Appendix for: 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** [7+]

Abstract

This document contains additional details and derivations for the main ICML 2020 paper.

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.[1] 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 1.

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, ℓ-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

[*]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 [7]Microsoft Research. Correspondence to: Florian Wenzel <florianwenzel@google.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

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

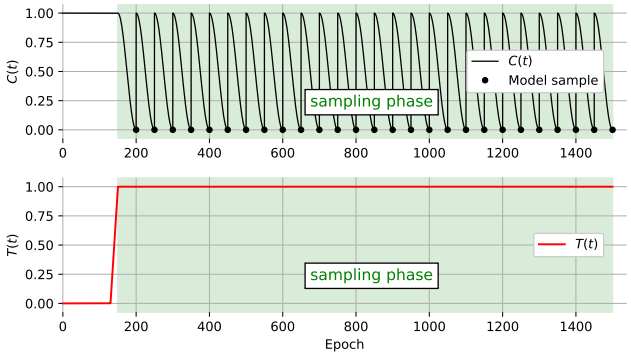


Figure 1. Cyclical time stepping $C(t)$, and temperature ramp-up $T(t)$, as proposed by Zhang et al. (2020) and used in Algorithm 1, for our ResNet-20 CIFAR-10 model (Section A.1). We sample one model at the end of each cycle when the inference accuracy is best, obtaining an ensemble of 27 models.

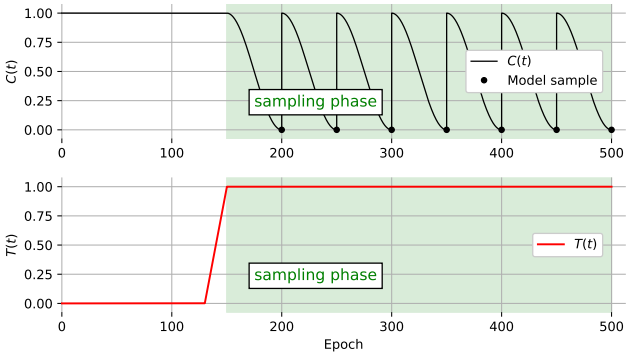


Figure 2. Cyclical time stepping $C(t)$, and temperature ramp-up $T(t)$ for our CNN-LSTM IMDB model (Section A.3). We sample one model at the end of each cycle when the inference accuracy is best, obtaining an ensemble of 7 models.

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[2] 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 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 2.

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 ℓ, momentum decay β) and (timestep h, friction γ) by considering the *instantaneous gradient effect* α 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 ℓ 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 α and β we can derive two non-linear equations that depend on the particular time discretization used; for the symplectic Euler Langevin scheme these are

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

Solving these equations for h and γ simultaneously, given ℓ, n, and β yields the bijection

$$h = \sqrt{\ell/n}, \tag{2}$$

$$\gamma = (1 - \beta)\sqrt{n/\ell}. \tag{3}$$

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

Algorithm 1: Stochastic Gradient Descent with Momentum (SGD) in Tensorflow.

1 **Function** SGD ($\tilde{G}$, $\boldsymbol{\theta}^{(0)}$, ℓ, β)
  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 $t = 1, 2, \ldots$
2 $\mathbf{m}^{(0)} \leftarrow \mathbf{0}$ // Initialize momentum
3 **for** $t = 1, 2, \ldots$ **do**
4 $\mathbf{m}^{(t)} \leftarrow \beta\, \mathbf{m}^{(t-1)} - \ell\, \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta}^{(t-1)})$ // Update momentum
5 $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + \mathbf{m}^{(t)}$ // Update parameters
6 **yield** $\boldsymbol{\theta}^{(t)}$ // Parameter at step t

Algorithm 2: Stochastic Gradient Descent with Momentum (SGD), reparameterized.

1 **Function** SGDEquivalent ($\tilde{G}$, $\boldsymbol{\theta}^{(0)}$, ℓ, β)
  Input: $\tilde{G} : \Theta \to \mathbb{R}$ 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

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 1 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 1 we first perform an equivalent substitution of the moments,

$$\tilde{\mathbf{m}}^{(t)} := \sqrt{\frac{n}{\ell}}\, \mathbf{m}^{(t)}, \qquad \text{respectively,} \tag{4}$$

$$\mathbf{m}^{(t)} := \sqrt{\frac{\ell}{n}}\, \tilde{\mathbf{m}}^{(t)}, \tag{5}$$

we obtain the update from line 4 in Algorithm 1,

$$\sqrt{\frac{\ell}{n}}\, \tilde{\mathbf{m}}^{(t)} \leftarrow \beta \sqrt{\frac{\ell}{n}}\, \tilde{\mathbf{m}}^{(t-1)} - \ell\, \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta}^{(t-1)}. \tag{6}$$

Multiplying both sides of (6) by $\sqrt{n}/\sqrt{\ell}$ we obtain an equivalent form of Algorithm 1 with lines 4 and 5 replaced by

$$\tilde{\mathbf{m}}^{(t)} \leftarrow \beta\, \tilde{\mathbf{m}}^{(t-1)} - \sqrt{\ell n}\, \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta}^{(t-1)}), \tag{7}$$

$$\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + \sqrt{\frac{\ell}{n}}\, \tilde{\mathbf{m}}^{(t)}. \tag{8}$$

From the bijection (2–3) we have $h = \sqrt{\ell/n}$ and $\gamma = (1 - \beta)\sqrt{n/\ell}$. Solving for β gives

$$\beta = 1 - \gamma \sqrt{\frac{\ell}{n}} = 1 - \gamma h. \tag{9}$$

We also have

$$\sqrt{\ell n} = \sqrt{\frac{\ell}{n} n^2} = n \sqrt{\frac{\ell}{n}} = h\, n. \tag{10}$$

Substituting (9) and (10) into (7) and (8) gives the equivalent updates

$$\tilde{\mathbf{m}}^{(t)} \leftarrow (1 - \gamma h)\, \tilde{\mathbf{m}}^{(t-1)} - h\, n\, \nabla_{\boldsymbol{\theta}} \tilde{G}(\boldsymbol{\theta}^{(t-1)}), \tag{11}$$

$$\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + h\, \tilde{\mathbf{m}}^{(t)}. \tag{12}$$

These equivalent changes produce Algorithm 2. Algorithm 1 and Algorithm 2 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 2 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.

Therefore, we estimate a simple diagonal preconditioner that ties together the scale of all parameter elements that belong

Algorithm 3: Estimate Layerwise Preconditioner.

Function `EstimateM`($\tilde{G}$, $\boldsymbol{\theta}$, K, ϵ)

Input: $\tilde{G} : \Theta \to \mathbb{R}$ mean energy function estimate; $(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_S) \in \mathbb{R}^{d_1 \times \cdots \times d_S}$ current model parameter variables; K number of minibatches (default $K = 32$); ϵ 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`
$\sigma_{\min} \leftarrow \min_s \sigma_s$ `// Least sensitive`
for $s = 1, 2, \dots, S$ **do**
  $\mathbf{M}_s \leftarrow \frac{\sigma_s}{\sigma_{\min}} I$
$\mathbf{M} \leftarrow \begin{bmatrix} \mathbf{M}_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \mathbf{M}_S \end{bmatrix}$
return $\mathbf{M}$

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 3.

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.[3] 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.

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

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 tempered 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 temper 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,

$$p_P(\boldsymbol{\theta}|\mathcal{D}) \propto \exp(-U_P(\boldsymbol{\theta})). \tag{13}$$

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

$$D_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p_P(\boldsymbol{\theta}|\mathcal{D})) \tag{14}$$
$$= \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta}) - \log p_P(\boldsymbol{\theta}|\mathcal{D})\right] \tag{15}$$
$$= \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta}) - \log \frac{\exp(-U_P(\boldsymbol{\theta}))}{\int \exp(-U_P(\boldsymbol{\theta}'))\,\mathrm{d}\boldsymbol{\theta}'}\right]. \tag{16}$$

The normalizing integral in (16) is not a function of $\boldsymbol{\theta}$ and thus does not depend on $q(\boldsymbol{\theta})$, allowing us to simplify the equation further:

$$= \mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) - \frac{1}{T}\sum_{i=1}^n \log p(y_i|x_i, \boldsymbol{\theta})\right] \tag{17}$$
$$+ \underbrace{\log \int \exp(-U_P(\boldsymbol{\theta}))\,\mathrm{d}\boldsymbol{\theta}}_{\text{constant, } =: \log E_P} \tag{18}$$
$$= D_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})) - \frac{1}{T}\sum_{i=1}^n \log p(y_i|x_i, \boldsymbol{\theta}) + \log E_P. \tag{19}$$

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 (19) 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 $\lambda := T$,

$$\operatorname*{argmin}_{q\in\mathcal{Q}} D_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p_P(\boldsymbol{\theta}|\mathcal{D})) \tag{20}$$

$$= \operatorname*{argmin}_{q\in\mathcal{Q}} D_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})) - \frac{1}{T}\sum_{i=1}^{n} \log p(y_i|x_i,\boldsymbol{\theta}) \tag{21}$$

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,

$$= \operatorname*{argmin}_{q\in\mathcal{Q}} T D_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})) - \sum_{i=1}^{n} \log p(y_i|x_i,\boldsymbol{\theta}) \tag{22}$$

Substituting $\lambda := T$ yields

$$= \operatorname*{argmin}_{q\in\mathcal{Q}} \lambda D_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})) - \sum_{i=1}^{n} \log p(y_i|x_i,\boldsymbol{\theta}). \tag{23}$$

The last equation, (23) 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),

$$\mathbb{E}[C(S)] = C(\infty) + a_1\frac{1}{S} + a_2\frac{1}{S^2} + \dots. \tag{24}$$

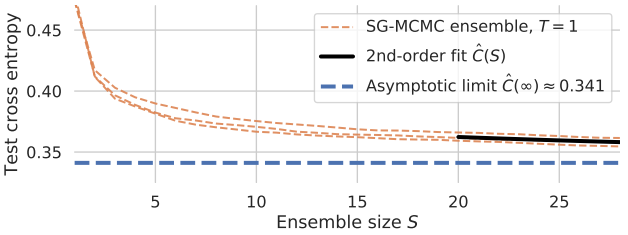


Figure 3. Regressing the limiting ResNet-20/CIFAR-10 ensemble performance: at temperature $T = 1$ an ensemble of size $S = \infty$ would achieve 0.341 test cross-entropy. For SG-MCMC we show three different runs with varying seeds.

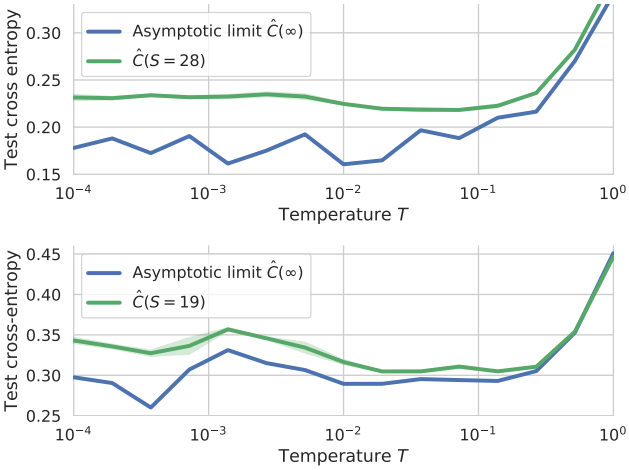


Figure 4. Ensemble variance for ResNet-20/CIFAR-10 (**top**) and CNN-LSTM/IMDB (**bottom**) does not explain poor performance at $T = 1$: even in the infinite limit the performance $C(\infty)$ remains poor compared to $T < 1$.

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 3 we show the regressed test cross-entropy metric obtained by fitting (24) to second order to all samples for $S \geq 20$ close to the asymptotic regime, and visualize the estimate $\hat{C}(\infty)$. In Figure 4 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 3); but *second*, the additional gain that could be obtained from longer sampling is too small to make $T = 1$ superior to $T < 1$ (Figure 4).

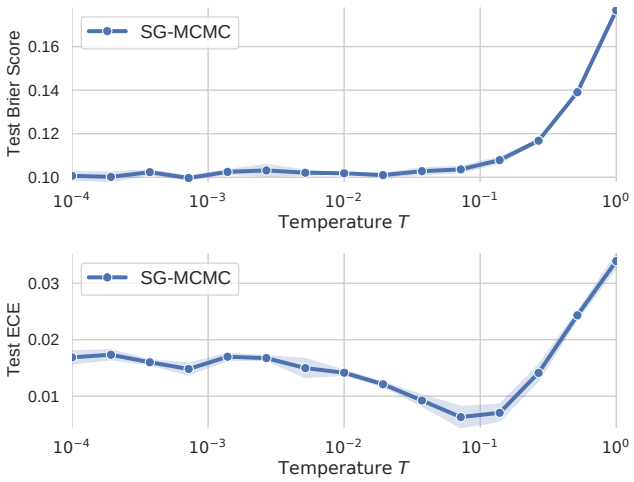


Figure 5. ResNet-20/CIFAR-10: In the main paper we show that cold posteriors improve prediction performance in terms of accuracy and cross entropy (Figure 1 and Figure 2). This plot shows that cold posteriors also improve the uncertainty metrics Brier score and expected calibration error (ECE) (lower is better).

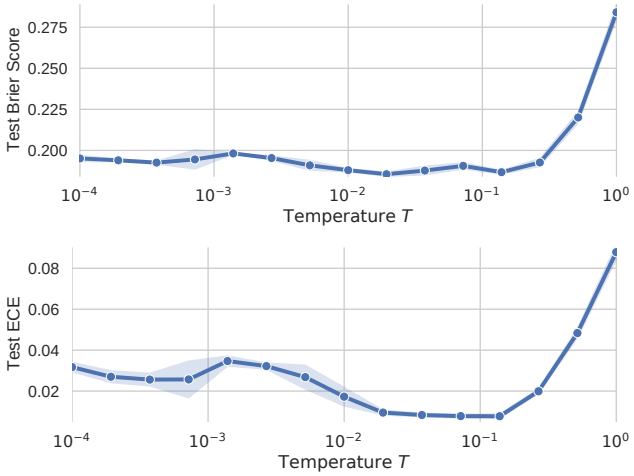


Figure 6. CNN-LSTM/IMDB: Cold posteriors also improve the uncertainty metrics Brier score and expected calibration error (ECE) (lower is better). The plots for accuracy and cross entropy are shown in Figure 3.

G. Cold posteriors improve uncertainty metrics.

In the main paper we show that cold posteriors improve prediction performance in terms of accuracy and cross entropy. Figure 5 and Figure 6 show that for both the ResNet-20 and the CNN-LSTM model, cold posteriors also improve the uncertainty metrics Brier score (Brier, 1950) and expected calibration error (ECE) (Naeini et al., 2015).

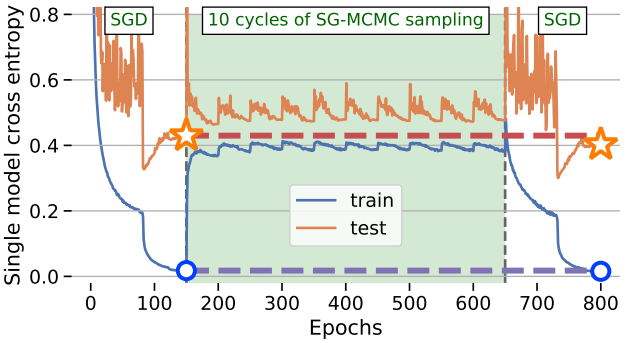


Figure 7. Do the SG-MCMC dynamics harm a beneficial initialization bias used by SGD? We first train a ResNet-20 on CIFAR-10 via SGD, then switch over to SG-MCMC sampling and finally switch back to SGD optimization. We report the single-model test cross entropy of SGD and the SG-MCMC chain as function of epochs. SGD recovers from being initialized by the SG-MCMC state.

H. 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 13. In Figure 7 we additionally report the test cross entropy for the same experiment. 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.

I. 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,*

$$H(\boldsymbol{\theta}, \mathbf{m}) = \frac{1}{T}U(\boldsymbol{\theta}) + \frac{1}{2}\mathbf{m}^T\mathbf{M}^{-1}\mathbf{m}, \tag{25}$$

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

$$T = \frac{\mathbb{E}_{(\boldsymbol{\theta},\mathbf{m})}[\langle B(\boldsymbol{\theta}, \mathbf{m}), \nabla H(\boldsymbol{\theta}, \mathbf{m})\rangle]}{\mathbb{E}_{(\boldsymbol{\theta},\mathbf{m})}[\langle \mathbf{1}_{2d}, \nabla B(\boldsymbol{\theta}, \mathbf{m})\rangle]}. \tag{26}$$

Note that for the Hamiltonian (25) we have, assuming a

symmetric preconditioner, $(\mathbf{M}^{-1})^T = \mathbf{M}^{-1}$,

$$\nabla_{\boldsymbol{\theta}} H(\boldsymbol{\theta}, \mathbf{m}) = \frac{1}{T}\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}), \quad (27)$$

$$\nabla_{\mathbf{m}} H(\boldsymbol{\theta}, \mathbf{m}) = \mathbf{M}^{-1}\,\mathbf{m}. \quad (28)$$

I.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),

$$\mathbf{m} \sim \mathcal{N}(0, \mathbf{M}). \quad (29)$$

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

$$\hat{T}_K(\mathbf{m}) := \frac{\mathbf{m}^T\,\mathbf{M}^{-1}\,\mathbf{m}}{d}, \quad (30)$$

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 χ^2-distribution with d degrees of freedom,

$$(\mathbf{m}^T\mathbf{M}^{-1}\mathbf{m}) \sim \chi^2(d). \quad (31)$$

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

$$J_{T_K}(d, c) := \left(\frac{T}{d}F^{-1}_{\chi^2(d)}\left(\frac{1-c}{2}\right), \frac{T}{d}F^{-1}_{\chi^2(d)}\left(\frac{1+c}{2}\right)\right), \quad (32)$$

where $F^{-1}_{\chi^2(d)}$ is the inverse cumulative distribution function of the χ^2 distribution with d degrees of freedom. By construction if (31) 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 (29) 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.

I.2. Configurational Temperature Estimation

The so called *configurational temperature*[4] is defined as

$$\hat{T}_C(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})) = \frac{\langle \boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})\rangle}{d}. \quad (33)$$

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

$$\mathbb{E}_{\boldsymbol{\theta}\sim\exp(-U(\boldsymbol{\theta})/T)}[\hat{T}_C(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}))] = T. \quad (34)$$

Because (33) 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

$$\hat{T}_C(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})) \sim \mathcal{N}(T, \sigma^2_{T_C}), \quad (35)$$

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 (33) 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.

J. Simulation Accuracy Ablation Study

Equipped with the diagnostics of Section I we can now study how accurate our algorithms simulate the Langevin dynamics. We will demonstrate that layerwise preconditioning and

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

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 8 to 11 and Figures 12a to 12d.

The results indicate that both cosine time stepping and layer-wise 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 8 and Figure 12a) 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 ≥ 88 percent even in cases where the simulation accuracy is poor. This indicates that optimization is different from accurate Langevin dynamics simulation.

K. 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 K.4 we perform an experiment to demonstrate that the dirty likelihood cannot explain cold posteriors.

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

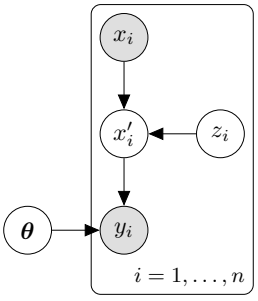


Figure 13. Augmented model with added latent variable z_i.

$$p(Y \mid X, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i \mid x_i, \boldsymbol{\theta}) \tag{36}$$

$$= \prod_{i=1}^{n} \mathbb{E}_{z_i \sim p(z_i)}[p(y_i \mid x_i, z_i, \boldsymbol{\theta})]. \tag{37}$$

Note that in (37) the latent variable z_i is integrated out and therefore the marginal likelihood is a deterministic function.

K.2. Log-likelihood Bound and Jensen Posterior

Given a prior $p(\boldsymbol{\theta})$ the log-posterior for the augmented model in Figure 13 takes the form

$$\log p(\boldsymbol{\theta} \mid \mathcal{D}) \tag{38}$$

$$= C + \log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \log \mathbb{E}_{z_i \sim p(z_i)}[p(y_i \mid x_i, z_i, \boldsymbol{\theta})], \tag{39}$$

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.

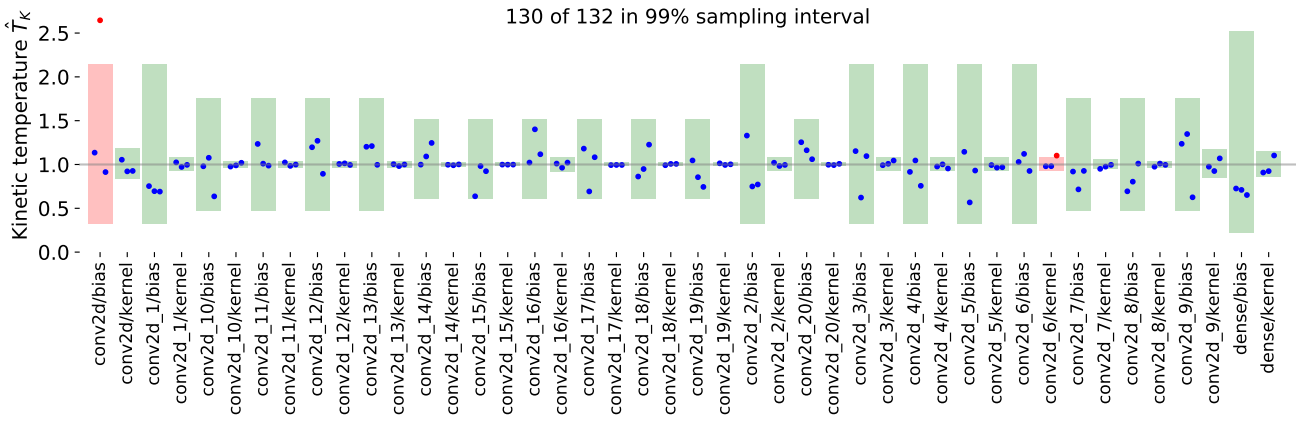


Figure 8. ResNet-20 CIFAR-10 Langevin per-variable kinetic temperature estimates **with preconditioning** and **with cosine time stepping schedule**. The green bars show the 99% true sampling distribution of the Kinetic temperature sample. The blue dots show the actual kinetic temperature samples at 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.

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

$$\geq C + \log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \mathbb{E}_{z_i \sim p(z_i)}[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})], \tag{40}$$

where $C = -\log p(Y|X)$ is the negative model evidence and is constant in $\boldsymbol{\theta}$. We call equation (40) the *Jensen bound* to the log-posterior $\log p(\boldsymbol{\theta}|\mathcal{D})$.

Jensen Posterior. Because we can estimate (40) in an unbiased manner, we will see that many popular methods

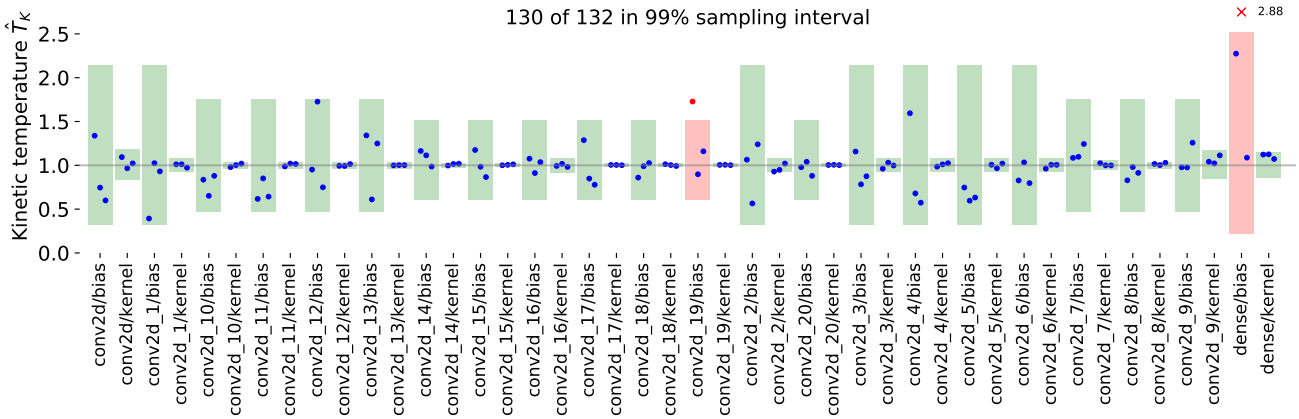


Figure 9. ResNet-20 CIFAR-10 Langevin per-variable kinetic temperature estimates **without preconditioning** but **with cosine time stepping schedule**. Two out of 132 variables are outside the 99% hpd region, indicating accurate simulation.

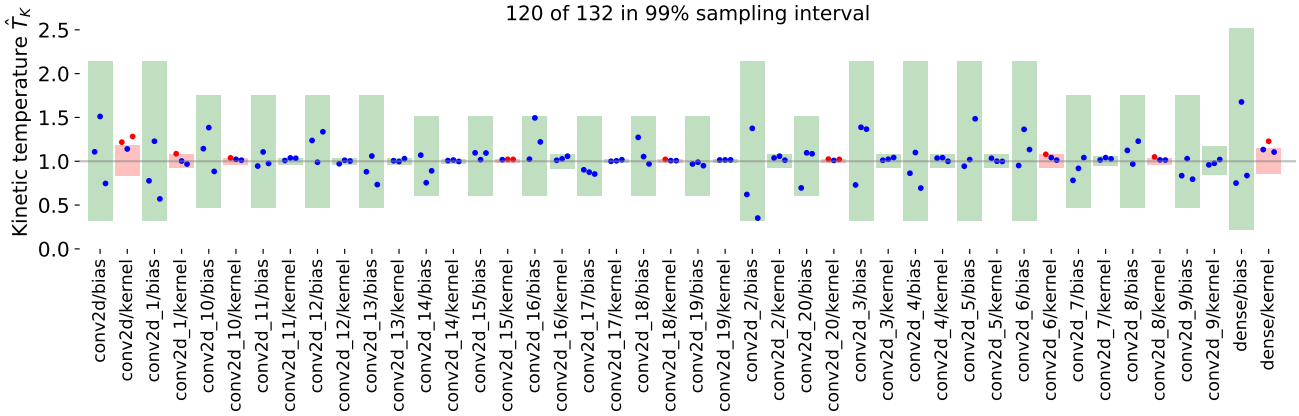


Figure 10. ResNet-20 CIFAR-10 Langevin per-variable kinetic temperature estimates **with preconditioning** but **without cosine time stepping schedule** (flat schedule). 12 out of 132 variables are too hot (boxes in red) and lie outside the acceptable region, indicating an inaccurate simulation of the Langevin dynamics. However, there is a marked improvement due to preconditioning compared to no preconditioning (Figure 11).

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 (40). Formally, the Jensen posterior is

$$p_J(\boldsymbol{\theta} \,|\, \mathcal{D}) :\propto \tag{41}$$

$$p(\boldsymbol{\theta}) \prod_{i=1}^{n} \exp\left(\mathbb{E}_{z_i \sim p(z_i)} \left[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\right]\right). \tag{42}$$

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 (39–40), 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

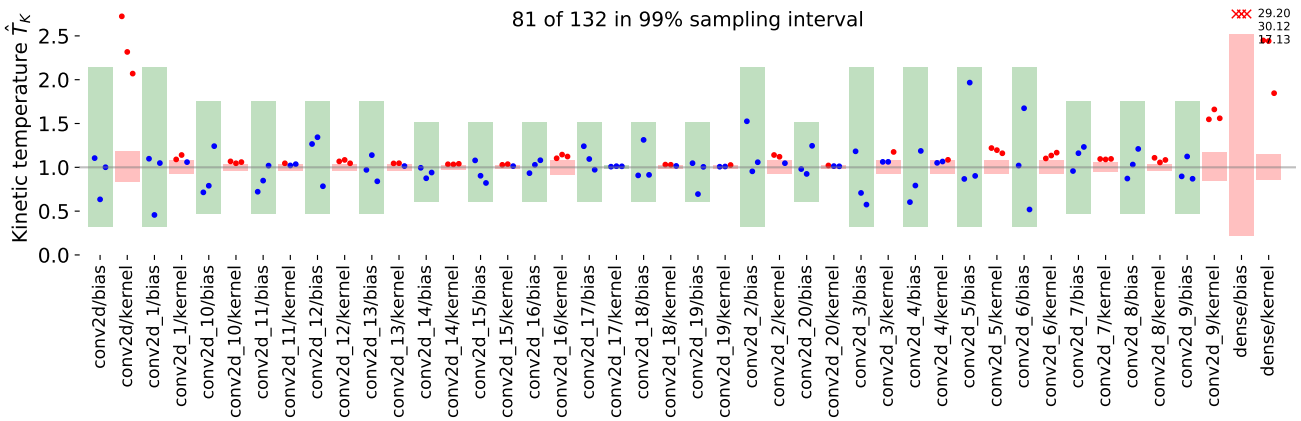


Figure 11. ResNet-20 CIFAR-10 Langevin per-variable kinetic temperature estimates **without preconditioning** and **without cosine time stepping schedule** (flat schedule). 51 out of 132 kinetic temperature samples are too hot (shaded in red) and lie outside the acceptable region, sometimes severely so, indicating a very poor simulation accuracy for the Langevin dynamics.

functions; each weighting function measures a local *Jensen gap* related to each instance. Although we did not pursue this line, the local Jensen gap (47) 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*

$$p_J(\boldsymbol{\theta} \,|\, \mathcal{D}) = p(\boldsymbol{\theta} \,|\, \mathcal{D}). \tag{43}$$

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

$$p(\boldsymbol{\theta} \,|\, \mathcal{D}) = p(\boldsymbol{\theta}) \prod_{i=1}^{n} \int p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\, p(z_i)\, \mathrm{d}z_i, \tag{44}$$

and the Jensen posterior as

$$p_J(\boldsymbol{\theta} \,|\, \mathcal{D}) := p(\boldsymbol{\theta}) \prod_{i=1}^{n} \exp\left(\mathbb{E}_{z_i \sim p(z_i)} \left[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\right]\right), \tag{45}$$

respectively. If we define the *Jensen prior*,

$$p_J(\boldsymbol{\theta}) :\propto w(\boldsymbol{\theta})\, p(\boldsymbol{\theta}), \tag{46}$$

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

$$w_i(\boldsymbol{\theta}) := \frac{\int p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\, p(z_i)\, \mathrm{d}z_i}{\exp\left(\mathbb{E}_{z_i \sim p(z_i)} [\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})]\right)}. \tag{47}$$

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 prior in (45) we obtain

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

$$\propto p_J(\boldsymbol{\theta}) \prod_{i=1}^{n} \exp\left(\mathbb{E}_{z_i \sim p(z_i)} \left[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\right]\right), \tag{49}$$

$$= p(\boldsymbol{\theta}) \left(\prod_{i=1}^{n} w_i(\boldsymbol{\theta})\right) \tag{50}$$

$$\prod_{i=1}^{n} \exp\left(\mathbb{E}_{z_i \sim p(z_i)} \left[\log p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\right]\right), \tag{51}$$

$$= p(\boldsymbol{\theta}) \prod_{i=1}^{n} \int p(y_i \,|\, x_i, z_i, \boldsymbol{\theta})\, p(z_i)\, \mathrm{d}z_i \tag{52}$$

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

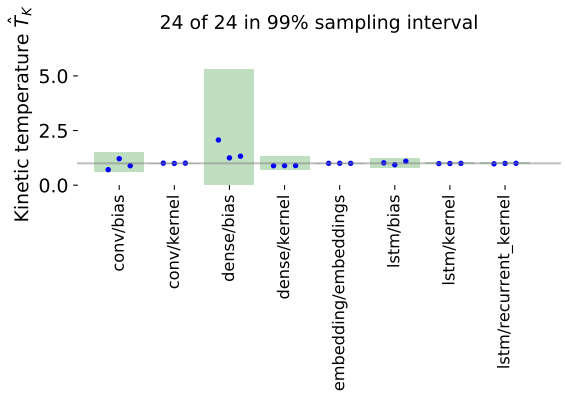
This constructively demonstrates the result (43). □

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

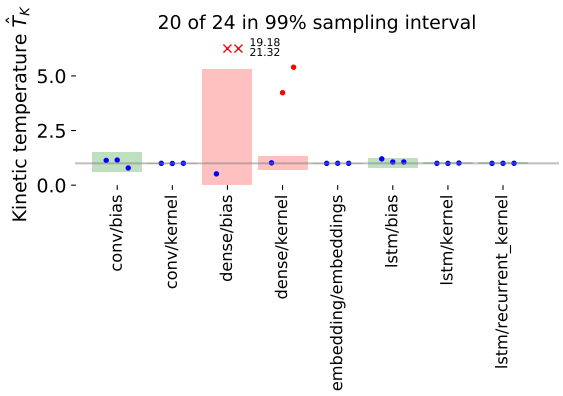
K.3. Deep Learning Techniques Optimize Jensen Posteriors

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 $p_J(\boldsymbol{\theta} \,|\, X, Y)$.

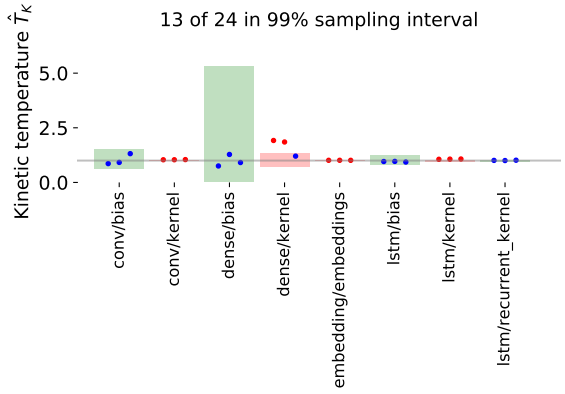
The connection between dropout and applying Jensen's bound has been discovered before by several groups (Noh et al., 2017), (Nalisnick et al., 2019), (Inoue, 2019), and contrasts sharply with the variational inference interpretation of dropout, (Kingma et al., 2015; Gal & Ghahramani,



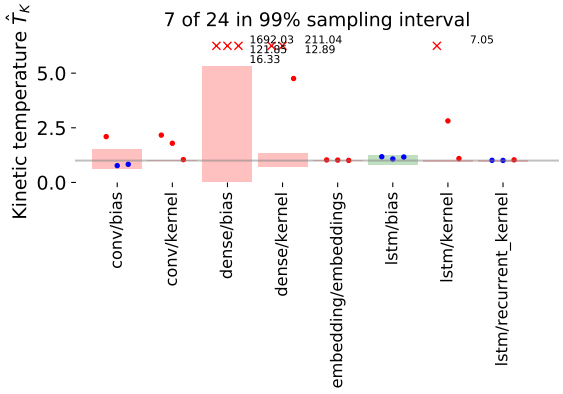
(a) Preconditioning, cosine stepping



(b) No preconditioning, cosine stepping



(c) Preconditioning, no cosine stepping



(d) No preconditioning, no cosine stepping

Figure 12. 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.

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 13 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 14, 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).

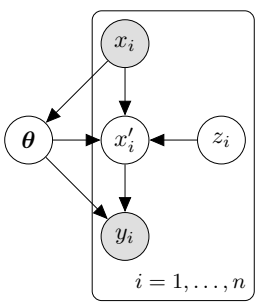


Figure 14. Augmented model for batch normalization.

With the above modifications all derivations in Section K.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.

K.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 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 15 on page 14 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 ± 0.015 is comparable to 0.3918 ± 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.

L. 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 sig-

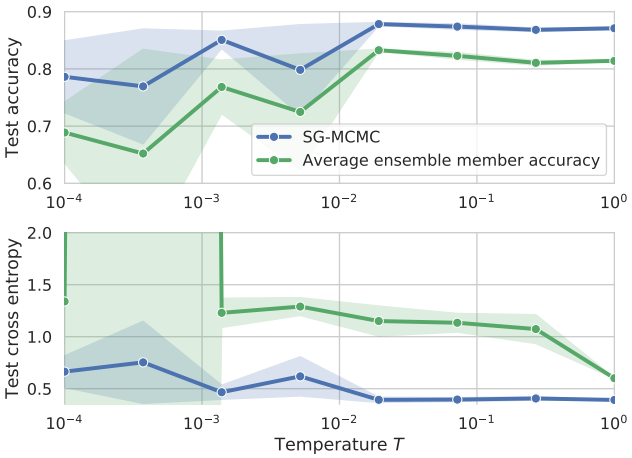


Figure 15. ResNet-20 with filter response normalization (FRN) instead of batch normalization and without any use of data augmentation.

nificant 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. Figure 12 shows that the cold posterior effect is present regardless of the scaling of the variance of the Normal prior. In the following we investigate certain scaling laws of the prior more detailed.

L.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).

$$p(\boldsymbol{\theta}_j) = \mathcal{N}\left(0, \frac{2}{b_j}\right), \tag{54}$$

where b_j is the *fan-in* of the j'th layer.[5]

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

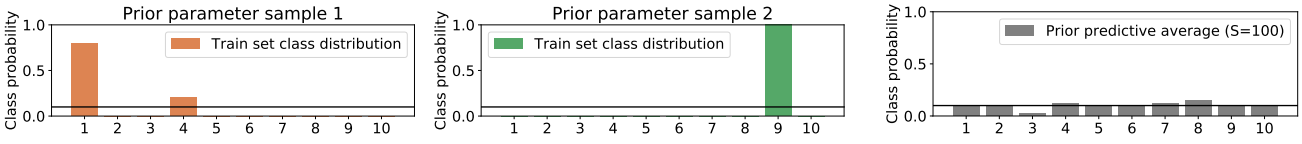
He-scaled Prior ResNet-20 CIFAR-10 Experiment: We also perform the original cold posterior experiment from the main paper with the He-scaling Normal prior. We show the temperature-dependence curves for test accuracy and test cross-entropy in Figure 17. The overall performance drops compared to the $\mathcal{N}(0, I)$ 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.

L.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).

[5] 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 $b_j = k^2 d$.



(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 16. 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.

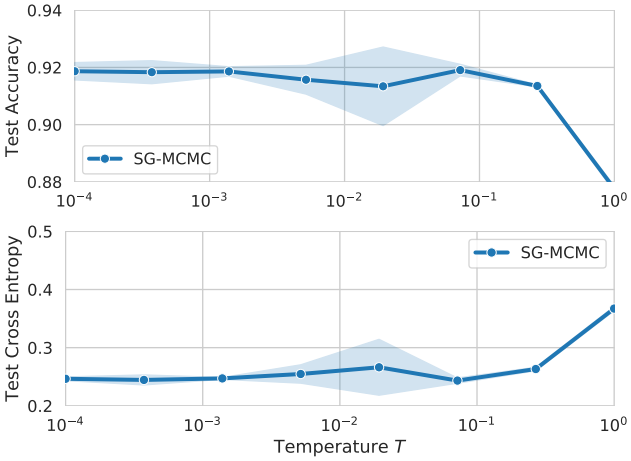


Figure 17. ResNet-20 on CIFAR-10 with He-scaling Normal prior (He-scaled Normal for `Conv2D` and `Dense` layers, and $\mathcal{N}(0, I)$ for all bias terms). The cold posterior effect remains: the poor predictive performance of the Bayes posterior at $T = 1$ holds for both accuracy and cross-entropy.

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 18a shows two prior draws and the resulting class distributions marginalized over the entire training set. Figure 18b 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.

M. Tempering the Observation Model?

In (Wilson & Izmailov, 2020), Equation (4) a proposal is made to use a different likelihood function of the form

$$p_T(y|x,\boldsymbol{\theta}) :\propto p(y|x,\boldsymbol{\theta})^{1/T}. \tag{55}$$

It is claimed that with this adjusted observation model the cold posterior is simply the ordinary Bayes posterior of the modified model. Indeed, if we are to plug the right hand side of (55) directly into our posterior energy function (2) we obtain the cold posterior energy function,

$$U_T(\boldsymbol{\theta}) := -\sum_{i=1}^{n} \frac{1}{T} \log p(y_i|x_i,\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}). \tag{56}$$

The mistake in this derivation is to ignore that renormalization of $p_T(y|x,\boldsymbol{\theta})$ must be carried out because the normalizing constant is not invariant of $\boldsymbol{\theta}$. In particular, this is in contrast to typical applications of Bayes rule for posteriors, where we can indeed write $p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$ without worries, as here the normalizing constant does not depend on $\boldsymbol{\theta}$. One consequence of this mistake is that $U_T(\boldsymbol{\theta})$ is *not* necessarily the energy function of a Bayes posterior.

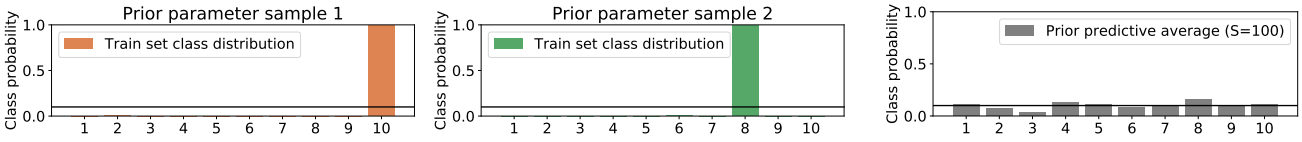
Instead, for the tempered observation model proposed by (Wilson & Izmailov, 2020) the correctly normalized observation likelihood is

$$p_T(y|x,\boldsymbol{\theta}) = \frac{p(y|x,\boldsymbol{\theta})^{1/T}}{\int p(y|x,\boldsymbol{\theta})^{1/T}\,\mathrm{d}y}. \tag{57}$$

Using this normalized observation model, the correct Bayes posterior energy corresponding to $p_T(y|x,\boldsymbol{\theta})$ is

$$\begin{aligned}\tilde{U}_T(\boldsymbol{\theta}) \quad := \quad & -\sum_{i=1}^{n} \frac{1}{T} \log p(y_i|x_i,\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) \\ & + \sum_{i=1}^{n} \log \int p(y|x_i,\boldsymbol{\theta})^{1/T}\,\mathrm{d}y. \end{aligned} \tag{58}$$

Therefore, when the observation model is transformed as in (55) and as suggested by (Wilson & Izmailov, 2020), then in order to obtain a normalized observation model we must include the correction term (58) and this produces a modified energy function, $\tilde{U}_T(\boldsymbol{\theta})$, that differs from the actual cold posterior energy function $U_T(\boldsymbol{\theta})$.



(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 18. 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.

Is there a way to "fix" this mistake? I.e. can one construct an observation model such that the resulting Bayes posterior corresponds to a tempered version of the Bayes posterior of the original observation model? For classification we found a way: we assign probability to a pseudo event "$\emptyset$" which cannot occur. To see this, assume a classification model $p(y|x)$ where $y \in \{1, 2, \ldots, K\}$. Clearly $\sum_{k=1}^{K} p(y = k|x) = 1$. Given a temperature $T \leq 1$ we define

$$\tilde{p}(y = k|x) := p(y = k|x)^{1/T}. \tag{59}$$

Clearly for $0 < T \leq 1$ we have that $f(x) = x^{1/T}$ is a monotonic function in $x \in [0, 1]$ and $f(x) = x^{1/T} \leq x$, and therefore $\sum_k \tilde{p}(y = k|x) \leq 1$. We absorb the remaining probability mass into a pseudo event "$\emptyset$",

$$\tilde{p}(y = \emptyset|x) := 1 - \sum_{k=1}^{K} \tilde{p}(y = k|x), \tag{60}$$

such that the resulting distribution $\tilde{p}$ over $K+1$ basic events sums to one,

$$\sum_{k\in\{1,2,\ldots,K,\emptyset\}} \tilde{p}(y = k|x) = 1. \tag{61}$$

Now observe that for any event in $\{1, 2, \ldots, K\}$ that actually *can* occur we have

$$\log \tilde{p}(y = k|x) = \frac{1}{T} \log p(y = k|x), \tag{62}$$

that is, we have achieved the effect of temperature scaling when using $\tilde{p}$ as observation model. While formally possible, can we make sense of this transformation and introduction of a pseudo event?

To us it seems entirely non-Bayesian to artificially introduce events into a model while knowing with perfect certainty that these events cannot happen and then allow the model to assign probability mass to those events. It is non-Bayesian because our knowledge with respect to the new event is perfect: it cannot occur. Therefore a model should respect this knowledge of the world.

N. 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 4.2 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[6], 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.

O. Details about Hamiltonian Monte Carlo

In this section, we describe practical considerations about Hamiltonian Monte Carlo (HMC) and present further results about its comparison with SG-MCMC (see Section 4.2).

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

- The number L of steps of the leapfrog integrator,

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

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

O.1. 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 20, 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 et al., 2017).[7]

O.2. Convergence monitoring

We monitor convergence by first inspecting trace plots and second by computing standard diagnostics, namely the effective sample size (ESS) (Brooks et al., 2011) and the potential scale reduction (PSRF) (Gelman & Rubin, 1992).

Trace plots. In Figures 21-22-23, 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[8] 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 21), 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 22 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 23, which we refer to as the standard deviation of the predictive entropy.

As a general observation, we can see on Figures 21-22-23 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 21-22-23 that get wider as T and the depth become larger).

ESS and PSRF. The effective sample size (ESS) (Brooks et al., 2011) measures how independent the samples are in terms of the auto-correlations within the sequence at different lags. The potential scale reduction factor (PSRF) (Gelman & Rubin, 1992) assesses the convergence of the chains (to the same target distribution) by testing for equality of means.

We computed ESS and PSRF for our HMC simulation (with 100 leapfrog steps and a step size of 0.1, as reported in the main paper). We used the TFP implementations `tfp.mcmc.{effective_sample_size, potential_scale_reduction}`. Figure 19 (left, middle) displays the ESS and PSRF with respect to the different temperature levels, for the 3 MLP depths. Both ESS and PSRF were averaged over the model parameters.

We observe that in the regime T in $[0.05, 1]$, the diagnostics indicate an approximate convergence (PSRF < 1.05 and ESS

[7] `tfp.mcmc.DualAveragingStepSizeAdaptation`.
[8] We limit ourselves to four temperatures to avoid clutter.

in $[1800, S]$, with $S = 2500$ total samples) for the 3 MLP depths. On the other hand, in the regime T in [0.001, 0.05], the diagnostics only continue to indicate an approximate convergence for the depth 1. For depths 2 and 3, both diagnostics substantially degrade, e.g., ESS down to ≈ 189 for depth 3.

O.3. KL divergence between predictive distributions

In Section 4.2, we compare side by side the cross-entropy of SG-MCMC and HMC for the different temperature levels, exhibiting a close agreement.

As an alternative visualization of this comparison, we computed the (symmetrized) KL divergence between the SG-MCMC and HMC predictive distributions (i.e., in our setting, categorical distributions with 3 classes).

For SG-MCMC and HMC (instantiated with 100 leapfrog steps and a step size of 0.1, as reported in the main paper), Figure 19 (right) displays the (symmetrized) KL with respect to the different temperature levels, for the 3 MLP depths (averaged over the seeds). We observe that all KLs are small (in the order of $\approx 10^{-5}$ for depth 1, and $\approx 10^{-3}$ for depths 2 and 3).

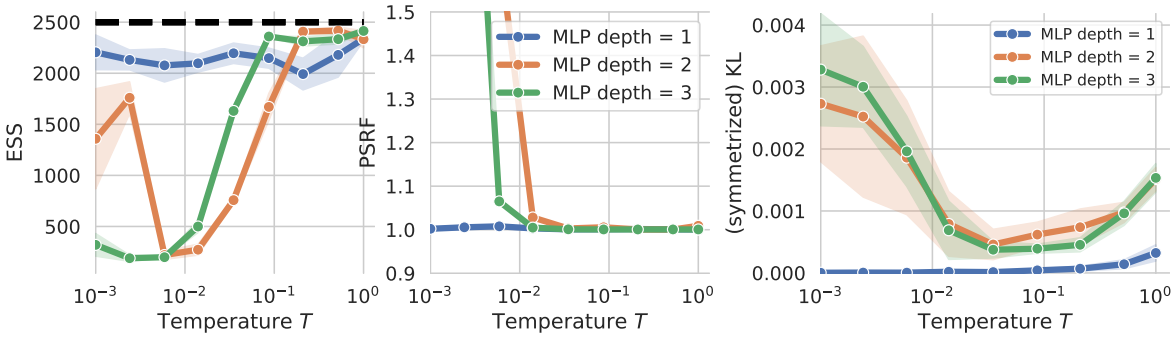


Figure 19. For HMC (instantiated with 100 leapfrog steps and a step size of 0.1, as reported in the main paper Section 4.2), we report the effective sample size (left) and potential scale reduction factor (middle) with respect to the different temperature levels. On the left plot, the black dash line corresponds to the $S = 2500$ samples and ESS=S indicates no correlation in the sequences. For PSRF, approximate convergence is generally considered when PSRF < 1.2 (Gelman & Rubin, 1992). (right) KL divergence between the predictive distributions of HMC and SG-MCMC.

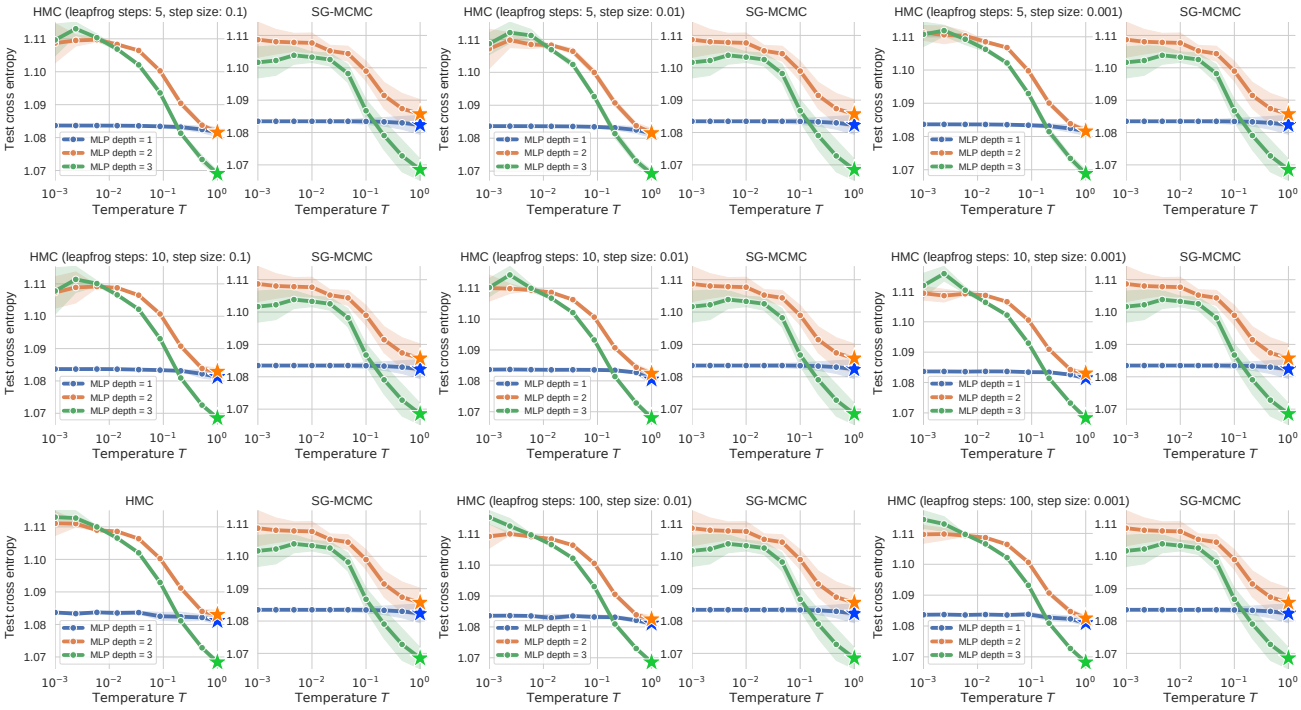


Figure 20. Comparisons between SG-MCMC and HMC instantiated with different choices of leapfrog steps L in $\{5, 10, 100\}$ and step sizes ε in $\{0.001, 0.01, 0.1\}$. The curves show the (held-out) cross entropy versus different temperature levels, aggregated over 5 different runs, for MLPs of various 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.

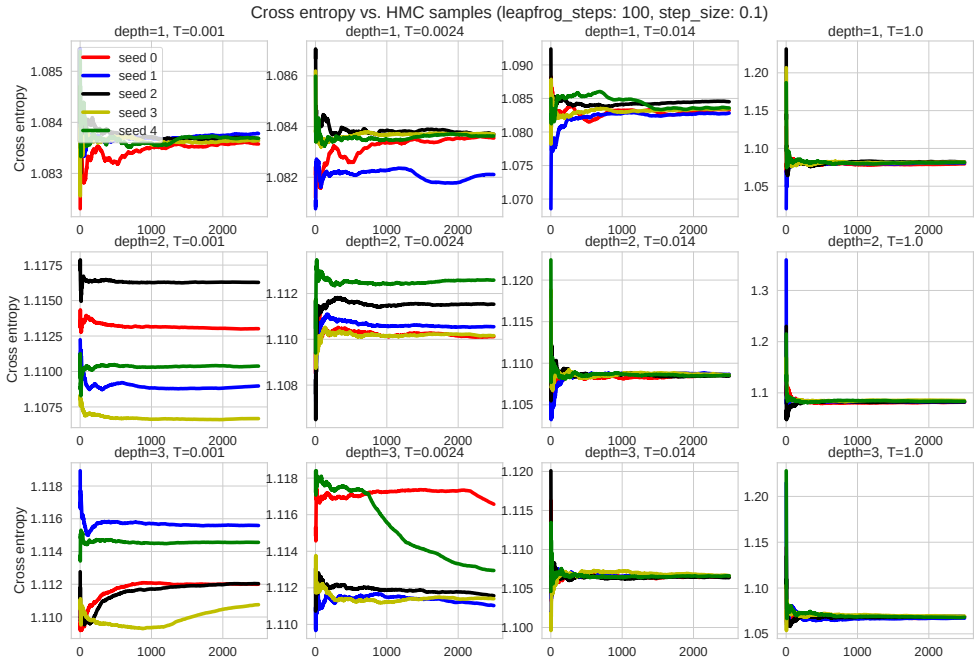


Figure 21. Trace plots of the cross entropy: We display the evolution of 5 different chains with respect to the $S = 2500$ HMC samples collected after the burn-in phase, for various depths (rows) and temperatures (columns). Overall, the chains exhibit a converging behavior, with typically more dispersion 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).

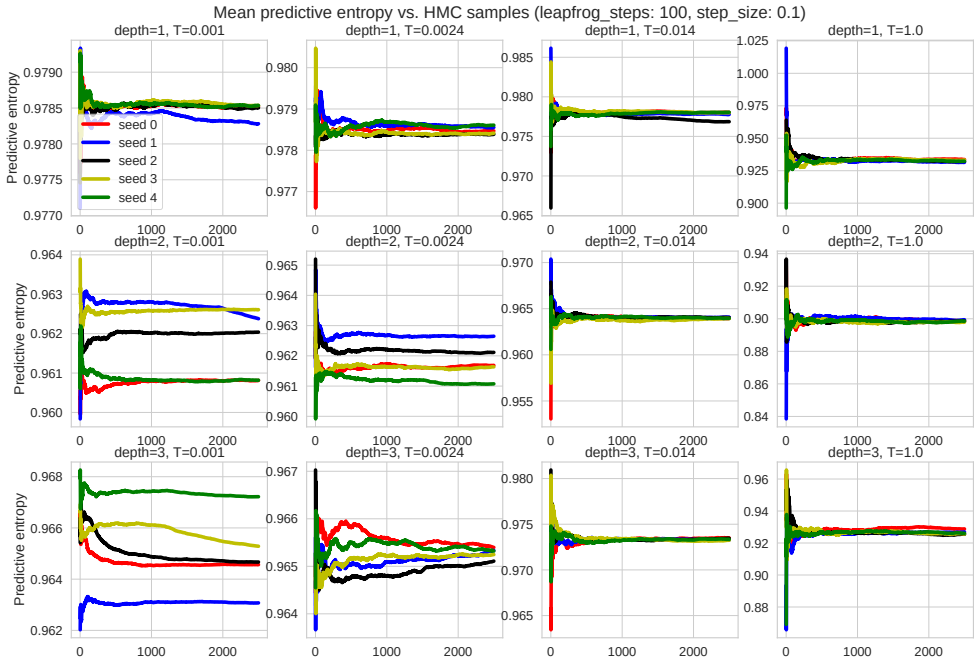


Figure 22. Trace plots of the mean predictive entropy (see definition in Section O). We display the evolution of 5 different chains with respect to the $S = 2500$ HMC samples collected after the burn-in phase, for various depths (rows) and temperatures (columns). See further discussions in Figure 21.

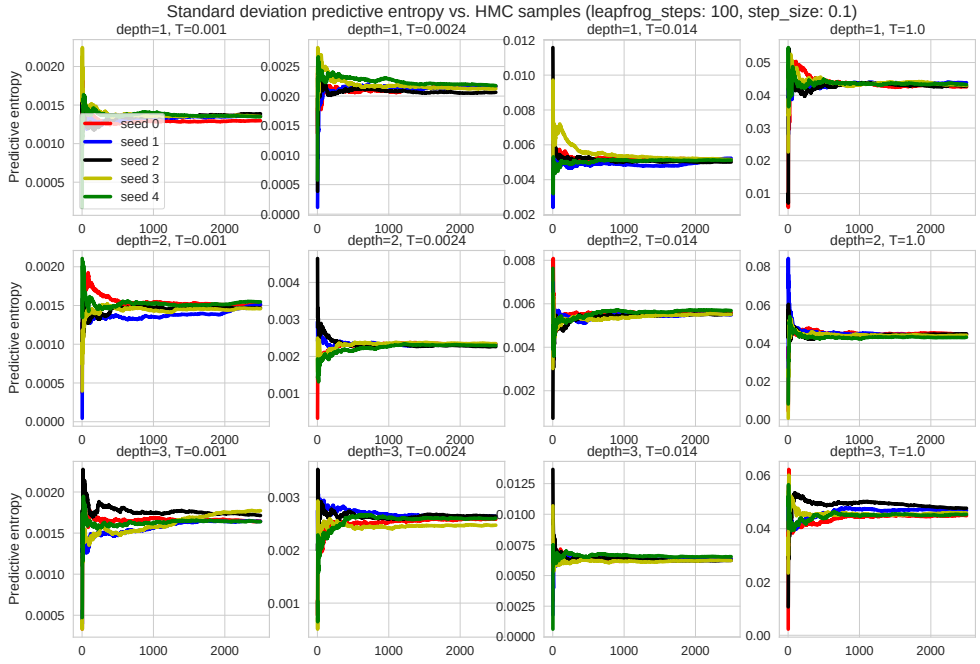


Figure 23. Trace plots of the standard deviation of the predictive entropy (see definition in Section O). We display the evolution of 5 different chains with respect to the $S = 2500$ HMC samples collected after the burn-in phase, for various depths (rows) and temperatures (columns). See further discussions in Figure 21.

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.

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

Brier, G. W. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

Brooks, S., Gelman, A., Jones, G., and Meng, X. *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press, 2011. ISBN 9781420079425. URL https://books.google.de/books?id=qfRsAIKZ4rIC.

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.

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.

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

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.

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.

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.

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.

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

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.

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

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

Naeini, M. P., Cooper, G., and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

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.

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.

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

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.

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

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.

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

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, 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.