally, the predictive distributions for the two chains are almost identical. This result suggests that on the synthetic problem under consideration HMC is able to mix in the space of predictions, and provides similar results independent of initialization and random seed. We come to the same conclusion for more realistic problems in Section 5.

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 $\{\theta_{mn} | m \in \{1, \dots, M\}, n \in \{1, \dots, N\}\}$, 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 Section 5.1 we evaluated it for each weight and each predicted softmax probability in the test set.

E. Posterior Visualizations

To further investigate 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

$$\mathcal{S} = \{w | w = w_1 \cdot a + w_2 \cdot b + w_3 \cdot (1 - a - b)\}. \tag{8}$$

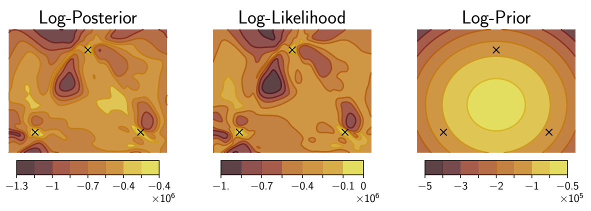
$\mathcal{S}$ is the unique two-dimensional affine subspace (plane) of the parameter space that includes parameter vectors w_1, w_2 and w_3.

In Figure 9(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. We note that popular approximate inference procedures, such as variational methods, and Laplace approximations, are typically constrained to unimodal Gaussian approximations to the posterior, which we indeed expect to miss a large space of compelling solutions in the posterior.

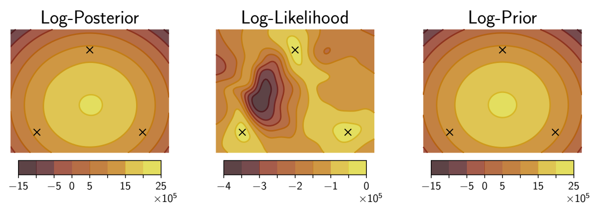
In Figure 9(b) we provide a visualization for the samples produced by 3 different HMC chains at iteration 51 after burn-in. Comparing the visualizations for samples from the same chain and samples from independent chains in Figure 9, we see that the shapes of the posterior surfaces are different, with the latter appearing more regular and symmetric. The qualitative differences between (a) and (b) suggest that while each HMC chain is able to navigate the posterior geometry the chains do not mix perfectly in the weight space, confirming our results in Section 5.1.

In Figure 9(c, d) we provide analogous visualizations for the CNN-LSTM architecture on IMDB. On IMDB, the posterior log-density is dominated by the prior, and the corresponding panels (c, d) are virtually indistinguishable in Figure 9. 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 9 (a, b). 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).

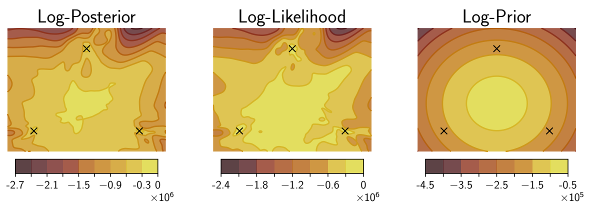
In Figure 9 (e), we visualize the likelihood cross-sections using our runs with varying posterior temperature on IMDB. The visualizations show that, as expected, low temperature leads to a sharp likelihood, while the high-temperature likelihood appear soft. In particular, the scale of the lowest



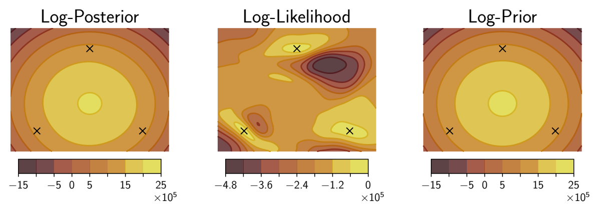
(a) CIFAR-10, Same chain



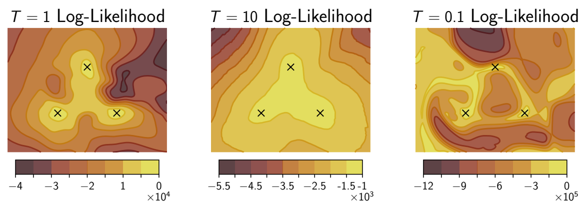
(c) IMDB, same chain



(b) CIFAR-10, Independent chains



(d) IMDB, independent chain



(e) IMDB, Log-Likelihood at different T

Figure 9. **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 on IMDB using CNN-LSTM. **(a, c):** samples from the same chain and **(b, d)**: independent chains; **(e):** Log-likelihood surfaces for samples from the same chain at posterior temperatures $T = 1$, 10 and 0.1. We use **(a, b):** ResNet-20-FRN on CIFAR-10 and **(c, d, e):** CNN-LSTM on IMDB.

likelihood values at $T = 10$ is only 10^3 while the scale at $T = 0.1$ is 10^6.

How are the visualizations created? To create the visualizations we pick the points in the parameter space corresponding to three HMC samples: w_1, w_2, w_3. We construct a basis in the 2-dimensional affine subspace passing through these three points: $u = w_2 - w_1$ and $v = w_3 - w_1$. We then orthogonalize the basis: $\hat{u} = u/\|u\|$, $\hat{v} = (v - \hat{u}^T v)/\|v - \hat{u}^T v\|$. We construct a 2-dimensional uniform grid in the basis $\hat{u}, \hat{v}$. Each point in the grid corresponds to a vector of parameters of the network. We evaluate the log-likelihood, log-prior and posterior log-density for each of the points in the grid, converting them to the corresponding network parameters. Finally, we produce contour plots using the collected values. The procedure is analogous to that used by Garipov et al. (2018)[7].

F. 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 period. For a converged chain, the performance of the BMA and individual samples should be stationary and not show any visible trends after a sufficiently long 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

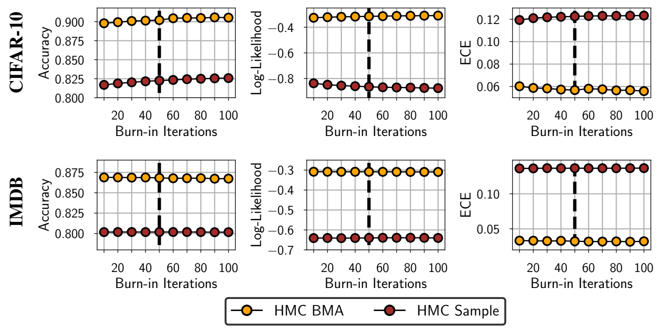


Figure 10. **HMC convergence.** The performance of an individual HMC sample and a BMA ensemble of 100 samples from each one of 3 HMC chains after the burn-in as a function of burn-in length. The dashed line indicates the burn-in length of 50 that we used in the main experiments in this paper. We use ResNet-20-FRN on CIFAR-10 and CNN-LSTM on IMDB. On IMDB, there is no visible dependence of the results on the burn-in length; on CIFAR-10, there is a weak trend that slows down over time.

[7]See also the blogpost https://izmailovpavel.github.io/curves_blogpost/, Section "How to Visualize Loss Surfaces?".

METHOD	CIFAR-10			CIFAR-100			IMDB		
	ACC	LOG-LIK	ECE	ACC	LOG-LIK	ECE	ACC	LOG-LIK	ECE
SGD	83.4	-0.800	0.119	47.8	-2.364	0.193	82.9	-0.755	0.136
HMC	**90.7**	**-0.307**	0.059	**69.3**	**-1.134**	0.134	**86.8**	**-0.308**	0.033
SGLD	89.3	-0.341	0.052	63.6	-1.404	0.148	86.1	-0.314	0.007
DE	89.2	-0.331	0.028	65.6	-1.385	0.170	85.8	-0.358	0.042
SGLD (5 CHAINS)	90.1	-0.327	0.066	66.9	-1.342	0.195	**86.6**	**-0.306**	**0.006**
MFVI	86.5	-0.409	**0.019**	54.9	-1.749	**0.032**	85.4	-0.341	0.038

Table 6. **Detailed results on CIFAR and IMDB.** Accuracy, log-likelihood and expected calibration error for Hamiltonian Monte Carlo (HMC), stochastic gradient Langevin dynamics (SGLD) with 1 and 5 chains, mean field variational inference (MFVI), stochastic gradient descent (SGD), and deep ensembles. We use ResNet-20-FRN on CIFAR datasets, and CNN-LSTM on IMDB. Bayesian neural networks via HMC outperform all baselines on all datasets in terms of accuracy and log-likelihood; on IMDB the performance of 5-chain SGLD is similar. All methods perform similarly well on ECE, except for SGD which is consistently poorly calibrated.

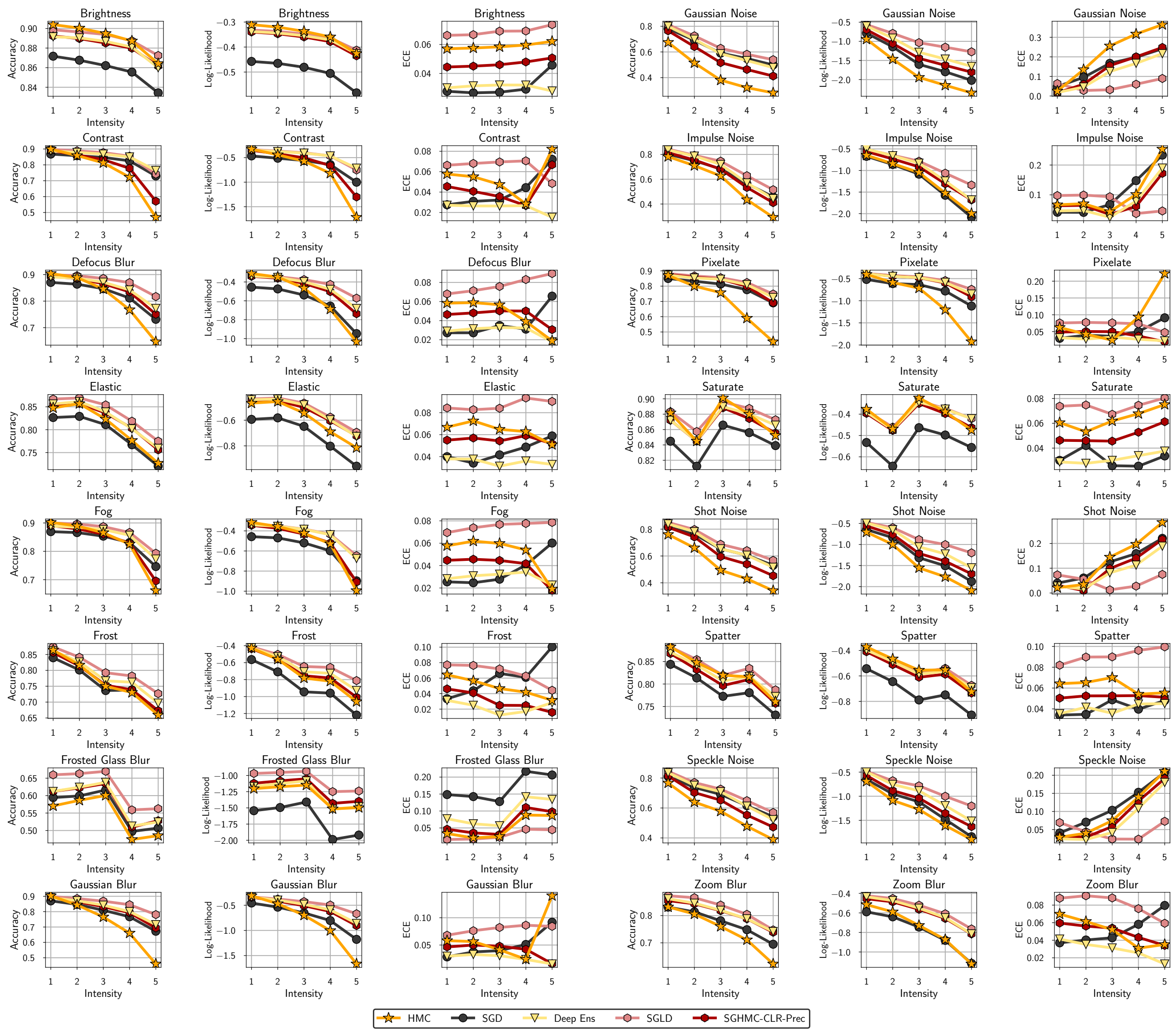


Figure 11. **Performance under corruption.** We show accuracy, log-likelihood and ECE of HMC, SGD, Deep Ensembles, SGLD and SGHMC-CLR-Prec for all 16 CIFAR-10-C corruptions as a function of corruption intensity. HMC shows poor accuracy on most of the corruptions with a few exceptions. SGLD provides the best robustness on average.

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 for ResNet-20-FRN on CIFAR-10 and CNN-LSTM on IMDB in Figure 10. On IMDB, there is no visible trend in performance, so a burn-in of just 10 HMC iterations should be sufficient. On CIFAR-10, we observe a slowly rising trend that saturates at about 50 iterations, indicating that a longer burn-in period is needed compared to IMDB. We therefore use a burn-in period of 50 HMC iterations on both CIFAR and IMDB for the remainder of the paper.

Is HMC Converging? In general, it is not possible to ensure that an MCMC method has converged to sampling from the true posterior distribution: theoretically, there may always remain regions of the posterior that cannot be discovered by the method but that contain most of the posterior mass. To maximize the performance of HMC, we choose the hyper-parameters that are the most likely to provide convergence: long trajectory lengths, and multiple long chains. In Section 5, we study the convergence of HMC using the available convergence diagnostics. We find that while HMC does not mix perfectly in weight space, in the space of predictions we cannot find evidence of non-mixing.

G. Detailed Results on CIFAR and IMDB

In Table 6 we report the accuracy, log-likelihood and expected calibration error for each of the methods we consider on CIFAR and IMDB datasets. BNNs via HMC provide the best performance on all datasets in terms of accuracy and log-likelihood (with 5-chain SGLD achieving competitive results on IMDB), and all methods except SGD show good calibration in terms of ECE.

H. BNNs are not Robust to Domain Shift

In Section 6.2, Figure 4 we have seen that surprisingly BNNs via HMC underperform significantly on corrupted data from CIFAR-10-C compared to SGLD, deep ensembles and even MFVI and SGD. We provide detailed results in Figure 11. HMC shows surprisingly poor robustness in terms of accuracy and log-likelihood across the corruptions. The ECE results are mixed. In most cases, the HMC ensemble of 720 models loses to a single SGD solution!

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 are often explicitly evaluated on CIFAR-10-C (e.g. Wilson & Izmailov, 2020; Dusenberry et al., 2020). Our results suggest that the improvements

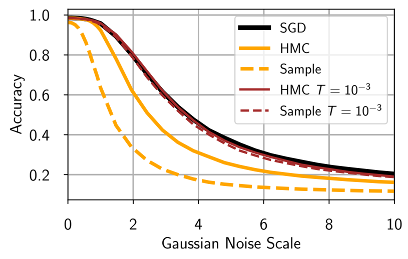


Figure 12. **Robustness on MNIST.** Performance of SGD, BMA ensembles and individual samples constructed by HMC at temperatures $T = 1$ and $T = 10^{-3}$ on the MNIST test set corrupted by Gaussian noise. We use a fully-connected network. Temperature 1 HMC shows very poor robustness, while lowering the temperature allows us to close the gap to SGD.

achieved by Bayesian methods on corrupted data may be a sign of *poor* posterior approximation.

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 12. 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!

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 provide evidence for this explanation, we run evaluate HMC at a very low temperature $T = 10^{-3}$, as low temperature posteriors concentrate on high-performance solutions similar to the ones found by SGD. We find that at this temperature, HMC performs comparably with SGD, closing the gap in robustness We have also experimented with varying the prior scale but were unable to close the gap in robustness at temperature $T = 1$.

We hypothesize that using a lower temperature with HMC

	AUC-ROC			
OOD Dataset	HMC	Deep Ens	ODIN	Mahalanobis
CIFAR-100	0.857	0.853	0.858	**0.882**
SVHN	0.8814	0.8529	0.967	**0.991**

Table 7. **Out-of-distribution detection.** We use a ResNet-20-FRN model trained on CIFAR-10 to detect out-of-distribution data coming from SVHN or CIFAR-100. We report the results for HMC, deep ensembles and specialized ODIN (Liang et al., 2017) and Mahalanobis (Lee et al., 2018) methods. We report the AUC-ROC score (higher is better) evaluating the ability of each method to distinguish between in-distribution and OOD data. The predictive uncertainty from Bayesian neural networks allows us to detect OOD inputs: HMC outpeforms deep ensembles on both datasets. Moreover, 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.

	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

Table 8. **Role of data augmentation in the cold posterior effect.** Results of a single chain ensemble constructed with the SGHMC-CLR-Prec 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). We use the ResNet-20 architecture on CIFAR-10. Regardless of the normalization technique, the cold posteriors effect is present when data augmentation is used, and not present otherwise.

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[8].

I. Out-of-Distribution Detection

Bayesian deep learning methods are often evaluated on out-of-distribution detection. In Table 7 we report the performance of an 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 train the methods on CIFAR-10 and use CIFAR-100 and SVHN as OOD data sources.

We find that BNNs perform competitively with the specialized ODIN method in the challenging near-OOD detection setting (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).

[8]In a follow-up work, Izmailov et al. (2021) provide a detailed explanation for why Bayesian neural networks can fail under covariate shift. In particular, they find that tempering does not generally improve robustness, and propose alternative resolutions.

J. 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 H 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 why the cold posteriors effect was observed in Wenzel et al. (2020).

J.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, 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: 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 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.

Furthermore, Wenzel et al. (2020) show in Figure 6 that with a high batch size they achieve good performance at

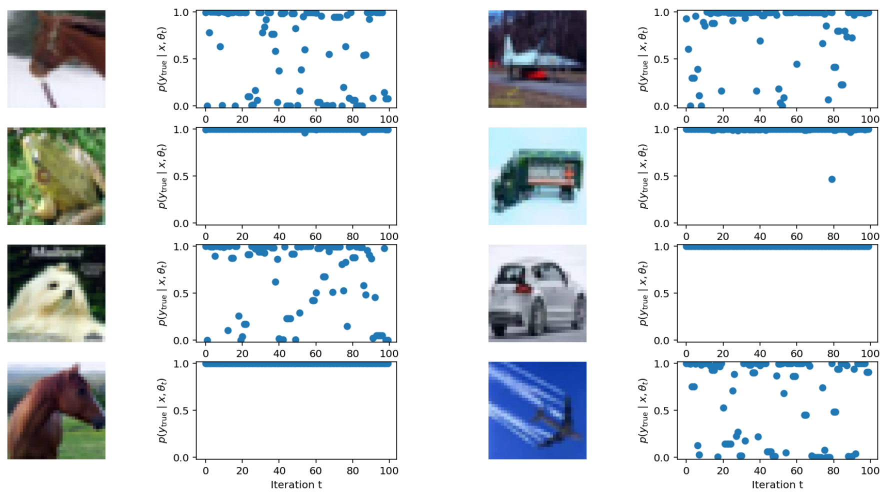


Figure 13. **HMC samples are (over)confident classifiers.** Plots show the probability assigned by a series of HMC samples to the true label of a held-out CIFAR-10 image. In many cases these probabilities are overconfident (i.e., assign the right answer probability near 0), but there are always *some* 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 $\frac{1}{5}$.

$T = 1$ for the CNN-LSTM. Using the code provided by the autors[9] with default hyper-parameters we achieved strong 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 setup and the configuration in 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 8 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. It is thus likely that the results in Wenzel et al. (2020) are at least partly affected by the use of data augmentation.

K. Effect of Gaussian Prior Scale

We use priors of the form $\mathcal{N}(0, \alpha^2 I)$ and vary the prior variance α^2. For all cases, we use a single HMC chain producing 40 samples. These are much shorter chains than the ones we used in Section 6, so the results are not as good; the purpose of this section is to explore the *relative* performance of BNNs under different priors.

We report the results for the CIFAR-10 and IMDB datasets

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

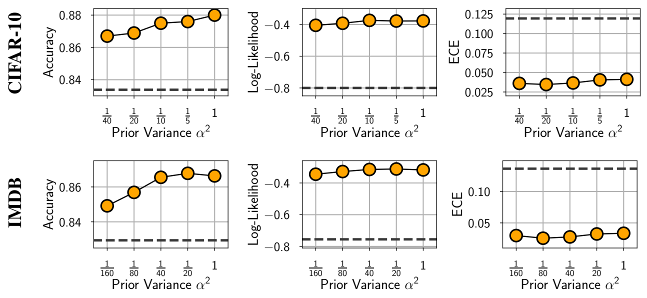


Figure 14. **Effect of prior variance.** The effect of prior variance on BNN performance. In each panel, the dashed line shows the performance of the SGD model from Section 6. While low prior variance may lead to over-regularization and hurt performance, all the considered prior scales lead to better results than the performance of an SGD-trained neural network of the same architecture.

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 are better than those of SGD training.

Why are BNNs so robust to the prior scale? 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. Consequently, nearly every member of the BMA may be highly overconfident. But the *ensemble* does not have to be overconfident—a mixture of overconfident experts can still make well-calibrated predictions. Figure 13 provides some qualitative evidence for this explanation; for some CIFAR-10 test set images, the HMC chain oscillates between assigning the true label probabilities near 1 and probabilities near 0.

L. Predictive Entropy and Calibration Curves for HMC and Scalable BDL Methods

In Figure 15 we visualize the distribution of predictive entropies and the calibration curves for HMC, SGD, deep ensembles, MFVI, SGLD and SGHMC-CLR-Prec on CIFAR-10 using ResNet-20-FRN.

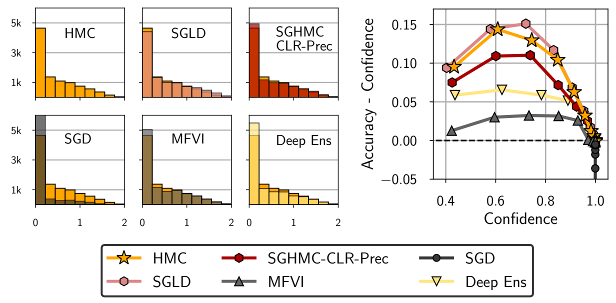


Figure 15. Distribution of predictive entropies (**left**) and calibration curve (**right**) of posterior predictive distributions for HMC, SGD, deep ensembles, MFVI, SGLD and SGHMC-CLR-Prec for ResNet20-FRN on CIFAR-10. On the left, for all methods, except HMC we plot a pair of histograms: for HMC and for the corresponding method. SGD, Deep ensembles and MFVI provide more confident predictions than HMC. SGMCMC methods appear to fit the predictive distribution of HMC better: SGLD is slightly underconfident relative to HMC while SGHMC-CLR-Prec is slightly over-confident.

All methods except fot SGD make conservative predictions: their confidences tend to underestimate their accuracies (Figure 15, right); SGD on the other hand is very over-confident (in agreement with the results in Guo et al., 2017). Deep ensembles and MFVI provide the most calibrated predictions, while SGLD and SGHMC-CLR-Prec match the HMC entropy distribution and calibration curve closer.