# How Good is the Bayes Posterior in Deep Neural Networks Really?

Florian Wenzel [*1] Kevin Roth [*+2] Bastiaan S. Veeling [*+3 1] Jakub Świątkowski [4+] Linh Tran [5+]
Stephan Mandt [6+] Jasper Snoek [1] Tim Salimans [1] Rodolphe Jenatton [1] Sebastian Nowozin [1]

Only classificatio

## Abstract

During the past five years the Bayesian deep learning community has developed increasingly accurate and efficient approximate inference procedures that allow for Bayesian inference in deep neural networks. However, despite this algorithmic progress and the promise of improved uncertainty quantification and sample efficiency there are—as of early 2020—no publicized deployments of Bayesian neural networks in industrial practice. In this work we cast doubt on the current understanding of Bayes posteriors in popular deep neural networks: we demonstrate through careful MCMC sampling that the posterior predictive induced by the Bayes posterior yields systematically worse predictions compared to simpler methods including point estimates obtained from SGD. Furthermore, we demonstrate that predictive performance is improved significantly through the use of a "cold posterior" that overcounts evidence. Such cold posteriors sharply deviate from the Bayesian paradigm but are commonly used as heuristic in Bayesian deep learning papers. We put forward several hypotheses that could explain cold posteriors and evaluate the hypotheses through experiments. Our work questions the goal of accurate posterior approximations in Bayesian deep learning: If the true Bayes posterior is poor, what is the use of more accurate approximations? Instead, we argue that it is timely to focus on understanding the origin of the improved performance of cold posteriors.

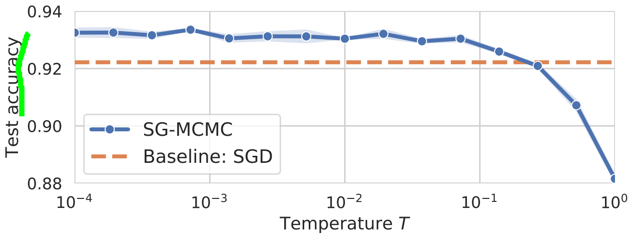


*Figure 1.* The "**cold posterior**" effect: for a ResNet-20 on CIFAR-10 we can improve the generalization performance significantly by cooling the posterior with a temperature $T \ll 1$, deviating from the Bayes posterior $p(\boldsymbol{\theta}|\mathcal{D}) \propto \exp(-U(\boldsymbol{\theta})/T)$ at $T = 1$.

## 1. Introduction

In supervised deep learning we use a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1,\dots,n}$ and a probabilistic model $p(y|x, \boldsymbol{\theta})$ to minimize the regularized cross-entropy objective,

$$L(\boldsymbol{\theta}) := -\frac{1}{n}\sum_{i=1}^{n} \log p(y_i|x_i, \boldsymbol{\theta}) + \Omega(\boldsymbol{\theta}), \qquad (1)$$

where $\Omega(\boldsymbol{\theta})$ is a regularizer over model parameters. We approximately optimize (1) using variants of stochastic gradient descent (SGD), (Sutskever et al., 2013). Beside being efficient, the SGD minibatch noise also has generalization benefits (Masters & Luschi, 2018; Mandt et al., 2017).

### 1.1. Bayesian Deep Learning

In Bayesian deep learning we do not optimize for a *single* likely model but instead want to discover *all* likely models. To this end we approximate the *posterior distribution* over model parameters, $p(\boldsymbol{\theta}|\mathcal{D}) \propto \exp(-U(\boldsymbol{\theta})/T)$, where $U(\boldsymbol{\theta})$ is the *posterior energy function*,

$$U(\boldsymbol{\theta}) := -\sum_{i=1}^{n} \log p(y_i|x_i, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta}), \qquad (2)$$

and $T$ is a *temperature*. Here $p(\boldsymbol{\theta})$ is a *proper* prior density function, for example a Gaussian density. If we scale $U(\boldsymbol{\theta})$ by $1/n$ and set $\Omega(\boldsymbol{\theta}) = -\frac{1}{n}\log p(\boldsymbol{\theta})$ we recover $L(\boldsymbol{\theta})$ in (1). Therefore $\exp(-U(\boldsymbol{\theta}))$ simply gives high probability to models which have low loss $L(\boldsymbol{\theta})$. Given $p(\boldsymbol{\theta}|\mathcal{D})$ we *predict* on a new instance $x$ by averaging over all likely models,

$$p(y|x, \mathcal{D}) = \int p(y|x, \boldsymbol{\theta})\, p(\boldsymbol{\theta}|\mathcal{D})\, \mathrm{d}\boldsymbol{\theta}, \qquad (3)$$

where (3) is also known as *posterior predictive* or *Bayes ensemble*. Solving the integral (3) exactly is not possible. Instead, we approximate the integral using a sample

predictive posterior

[*] Equal contribution [+] Work done while at Google [1] Google Research [2] ETH Zurich [3] University of Amsterdam [4] University of Warsaw [5] Imperial College London [6] University of California, Irvine. Correspondence to: Sebastian Nowozin <nowozin@google.com>.

approximation, $p(y|x, \mathcal{D}) \approx \frac{1}{S}\sum_{s=1}^{S} p(y|x, \boldsymbol{\theta}^{(s)})$, where $\boldsymbol{\theta}^{(s)}$, $s = 1, \dots, S$, is approximately sampled from $p(\boldsymbol{\theta}|\mathcal{D})$.

The remainder of this paper studies a surprising effect shown in Figure 1, the "*Cold Posteriors*" effect: for deep neural networks the Bayes posterior (at temperature $T = 1$) works poorly but by cooling the posterior using a temperature $T < 1$ we can significantly improve the prediction performance.

> **Cold Posteriors**: among all temperized posteriors the best posterior predictive performance on holdout data is achieved at temperature $T < 1$.

### 1.2. Why Should Bayes ($T = 1$) be Better?

Why would we expect that predictions made by the *ensemble model* (3) could improve over predictions made at a single well-chosen parameter? There are three reasons: 1. *Theory*: for several models where the predictive performance can be analyzed it is known that the posterior predictive (3) can dominate common point-wise estimators based on the likelihood, (Komaki, 1996), even in the case of misspecification, (Fushiki et al., 2005; Ramamoorthi et al., 2015); 2. *Classical empirical evidence*: for classical statistical models, averaged predictions (3) have been observed to be more robust in practice, (Geisser, 1993); and 3. *Model averaging*: recent deep learning models based on deterministic model averages, (Lakshminarayanan et al., 2017; Ovadia et al., 2019), have shown good predictive performance.

Note that a large body of work in the area of Bayesian deep learning in the last five years is motivated by the assertion that predicting using (3) is desirable. We will confront this assertion through a simple experiment to show that our understanding of the Bayes posterior in deep models is limited. Our work makes the following **contributions**:

- We demonstrate for two models and tasks (ResNet-20 on CIFAR-10 and CNN-LSTM on IMDB) that the Bayes posterior predictive has poor performance compared to SGD-trained models.
- We put forth and systematically examine hypotheses that could explain the observed behaviour.
- We introduce two new diagnostic tools for assessing the approximation quality of stochastic gradient Markov chain Monte Carlo methods (SG-MCMC) and demonstrate that the posterior is accurately simulated by existing SG-MCMC methods.

## 2. Cold Posteriors Perform Better

We now examine the quality of the posterior predictive for two simple deep neural networks. We will describe details of the models, priors, and approximate inference methods in Section 3 and Appendix A.1 to A.3. In particular, we will study the accuracy of our approximate inference and the influence of the prior in great detail in Section 4 and Section 5.2, respectively. Here we show that temperized Bayes ensembles obtained via low temperatures $T < 1$ outperform the true Bayes posterior at temperature $T = 1$.

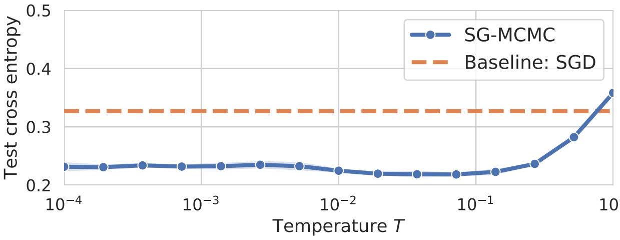


*Figure 2.* Predictive performance on the CIFAR-10 test set for a cooled ResNet-20 Bayes posterior. The SGD baseline is separately tuned for the same model (Appendix A.2).

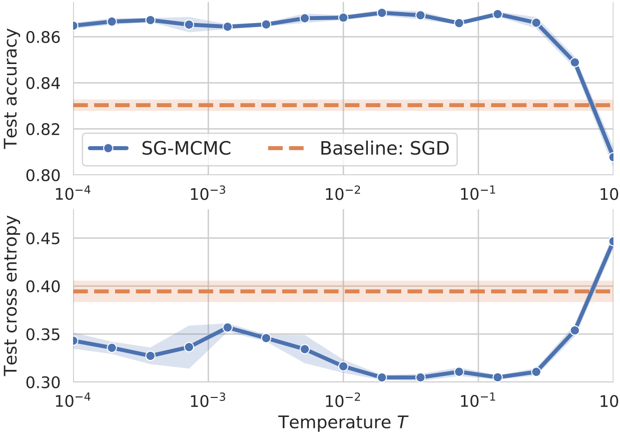


*Figure 3.* Predictive performance on the IMDB sentiment task test set for a tempered CNN-LSTM Bayes posterior. Error bars are ± one standard error over three runs. See Appendix A.4.

### 2.1. Deep Learning Models: ResNet-20 and LSTM

**ResNet-20 on CIFAR-10.** Figure 1 and 2 show the test accuracy and test cross-entropy of a Bayes prediction (3) for a ResNet-20 on the CIFAR-10 classification task.[1] We can clearly see that both accuracy and cross-entropy are significantly improved for a temperature $T < 1/10$ and that this trend is consistent. Also, surprisingly this trend holds all the way to small $T = 10^{-4}$: the test performance obtained from an ensemble of models at temperature $T = 10^{-4}$ is superior to the one obtained from $T = 1$ and better than the performance of a single model trained with SGD.

**CNN-LSTM on IMDB text classification.** Figure 3 shows the test accuracy and test cross-entropy of the tempered prediction (3) for a CNN-LSTM model on the IMDB sentiment classification task. The optimal predictive performance is again achieved for a tempered posterior with a temperature range of approximately $0.01 < T < 0.2$.

[1] A similar plot is in the appendix of (Zhang et al., 2020).

Should be p(theta)^{T}?

### 2.2. Why is a Temperature of $T < 1$ a Problem?

There are two reasons why cold posteriors are problematic. *First*, $T < 1$ corresponds to artificially sharpening the posterior, which can be interpreted as overcounting the data by a factor of $1/T$ and a rescaling[2] of the prior as $p(\theta)^{\frac{1}{T}}$. This is equivalent to a Bayes posterior obtained from a dataset consisting of $1/T$ replications of the original data, giving too strong evidence to individual models. For $T = 0$, all posterior probability mass is concentrated on the set of maximum a posteriori (MAP) point estimates. *Second*, $T = 1$ corresponds to the true Bayes posterior and performance gains for $T < 1$ point to a deeper and potentially resolvable problem with the prior, likelihood, or inference procedure.

### 2.3. Confirmation from the Literature

Should the strong performance of tempering the posterior with $T \ll 1$ surprise us? It certainly is an observation that needs to be explained, but it is not new: if we comb the literature of Bayesian inference in deep neural networks we find broader evidence for this phenomenon.

**Related work that uses $T < 1$ posteriors in SG-MCMC.** The following table lists work that uses SG-MCMC on deep neural networks and tempers the posterior.[3]

| Reference | Temperature $T$ |
|---|---|
| (Li et al., 2016) | $1/\sqrt{n}$ |
| (Leimkuhler et al., 2019) | $T < 10^{-3}$ |
| (Heek & Kalchbrenner, 2020) | $T = 1/5$ |
| (Zhang et al., 2020) | $T = 1/\sqrt{50000}$ |

c-SG-MCMC p

**Related work that uses $T < 1$ posteriors in Variational Bayes.** In the variational Bayes approach to Bayesian neural networks, (Blundell et al., 2015; Hinton & Van Camp, 1993; MacKay et al., 1995; Barber & Bishop, 1998) we optimize the parameters $\tau$ of a variational distribution $q(\boldsymbol{\theta}|\tau)$ by maximizing the evidence lower bound (ELBO),

$$\mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta}|\tau)}\left[\sum_{i=1}^{n} \log p(y_i|x_i, \boldsymbol{\theta})\right] - \lambda D_{\mathrm{KL}}(q(\boldsymbol{\theta}|\tau)\|p(\boldsymbol{\theta})). \quad (4)$$

For $\lambda = 1$ this directly minimizes $D_{\mathrm{KL}}(q(\boldsymbol{\theta}|\tau)\,\|\,p(\boldsymbol{\theta}|\mathcal{D}))$ and thus for sufficiently rich variational families will closely approximate the true Bayes posterior $p(\boldsymbol{\theta}|\mathcal{D})$. However, in practice researchers discovered that using values $\lambda < 1$ provides better predictive performance, with common values shown in the following table.[4]

?

| Reference | KL term weight $\lambda$ in (4) |
|---|---|
| (Zhang et al., 2018) | $\lambda \in \{1/2, 1/10\}$ |
| (Bae et al., 2018) | tuning of $\lambda$, unspecified |
| (Sun et al., 2019) | $\lambda = 1/n$ |
| (Osawa et al., 2019) | $\lambda \in \{1/5, 1/10\}$ |
| (Ashukha et al., 2020) | $\lambda$ from $10^{-5}$ to $10^{-3}$ |

In Appendix E we show that the KL-weighted ELBO (4) arises from tempering the likelihood part of the posterior.

From the above list we can see that the cold posterior problem has left a trail in the literature, and in fact we are not aware of *any* published work demonstrating well-performing Bayesian deep learning at temperature $T = 1$. We now give details on how we perform accurate Bayesian posterior inference in deep learning models.

## 3. Bayesian Deep Learning in Practice

In this section we describe how we achieve efficient and accurate simulation of Bayesian neural network posteriors. This section does not contain any major novel contribution but instead combines existing work.

### 3.1. Posterior Simulation using Langevin Dynamics

To generate approximate parameter samples $\boldsymbol{\theta} \sim p(\boldsymbol{\theta}\,|\,\mathcal{D})$ we consider *Langevin dynamics* over parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ and momenta $\mathbf{m} \in \mathbb{R}^d$, defined by the Langevin stochastic differential equation (SDE),

$$\mathrm{d}\,\boldsymbol{\theta} = \mathbf{M}^{-1}\,\mathbf{m}\,\mathrm{d}t, \quad (5)$$

$$\mathrm{d}\,\mathbf{m} = -\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})\,\mathrm{d}t - \gamma\mathbf{m}\,\mathrm{d}t + \sqrt{2\gamma \underline{T}}\,\mathbf{M}^{1/2}\,\mathrm{d}\mathbf{W}. \quad (6)$$

Here $U(\boldsymbol{\theta})$ is the *posterior energy* defined in (2), and $T > 0$ is the *temperature*. We use $\mathbf{W}$ to denote a standard multivariate Wiener process, which we can loosely understand as a generalized Gaussian distribution (Särkkä & Solin, 2019; Leimkuhler & Matthews, 2016). The *mass matrix* $\mathbf{M}$ is a preconditioner, and if we use no preconditioner then $\mathbf{M} = I$, such that all $\mathbf{M}$-related terms vanish from the equations. The *friction* parameter $\gamma > 0$ controls both the strength of coupling between the moments $\mathbf{m}$ and parameters $\boldsymbol{\theta}$ as well as the amount of injected noise (Langevin, 1908; Leimkuhler & Matthews, 2016). For any friction $\gamma > 0$ the SDE (5–6) has the same limiting distribution, but the choice of friction *does* affect the speed of convergence to this distribution. Simulating the continuous Langevin SDE (5–6) produces a trajectory distributed according to $\exp(-U(\boldsymbol{\theta})/T)$ and the Bayes posterior is recovered for $T = 1$.

[2] E.g., using a Normal prior with temperature $T$ results in a Normal distribution with scaled variance by a factor of $T$.

[3] For (Li et al., 2016) the tempering with $T = 1/\sqrt{n}$ arises due to an implementation mistake. For (Heek & Kalchbrenner, 2020) we communicated with the authors, and tempering arises due to overcounting data by a factor of 5, approximately justified by data augmentation, corresponding to $T = 1/5$. For (Zhang et al., 2020) the original implementation contains inadvertent tempering, however, the authors added a study of tempering in a revision.

[4] For (Osawa et al., 2019) scaling with $\lambda$ arises due to their use of a "data augmentation factor" $\rho \in \{5, 10\}$.

So, tempering scales the noise term in SG-MCMC with T? A

### 3.2. Stochastic Gradient MCMC (SG-MCMC)

Bayesian inference now corresponds to simulating the above SDE (5–6) and this requires numerical discretization. For efficiency *stochastic gradient Markov chain Monte Carlo* (SG-MCMC) methods further approximate $\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$ with a minibatch gradient (Welling & Teh, 2011; Chen et al., 2014). For a minibatch $B \subset \{1, 2, \ldots, n\}$ we first compute the minibatch average gradient $\tilde{G}(\boldsymbol{\theta})$,

$$\nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta}) := -\frac{1}{|B|} \sum_{i \in B} \nabla_{\boldsymbol{\theta}} \log p(y_i | x_i, \boldsymbol{\theta}) - \frac{1}{n} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}), \tag{7}$$

and approximate $\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$ with the unbiased estimate $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}) = n \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta})$. Here $|B|$ is the minibatch size and $n$ is the training set size; in particular, note that the log prior scales with $1/n$ regardless of the batch size.

The SDE (5–6) is defined in continuous time ($\mathrm{d}t$), and in order to solve the dynamics numerically we have to discretize the time domain (Särkkä & Solin, 2019). In this work we use a simple first-order symplectic Euler discretization, (Leimkuhler & Matthews, 2016), as first proposed for (5–6) by (Chen et al., 2014). Recent work has used more sophisticated discretizations, (Chen et al., 2015; Shang et al., 2015; Heber et al., 2019; Heek & Kalchbrenner, 2020). Applying the symplectic Euler scheme to (5–6) gives the discrete time update equations,

$$\begin{aligned}
\mathbf{m}^{(t)} &= (1 - h\gamma)\, \mathbf{m}^{(t-1)} - hn \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta}^{(t-1)}) && (8)\\
&\quad + \sqrt{2\gamma h T}\, \mathbf{M}^{1/2}\, \mathbf{R}^{(t)}, && (9)\\
\boldsymbol{\theta}^{(t)} &= \boldsymbol{\theta}^{(t-1)} + h\, \mathbf{M}^{-1} \mathbf{m}^{(t)}, && (10)
\end{aligned}$$

where $\mathbf{R}^{(t)} \sim \mathcal{N}_d(0, I_d)$ is a standard Normal vector.

In (8–10), the parameterization is in terms of step size $h$ and friction $\gamma$. These quantities are different from typical SGD parameters. In Appendix B we establish an exact correspondence between the SGD learning rate $\ell$ and momentum decay parameters $\beta$ and SG-MCMC parameters. For the symplectic Euler discretization of Langevin dynamics, we derive this relationship as $h := \sqrt{\ell/n}$, and $\gamma := (1 - \beta)\sqrt{n/\ell}$, where $n$ is the total training set size.

### 3.3. Accurate SG-MCMC Simulation

In practice there remain two sources of error when following the dynamics (8–10):

- *Minibatch noise*: $\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta})$ is an unbiased estimate of $\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$ but contains additional estimation variance.
- *Discretization error*: we incur error by following a continuous-time path (5–6) using discrete steps (8–10).

We use two methods to reduce these errors: *preconditioning* and *cyclical time stepping*.

SGHMC?

**Layerwise Preconditioning.** Preconditioning through a choice of matrix $\mathbf{M}$ is a common way to improve the behavior of optimization methods. Li et al. (2016) and Ma et al. (2015) proposed preconditioning for SG-MCMC methods, and in the context of molecular dynamics the use of a matrix $\mathbf{M}$ has a long tradition as well, (Leimkuhler & Matthews, 2016). Li's proposal is an adaptive preconditioner inspired by RMSprop, (Tieleman & Hinton, 2012). Unfortunately, using the discretized Langevin dynamics with a preconditioner $\mathbf{M}(\boldsymbol{\theta})$ that depends on $\boldsymbol{\theta}$ compromises the correctness of the dynamics.[5] We propose a simpler preconditioner that limits the frequency of adaptating $\mathbf{M}$: after a number of iterations we estimate a new preconditioner $\mathbf{M}$ using a small number of batches, say 32, but without updating any model parameters. This preconditioner then remains fixed for a number of iterations, for example, the number of iterations it takes to visit the training set once, i.e. one epoch. We found this strategy to be highly effective at improving simulation accuracy. For details, please see Appendix D.

**Cyclical time stepping.** The second method to improve simulation accuracy is to decrease the discretization step size $h$. Chen et al. (2015) studied the consequence of both minibatch noise and discretization error on simulation accuracy and showed that the overall simulation error goes to zero for $h \searrow 0$. While lowering the step size $h$ to a small value would also make the method slow, recently Zhang et al. (2020) propose to perform *cycles* of iterations $t = 1, 2, \ldots$ with a high-to-low step size schedule $h_0\, C(t)$ described by an initial step size $h_0$ and a function $C(t)$ that starts at $C(1) = 1$ and has $C(L) = 0$ for a cycle length of $L$ iterations. Such cycles retain fast simulation speed in the beginning while accepting simulation error. Towards the end of each cycle however, a small step size ensures an accurate simulation. We use the cosine schedule from (Zhang et al., 2020) for $C(t)$, see Appendix A.

We integrate these two techniques together into a practical SG-MCMC procedure, Algorithm 1. When no preconditioning and no cosine schedule is used ($\mathbf{M} = I$ and $C(t) = 1$ in all iterations) and $T(t) = 0$ this algorithm is equivalent to *Tensorflow*'s SGD with momentum (Appendix C).

Coming back to the Cold Posteriors effect, what could explain the poor performance at temperature $T = 1$? With our Bayesian hearts, there are only three possible areas to examine: the inference, the prior, or the likelihood function.

## 4. Inference: Is it Accurate?

Both the Bayes posterior and the cooled posteriors are all intractable. Moreover, it is plausible that the high-dimensional posterior landscape of a deep network may lead to difficult-

[5] Li et al. (2016) derives the required correction term, which however is expensive to compute and omitted in practice.

**Algorithm 1:** Symplectic Euler Langevin scheme.

```
Function SymEulerSGMCMC($\tilde{G}$, $\boldsymbol{\theta}^{(0)}$, $\ell$, $\beta$, $n$, $T$)
    Input: $\tilde{G} : \Theta \to \mathbb{R}$ mean energy function estimate;
           $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^d$ initial parameter; $\ell > 0$ learning
           rate; $\beta \in [0, 1)$ momentum decay; $n$ total
           training set size; $T(t) \geq 0$ temperature
           schedule
    Output: Sequence $\boldsymbol{\theta}^{(t)}$, $t = 1, 2, \ldots$
    $h_0 \leftarrow \sqrt{\ell/n}$                         // SDE time step
    $\gamma \leftarrow (1 - \beta)\sqrt{n/\ell}$              // friction
    Sample $\mathbf{m}^{(0)} \sim \mathcal{N}_d(0, I_d)$
    $\mathbf{M} \leftarrow I$                               // Initial $\mathbf{M}$
    for $t = 1, 2, \ldots$ do
        if new epoch then
            $\mathbf{m}_c \leftarrow \mathbf{M}^{-1/2}\,\mathbf{m}^{(t-1)}$
            $\mathbf{M} \leftarrow$ EstimateM($\tilde{G}, \boldsymbol{\theta}^{(t-1)}$)
            $\mathbf{m}^{(t-1)} \leftarrow \mathbf{M}^{1/2}\,\mathbf{m}_c$
        $h \leftarrow C(t)\,h_0$          // Cyclic modulation
        Sample $\mathbf{R}^{(t)} \sim \mathcal{N}_d(0, I_d)$     // noise
        $\mathbf{m}^{(t)} \leftarrow (1 - h\gamma)\,\mathbf{m}^{(t-1)} - hn\nabla_{\boldsymbol{\theta}}\tilde{G}(\boldsymbol{\theta}^{(t-1)}) +$
            $\sqrt{2\gamma h T(t)}\,\mathbf{M}^{1/2}\,\mathbf{R}^{(t)}$
        $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + h\,\mathbf{M}^{-1}\mathbf{m}^{(t)}$
        if end of cycle then
            yield $\boldsymbol{\theta}^{(t)}$          // Parameter sample
```

to-simulate SDE dynamics (5–6). Our approximate SG-MCMC inference method further has to deal with minibatch noise and produces only a finite sample approximation to the predictive integral (3). Taken together, could the Cold Posteriors effect arise from a poor inference accuracy?

### 4.1. Hypothesis: Inaccurate SDE Simulation

**Inaccurate SDE Simulation Hypothesis**: the SDE (5–6) is poorly simulated.

To gain confidence that our SG-MCMC method simulates the posterior accurately, we introduce diagnostics that previously have not been used in the SG-MCMC context:

- **Kinetic temperatures** (Appendix H.1): we report per-variable statistics derived from the moments $\mathbf{m}$. For these so called *kinetic temperatures* we know the exact sampling distribution under Langevin dynamics and compute their 99% confidence intervals.
- **Configurational temperatures** (Appendix H.2): we report per-variable statistics derived from $\langle \boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) \rangle$. For these *configurational temperatures* we know the expected value under Langevin dynamics.

We propose to use these diagnostics to assess simulation accuracy of SG-MCMC methods. We introduce the diagnostics and our new results in detail in Appendix H.

**Inference Diagnostics Experiment:** In Appendix I we report a detailed study of simulation accuracy for both models. This study reports accurate simulation for both models when both preconditioning and cyclic time stepping are used. We can therefore with reasonably high confidence rule out a poor simulation of the SDE. All remaining experiments in this paper also pass the simulation accuracy diagnostics.

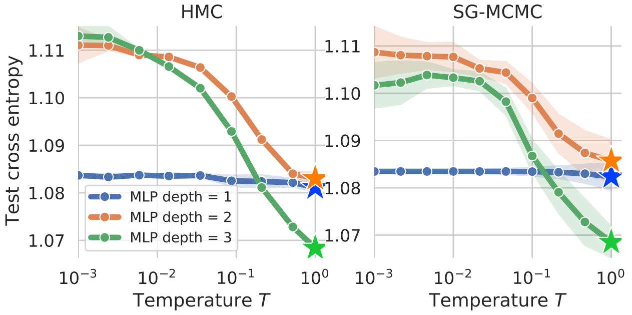


*Figure 4.* HMC (left) agrees closely with SG-MCMC (right) for synthetic data on multilayer perceptrons. A star indicates the optimal temperature for each model: for the synthetic data sampled from the prior there are no cold posteriors and both sampling methods perform best at $T = 1$.

### 4.2. Hypothesis: Biased SG-MCMC

**Biased SG-MCMC Hypothesis**: Lack of accept/reject Metropolis-Hastings corrections in SG-MCMC introduces bias.

In Markov chain Monte Carlo it is common to use an additional accept-reject step that corrects for bias in the sampling procedure. For MCMC applied to deep learning this correction step is too expensive and therefore omitted in SG-MCMC methods, which is valid for small time steps only, (Chen et al., 2015). If accept-reject is computationally feasible the resulting procedure is called *Hamiltonian Monte Carlo* (HMC) (Neal et al., 2011; Betancourt & Girolami, 2015; Duane et al., 1987; Hoffman & Gelman, 2014). Because it provides unbiased simulation, we can consider HMC the *gold standard*, (Neal, 1995). We now compare gold standard HMC against SG-MCMC on a small example where comparison is feasible. We provide details of our HMC setup in Appendix M.

**HMC Experiment:** we construct a simple setup using a multilayer perceptron (MLP) where by construction $T = 1$ is optimal; such Bayes optimality must hold in expectation if the data is generated by the prior and model that we use for inference, (Berger, 1985). Thus, we can ensure that if the cold posterior effect is observed it must be due to a problem in our inference method. We perform all inference without minibatching ($|B| = n$) and test MLPs of varying number of one to three layers, ten hidden units each, and using the ReLU activation. As HMC implementation we use `tfp.mcmc.HamiltonianMonteCarlo` from *Tensorflow Probability* (Dillon et al., 2017; Lao et al., 2020): Details for our data and HMC are in Appendix L–M.

W?

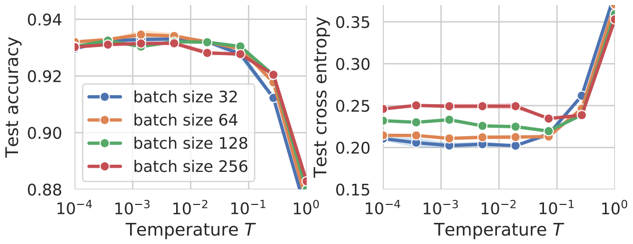


*Figure 5.* Batch size dependence of the ResNet-20/CIFAR-10 ensemble performance, reporting mean and standard error (3 runs): for all batch sizes the optimal predictions are obtained for $T < 1$.

In Figure 4 the SG-MCMC results agree very well with the HMC results with optimal predictions at $T = 1$, i.e. no cold posteriors are present. For the cases tested we conclude that SG-MCMC is almost as accurate as HMC and the lack of accept-reject correction cannot explain cold posteriors.

### 4.3. Hypothesis: Stochastic Gradient Noise

> **Minibatch Noise Hypothesis**: gradient noise from minibatching causes inaccurate sampling at $T = 1$.

Gradient noise due to minibatching can be heavy-tailed and non-Gaussian even for large batch sizes, (Simsekli et al., 2019). Our SG-MCMC method is only justified if the effect of noise will diminish for small time steps. We therefore study the influence of batch size on predictive performance through the following experiment.

**Batchsize Experiment:** we repeat the original ResNet-20/CIFAR-10 experiment at different temperatures for batch sizes in $\{32, 64, 128, 256\}$ and study the variation of the predictive performance as a function of batch size. Figure 5 and Figure 6 show that while there is a small variation between different batch sizes $T < 1$ remains optimal for all batch sizes. Therefore minibatch noise alone cannot explain the observed poor performance at $T = 1$.

For both ResNet and CNN-LSTM the best cross-entropy is achieved by the smallest batch size of 32 and 16, respectively. The smallest batch size has the *largest* gradient noise. We can interpret this noise as an additional heat source that increases the effective simulation temperature. However, the noise distribution arising from minibatching is anisotropic, (Zhu et al., 2019), and this could perhaps aid generalization. We will not study this hypothesis further here.

### 4.4. Hypothesis: Bias-Variance Trade-off

> **Bias-variance Tradeoff Hypothesis**: For $T = 1$ the posterior is diverse and there is high variance between model predictions. For $T \ll 1$ we sample nearby modes and reduce prediction variance but increase bias; the variance dominates the error and reducing variance ($T \ll 1$) improves predictive performance.

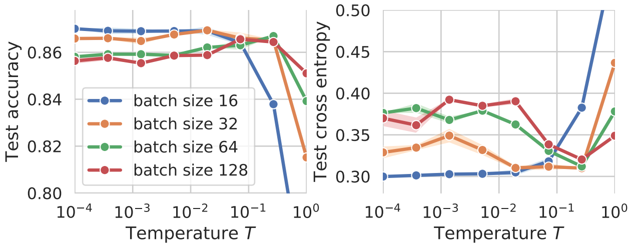


*Figure 6.* Batch size dependence of the CNN-LSTM/IMDB ensemble performance, reporting mean and standard error (3 runs): for all batch sizes, the optimal performance is achieved at $T < 1$.

If this hypothesis were true then simply collecting more ensemble members, $S \to \infty$, would reduce the variance to arbitrary small values and thus fix the poor predictive performance we observe at $T = 1$. Doing so would require running our SG-MCMC schemes for longer—potentially for much longer. We study this question in detail in Appendix F and conclude by an asymptotic analysis that the amount of variance cannot explain cold posteriors.

## 5. Why Could the Bayes Posterior be Poor?

With some confidence in our approximate inference procedure what are the remaining possibilities that could explain the cold posterior effect? The remaining two places to look at are the likelihood function and the prior.

### 5.1. Problems in the Likelihood Function?

For Bayesian deep learning we use the same likelihood function $p(y|x, \boldsymbol{\theta})$ as we use for SGD. Therefore, because the same likelihood function works well for SGD it appears an unlikely candidate to explain the cold posterior effect. However, current deep learning models use a number of techniques—such as data augmentation, dropout, and batch normalization—that are not formal likelihood functions. This observations brings us to the following hypothesis. ?

> **Dirty Likelihood Hypothesis**: Deep learning practices that violate the likelihood principle (batch normalization, dropout, data augmentation) cause deviation from the Bayes posterior.

In Appendix J we give a theory of "*Jensen posteriors*" which describes the likelihood-like functions arising from modern deep learning techniques. We report an experiment (Appendix J.4) that—while slightly inconclusive—demonstrates that cold posteriors remain when a clean likelihood is used in a suitably modified ResNet model; the CNN-LSTM model already had a clean likelihood function.

### 5.2. Problems with the Prior $p(\boldsymbol{\theta})$?

So far we have used a simple Normal prior, $p(\boldsymbol{\theta}) = \mathcal{N}(0, I)$, as was done in prior work (Zhang et al., 2020; Heek &

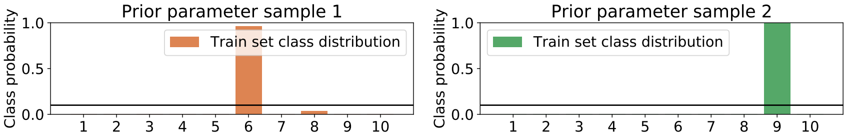


*Figure 7.* ResNet-20/CIFAR-10 typical prior predictive distributions for 10 classes under a $\mathcal{N}(0, I)$ prior averaged over the entire training set, $\mathbb{E}_{x \sim p(x)}[p(y|x, \boldsymbol{\theta}^{(i)})]$. Each plot is for one sample $\boldsymbol{\theta}^{(i)} \sim \mathcal{N}(0, I)$ from the prior. Given a sample $\boldsymbol{\theta}^{(i)}$ the average training data class distribution is highly concentrated around the same classes for all $x$.

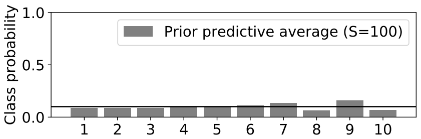


*Figure 8.* ResNet-20/CIFAR-10 prior predictive $\mathbb{E}_{x \sim p(x)}[\mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta})}[p(y|x, \boldsymbol{\theta})]]$ over 10 classes, estimated using $S = 100$ prior samples $\boldsymbol{\theta}^{(i)}$ and all training images.

Kalchbrenner, 2020; Ding et al., 2014; Li et al., 2016; Zhang et al., 2018). But is this a good prior?

One could hope, that perhaps with an informed and structured model architecture, a simple prior could be sufficient in placing prior beliefs on suitable functions, as argued by Wilson (2019). While plausible, we are mildly cautious because there are known examples where innocent looking priors have turned out to be unintentionally highly informative.[6] Therefore, with the cold posterior effect having a track record in the literature, perhaps $p(\boldsymbol{\theta}) = \mathcal{N}(0, I)$ could have similarly unintended effects of placing large prior mass on undesirable functions. This leads us to the next hypothesis.

**Bad Prior Hypothesis**: The current priors used for BNN parameters are inadequate, unintentionally informative, and their effect becomes stronger with increasing model depths and capacity.

To study the quality of our prior, we study typical functions obtained by sampling from the prior, as is good practice in model criticism, (Gelman et al., 2013).

**Prior Predictive Experiment:** for our ResNet-20 model we generate samples $\boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta}) = \mathcal{N}(0, I)$ and look at the induced predictive distribution $\mathbb{E}_{x \sim p(x)}[p(y|x, \boldsymbol{\theta}^{(i)})]$ for each parameter sample, using the real CIFAR-10 training images. From Figure 7 we see that typical prior draws produce concentrated class distributions, indicating that the $\mathcal{N}(0, I)$ distribution is a poor prior for the ResNet-20 likelihood. From Figure 8 we can see that the average predictions obtained from such concentrated functions remain close to the uniform class distribution. Taken together, from a subjective Bayesian view $p(\boldsymbol{\theta}) = \mathcal{N}(0, I)$ is a *poor prior*: typical functions produced by this prior place a high probability the same few classes for all $x$. In Appendix K we carry out another prior predictive study using He-scaling priors, (He et al., 2015), which leads to similar results.

**Training Set Size $n$ Scaling Experiment:** the posterior energy $U(\boldsymbol{\theta})$ in (2) sums over all $n$ data log-likelihoods but adds $\log p(\boldsymbol{\theta})$ only once. This means that the influence of $\log p(\boldsymbol{\theta})$ vanishes at a rate of $1/n$ and thus the prior will exert its strongest influence for small $n$. We now study what happens for small $n$ by comparing the Bayes predictive under a $\mathcal{N}(0, I)$ prior against performing SGD maximum a posteriori (MAP) estimation on the *same* log-posterior.[7]

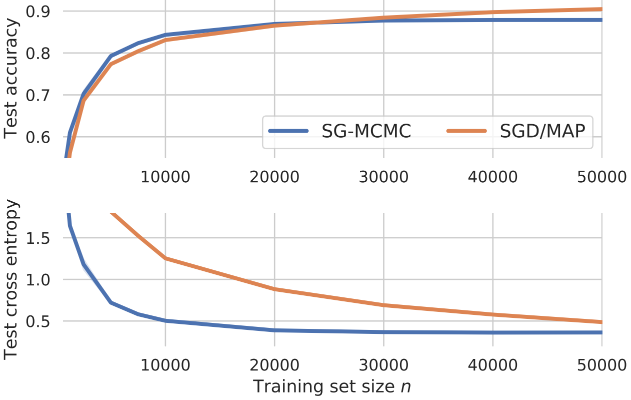


*Figure 9.* ResNet-20/CIFAR-10 predictive performance as a function of training set size $n$. The Bayes posterior ($T = 1$) degrades gracefully as $n$ decreases, whereas SGD/MAP performs worse.

Figure 9 and Figure 10 show the predictive performance for ResNet-20 on CIFAR-10 and CNN-LSTM on IMDB, respectively. These results differ markedly between the two models and datasets: for ResNet-20 / CIFAR-10 the Bayes posterior at $T = 1$ degrades gracefully for small $n$, whereas SGD suffers large losses in test cross-entropy for small $n$. For CNN-LSTM / IMDB predictions from the Bayes posterior at $T = 1$ deteriorate quickly in both test accuracy and cross entropy. In all these runs SG-MCMC and SGD/MAP work with the same $U(\boldsymbol{\theta})$ and the difference is between integration and optimization. The results are inconclusive but somewhat implicate the prior in the cold posterior effect: as $n$ becomes small there is an increasing difference between the cross-entropy achieved by the Bayes prediction and the SGD estimate, for large $n$ the SGD estimate performs better.

**Capacity Experiment:** we consider a MLP using a $\mathcal{N}(0, I)$ prior and study the relation of the network capacity to the cold posterior effect. We train MLPs of varying depth (number of layers) and width (number of units per layer) at different temperatures on CIFAR-10. Figure 11 shows that for increasing capacity the cold posterior effect becomes more prominent. This indicates a connection between model capacity and strength of the cold posterior effect.

[6] A shocking example in the Dirichlet-Multinomial model is given by Nemenman et al. (2002). Importantly the unintended effect of the prior was not recognized when the model was originally proposed by Wolpert & Wolf (1995).

[7] For SGD we minimize $U(\boldsymbol{\theta})/n$.

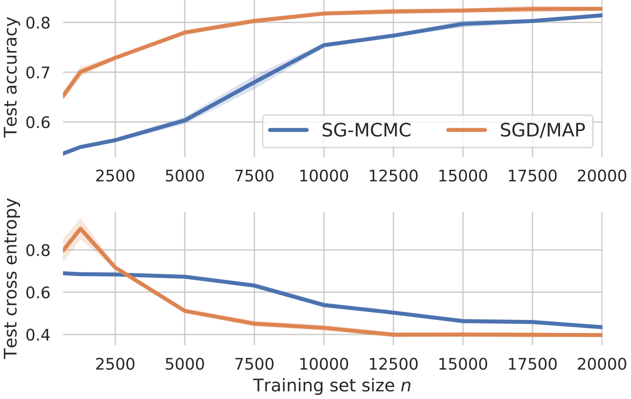


*Figure 10.* CNN-LSTM/IMDB predictive performance as a function of training set size $n$. The Bayes posterior ($T = 1$) suffers more than the SGD performance, indicating a problematic prior.

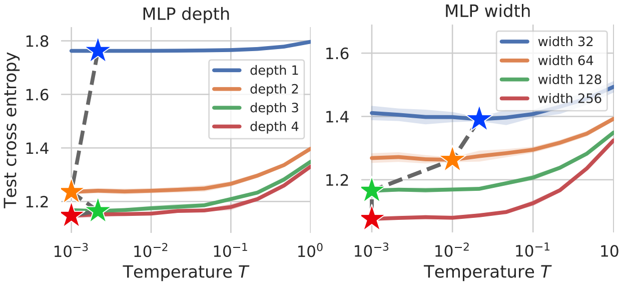


*Figure 11.* MLP of different capacities (depth and width) on CIFAR-10. Left: we fix the width to 128 and vary the depth. Right: we fix the depth to 3 and vary the width. Increasing capacity lowers the optimal temperature.

### 5.3. Inductive Bias due to SGD?

> **Implicit Initialization Prior in SGD**: The inductive bias from initialization is strong and beneficial for SGD but harmed by SG-MCMC sampling.

Optimizing neural networks via SGD with a suitable initialization is known to have a beneficial inductive bias leading to good local optima, (Masters & Luschi, 2018; Mandt et al., 2017). Does SG-MCMC perform worse due to decreasing the influence of that bias? We address this question by the following experiment. We first run SGD until convergence, then switch over to SG-MCMC sampling for 500 epochs (10 cycles), and finally switch back to SGD again. Figure 12 shows that SGD initialized by the last model of the SG-MCMC sampling dynamics recovers the same performance as vanilla SGD. This indicates that the beneficial initialization bias for SGD is not destroyed by SG-MCMC. Details can be found in Appendix G.

## 6. Related Work on Tempered Posteriors

Statisticians have studied *tempered* or *fractional* posteriors for $T > 1$. Motivated by the behavior of Bayesian inference in *misspecified* models (Grünwald et al., 2017; Jansen, 2013) develop the *SafeBayes* approach and Bhattacharya et al. (2019) develops *fractional posteriors* with the goal of slowing posterior concentration. The use of multiple temperatures $T > 1$ is also common in Monte Carlo simulation in the presence of rough energy landscapes, e.g. (Earl & Deem, 2005; Sugita & Okamoto, 1999; Swendsen & Wang, 1986). However, the purpose of such tempering is to aid in accurate sampling at a desired target temperature, but not in changing the target distribution.

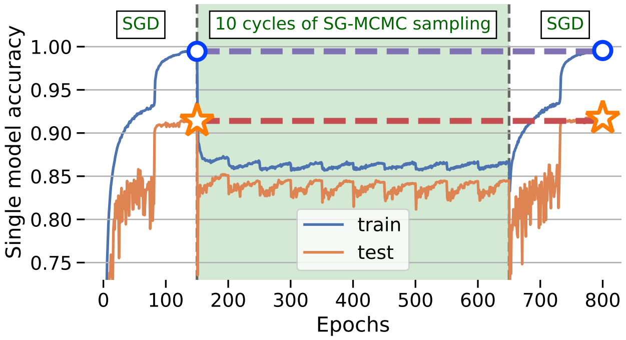


*Figure 12.* Do the SG-MCMC dynamics harm a beneficial initialization bias used by SGD? We first train a ResNet-20 on CIFAR-10 via SGD, then switch over to SG-MCMC sampling and finally switch back to SGD optimization. We report the single-model test accuracy of SGD and the SG-MCMC chain as function of epochs. SGD recovers from being initialized by the SG-MCMC state.

## 7. Conclusion

Our work has raised the question of cold posteriors but we did not fully resolve nor fix the cause for the cold posterior phenomenon. Yet our experiments suggest the following.

**SG-MCMC is accurate enough:** our experiments (Section 4–5) and novel diagnostics (Appendix H) indicate that current SG-MCMC methods are robust, scalable, and accurate enough to provide good approximations to parameter posteriors in deep nets.

**Cold posteriors work:** while we do not fully understand cold posteriors, tempered SG-MCMC ensembles provide a way to train ensemble models with improved predictions compared to individual models. However, taking into account the added computation from evaluating ensembles, there may be more practical methods, (Lakshminarayanan et al., 2017; Wen et al., 2019; Ashukha et al., 2020).

**More work on priors for deep nets is needed:** the experiments in Section 5.2 implicate the prior $p(\boldsymbol{\theta})$ in the cold posterior effect, although the prior may not be the only cause. Further investigations in Appendix K fail to produce a “simple” fix based on known scaling laws in deep networks. Future work on suitable priors for Bayesian neural networks is needed, building on recent advances, (Sun et al., 2019; Pearce et al., 2019; Flam-Shepherd et al., 2017; Hafner et al., 2018).

# A. Model Details

We now give details regarding the models we use in all our experiments. We use Tensorflow version 2.1 and carry out all experiments on Nvidia P100 accelerators.

## A.1. ResNet-20 CIFAR-10 Model

We use the CIFAR-10 dataset from (Krizhevsky et al., 2009), in “version 3.0.0” provided in Tensorflow Datasets.[8] We use the Tensorflow Datasets training/testing split of 50,000 and 10,000 images, respectively.

We use the ResNet-20 model from `https://keras.io/examples/cifar10_resnet/` as a starting point. For our SGD baseline we use the exact same setup as in the Keras example (200 epochs, learning rate schedule, SGD with Nesterov acceleration). Notably the Keras example uses bias terms in all convolution layers, whereas some other implementations do not.

The Keras example page reports a reference test accuracy of 92.16 percent for the CIFAR-10 model, compared to our 92.22 percent accuracy. This is consistent with the larger literature, collected for example at `https://github.com/google/edward2/tree/master/baselines/cifar10`, with even higher accuracy achieved for variations of the ResNet model such as using wide layers, removing bias terms in the convolution layers, or additional regularization.

In this paper we study the phenomenon of poor $T = 1$ posteriors obtained by SG-MCMC and therefore use an accurate simulation and sampling setup at the cost of runtime. In order to obtain accurate simulations we use the following settings for SG-MCMC in every experiment, except where noted otherwise:

- Number of epochs: 1500
- Initial learning rate: $\ell = 0.1$
- Momentum decay: $\beta = 0.98$
- Batch size: $|B| = 128$
- Sampling start: begin at epoch 150
- Cycle length: 50
- Cycle schedule: cosine
- Prior: $p(\boldsymbol{\theta}) = \mathcal{N}(0, I)$

For experiments on CIFAR-10 we use data augmentation as follows:

- random left/right flipping of the input image;
- border-padding by zero values, four pixels in horizontal and vertical direction, followed by a random cropping of the image to its original size.

We visualize the cyclic schedule used in our ResNet-20 CIFAR-10 experiments in Figure 13.

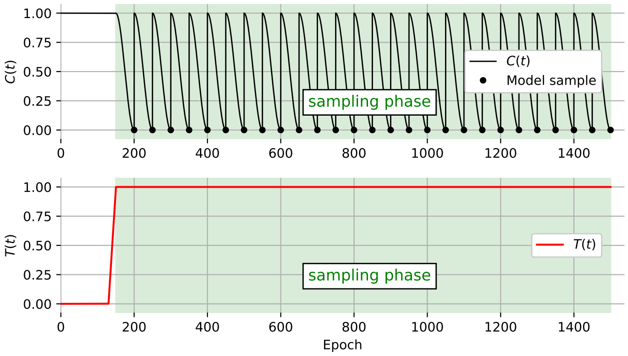


*Figure 13.* Cyclical time stepping $C(t)$, and temperature ramp-up $T(t)$, as proposed by Zhang et al. (2020) and used in Algorithm 1, for our ResNet-20 CIFAR-10 model (Section A.1). We sample one model at the end of each cycle when the inference accuracy is best, obtaining an ensemble of 27 models.

## A.2. ResNet-20 CIFAR-10 SGD Baseline

For the SGD baseline we follow the best practice from the existing Keras example which was tuned for generalization performance. In particular we use:

- Number of epochs: 200
- Initial learning rate: $\ell = 0.1$
- Momentum term: 0.9
- L2 regularization coefficient: 0.002
- Batch size: 128
- Optimizer: SGD with Nesterov momentum
- Learning rate schedule (epoch, $\ell$-multiplier): $(80, 0.1)$, $(120, 0.01)$, $(160, 0.001)$, $(180, 0.0005)$.

Data augmentation is the same as described in Section A.1. We report the final validation performance and over the 200 epochs do not observe any overfitting.

## A.3. CNN-LSTM IMDB Model

We use the IMDB sentiment classification text dataset provided by the `tensorflow.keras.datasets` API in Tensorflow version 2.1. We use 20,000 words and a maximum sequence length of 100 tokens. We use 20,000 training sequences and 25,000 testing sequences.

We use the CNN-LSTM example[9] as a starting point. For our SGD baseline we use the Keras model but add a prior $p(\theta) = \mathcal{N}(0, I)$ as used for the Bayesian posterior. We then use the Tensorflow SGD implementation to optimize the resulting $U(\theta)$ function. For SGD the model overfits and we therefore report the best end-of-epoch test accuracy and

[8] See `https://www.tensorflow.org/datasets/catalog/cifar10`

[9] Available at `https://github.com/keras-team/keras/blob/master/examples/imdb_cnn_lstm.py`

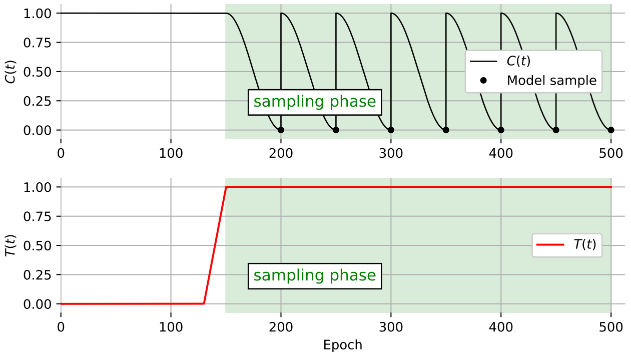


*Figure 14.* Cyclical time stepping $C(t)$, and temperature ramp-up $T(t)$ for our CNN-LSTM IMDB model (Section A.3). We sample one model at the end of each cycle when the inference accuracy is best, obtaining an ensemble of 7 models.

test cross-entropy achieved.

For all experiments, except where explicitly noted otherwise, we use the following parameters:

- Number of epochs: 500
- Initial learning rate: $\ell = 0.1$
- Momentum decay: $\beta = 0.98$
- Batch size: $|B| = 32$
- Sampling start: begin at epoch 50
- Cycle length: 25
- Cycle schedule: cosine
- Prior: $p(\boldsymbol{\theta}) = \mathcal{N}(0, I)$

We visualize the cyclic schedule used in our CNN-LSTM IMDB experiments in Figure 14.

### A.4. CNN-LSTM IMDB SGD Baseline

The SGD baseline follows the Keras example settings:

- Number of epochs: 50
- Initial learning rate: $\ell = 0.1$
- Momentum term: 0.98
- Regularization: MAP with $\mathcal{N}(0, I)$ prior
- Batch size: 32
- Optimizer: SGD with Nesterov momentum
- Learning rate schedule: None

We report the optimal test set performance from all end-of-epoch test evaluations. This is necessary because there is significant overfitting after the first ten epochs.

## B. Deep Learning Parameterization of SG-MCMC Methods

We derive the bijection between (learning rate $\ell$, momentum decay $\beta$) and (timestep $h$, friction $\gamma$) by considering the *instantaneous gradient effect* $\alpha$ on the parameter, i.e. the amount by which the current gradient at time $t$ affects the current gradient update update at time $t$. We set $\alpha = \ell/n$, where $\ell$ is the familiar learning rate parameter used in SGD and the factor $1/n$ is to convert $\nabla_{\boldsymbol{\theta}} U$ to $\nabla_{\boldsymbol{\theta}} G$, as $\nabla_{\boldsymbol{\theta}} G = \nabla_{\boldsymbol{\theta}} U/n$ is the familiar minibatch mean gradient. Likewise, the *momentum decay* is the factor $\beta < 1$ by which the momentum vector $\mathbf{m}^{(t)}$ is shrunk in each discretized time step. Having determined $\alpha$ and $\beta$ we can derive two non-linear equations that depend on the particular time discretization used; for the symplectic Euler Langevin scheme these are

$$h^2 = \alpha \quad \left(= \frac{\ell}{n}\right), \qquad \text{and} \qquad 1 - h\gamma = \beta. \tag{11}$$

Solving these equations for $h$ and $\gamma$ simultaneously, given $\ell$, $n$, and $\beta$ yields the bijection

$$h = \sqrt{\ell/n}, \tag{12}$$

$$\gamma = (1-\beta)\sqrt{n/\ell}. \tag{13}$$

**Algorithm 2:** Stochastic Gradient Descent with Momentum (SGD) in Tensorflow.

```
Function SGD ($\tilde{G}$, $\boldsymbol{\theta}^{(0)}$, $\ell$, $\beta$)
    Input: $\tilde{G} : \Theta \to \mathbb{R}$ average batch loss function, cf
           equation (7); $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^d$ initial parameter;
           $\ell > 0$ learning rate parameter; $\beta \in [0, 1)$
           momentum decay parameter.
    Output: Parameter sequence $\boldsymbol{\theta}^{(t)}$, at step $t = 1, 2, \ldots$
    $\mathbf{m}^{(0)} \leftarrow \mathbf{0}$        // Initialize momentum
    for $t = 1, 2, \ldots$ do
        $\mathbf{m}^{(t)} \leftarrow \beta\, \mathbf{m}^{(t-1)} - \ell\, \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta}^{(t-1)})$
                                        // Update momentum
        $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + \mathbf{m}^{(t)}$        // Update
         parameters
        yield $\boldsymbol{\theta}^{(t)}$        // Parameter at step $t$
```

## C. Connection to Stochastic Gradient Descent (SGD)

We now give a precise connection between stochastic gradient descent (SGD) and the symplectic Euler SG-MCMC method, Algorithm 1 from the main paper.

Algorithm 2 gives the stochastic gradient descent (SGD) with momentum algorithm as implemented in *Tensorflow*'s version 2.1 optimization methods, `tensorflow.keras.optimizers.SGD` and `tensorflow.train.MomentumOptimizer`, (Abadi et al., 2016).

Starting with Algorithm 2 we first perform an equivalent substitution of the moments,

$$\tilde{\mathbf{m}}^{(t)} := \sqrt{\frac{n}{\ell}}\,\mathbf{m}^{(t)}, \qquad \text{respectively,} \tag{14}$$

$$\mathbf{m}^{(t)} := \sqrt{\frac{\ell}{n}}\,\tilde{\mathbf{m}}^{(t)}, \tag{15}$$

we obtain the update from line 4 in Algorithm 2,

$$\sqrt{\frac{\ell}{n}}\,\tilde{\mathbf{m}}^{(t)} \leftarrow \beta\sqrt{\frac{\ell}{n}}\,\tilde{\mathbf{m}}^{(t-1)} - \ell\,\nabla_{\boldsymbol{\theta}}\tilde{G}(\boldsymbol{\theta}^{(t-1)}). \tag{16}$$

Multiplying both sides of (16) by $\sqrt{n}/\sqrt{\ell}$ we obtain an equivalent form of Algorithm 2 with lines 4 and 5 replaced by

$$\tilde{\mathbf{m}}^{(t)} \leftarrow \beta\,\tilde{\mathbf{m}}^{(t-1)} - \sqrt{\ell n}\,\nabla_{\boldsymbol{\theta}}\tilde{G}(\boldsymbol{\theta}^{(t-1)}), \tag{17}$$

$$\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + \sqrt{\frac{\ell}{n}}\,\tilde{\mathbf{m}}^{(t)}. \tag{18}$$

From the bijection (12–13) we have $h = \sqrt{\ell/n}$ and $\gamma = (1-\beta)\sqrt{n/\ell}$. Solving for $\beta$ gives

$$\beta = 1 - \gamma\sqrt{\frac{\ell}{n}} = 1 - \gamma h. \tag{19}$$

We also have

$$\sqrt{\ell n} = \sqrt{\frac{\ell}{n}n^2} = n\sqrt{\frac{\ell}{n}} = h\,n. \tag{20}$$

Substituting (19) and (20) into (17) and (18) gives the equivalent updates

$$\tilde{\mathbf{m}}^{(t)} \leftarrow (1-\gamma h)\,\tilde{\mathbf{m}}^{(t-1)} - h\,n\,\nabla_{\boldsymbol{\theta}}\tilde{G}(\boldsymbol{\theta}^{(t-1)}), \tag{21}$$

$$\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + h\,\tilde{\mathbf{m}}^{(t)}. \tag{22}$$

These equivalent changes produce Algorithm 3. Algorithm 2 and Algorithm 3 generate equivalent trajectories $\boldsymbol{\theta}^{(t)}$, $t = 1, 2, \ldots$, but differ in the scaling of their momenta, $\mathbf{m}^{(t)}$ and $\tilde{\mathbf{m}}^{(t)}$.

Comparing lines 4–5 in Algorithm 3 with lines 13–14 in Algorithm 1 from the main paper we see that when $\mathbf{M} = I$ and $C(t) = 1$ the only remaining difference between the updates is the additional noise $\sqrt{2\gamma h T}\,\mathbf{M}^{1/2}\mathbf{R}^{(t)}$ in the SG-MCMC method. In this *precise* sense the SG-MCMC Algorithm 1 from the main paper is just "SGD with noise".

## D. Semi-Adaptive Estimation of Layerwise Preconditioner $\mathbf{M}$

During our experiments with deep learning models we noticed that both minibatch noise as well as gradient magnitudes tend to behave similar within a set of related parameters. For example, for a given learning iteration, all gradients related to convolution kernel weights of the same convolution layer of a network tend to have similar magnitudes and minibatch noise variance. At the same iteration they may be different from the magnitudes and minibatch noise variance of gradients of the parameters of another layer in the same network.

**Algorithm 3:** Stochastic Gradient Descent with Momentum (SGD), reparameterized.

```
1 Function SGDEquivalent ($\tilde{G}$, $\boldsymbol{\theta}^{(0)}$, $\ell$, $\beta$)
      Input: $\tilde{G}:\Theta\rightarrow\mathbb{R}$ average batch loss function, cf
             equation (7); $\boldsymbol{\theta}^{(0)}\in\mathbb{R}^d$ initial parameter;
             $h>0$ discretization step size parameter; $\gamma>0$
             friction parameter.
      Output: Parameter sequence $\boldsymbol{\theta}^{(t)}$, $t=1,2,\ldots$, at
              step $t$
2     $\tilde{\mathbf{m}}^{(0)}\leftarrow\mathbf{0}$           // Initialize momentum
3     for $t=1,2,\ldots$ do
4         $\tilde{\mathbf{m}}^{(t)}\leftarrow(1-\gamma h)\,\tilde{\mathbf{m}}^{(t-1)}-h\,n\,\nabla_{\boldsymbol{\theta}}\tilde{G}(\boldsymbol{\theta}^{(t-1)})$
           // Update momentum
5         $\boldsymbol{\theta}^{(t)}\leftarrow\boldsymbol{\theta}^{(t-1)}+h\,\tilde{\mathbf{m}}^{(t)}$        // Update
           parameters
6         yield $\boldsymbol{\theta}^{(t)}$        // Parameter at step t
```

Therefore, we estimate a simple diagonal preconditioner that ties together the scale of all parameter elements that belong to the same model variable. Moreover, we normalize the preconditioner so that the least sensitive variable always has scale one. With such normalization, if all variables would be equally sensitive the preconditioner becomes $\mathbf{M} = I$, the identity preconditioner.

We estimate the layerwise preconditioner using Algorithm 4.

**Updating the preconditioner.** In Langevin schemes the preconditioner couples the moment space to the parameter space. If we use a new estimate $\mathbf{M}'$ to replace the old preconditioner $\mathbf{M}$ then we change this coupling and if left unchanged then the old moments $\mathbf{m}$ would no longer have the correct distribution.[10] We therefore posit that upon changing the preconditioner the effect of the moments should remain the same. To retain the full information in the current moments we set $\mathbf{m}' = \mathbf{M}'^{1/2}\mathbf{M}^{-1/2}\mathbf{m}$ which we can understand as $\mathbf{M}'^{1/2}(\mathbf{M}^{-1/2}\mathbf{m})$, where the bracketed part canonicalizes the moments $\mathbf{m}$ to the identity preconditioner, and $\mathbf{M}'^{1/2}$ transfers the canonical moments to the new preconditioner.

[10] More precisely, $\mathbf{M}^{-1/2}\mathbf{m}$ should always be distributed according to $\mathcal{N}(0, I)$.

**Algorithm 4:** Estimate Layerwise Preconditioner.

```
Function EstimateM($\tilde{G}$, $\boldsymbol{\theta}$, $K$, $\epsilon$)
    Input: $\tilde{G}: \Theta \to \mathbb{R}$ mean energy function estimate;
        $(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_S) \in \mathbb{R}^{d_1 \times \dots \times d_S}$ current model
        parameter variables; $K$ number of minibatches
        (default $K = 32$); $\epsilon$ regularization value
        (default $\epsilon = 10^{-7}$)
    Output: Preconditioning matrix $\mathbf{M}$
    for $s = 1, 2, \dots, S$ do
        $\mathbf{v}_s \leftarrow \mathbf{0}$
    for $k = 1, 2, \dots, K$ do
        $\mathbf{g}^{(k)} \leftarrow \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta})$    // Noisy gradient
        for $s = 1, 2, \dots, S$ do
            $\mathbf{v}_s \leftarrow \mathbf{v}_s + \mathbf{g}_s^{(k)} \cdot \mathbf{g}_s^{(k)}$
    for $s = 1, 2, \dots, S$ do
        $\sigma_s \leftarrow \sqrt{\epsilon + \frac{1}{d_s K} \sum_i \mathbf{v}_{s,i}}$    // RMSprop
    $\sigma_{\text{min}} \leftarrow \min_s \sigma_s$    // Least sensitive
    for $s = 1, 2, \dots, S$ do
        $\mathbf{M}_s \leftarrow \frac{\sigma_s}{\sigma_{\text{min}}} I$
    $\mathbf{M} \leftarrow \begin{bmatrix} \mathbf{M}_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \mathbf{M}_S \end{bmatrix}$
    return $\mathbf{M}$
```

# E. Kullback-Leibler Scaling in Variational Bayesian Neural Networks

With the posterior energy $U(\boldsymbol{\theta})$ defined in the main paper we define two variants of temperized posterior energies:

- Fully tempered energy: $U_F(\boldsymbol{\theta}) = U(\boldsymbol{\theta})/T$, and
- Partially tempered energy: $U_P(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta}) - \frac{1}{T}\sum_{i=1}^n \log p(y_i|x_i, \boldsymbol{\theta})$.

Note that $U_F(\boldsymbol{\theta})$ is used for all experiments in the paper and temperizes both the log-likelihood as well as the log-prior terms, whereas $U_P(\boldsymbol{\theta})$ only scales the log-likelihood terms while leaving the log-prior untouched.

We now show that Kullback-Leibler scaling as commonly done in variational Bayesian neural networks corresponds to approximating the partially tempered posterior,

$$p_P(\boldsymbol{\theta}|\mathcal{D}) \propto \exp(-U_P(\boldsymbol{\theta})). \tag{23}$$

For any distribution $q(\boldsymbol{\theta})$ we consider the Kullback-Leibler divergence,

$$\begin{aligned} & D_{\text{KL}}(q(\boldsymbol{\theta}) \,\|\, p_P(\boldsymbol{\theta}|\mathcal{D})) && (24) \\ &= \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta}) - \log p_P(\boldsymbol{\theta}|\mathcal{D})\right] && (25) \\ &= \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta}) - \log \frac{\exp(-U_P(\boldsymbol{\theta}))}{\int \exp(-U_P(\boldsymbol{\theta}'))\,\mathrm{d}\boldsymbol{\theta}'}\right]. && (26) \end{aligned}$$

The normalizing integral in (26) is not a function of $\boldsymbol{\theta}$ and thus does not depend on $q(\boldsymbol{\theta})$, allowing us to simplify the equation further:

$$= \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) - \frac{1}{T}\sum_{i=1}^n \log p(y_i|x_i, \boldsymbol{\theta})\right] \tag{27}$$

$$\underbrace{+ \log \int \exp(-U_P(\boldsymbol{\theta}))\,\mathrm{d}\boldsymbol{\theta}}_{\text{constant, } =: \log E_P} \tag{28}$$

$$= D_{\text{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})) - \frac{1}{T}\sum_{i=1}^n \log p(y_i|x_i, \boldsymbol{\theta}) + \log E_P. \tag{29}$$

Here we defined $E_P$ as the *partial temperized evidence* which does not depend on $\boldsymbol{\theta}$ and therefore becomes a constant. The global minimizer of (29) over all distributions $q \in \mathcal{Q}$ is the unique distribution $p_P(\boldsymbol{\theta}|\mathcal{D})$, (MacKay et al., 1995).

We now consider this minimizer, substituting $\lambda := T$,

$$\begin{aligned} & \underset{q\in\mathcal{Q}}{\operatorname{argmin}}\; D_{\text{KL}}(q(\boldsymbol{\theta}) \,\|\, p_P(\boldsymbol{\theta}|\mathcal{D})) && (30) \\ &= \underset{q\in\mathcal{Q}}{\operatorname{argmin}}\; D_{\text{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})) - \frac{1}{T}\sum_{i=1}^n \log p(y_i|x_i, \boldsymbol{\theta}) && (31) \end{aligned}$$

The minimizing $q \in \mathcal{Q}$ does not depend on the overall scaling of the optimizing function. We can therefore scale the function by a factor of $T$,

$$= \underset{q\in\mathcal{Q}}{\operatorname{argmin}}\; T D_{\text{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})) - \sum_{i=1}^n \log p(y_i|x_i, \boldsymbol{\theta}) \tag{32}$$

Substituting $\lambda := T$ yields

$$= \underset{q\in\mathcal{Q}}{\operatorname{argmin}}\; \lambda D_{\text{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})) - \sum_{i=1}^n \log p(y_i|x_i, \boldsymbol{\theta}). \tag{33}$$

The last equation, (33) is the KL-weighted negative evidence lower bound (ELBO) objective commonly used in variational Bayes for Bayesian neural networks, confer the ELBO equation (4) from the main paper.

## F. Inference Bias-Variance Trade-off Hypothesis

**Bias-variance Tradeoff Hypothesis**: For $T = 1$ the posterior is diverse and there is high variance between model predictions. For $T \ll 1$ we sample nearby modes and reduce prediction variance but increase bias; the variance dominates the error and reducing variance ($T \ll 1$) improves predictive performance.

We approach the hypothesis using a simple asymptotic argument. We consider the SG-MCMC method we use, including preconditioning and cyclical time stepping. Whereas within a cycle the Markov chain is non-homogeneous, if we consider only the end-of-cycle iterates that emit a parameter $\boldsymbol{\theta}^{(t)}$, then this coarse-grained process is a homogeneous Markov chain. For such Markov chains we can leverage generalized central limit theorems for functions of $\boldsymbol{\theta}$, see e.g. (Jones et al., 2004; Häggström & Rosenthal, 2007), and because of existence of limits we can consider the asymptotic behavior of the test cross-entropy performance measure $C(S)$ as we increase the ensemble size $S \to \infty$.

In particular, expectations of smooth functions of empirical means of $S$ samples have an expansion of the form, (Nowozin, 2018; Schucany et al., 1971),

$$\mathbb{E}[C(S)] = C(\infty) + a_1 \frac{1}{S} + a_2 \frac{1}{S^2} + \dots . \tag{34}$$

**Risk Asymptotics Experiment:** if we can estimate $C(\infty)$ we know what performance we could achieve if we were to keep sampling. To this end we apply a simple linear regression estimate, (Schucany et al., 1971), to the empirically observed performance estimates $\hat{C}(S)$ for different ensemble sizes $S$. By truncation at second order, we obtain estimates for $C(\infty)$, $a_1$, and $a_2$.

In Figure 15 we show the regressed test cross-entropy metric obtained by fitting (34) to second order to all samples for $S \geq 20$ close to the asymptotic regime, and visualize the estimate $\hat{C}(\infty)$. In Figure 16 we visualize our estimated $\hat{C}(\infty)$ as a function of the temperature $T$. The results indicate two things: *first*, we could gain better predictive performance from running our SG-MCMC method for longer (Figure 15); but *second*, the additional gain that could be obtained from longer sampling is too small to make $T = 1$ superior to $T < 1$ (Figure 16).

## G. Details on the Experiment for the Implicit Initialization Prior in SGD Hypothesis

SGD and SG-MCMC are setup as described in Appendix A.1. In the main paper the test accuracy as function of epochs is shown in Figure 12. In Figure 17 we additionally report the test cross entropy for the same experiment.

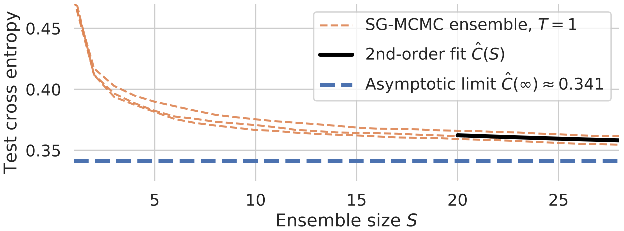


*Figure 15.* Regressing the limiting ResNet-20/CIFAR-10 ensemble performance: at temperature $T = 1$ an ensemble of size $S = \infty$ would achieve 0.341 test cross-entropy. For SG-MCMC we show three different runs with varying seeds.

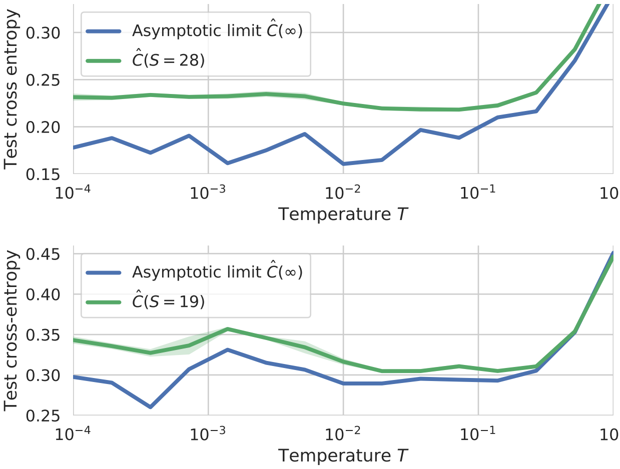


*Figure 16.* Ensemble variance for ResNet-20/CIFAR-10 (**top**) and CNN-LSTM/IMDB (**bottom**) does not explain poor performance at $T = 1$: even in the infinite limit the performance $C(\infty)$ remains poor compared to $T < 1$.

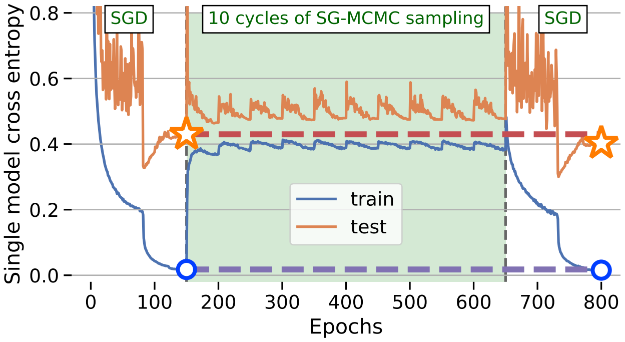


*Figure 17.* Do the SG-MCMC dynamics harm a beneficial initialization bias used by SGD? We first train a ResNet-20 on CIFAR-10 via SGD, then switch over to SG-MCMC sampling and finally switch back to SGD optimization. We report the single-model test cross entropy of SGD and the SG-MCMC chain as function of epochs. SGD recovers from being initialized by the SG-MCMC state.

SGD initialized by the last model of the SG-MCMC sampling dynamics also recovers the same performance in terms of cross entropy as vanilla SGD.

# H. Diagnostics: Temperatures

The following proposition adapted from (Leimkuhler & Matthews, 2016, Section 6.1.5) provides a general way to construct temperature observables.

**Proposition 1** (Constructing Temperature Observables)**.** *Given a Hamiltonian $H(\boldsymbol{\theta}, \mathbf{m})$ corresponding to Langevin dynamics,*

$$H(\boldsymbol{\theta}, \mathbf{m}) = \frac{1}{T} U(\boldsymbol{\theta}) + \frac{1}{2} \mathbf{m}^T \mathbf{M}^{-1} \mathbf{m}, \tag{35}$$

*and an arbitrary smooth vector field $B : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^d$ satisfying*

- $0 < \mathbb{E}_{(\boldsymbol{\theta},\mathbf{m})}[\langle B(\boldsymbol{\theta}, \mathbf{m}), \nabla H(\boldsymbol{\theta}, \mathbf{m})\rangle] < \infty$,
- $0 < \mathbb{E}_{(\boldsymbol{\theta},\mathbf{m})}[\langle \mathbf{1}_{2d}, \nabla B(\boldsymbol{\theta}, \mathbf{m})\rangle] < \infty$*, and*
- $\|B(\boldsymbol{\theta}, \mathbf{m}) \exp(-H(\boldsymbol{\theta}, \mathbf{m}))\| < \infty$ *for all* $(\boldsymbol{\theta}, \mathbf{m}) \in \mathbb{R}^d \times \mathbb{R}^d$,

*then*

$$T = \frac{\mathbb{E}_{(\boldsymbol{\theta},\mathbf{m})}[\langle B(\boldsymbol{\theta}, \mathbf{m}), \nabla H(\boldsymbol{\theta}, \mathbf{m})\rangle]}{\mathbb{E}_{(\boldsymbol{\theta},\mathbf{m})}[\langle \mathbf{1}_{2d}, \nabla B(\boldsymbol{\theta}, \mathbf{m})\rangle]}. \tag{36}$$

Note that for the Hamiltonian (35) we have, assuming a symmetric preconditioner, $(\mathbf{M}^{-1})^T = \mathbf{M}^{-1}$,

$$\nabla_{\boldsymbol{\theta}} H(\boldsymbol{\theta}, \mathbf{m}) = \frac{1}{T} \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}), \tag{37}$$

$$\nabla_{\mathbf{m}} H(\boldsymbol{\theta}, \mathbf{m}) = \mathbf{M}^{-1} \mathbf{m}. \tag{38}$$

### H.1. Kinetic Temperature Estimation

Simulating the Langevin dynamics, equations (5–6) from the main paper, produces moments $\mathbf{m}$ which are jointly distributed according to a multivariate Normal distribution, (Leimkuhler & Matthews, 2016),

$$\mathbf{m} \sim \mathcal{N}(0, \mathbf{M}). \tag{39}$$

The *kinetic temperature* $\hat{T}_K(\mathbf{m})$ is derived from the moments as

$$\hat{T}_K(\mathbf{m}) := \frac{\mathbf{m}^T \mathbf{M}^{-1} \mathbf{m}}{d}, \tag{40}$$

and we have that for a perfect simulation of the dynamics we achieve $\mathbb{E}[\hat{T}_K(\mathbf{m})] = T$, where $T$ is the target temperature of the system, (Leimkuhler & Matthews, 2016). This can be seen by instantiating Proposition 1 for the Langevin Hamiltonian and $B_K(\boldsymbol{\theta}, \mathbf{m}) = \begin{bmatrix} \mathbf{0} \\ \mathbf{m} \end{bmatrix}$.

In general we only approximately solve the SDE and errors in the solution arise due to discretization, minibatch noise, or lack of full equilibration to the stationary distribution. Therefore, we can use $\hat{T}_K(\mathbf{m})$ as a diagnostic to measure the temperature of the current system state, and a deviation from the target temperature could diagnose poor solution accuracy. To this end, we know that if $\mathbf{m} \sim \mathcal{N}(0, \mathbf{M})$ then $(\mathbf{M}^{-1/2} \mathbf{m}) \sim \mathcal{N}(0, I_d)$ and thus the inner product $(\mathbf{M}^{-1/2} \mathbf{m})^T (\mathbf{M}^{-1/2} \mathbf{m}) = \mathbf{m}^T \mathbf{M}^{-1} \mathbf{m}$ is distributed according to a standard $\chi^2$-distribution with $d$ degrees of freedom,

$$(\mathbf{m}^T \mathbf{M}^{-1} \mathbf{m}) \sim \chi^2(d). \tag{41}$$

The $\chi^2(d)$ distribution has mean $d$ and variance $2d$ and we can use the tail probabilities to test whether the observed temperature could arise from an accurate discretization of the SDE (5–6). For a given confidence level $c \in (0, 1)$, e.g. $c = 0.99$, we define the confidence interval

$$J_{T_K}(d, c) := \left( \frac{T}{d} F^{-1}_{\chi^2(d)} \left( \frac{1-c}{2} \right), \frac{T}{d} F^{-1}_{\chi^2(d)} \left( \frac{1+c}{2} \right) \right), \tag{42}$$

where $F^{-1}_{\chi^2(d)}$ is the inverse cumulative distribution function of the $\chi^2$ distribution with $d$ degrees of freedom. By construction if (41) holds, then $\hat{T}_K(\mathbf{m}) \in J_{T_K}(d, c)$ with probability $c$ exactly.

Therefore, if $c$ is close to one, say $c = 0.99$, and we find that $\hat{T}_K(\mathbf{m}) \notin J(d, c)$ this indicates issues of discretization error or convergence of the SDE (5–6).

Because (39) holds for any subvector of $\mathbf{m}$, we can create one kinetic temperature estimate for each model variable separately, such as one or two scalar temperature estimates for each layer (e.g. one for the weights and one for the bias of a `Dense` layer). We found per-layer temperature estimates helpful in diagnosing convergence issues and this directly led to the creation of our layerwise preconditioner.

### H.2. Configurational Temperature Estimation

The so called *configurational temperature*[11] is defined as

$$\hat{T}_C(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})) = \frac{\langle \boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})\rangle}{d}. \tag{43}$$

For a perfect simulation of SDE (5–6) we have $\mathbb{E}[\hat{T}_C] = T$, where $T$ is the target temperature of the system. This can be seen by instantiating Proposition 1 for the Langevin Hamiltonian and $B_C(\boldsymbol{\theta}, \mathbf{m}) = \begin{bmatrix} \boldsymbol{\theta} \\ \mathbf{0} \end{bmatrix}$.

As for the kinetic temperature diagnostic, we can instantiate Proposition 1 for arbitrary subsets of parameters by a suitable choice of $B_C(\boldsymbol{\theta}, \mathbf{m})$. However, whereas for the kinetic temperature the exact sampling distribution of the estimate is known in the form of a scaled $\chi^2$ distribution, we are not aware of a characterization of the sampling distribution of configurational temperature estimates. It is likely this sampling distribution depends on $U(\boldsymbol{\theta})$ and thus does not

[11] Sometimes other quantities are also refered to as configurational temperature, see (Leimkuhler & Matthews, 2016, Section 6.1.5).

have a simple form. Proposition 1 only asserts that under the true target distribution we have

$$\mathbb{E}_{\boldsymbol{\theta}\sim\exp(-U(\boldsymbol{\theta})/T)}[\hat{T}_C(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}))] = T. \tag{44}$$

Because (1) is the empirical average of per parameter random variables, if all these variables have finite variance the central limit theorem asserts that for large $d$ we can expect

$$\hat{T}_C(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})) \sim \mathcal{N}(T, \sigma^2_{T_C}), \tag{45}$$

with unknown variance $\sigma^2_{T_C}$.

Recent work of Yaida (2018) provides a similar diagnostic, equation (FDR1') in their work, to the configurational temperature (43) for the SGD equilibrium distribution under finite time dynamics. However, our goal here is different: whereas Yaida (2018) is interested in diagnosing convergence to the SGD equilibrium distribution in order to adjust learning rates we instead want to diagnose discrepancy of our current dynamics against the true target distribution.

# I. Simulation Accuracy Ablation Study

Equipped with the diagnostics of Section H we can now study how accurate our algorithms simulate the Langevin dynamics. We will demonstrate that layerwise preconditioning and cyclical time stepping are individually effective at improving simulation accuracy, however, only by combining these two methods we can achieve high simulation accuracy on the CNN-LSTM model as measured by our diagnostics.

**Setup.** We perform the same ResNet-20 CIFAR-10 and CNN-LSTM IMDB experiments as in the main paper, but consider four variations of our algorithm: with and without preconditioning, and with and without cosine time stepping schedules. In case no preconditioner is used we simply set $\mathbf{M} = I$ for all iterations. In case no cosine time stepping is used we simply set $C(t) = 1$ for all iterations.

Independent of whether cosine time stepping is used we divide the iterations into cycles and for each method consider all models at the end of a cycle, where we hope simulation accuracy is the highest. We then evaluate the temperature diagnostics for all model variables. For the kinetic temperatures, if simulation is accurate then 99 percent of the variables should on average lie in the 99% high probability region under the sampling distribution. For the configurational temperature we can only report the average configurational temperature across all the end-of-cycle models.

**Results.** We report the results in Table 1 and Table 2 and visualize the kinetic temperatures in Figures 18 to 21 and Figures 22a to 22d.

The results indicate that both cosine time stepping and layerwise preconditioning have a beneficial effect on simulation accuracy. For ResNet-20 cyclical time stepping is sufficient for high simulation accuracy, but it is by itself not able to achieve high accuracy on the CNN-LSTM model. For both models the combination of cyclical time stepping and preconditioning (Figure 18 and Figure 22a) achieves a high simulation accuracy, that is, all kinetic temperatures match the sampling distribution of the Langevin dynamics, indicating—at least with respect to the power of our diagnostics—accurate simulation.

Another interesting observation can be seen in Table 1: we can achieve a high accuracy of $\geq 88$ percent even in cases where the simulation accuracy is poor. This indicates that optimization is different from accurate Langevin dynamics simulation.

# J. Dirty Likelihood Functions

**Dirty Likelihood Hypothesis**: Deep learning practices that violate the likelihood principle (batch normalization, dropout, data augmentation) cause deviation from the Bayes posterior.

We now discuss how batch normalization, dropout, and data augmentation produce non-trivial modifications to the likelihood function. We call the resulting likelihood functions "dirty" to distinguish them from clean likelihood functions without such modifications. Our discussion will suggest that these techniques can be seen as a computational efficient "*Jensen posterior*" approximation of a proper Bayesian posterior of another model. Our analysis builds on and generalizes previous Bayesian interpretations, (Noh et al., 2017; Atanov et al., 2018; Shekhovtsov & Flach, 2018; Nalisnick et al., 2019; Inoue, 2019). In Section J.4 we perform an experiment to demonstrate that the dirty likelihood cannot explain cold posteriors.

## J.1. Augmented Latent Model

To accommodate popular deep learning methods we first augment the probabilistic model $p(y|x, \boldsymbol{\theta})$ itself by adding a *latent variable* $z$. The augmented model is $p(y|x, z, \boldsymbol{\theta})$ and we can obtain the *effective model* $p(y|x, \boldsymbol{\theta}) = \int p(y|x, z, \boldsymbol{\theta})\, p(z) dz$. For a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1,\ldots,n}$, where we denote $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$, the resulting model has as likelihood

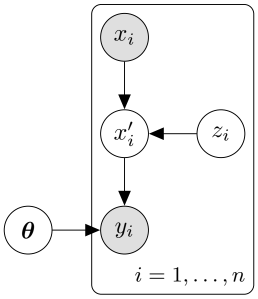


*Figure 23.* Augmented model with added latent variable $z_i$.

| Precond | Cyclic | $\hat{\mathbb{E}}[\hat{T}_K \in \mathcal{R}_{99}]$ | $\hat{\mathbb{E}}[\hat{T}_C]$ | Accuracy (%) | Cross-entropy |
|---|---|---|---|---|---|
| ✓ | ✓ | 0.989±0.0014 | 0.94±0.011 | 88.2±0.11 | 0.358±0.0011 |
| ✗ | ✓ | 0.9772±0.00059 | 1.02±0.018 | 88.49±0.014 | 0.3500±0.00064 |
| ✓ | ✗ | 0.905±0.0019 | 1.23±0.046 | 88.0±0.10 | 0.3808±0.00064 |
| ✗ | ✗ | 0.676±0.0052 | 1.7±0.18 | 86.86±0.072 | 0.507±0.0080 |

*Table 1.* ResNet-20 CIFAR-10 simulation accuracy ablation at $T = 1$: layerwise preconditioning and cyclical time stepping each have a beneficial effect on improving inference accuracy and the effect is complementary. $\hat{\mathbb{E}}[\hat{T}_K \in \mathcal{R}_{99}]$ is the empirically estimated probability that the kinetic temperature statistics are in the 99% confidence interval, the ideal value is 0.99. $\hat{\mathbb{E}}[\hat{T}_C]$ is the empirical average of the configurational temperature estimates, the ideal value is 1.0. For both quantities we take the value achieved at the end of each cycle, that is, whenever $C(t) = 0$ and average all the resulting values. The deviation is given in ±SEM where SEM is the standard error of the mean estimated from three independent experiment replicates. Both preconditioning and cyclical time stepping are effective at improving the simulation accuracy.

| Precond | Cyclic | $\hat{\mathbb{E}}[\hat{T}_K \in \mathcal{R}_{99}]$ | $\hat{\mathbb{E}}[\hat{T}_C]$ | Accuracy (%) | Cross-entropy |
|---|---|---|---|---|---|
| ✓ | ✓ | 0.954±0.0053 | 0.99122±0.000079 | 81.95±0.22 | 0.425±0.0032 |
| ✗ | ✓ | 0.761±0.0095 | 1.012±0.0088 | 51.3±0.65 | 0.6925±0.00019 |
| ✓ | ✗ | 0.49±0.012 | 0.9933±0.00019 | 74.5±0.49 | 0.579±0.0048 |
| ✗ | ✗ | 0.384±0.0018 | 1.0141±0.00066 | 0.49997±0.000039 | 0.698±0.0013 |

*Table 2.* CNN-LSTM IMDB simulation accuracy ablation at $T = 1$: with *both* layerwise preconditioning and cyclical time stepping we can achieve high inference accuracy as measured by configurational and kinetic temperature diagnostics. Just using one (either preconditioning or cyclical time stepping) is insufficient for high inference accuracy. This is markedly different from the results obtained for ResNet-20 CIFAR-10 (Table 1), indicating that perhaps the ResNet posterior is easier to sample from.

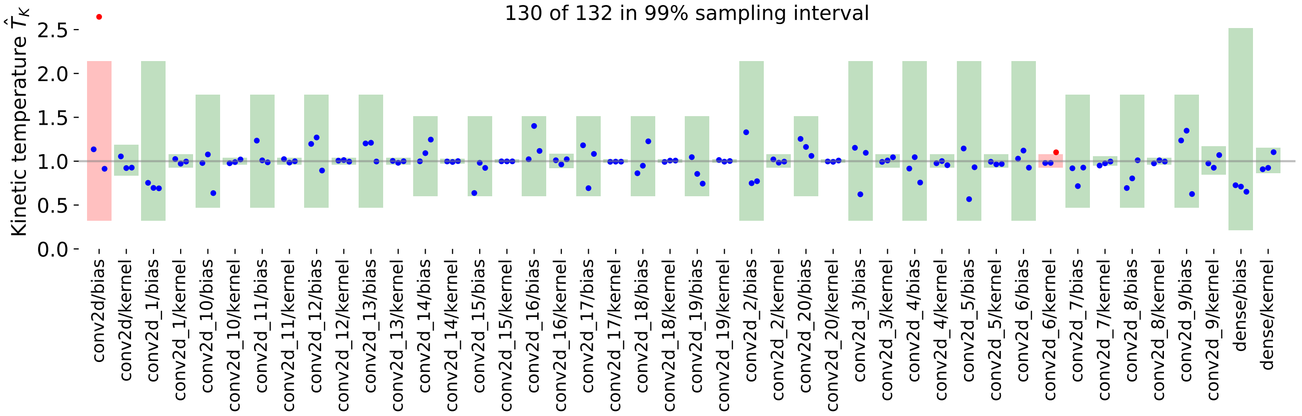


*Figure 18.* ResNet-20 CIFAR-10 Langevin per-variable kinetic temperature estimates **with preconditioning** and **with cosine time stepping schedule**. The green bars show the 99% true sampling distribution of the Kinetic temperature sample. The blue dots show the actual kinetic temperature samples at the end of sampling. About 1% of variables should be outside the green boxes, which matches the empirical count (2 out of 132 samples), indicating an accurate simulation of the Langevin dynamics at the end of each cycle.

function in $\boldsymbol{\theta}$ that is the *marginal likelihood*, obtained by integrating over all $z_i$ variables,

$$
\begin{aligned}
p(Y \mid X, \boldsymbol{\theta}) &= \prod_{i=1}^{n} p(y_i \mid x_i, \boldsymbol{\theta}) && (46) \\
&= \prod_{i=1}^{n} \mathbb{E}_{z_i \sim p(z_i)} [p(y_i \mid x_i, z_i, \boldsymbol{\theta})]. && (47)
\end{aligned}
$$

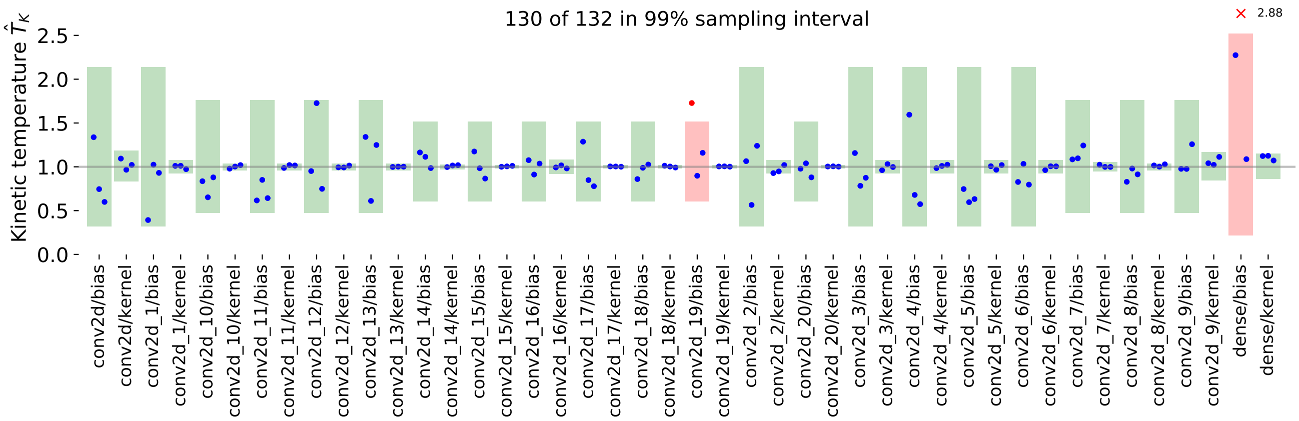


*Figure 19.* ResNet-20 CIFAR-10 Langevin per-variable kinetic temperature estimates **without preconditioning** but **with cosine time stepping schedule**. Two out of 132 variables are outside the 99% hpd region, indicating accurate simulation.

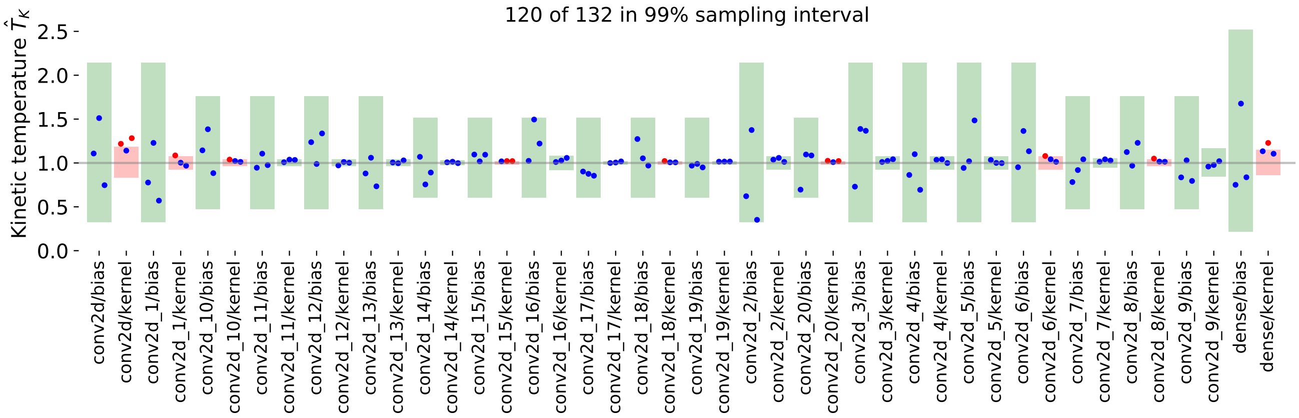


*Figure 20.* ResNet-20 CIFAR-10 Langevin per-variable kinetic temperature estimates **with preconditioning** but **without cosine time stepping schedule** (flat schedule). 12 out of 132 variables are too hot (boxes in red) and lie outside the acceptable region, indicating an inaccurate simulation of the Langevin dynamics. However, there is a marked improvement due to preconditioning compared to no preconditioning (Figure 21).

Note that in (47) the latent variable $z_i$ is integrated out and therefore the marginal likelihood is a deterministic function.

## J.2. Log-likelihood Bound and Jensen Posterior

Given a prior $p(\boldsymbol{\theta})$ the log-posterior for the augmented model in Figure 23 takes the form

$$\log p(\boldsymbol{\theta} \,|\, \mathcal{D}) \tag{48}$$

$$= C + \log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \log \mathbb{E}_{z_i \sim p(z_i)}[p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})], \tag{49}$$

where we can now apply *Jensen's inequality*, $f(\mathbb{E}[x]) \geq \mathbb{E}[f(x)]$ for concave $f = \log$,

$$\geq C + \log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \mathbb{E}_{z_i \sim p(z_i)}[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})], \tag{50}$$

where $C = -\log p(Y|X)$ is the negative model evidence and is constant in $\boldsymbol{\theta}$. We call equation (50) the *Jensen bound* to the log-posterior $\log p(\boldsymbol{\theta}|\mathcal{D})$.

**Jensen Posterior.** Because we can estimate (50) in an unbiased manner, we will see that many popular methods such as dropout and data augmentation can be cast as special cases of the Jensen bound. We also define the *Jensen posterior* as the posterior distribution associated with (50).

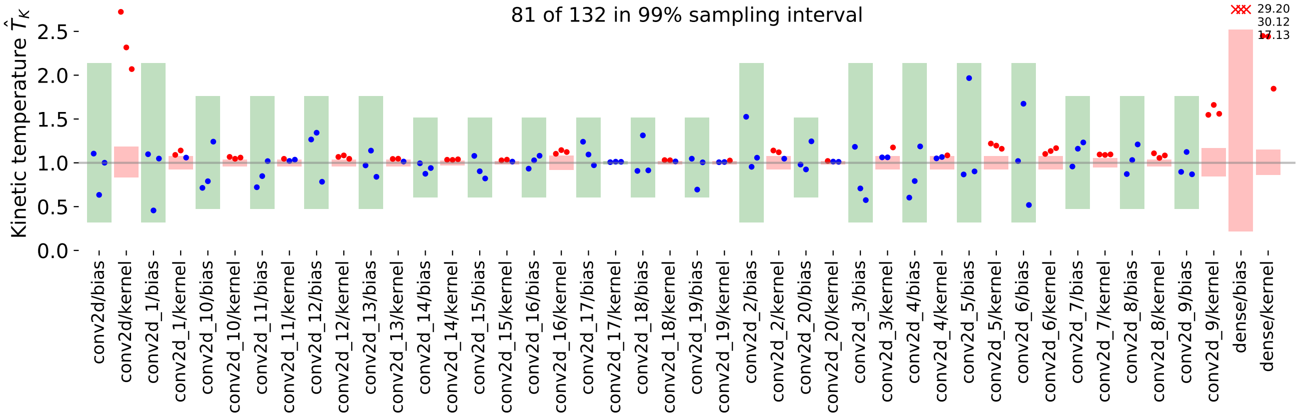


*Figure 21.* ResNet-20 CIFAR-10 Langevin per-variable kinetic temperature estimates **without preconditioning** and **without cosine time stepping schedule** (flat schedule). 51 out of 132 kinetic temperature samples are too hot (shaded in red) and lie outside the acceptable region, sometimes severely so, indicating a very poor simulation accuracy for the Langevin dynamics.

Formally, the Jensen posterior is

$$p_J(\boldsymbol{\theta} \,|\, \mathcal{D}) :\propto \tag{51}$$

$$p(\boldsymbol{\theta}) \prod_{i=1}^{n} \exp\left(\mathbb{E}_{z_i \sim p(z_i)} \left[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\right]\right). \tag{52}$$

Given this object, can we relate its properties to the properties of the full posterior, and can the Jensen posterior serve as a meaningful surrogate to the true posterior? We first observe that $p_J(\boldsymbol{\theta} \,|\, \mathcal{D})$ indeed defines a probability distribution over parameters: with a proper prior $p(\boldsymbol{\theta})$, we have $p(\boldsymbol{\theta} \,|\, \mathcal{D}) \geq p_J(\boldsymbol{\theta} \,|\, \mathcal{D})$ by (49–50), thus $\int p_J(\boldsymbol{\theta} \,|\, \mathcal{D}) \,\mathrm{d}\boldsymbol{\theta} \leq \int p(\boldsymbol{\theta} \,|\, \mathcal{D}) \,\mathrm{d}\boldsymbol{\theta} < \infty$.

**Jensen Prior.** We now show that the Jensen posterior can be interpreted as a full Bayesian posterior in a different model. In particular, we give a construction which retains the likelihood of the original model but modifies the prior. In the function that re-weights the prior the data set appears; this is not to be understood as a prior which depends on the observed data. Instead, we can think of this as an existence proof, that is, if we were to have chosen this modified prior then the resulting Jensen posterior under the modified Jensen prior corresponds to the full Bayesian posterior under the original prior.

In a sense the result is vacuous because any desirable posterior can be obtained by such re-weighting. However, the proof illustrates the structure of how the Jensen posterior deviates from the true posterior through a set of weighting functions; each weighting function measures a local *Jensen gap* related to each instance. Although we did not pursue this line, the local Jensen gap (57) can be numerically estimated and may prove to be a useful quantity in itself.

**Proposition 2** (Jensen Prior)**.** *For a proper prior $p(\boldsymbol{\theta})$ and a fixed dataset $\mathcal{D}$, we can define a prior $p_J(\boldsymbol{\theta})$ such that when using this modified prior in the Jensen posterior we have*

$$p_J(\boldsymbol{\theta} \,|\, \mathcal{D}) = p(\boldsymbol{\theta} \,|\, \mathcal{D}). \tag{53}$$

*In particular, this implies that any Jensen posterior can be interpreted as the posterior distribution of the same model under a different prior.*

*Proof.* We have the true posterior

$$p(\boldsymbol{\theta} \,|\, \mathcal{D}) = p(\boldsymbol{\theta}) \prod_{i=1}^{n} \int p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\, p(z_i) \,\mathrm{d}z_i, \tag{54}$$

and the Jensen posterior as

$$p_J(\boldsymbol{\theta} \,|\, \mathcal{D}) := p(\boldsymbol{\theta}) \prod_{i=1}^{n} \exp\left(\mathbb{E}_{z_i \sim p(z_i)} \left[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\right]\right), \tag{55}$$

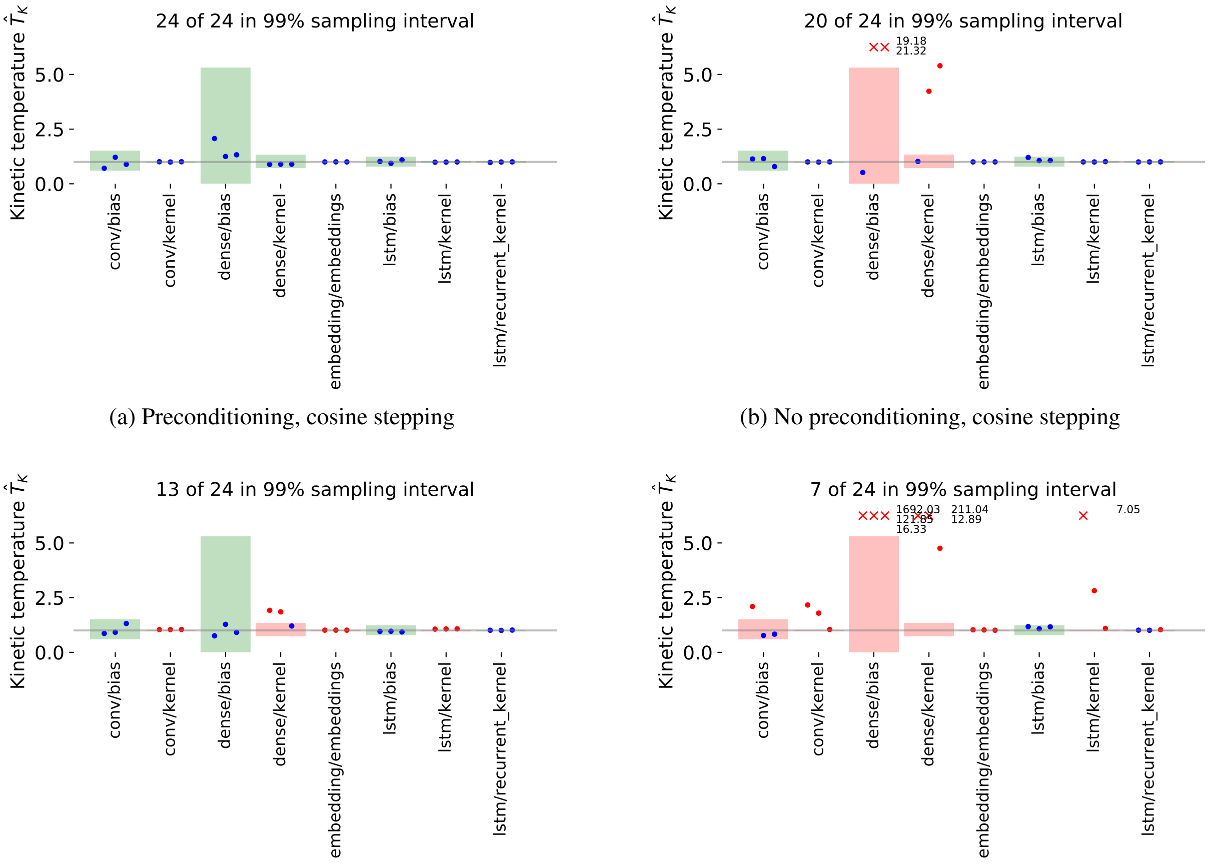
respectively. If we define the *Jensen prior*,

$$p_J(\boldsymbol{\theta}) :\propto w(\boldsymbol{\theta})\, p(\boldsymbol{\theta}), \tag{56}$$

where we set the weighting function $w(\boldsymbol{\theta}) := \prod_{i=1}^{n} w_i(\boldsymbol{\theta})$, with the individual weighting functions defined as

$$w_i(\boldsymbol{\theta}) := \frac{\int p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\, p(z_i) \,\mathrm{d}z_i}{\exp\left(\mathbb{E}_{z_i \sim p(z_i)}[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})]\right)}. \tag{57}$$

Due to Jensen's inequality we have $w_i(\boldsymbol{\theta}) \leq 1$ and hence $w(\boldsymbol{\theta}) \leq 1$ and thus $p_J(\boldsymbol{\theta})$ is normalizable. Using $p_J(\boldsymbol{\theta})$ as



(a) Preconditioning, cosine stepping

(b) No preconditioning, cosine stepping

(c) Preconditioning, no cosine stepping

(d) No preconditioning, no cosine stepping

*Figure 22.* CNN-LSTM IMDB Langevin per-variable kinetic temperature estimates at temperature $T = 1$ for four different simulation settings: with and without preconditioning, with and without cosine time stepping. The only accurate simulation is obtained with *both* preconditioning *and* cosine time stepping.

prior in (55) we obtain

$$
\begin{aligned}
& p_J(\boldsymbol{\theta} \,|\, \mathcal{D}) && (58)\\
& \propto p_J(\boldsymbol{\theta}) \prod_{i=1}^{n} \exp\left(\mathbb{E}_{z_i \sim p(z_i)}\left[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\right]\right), && (59)\\
& = p(\boldsymbol{\theta}) \left(\prod_{i=1}^{n} w_i(\boldsymbol{\theta})\right) && (60)\\
& \qquad\qquad \prod_{i=1}^{n} \exp\left(\mathbb{E}_{z_i \sim p(z_i)}\left[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\right]\right), && (61)\\
& = p(\boldsymbol{\theta}) \prod_{i=1}^{n} \int p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\, p(z_i)\, \mathrm{d}z_i && (62)\\
& \propto p(\boldsymbol{\theta} \,|\, \mathcal{D}). && (63)
\end{aligned}
$$

This constructively demonstrates the result (53). □

We now interpret current deep learning methods as optimizing the Jensen posterior.

### J.3. Deep Learning Techniques Optimize Jensen Posteriors

**Dropout.** In *dropout* we sample random binary masks $z_i \sim p(z_i)$ and multiply network activations with such masks (Srivastava et al., 2014). Specializing the above latent variable model to dropout gives an interpretation of doing maximum aposteriori (MAP) estimation on the Jensen posterior $p_J(\boldsymbol{\theta} \,|\, X, Y)$.

The connection between dropout and applying Jensen's bound has been discovered before by several groups (Noh et al., 2017), (Nalisnick et al., 2019), (Inoue, 2019), and contrasts sharply with the variational inference interpretation of dropout, (Kingma et al., 2015; Gal & Ghahramani, 2016). Recent variants of dropout such as *noise-in* (Dieng et al., 2018) can also be interpreted in the same way.

The Jensen prior interpretation justifies the use of standard dropout in Bayesian neural networks: the inferred posterior is the Jensen posterior which is also a Bayesian posterior under the Jensen prior.

**Data Augmentation.** Data augmentation is a simple and intuitive way to insert high-level prior knowledge into neural networks: by targeted augmentation of the available training data we can encode invariances with respect to natural transformation or noise, leading to better generalization, (Perez & Wang, 2017).

Data augmentation is also an instance of the above latent variable model, where $z_i$ now corresponds to randomly sampled parameters of an augmentation, for example, whether to flip an image along the vertical axis or not.

Interestingly, the above model suggests that to obtain better predictive performance at test time, the posterior predictive should be obtained by averaging the individual posterior predictive distributions over multiple latent variable realizations. Indeed this is what early work on convolutional networks did, (He et al., 2015; 2016), improving predictive performance significantly.

The Jensen prior interpretation again justifies the use of approximate Bayesian inference techniques targeting the Jensen posterior. In particular, our theory suggests that the dataset size $n$ should *not* be adjusted to account for augmentation.

**Batch Normalization.** As a practical technique batch normalization (Ioffe & Szegedy, 2015) accelerates and stabilizes learning in deep neural networks. The model of Figure 23 cannot directly serve to interpret batch normalization due to the dependence of batch normalization statistics on the batch. We therefore need to extend the model to incorporate a random choice of batches yielding continuous random batch normalization statistics as proposed earlier (Atanov et al., 2018; Shekhovtsov & Flach, 2018).

Formally such variation of batch normalization corresponds to the model shown in Figure 24, where $(x_i)_i \to \boldsymbol{\theta}$ signifies the additional randomness in $p(\boldsymbol{\theta}|X)$ due to random batches, and $(\boldsymbol{\theta}, x_i, z_i) \to x'_i$ are the resulting random outputs of the network, where $z_i$ is a per-instance randomness source (Atanov et al., 2018).

With the above modifications all derivations in Section J.2 hold and batch normalization has a Jensen posterior. In particular, the Jensen interpretation also suggests to perform batch normalization at test-time, averaging over multiple different batches composed of training set samples.

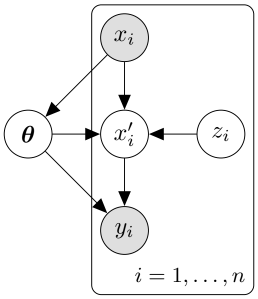


*Figure 24.* Augmented model for batch normalization.

### J.4. Dirty Likelihood Experiment

The dirty likelihood hypothesis is plausible for the ResNet-20 experiments which use data augmentation and batch normalization, however, our CNN-LSTM model does have a clean likelihood function already.

To gain further confidence that this hypothesis cannot explain cold posterior we train a ResNet-20 without batch normalization or data augmentation.

**Clean Likelihood ResNet Experiment**: we disable data augmentation and replace batch normalization with filter

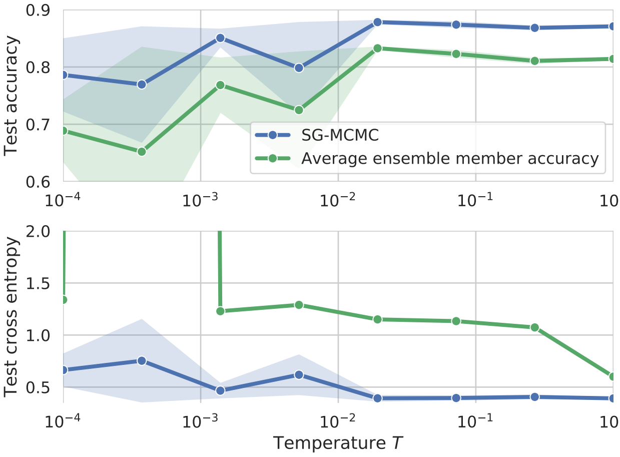


*Figure 25.* ResNet-20 with filter response normalization (FRN) instead of batch normalization and without any use of data augmentation.

response normalization, (Singh & Krishnan, 2019). Without data augmentation and without batch normalization we now have a clean likelihood function and SG-MCMC targets a true underlying Bayes posterior.

Figure 25 on page 21 shows the predictive test performance as a function of temperature. We clearly see that for small temperatures $T \ll 1$ the removal of data augmentation and batch normalization leads to a higher standard error over the three runs, so that indeed data augmentation and batch normalization had a stabilizing effect on training and mitigated overfitting. However, for test accuracy the best performance by the SG-MCMC ensemble model is still achieved for $T < 1$. In particular, for test accuracy the best accuracy of $87.8 \pm 0.16\%$ is achieved at $T = 0.0193$, comparing to a worse predictive accuracy of $87.1 \pm 0.13\%$ at temperature $T = 1$. For test cross entropy the performance achieved at $T = 0.0193$ with $0.393 \pm 0.015$ is comparable to $0.3918 \pm 0.0021$ achieved at $T = 1$.

The clean likelihood ResNet experiment is slightly inconclusive as there is now a less marked improvement when going to lower temperatures. However, our CNN-LSTM IMDB model already had a clean likelihood function. Therefore, while dirty likelihoods may play a role in shaping the posterior that SG-MCMC methods simulate from they likely do not account for the cold posterior effect.

# K. Prior Predictive Analysis for Different Prior Scales

Our experiments in the main paper (Section 5.2) clearly demonstrate that the prior $p(\boldsymbol{\theta}) = \mathcal{N}(0, I)$ is bad in that it places prior mass on the same highly concentrated class probabilities for all training instances.

What other priors could we use? The literature contains significant prior work on this question. Neal (1995) examined priors for shallow neural networks and identified scaling laws and correspondence to Gaussian process kernels. Recently a number of works added to Neal's analysis by extending the results to deep and wide neural networks (Lee et al., 2018; de G. Matthews et al., 2018; Yang, 2019), convolutional networks (Garriga-Alonso et al., 2019), and Bayesian neural networks (Novak et al., 2019).

A related line of work explores random functions defined by the initialization process of a deep neural network. Glorot & Bengio (2010) and He et al. (2015) developed efficient random initialization schemes for deep neural networks and a more formal analysis of information flow in random functions defined by neural networks is given by Schoenholz et al. (2017) and Hayou et al. (2018). All these works derive variance-scaling laws for independent Gaussian priors. The precise scaling law depends on the network layer and the activation function being used. For the same architecture and activation the scaling laws generally agree with those obtained from the Gaussian process perspective.

## K.1. He-Scaled Normal Prior, $\mathcal{N}(0, I)$ for Biases

To remain as close as possible to our existing setup we investigate a He-scaling prior, equation (14) in (He et al., 2015).

$$p(\boldsymbol{\theta}_j) = \mathcal{N}\left(0, \frac{2}{b_j}\right), \tag{64}$$

where $b_j$ is the *fan-in* of the $j$'th layer.[12]

The scaling law derived by He et al. (2015) does not cover the bias terms in a model. This is due to the work considering only initialization—(He et al., 2015) initialized all biases to zero—whereas we would like to have proper priors for all model variables. We therefore choose the original $\mathcal{N}(0, I)$ prior for all bias variables in our model.

**He-scaled Prior Predictive Experiment:** For our ResNet-20 setup on CIFAR-10 we use our He-scaled-Normal prior to once again carry out the prior predictive experiment that was originally done in Section 5.2, Figures 7 and 8 of the main paper. Figure 26 show the prior predictive results for the new prior. The basic conclusion remains unchanged: despite scaling the convolution weights and dense layer weights by the He-scaling law in the prior the prior predictive distributions remain highly concentrated around the same distribution for all training instances.

Why do functions under this prior remain concentrated? Perhaps it is due to the loose $\mathcal{N}(0, I)$ prior for the bias terms

[12] For a `Dense` layer the fan-in is the number of input dimensions, for a `Conv2D` layer with a kernel of size $k$-by-$k$ and $d$ input channels the fan-in is $b_j = k^2 d$.

such that any concentration in early layers is amplified in later layers? We investigate this further in Section K.2.

**He-scaled Prior ResNet-20 CIFAR-10 Experiment:** We also perform the original cold posterior experiment from the main paper with the He-scaling Normal prior. We show the temperature-dependence curves for test accuracy and test cross-entropy in Figure 27. The overall performance drops compared to the $\mathcal{N}(0, I)$ prior, but the cold posterior effect clearly remains. With this result and the result from the prior predictive study we can conclude that a simple Normal scaling correction is not enough to yield a sensible prior.

### K.2. He-Scaled Normal Prior, $\mathcal{N}(0, \epsilon I)$ for Biases

In this section we experiment with He-scaling and a very small scale for the bias prior. There are two motivations for such experimentation: *first*, He-scaling was originally proposed by He et al. (2015) for initializing deep convolutional neural networks and in their initialization all bias terms were initialized to zero. *Second*, bias terms influence a large number of downstream activations and getting the scale wrong for our bias priors may have the large concentration effect that we observe in the previous prior predictive experiments.

We therefore propose to use a He-scaling Normal prior for all `Conv2D` and `Dense` layer weights and to use a $\mathcal{N}(0, \epsilon I)$ prior for all bias terms. Here we use $\epsilon = \sigma^2$ with $\sigma = 10^{-6}$, essentially sampling all bias terms close to zero as in the original initialization due to (He et al., 2015).

**He-scaled Prior, $\mathcal{N}(0, \epsilon I)$ Bias Prior Experiment:** We draw ResNet-20 models from the prior and evaluate the predicted class distributions on the entire CIFAR-10 training set. Figure 28a shows two prior draws and the resulting class distributions marginalized over the entire training set. Figure 28b shows a marginal prior predictive, marginalized over $S = 100$ prior draws and the entire training distribution of 50,000 images. The resulting marginal prior predictive approaches the uniform distribution. However, the He-scaled prior with $\mathcal{N}(0, \epsilon I)$ for bias terms remains a bad prior: random draws place prior mass on the same concentrated class distribution for all training instances.

## L. Details: Generation of a Synthetic Dataset Based on an MLP Drawn From its Prior Distribution

In this section, we describe how we generate a synthetic dataset based on a multi-layer perceptron (MLP) drawn from its prior distribution, as used in Section ???? of the main paper.

We generate synthetic data by (i) drawing a MLP from its prior distribution, i.e., $\texttt{mlp}_{\boldsymbol{\theta}}$ with $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$, (ii) sampling input data point $x$'s $\in \mathbb{R}^5$ from a standard normal distribution and (iii) sampling label $y$'s $\in \{1, 2, 3\}$ from the resulting logits $\texttt{mlp}_{\boldsymbol{\theta}}(x)$. We take $\texttt{mlp}_{\boldsymbol{\theta}}$ to be of depth 2, with 10 units and relu activation functions. We generate $n = 100$ points for inference and 10,000 for evaluations.

The choice of $p(\boldsymbol{\theta})$ requires some care. On the one hand, a naive choice of normal priors with unit standard deviation leads to a degenerated dataset that concentrates all its outputs on a single class. On the other hand, normal priors with a smaller standard deviation[13], e.g., 0.05, lead to a less spiky label distribution but with little dependence on the input $x$'s.

As a result, we considered a He normal prior (He et al., 2015) for the weights of $\texttt{mlp}_{\boldsymbol{\theta}}$ and a normal prior, with standard deviation 0.05, for the bias terms. We similarly adapted the choice of the priors for the MLPs used to learn over the data generated in this way.

## M. Practical Usage of Hamiltonian Monte Carlo

In this section, we describe practical considerations about Hamiltonian Monte Carlo (HMC).

HMC mainly exposes four hyperparameters that need to be set (Neal et al., 2011):

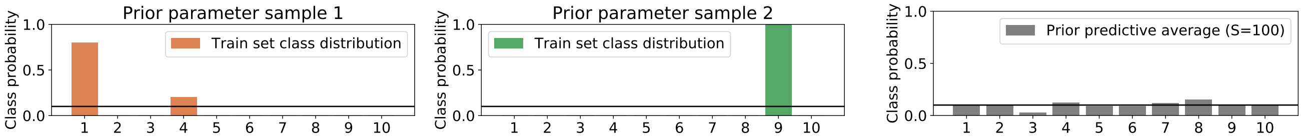
- The number $L$ of steps of the leapfrog integrator,
- The step size $\varepsilon$ in the leapfrog integrator,
- The number $b$ of steps of the burn-in phase,
- The number $S$ of samples to generate.

**Hyperparameter choices.** In our experiments with HMC, we have set $S = 2500$, generating a total of 25000 samples after the burn-in phase and keeping one sample every ten samples.

For the burn-in phase, we investigated in preliminary experiments the effect of varying the number of steps $b \in \{500, 1000, 5000\}$, noticing that our diagnostics (as later described) started to stabilize for $b = 1000$, so that we decided to use $b = 5000$ out of precaution (even though it may not be the most efficient option).

We thereafter searched a good combination of leapfrog steps and step size for $L \in \{5, 10, 100\}$ and $\varepsilon \in \{0.001, 0.01, 0.1\}$. The results of the nine possible combinations are reported in Figure 29, after aggregating 5 different runs (i.e., from 5 different random initial conditions). The influence of the step size in our experiments was likely reduced by the fact that we used the dual averaging step-size adaptation scheme from Hoffman & Gelman (2014), as implemented in *Tensorflow Probability* (Dillon

[13] Default value of `tf.random_normal_initializer`.



(a) Typical predictive distributions for 10 classes under the prior, averaged over the entire training set, $\mathbb{E}_{x\sim p(x)}[p(y|x,\boldsymbol{\theta}^{(i)})]$. Each plot is for one sample $\boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta})$. Given a sample $\boldsymbol{\theta}^{(i)}$ the average training data class distribution is still highly concentrated around the same classes for all $x$.

(b) Prior predictive $\mathbb{E}_{x\sim p(x)}[\mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta})}[p(y|x,\boldsymbol{\theta})]]$ over 10 classes for a Kaiming-scaling prior, estimated using $S = 100$ samples $\boldsymbol{\theta}^{(i)}$ and all training images.

*Figure 26.* ResNet-20/CIFAR-10 prior predictive study for a He-scaled Normal prior for `Conv2D` and `Dense` layers and a $\mathcal{N}(0, I)$ prior for all bias terms. This prior concentrates prior mass on functions which output the *same* concentrated label distribution for *all* training instances. It is therefore a bad prior.

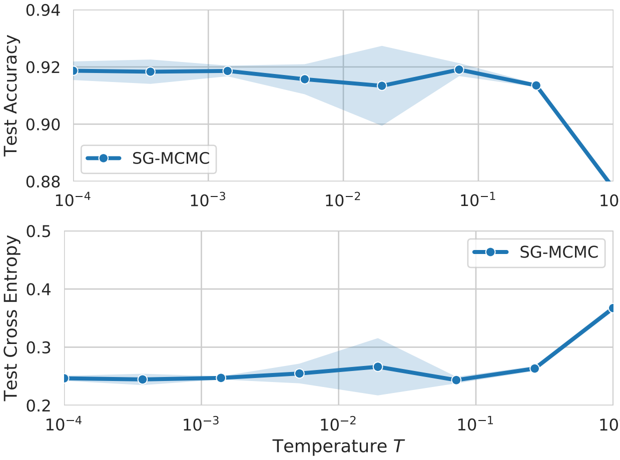


*Figure 27.* ResNet-20 on CIFAR-10 with He-scaling Normal prior (He-scaled Normal for `Conv2D` and `Dense` layers, and $\mathcal{N}(0, I)$ for all bias terms). The cold posterior effect remains: the poor predictive performance of the Bayes posterior at $T = 1$ holds for both accuracy and cross-entropy.

et al., 2017).[14]

**Convergence monitoring.** In Figures 30-31-32, we detail the inspection of the 5 different chains for the choice $L = 100$ and $\varepsilon = 0.1$ (which corresponds to the results of the sampler shown in the main paper). As practical diagnostic tools, we consider *trace plots* where we monitor the evolution of some statistics with respect to the generated HMC samples (e.g., see Section 24.4 in Murphy (2012), and references therein, for an introduction in a machine learning context). We compute trace plots for different depths of the MLP (in $\{1, 2, 3\}$) and different[15] temperatures, $T \in \{0.001, 0.0024, 0.014, 1.0\}$.

In addition to monitoring the evolution of the cross entropy for $S' \in \{1, 2, \ldots, S\}$ HMC samples (see Figure 30), we also consider the following statistics:

- **Mean of the predictive entropy:** Let us denote by $\mathcal{D}_{\text{held-out}}$ the held-out set of pairs $(x, y)$ and $\mathcal{E}_{\boldsymbol{\theta}}(x)$ the entropy of the softmax output at the input $x$

  $$\mathcal{E}_{\boldsymbol{\theta}}(x) = -\sum_c p(y = c|x, \boldsymbol{\theta}) \log p(y = c|x, \boldsymbol{\theta}),$$

  together with its average over the held-out set

  $$\mathcal{E}_{\boldsymbol{\theta}} = \frac{1}{|\mathcal{D}_{\text{held-out}}|} \sum_{(x,y)\in\mathcal{D}_{\text{held-out}}} \mathcal{E}_{\boldsymbol{\theta}}(x).$$

  For $S' \in \{1, 2, \ldots, S\}$ samples collected along the trajectory of HMC, we report in Figure 31 the estimate

  $$\hat{\mathcal{E}} = \frac{1}{S'} \sum_{s=1}^{S'} \mathcal{E}_{\boldsymbol{\theta}_s} \approx \bar{\mathcal{E}} = \int \mathcal{E}_{\boldsymbol{\theta}} \cdot p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta},$$

  which we refer to as the mean of the predictive entropy.

- **Standard deviation of the predictive entropy:** We also consider the monitoring of the second moment of the predictive entropy. With the above notation, we estimate

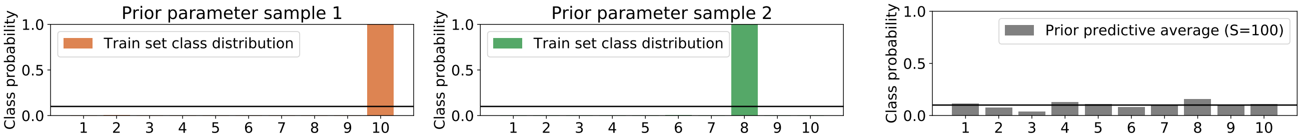
  $$\frac{1}{S'-1} \sum_{s=1}^{S'} (\mathcal{E}_{\boldsymbol{\theta}_s} - \hat{\mathcal{E}})^2 \approx \int (\mathcal{E}_{\boldsymbol{\theta}} - \bar{\mathcal{E}})^2 \cdot p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}$$

  and report its square root in Figure 32, which we refer to as the standard deviation of the predictive entropy.

As a general observation, we can see on Figures 30-31-32 that, overall, the 5 different chains tend to exhibit a converging behavior for the three examined statistics, with typically more dispersion as the depth and the temperature increase (which is reflected by the ranges of the y-axis in the plots of Figures 30-31-32 that get wider as $T$ and the depth become larger).

[14] `tfp.mcmc.DualAveragingStepSizeAdaptation`.

[15] We limit ourselves to four temperatures to avoid clutter.



(a) Typical predictive distributions for 10 classes under the prior, averaged over the entire training set, $\mathbb{E}_{x\sim p(x)}[p(y|x,\boldsymbol{\theta}^{(i)})]$. Each plot is for one sample $\boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta})$. Given a sample $\boldsymbol{\theta}^{(i)}$ the average training data class distribution is still highly concentrated around the same classes for all $x$ despite using a small $\mathcal{N}(0, \epsilon I)$ prior for biases.

(b) Prior predictive $\mathbb{E}_{x\sim p(x)}[\mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta})}[p(y|x,\boldsymbol{\theta})]]$ over 10 classes for a Kaiming-scaling prior with a $\mathcal{N}(0, \epsilon I)$ prior for bias terms. We estimate the marginal distribution using $S = 100$ samples $\boldsymbol{\theta}^{(i)}$ and all training images.

*Figure 28.* ResNet-20/CIFAR-10 prior predictive study for a He-scaled Normal prior for `Conv2D` and `Dense` layers and a $\mathcal{N}(0, \epsilon I)$ prior for all bias terms. This prior still concentrates prior mass on functions which output the *same* concentrated label distribution for *all* training instances. It is therefore a bad prior.

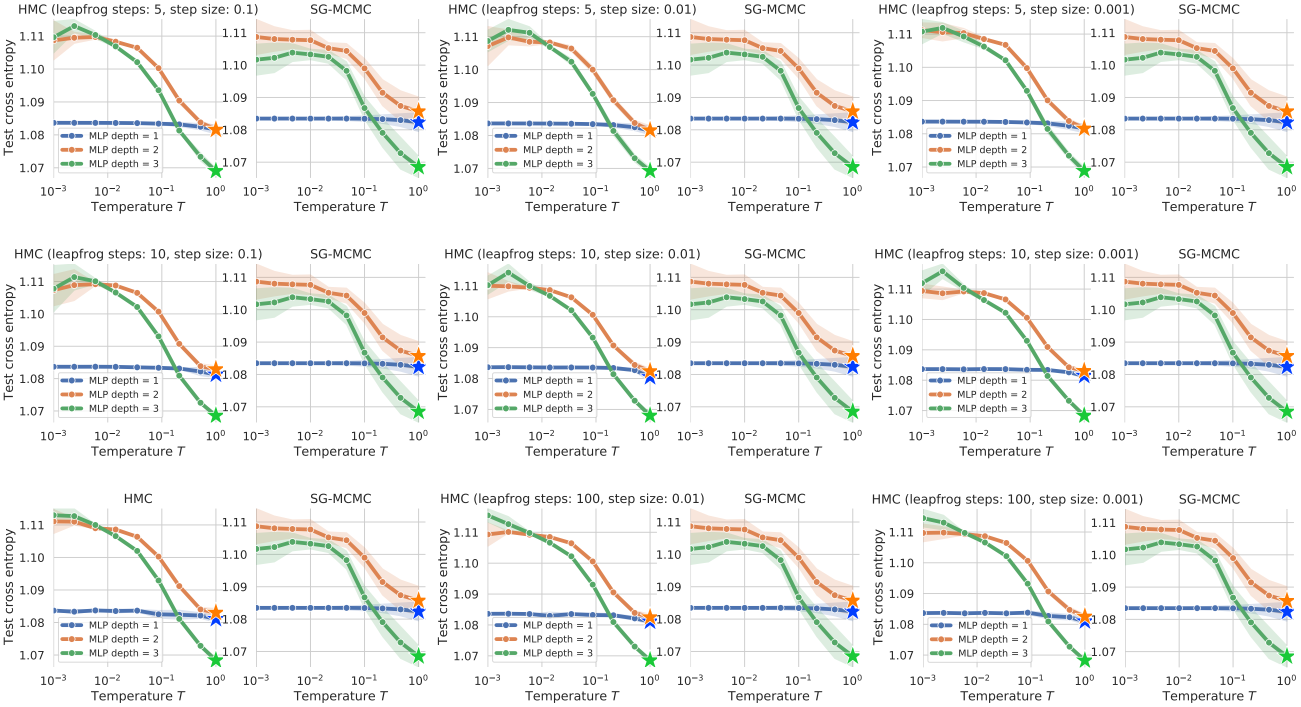


*Figure 29.* Comparisons between SG-MCMC and HMC instantiated with different choices of leapfrog steps $L$ in $\{5, 10, 100\}$ and step sizes $\varepsilon$ in $\{0.001, 0.01, 0.1\}$. The curves show the (held-out) cross entropy versus different temperature levels, aggregated over 5 different runs, for MLPs of various depths (in $\{1, 2, 3\}$ with fixed number of units 10 and relu activation functions). Details about the dataset used can be found in the core paper. The setting $L = 100$ and $\varepsilon = 0.1$ corresponds to the results reported in the main paper.

## References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016.

Ashukha, A., Lyzhov, A., Molchanov, D., and Vetrov, D. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *Eigth International Conference on Learning Representations (ICLR 2020)*, 2020.

Atanov, A., Ashukha, A., Molchanov, D., Neklyudov, K., and Vetrov, D. Uncertainty estimation via stochastic batch normalization. *arXiv preprint arXiv:1802.04893*, 2018.

Bae, J., Zhang, G., and Grosse, R. Eigenvalue corrected noisy natural gradient. *arXiv preprint arXiv:1811.12565*, 2018.

Barber, D. and Bishop, C. M. Ensemble learning for multilayer networks. In *Advances in neural information processing systems*, pp. 395–401, 1998.

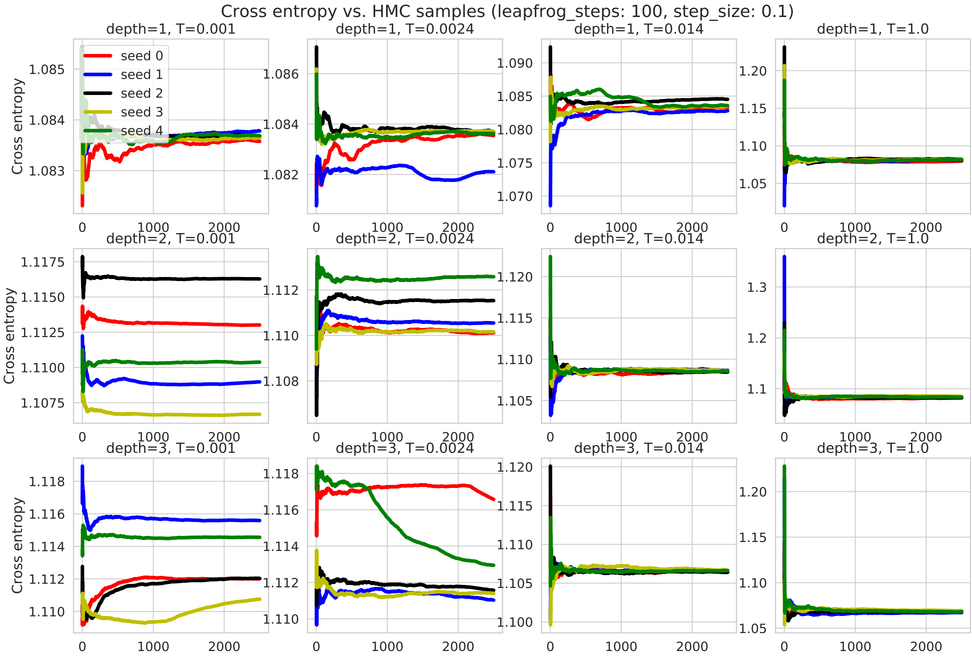


*Figure 30.* Trace plots of the cross entropy: We display the evolution of 5 different chains with respect to the $S = 2500$ HMC samples collected after the burn-in phase, for various depths (rows) and temperatures (columns). Overall, the chains exhibit a converging behavior, with typically more dispersion as the depth and the temperature increase (which is reflected by the ranges of the y-axis that get wider as $T$ and the depth increase).

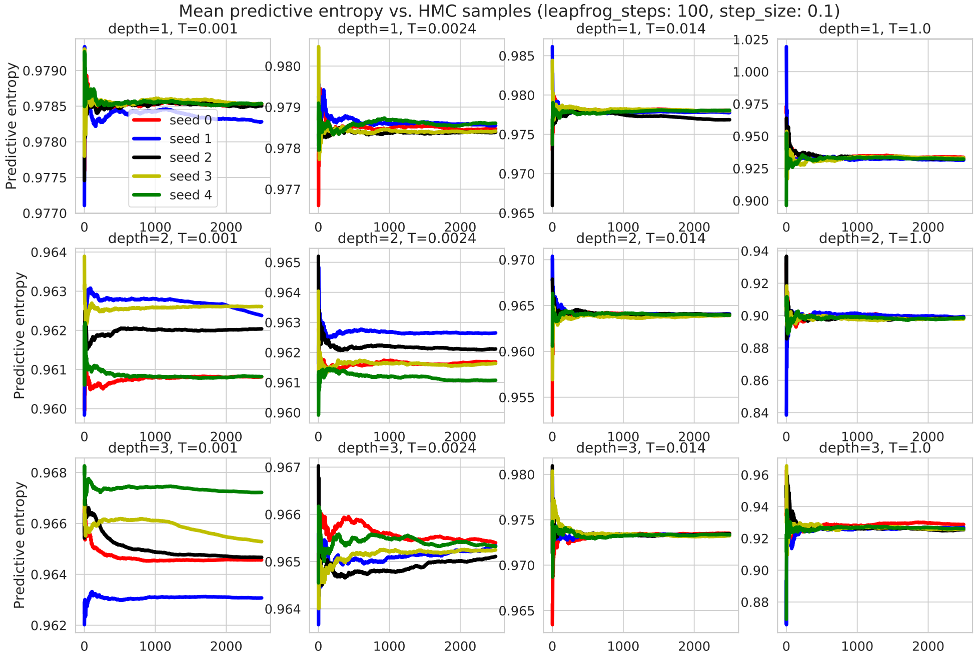


*Figure 31.* Trace plots of the mean predictive entropy (see definition in Section M). We display the evolution of 5 different chains with respect to the $S = 2500$ HMC samples collected after the burn-in phase, for various depths (rows) and temperatures (columns). See further discussions in Figure 30.

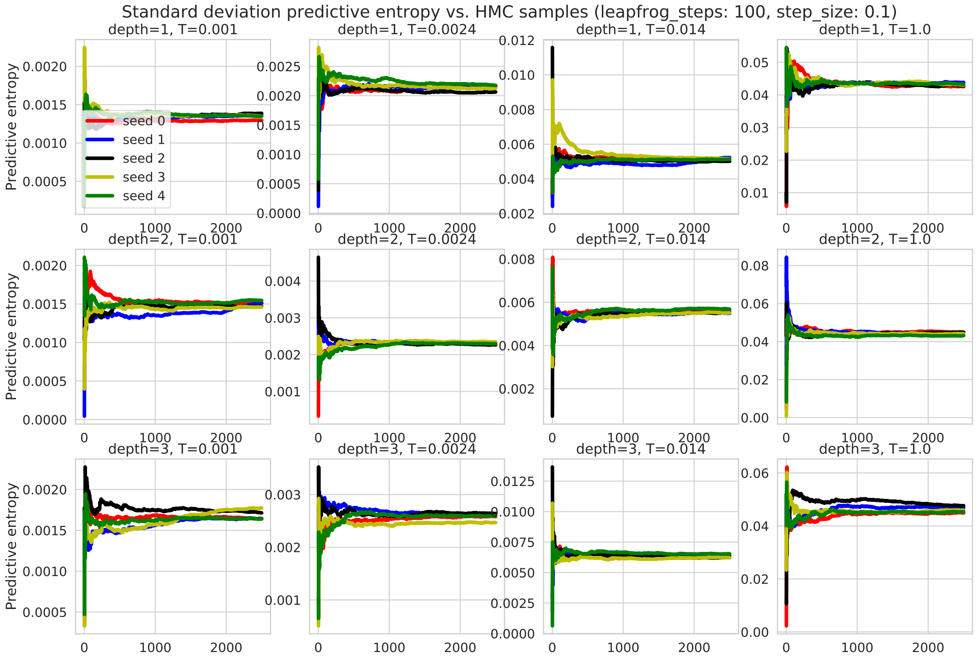


*Figure 32.* Trace plots of the standard deviation of the predictive entropy (see definition in Section M). We display the evolution of 5 different chains with respect to the $S = 2500$ HMC samples collected after the burn-in phase, for various depths (rows) and temperatures (columns). See further discussions in Figure 30.

Berger, J. O. *Statistical decision theory and Bayesian analysis*. Springer, 1985.

Betancourt, M. and Girolami, M. Hamiltonian Monte Carlo for hierarchical models. *Current trends in Bayesian methodology with applications*, 79:30, 2015.

Bhattacharya, A., Pati, D., Yang, Y., et al. Bayesian fractional posteriors. *The Annals of Statistics*, 47(1):39–66, 2019.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. 37:1613–1622, 2015.

Chen, C., Ding, N., and Carin, L. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *Advances in Neural Information Processing Systems*, pp. 2278–2286, 2015.

Chen, T., Fox, E., and Guestrin, C. Stochastic gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*, pp. 1683–1691, 2014.

de G. Matthews, A. G., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. In *ICLR*, 2018.

Dieng, A. B., Ranganath, R., Altosaar, J., and Blei, D. M. Noisin: Unbiased regularization for recurrent neural networks. *arXiv preprint arXiv:1805.01500*, 2018.

Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M., and Saurous, R. A. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017.

Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R. D., and Neven, H. Bayesian sampling using stochastic gradient thermostats. In *Advances in neural information processing systems*, pp. 3203–3211, 2014.

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

Earl, D. J. and Deem, M. W. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

Flam-Shepherd, D., Requeima, J., and Duvenaud, D. Mapping gaussian process priors to bayesian neural networks. In *NIPS Bayesian deep learning workshop*, 2017.

Fushiki, T. et al. Bootstrap prediction and Bayesian prediction under misspecified models. *Bernoulli*, 11(4):747–758, 2005.

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.

Garriga-Alonso, A., Rasmussen, C. E., and Aitchison, L. Deep convolutional networks as shallow gaussian processes. In *ICLR*, 2019.

Geisser, S. *An Introduction to Predictive Inference*. Chapman and Hall, New York, 1993.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Grünwald, P., Van Ommen, T., et al. Inconsistency of Bayesian inference for misspecified linear models, and a proposal for repairing it. *Bayesian Analysis*, 12(4): 1069–1103, 2017.

Hafner, D., Tran, D., Lillicrap, T., Irpan, A., and Davidson, J. Noise contrastive priors for functional uncertainty. *arXiv preprint arXiv:1807.09289*, 2018.

Häggström, O. and Rosenthal, J. On variance conditions for markov chain clts. *Electronic Communications in Probability*, 12:454–464, 2007.

Hayou, S., Doucet, A., and Rousseau, J. On the selection of initialization and activation function for deep neural networks. *arXiv preprint arXiv:1805.08266*, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

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.

Heber, F., Trstanova, Z., and Leimkuhler, B. Tati-thermodynamic analytics toolkit: Tensorflow-based software for posterior sampling in machine learning applications. *arXiv preprint arXiv:1903.08640*, 2019.

Heek, J. and Kalchbrenner, N. Bayesian inference for large scale image classification. In *International Conference on Learning Representations (ICLR 2020)*, 2020.

Hinton, G. and Van Camp, D. Keeping neural networks simple by minimizing the description length of the weights. In *in Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory*, 1993.

Hoffman, M. D. and Gelman, A. The no-u-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1): 1593–1623, 2014.

Inoue, H. Multi-sample dropout for accelerated training and better generalization. *arXiv preprint arXiv:1905.09788*, 2019.

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

Jansen, L. Robust Bayesian inference under model misspecification, 2013. Master thesis.

Jones, G. L. et al. On the Markov chain central limit theorem. *Probability surveys*, 1(299-320):5–1, 2004.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pp. 2575–2583, 2015.

Komaki, F. On asymptotic properties of predictive distributions. *Biometrika*, 83(2):299–313, 1996.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems 30*. 2017.

Langevin, P. Sur la théorie du mouvement brownien. *Compt. Rendus*, 146:530–533, 1908.

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, 2020.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. Deep neural networks as gaussian processes. In *ICLR*, 2018.

Leimkuhler, B. and Matthews, C. *Molecular Dynamics*. Springer, 2016.

Leimkuhler, B., Matthews, C., and Vlaar, T. Partitioned integrators for thermodynamic parameterization of neural networks. *arXiv preprint arXiv:1908.11843*, 2019.

Li, C., Chen, C., Carlson, D., and Carin, L. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Ma, Y.-A., Chen, T., and Fox, E. A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems*, pp. 2917–2925, 2015.

MacKay, D. J. et al. Ensemble learning and evidence maximization. In *Proc. Nips*, volume 10, pp. 4083. Citeseer, 1995.

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.

Masters, D. and Luschi, C. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Nalisnick, E., Hernandez-Lobato, J. M., and Smyth, P. Dropout as a structured shrinkage prior. In *International Conference on Machine Learning*, pp. 4712–4722, 2019.

Neal, R. M. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

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

Nemenman, I., Shafee, F., and Bialek, W. Entropy and inference, revisited. In *Advances in neural information processing systems*, pp. 471–478, 2002.

Noh, H., You, T., Mun, J., and Han, B. Regularizing deep neural networks by noise: Its interpretation and optimization. In *Advances in Neural Information Processing Systems*, pp. 5109–5118, 2017.

Novak, R., Xiao, L., Bahri, Y., Lee, J., Yang, G., Hron, J., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Bayesian deep convolutional networks with many channels are gaussian processes. In *ICLR*, 2019.

Nowozin, S. Debiasing evidence approximations: On importance-weighted autoencoders and jackknife variational inference. In *Sixth International Conference on Learning Representations (ICLR 2018)*, 2018.

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

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. In *Advances in Neural Information Processing Systems (NeurIPS 2019)*, 2019.

Pearce, T., Zaki, M., Brintrup, A., and Neely, A. Expressive priors in bayesian neural networks: Kernel combinations and periodic functions. *arXiv preprint arXiv:1905.06076*, 2019.

Perez, L. and Wang, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

Ramamoorthi, R. V., Sriram, K., and Martin, R. On posterior concentration in misspecified models. *Bayesian Anal.*, 10 (4):759–789, 12 2015. doi: 10.1214/15-BA941.

Särkkä, S. and Solin, A. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. Deep information propagation. In *ICLR*, 2017.

Schucany, W., Gray, H., and Owen, D. On bias reduction in estimation. *Journal of the American Statistical Association*, 66(335):524–533, 1971.

Shang, X., Zhu, Z., Leimkuhler, B., and Storkey, A. J. Covariance-controlled adaptive Langevin thermostat for large-scale Bayesian sampling. In *Advances in Neural Information Processing Systems*, pp. 37–45, 2015.

Shekhovtsov, A. and Flach, B. Stochastic normalizations as Bayesian learning. *arXiv preprint arXiv:1811.00639*, 2018.

Simsekli, U., Sagun, L., and Gurbuzbalaban, M. A tail-index analysis of stochastic gradient noise in deep neural networks. *arXiv preprint arXiv:1901.06053*, 2019.

Singh, S. and Krishnan, S. Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. *arXiv preprint arXiv:1911.09737*, 2019.

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.

Sugita, Y. and Okamoto, Y. Replica-exchange molecular dynamics method for protein folding. *Chemical physics letters*, 314(1-2):141–151, 1999.

Sun, S., Zhang, G., Shi, J., and Grosse, R. Functional variational Bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Swendsen, R. H. and Wang, J.-S. Replica Monte Carlo simulation of spin-glasses. *Physical review letters*, 57 (21):2607, 1986.

Tieleman, T. and Hinton, G. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. Coursera: Neural Networks for Machine Learning, 2012.

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.

Wen, Y., Tran, D., and Ba, J. BatchEnsemble: Efficient ensemble of deep neural networks via rank-1 perturbation. 2019. Bayesian deep learning workshop 2019.

Wilson, A. G. The case for Bayesian deep learning. *NYU Courant Technical Report*, 2019. Accessible at `https://cims.nyu.edu/~andrewgw/caseforbdl.pdf`.

Wolpert, D. H. and Wolf, D. R. Estimating functions of probability distributions from a finite set of samples. *Physical Review E*, 52(6):6841, 1995.

Yaida, S. Fluctuation-dissipation relations for stochastic gradient descent. *arXiv preprint arXiv:1810.00004*, 2018.

Yang, G. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

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

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 (ICLR 2020)*, 2020.

Zhu, Z., Wu, J., Yu, B., Wu, L., and Ma, J. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, 2019.