

What Are Bayesian Neural Network Posteriors Really Like?

1. INTRO: WHAT DOES IT MEAN TO BE BAYESIAN AND WHY DO WE CARE?

Basic case, consider linear regression,

$$y = wx + \epsilon$$

where $\epsilon \sim N(0, \sigma^2)$. The standard machine learning approach would find the setting of parameters w that maximizes the likelihood,

$$\max_w \sum_{i=1}^N \log N(y_i | w\phi(x_i), \sigma^2) \iff \min_w \frac{1}{N} \sum_{i=1}^N (y_i - w\phi(x_i))^2$$

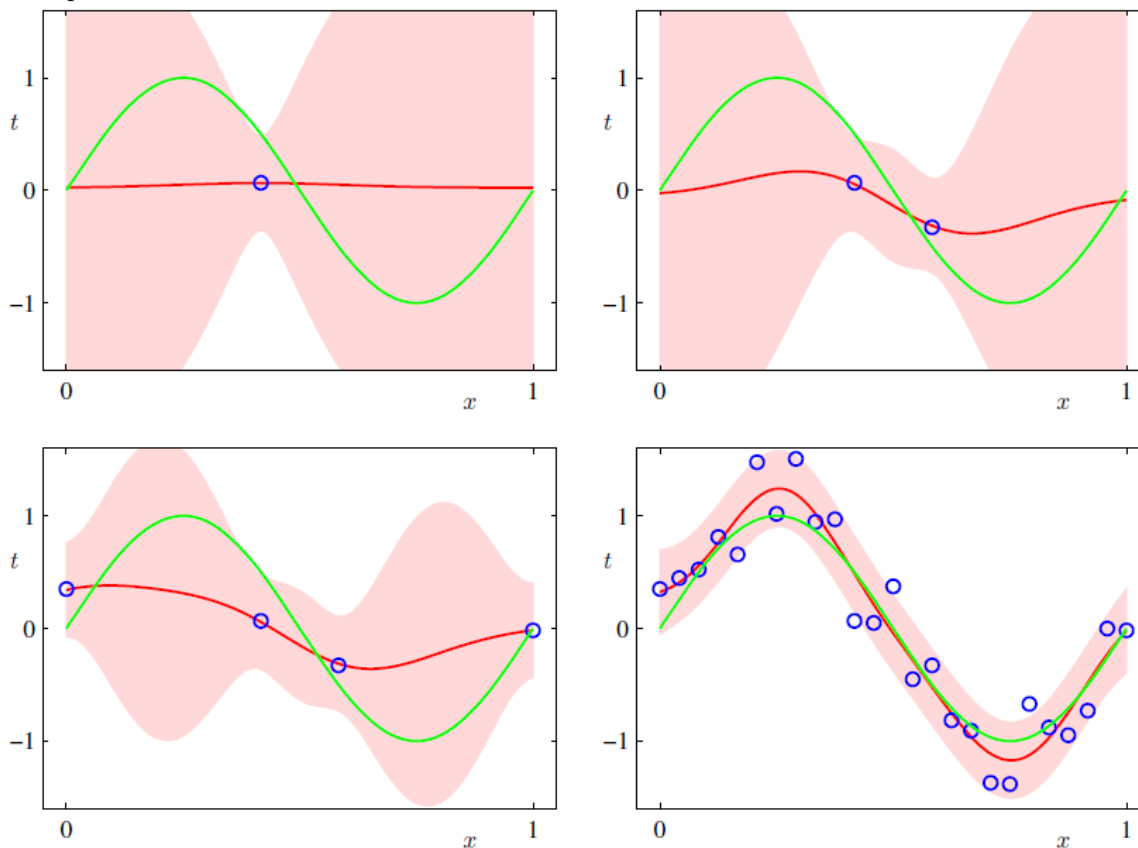
Imagine that we have a small number of data points, we have no way here of expressing our uncertainty about parameter settings. Introduce a prior $p(w)$ over parameters, now using Bayes rule,

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

This allows us to do *Bayesian model averaging* (BMA),

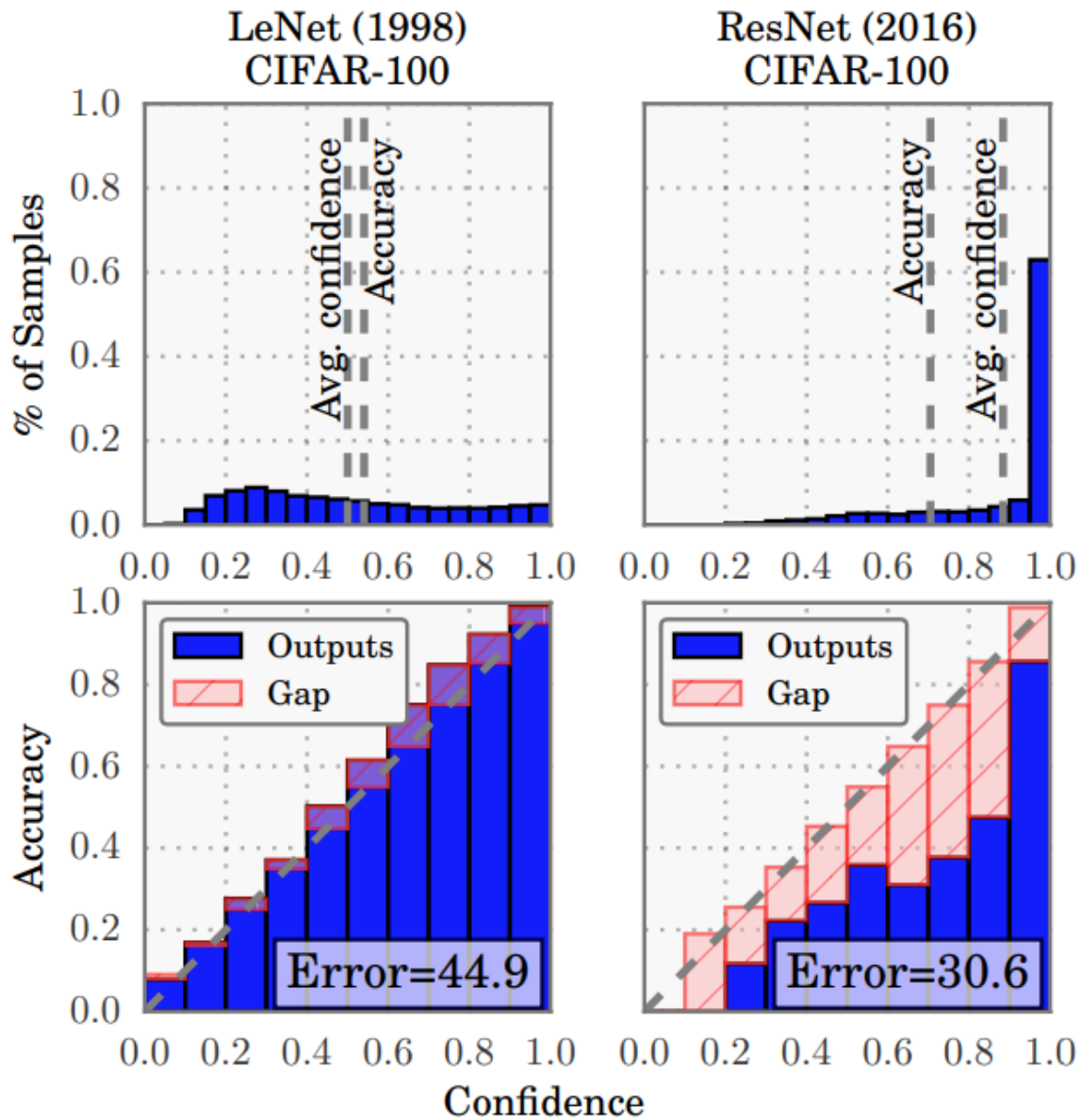
$$p(y^*|x^*, D) = \int_w p(y^*|x^*, w)p(w|D)dw$$

aka the predictive distribution.



Notice that the first term in the integral is our estimate for a single parameter setting. If we leave out the second term the consequence is ****overconfident predictions****, but it is important that

$p(y|x)$ should represent real probabilities, i.e. be calibrated. Calibration is increasingly a problem in modern neural networks (Weinberger 17)



2. BAYESIAN INFERENCE IS CHALLENGING

- (a) posterior is intractable
- (b) millions of parameters
- (c) what prior should we use?

3. WHAT IS INTRACTABLE EXACTLY?

I will show you that in even relatively simple cases the posterior is intractable. (Blei) Consider the inference problem where we want to estimate some latent variables z given some observations

x , $p(z|x)$. We write the conditional density,

$$p(z|x) = \frac{p(z, x)}{p(x)}$$

How do we get this denominator, "the evidence"? We marginalize out the latent variables,

$$p(x) = \int p(z, x) dz$$

This evidence integral in many cases does not have a closed form or takes exponential time to compute. As a simple example consider the Bayesian mixture of unit-variance univariate Gaussians. For the case of K Gaussians with means $\mu = \{\mu_1, \dots, \mu_K\}$, we draw mean parameters from a common prior $p(\mu_k)$ which is a Gaussian $N(0, \sigma^2)$ and let σ^2 be a hyperparameter. Generating an observation x_i entails choosing a cluster assignment c_i then drawing x_i from the gaussian corresponding to this cluster $N(c_i^T \mu, 1)$. The full model is

$$\mu_k \sim N(0, \sigma^2)$$

$$c_i \sim \text{Cat}(1/K, \dots, 1/K)$$

$$x_i | c_i, \mu \sim N(c_i^T \mu, 1)$$

The joint density (for n samples) is

$$p(\mu, c, x) = p(\mu) \prod_{i=1}^n p(c_i) p(x_i | c_i, \mu)$$

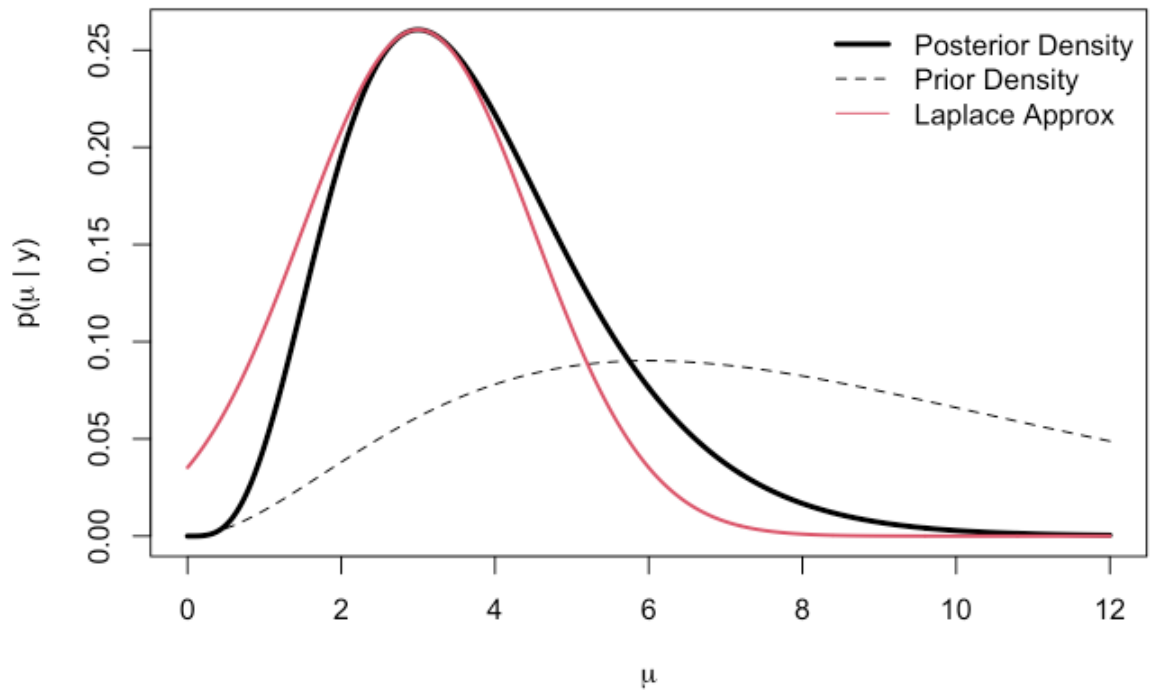
The latent variables here are $z = \{\mu, c\}$. The evidence is,

$$p(x) = \int p(\mu) \prod_{i=1}^n \sum_{c_i} p(c_i) p(x_i | c_i, \mu) d\mu$$

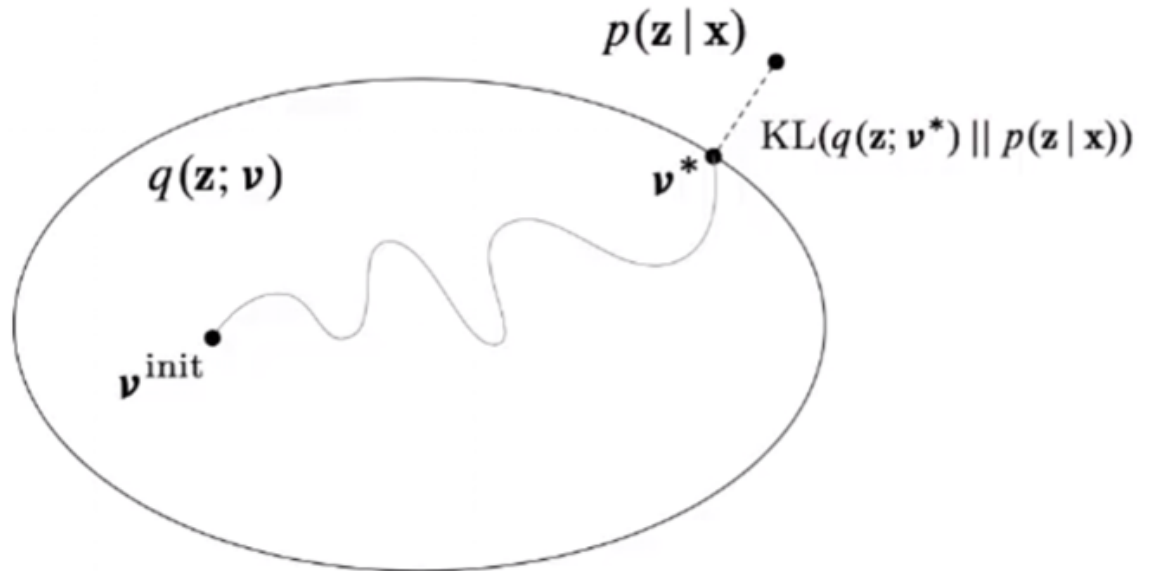
You can't factor this wrt μ_k (the μ_k appear in all n factors), so the complexity of evaluating this numerically is $O(K^n)$.

4. SO WHAT DO YOU DO? APPROXIMATE BAYESIAN INFERENCE

- (a) Laplace (second order taylor expand to get an estimate centered at the posterior mode, but hits only 1 mode, requires Hessian)



(b) Variational Inference (find the best approximation within a variational family)



(c) Expectation Propagation (Adams 15)

Minimize $KL(p||q)$

(d) Markov Chain Monte Carlo (define a Markov chain for the posterior) We will discuss this in detail

(e) Stochastic Gradient-Based Monte Carlo (Welling, Teh 11) (Mandt 17) (Appendix A) (1) omit Metropolis-Hastings correction step (2) noise from subsampling data

People have evaluated these approaches based on downstream performance metrics, but none have evaluated the extent to which they match the true posterior in practical datasets or architectures.

Here we introduce background required to understand the MCMC approach so that we can discuss HMC.

5. MCMC

To approximate the bayesian model average

$$p(y|x, D) = \int_w p(y|x, w)p(w|D)dw$$

we can sample $p(y|x, D) \approx \frac{1}{M} \sum_{i=1}^M p(y|x, w_i)$ where $w_i \sim p(w|D)$ are samples from the posterior.

How do we get samples from the posterior? We construct a Markov chain that generates approximate samples from the posterior (Metropolis and Ulam 49). Consider a sequence of samples of z . Let $q(z|z^t)$ be a proposal distribution for z that depends on the current state z^t so that z^1, z^2, \dots form a Markov chain. This distribution needs to be easy to sample from. Now we introduce a criterion that lets us judge a candidate sample from q , call it z^* , and accept or reject the sample.

The Metropolis algorithm exploits the fact that we can easily evaluate z^* under the true distribution up to a normalization constant $\hat{p}(z^*)$. So we can take ratios that cancel the normalization constant $R = \frac{\hat{p}(z^*)}{\hat{p}(z^t)}$. If we accept samples with probability $\min(1, R)$ (always keep the sample if it is more likely than z^t , otherwise keep with decaying probability) and the proposal distribution is symmetric, this Markov chain tends to generate samples from $p(z)$ as $t \rightarrow \infty$.

Put more simply: Suppose that our proposal distribution is Gaussian. We walk around the probability density, moving to higher density region whenever we sample a point that has higher probability, otherwise going to a lower probability region with a chance that is proportional to the ratio of lower probability/current probability. In this way I explore the probability density evenly if my proposal distribution is symmetric.

This argument requires symmetry, but admits a generalization in the Metropolis-Hastings algorithm (Hastings 70) to the case where the proposal distribution is not symmetric. Consider k possible transitions,

$$A_k(z^*, z^t) = \min \left(1, \frac{\hat{p}(z^*)q_k(z^t|z^*)}{\hat{p}(z^t)q_k(z^*|z^t)} \right)$$

It is possible to prove that $p(z)$ is an invariant distribution of this Markov chain by showing that it satisfies the sufficient condition of detailed balance (Bishop pg 541).

But notice that subsequent samples are highly correlated! In fact the chain has random walk characteristics and only explores a distance from the original z^0 proportional to \sqrt{t} .

6. HAMILTONIAN MONTE CARLO

(Neal 11) Introduces a variant on the Metropolis algorithm with dramatically reduced correlation between samples.

Flip our posterior space (or its log) $-\log p(\theta|x)$. If we treat this flipped posterior with an analogy to energy and explore based on Hamiltonian dynamics in that space, then we will move towards low energy regions. And these are exactly the places where $p(\theta|x)$ is highest.

Now we imagine an ensemble of millions of particles traversing this energy space based on the hamiltonian dynamics. If we sample the energies of this ensemble, we get a sample with density proportional to the posterior!

In practice, we introduce a momentum variable ρ and draw from the joint density,

$$p(\rho, \theta) = p(\rho|\theta)p(\theta)$$

where $\rho \sim \text{MultiNormal}(0, \Sigma)$. The joint density is a Hamiltonian,

$$H(\rho, \theta) = -\log p(\rho, \theta) = -\log p(\rho|\theta) - \log p(\theta) = T(\rho|\theta) + V(\theta)$$

At each iteration draw a value for momentum ρ and evolve the joint (θ, ρ) by Hamilton's equations,

$$\begin{aligned}\frac{d\theta}{dt} &= \frac{\partial H}{\partial \rho} = \frac{\partial T}{\partial \rho} \\ \frac{d\rho}{dt} &= -\frac{\partial H}{\partial \theta} = -\frac{\partial T}{\partial \theta} - \frac{\partial V}{\partial \theta} = -\frac{\partial V}{\partial \theta}\end{aligned}$$

The leapfrog integrator gives stable solutions to these Hamiltonian systems of equations. Since the numerical integration is not perfect (Leimkuhler 04 Simulating Hamiltonian Dynamics) there is a metropolis acceptance step.

Algorithm 1 Hamiltonian Monte Carlo

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

```

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

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

7. BAYESIAN NEURAL NETWORK

A neural network can be viewed as a probabilistic model $p(y|x, w)$ where $x \in \mathbb{R}^p$. For classification, $p(y|x, w)$ is categorical and the loss becomes categorical cross entropy. For regression, $p(y|x, w)$ is Gaussian and the loss becomes MSE.

We can make a point estimate of this model by choosing weights with Maximum Likelihood Estimation,

$$w^{MLE} = \arg \max_w \log p(D|w) = \arg \max_w \sum_i \log p(y_i|x_i, w)$$

and if $\log p(D|w)$ is differentiable in w we can find this w^{MLE} by gradient descent with backprop. A prior can be placed on weights and they can be estimated by,

$$w^{MAP} = \arg \max_w \log p(D|w) + \log p(w)$$

giving us L2 regularization if prior is Gaussian, L1 regularization if prior is Laplacian. But this is still point estimation.

For fully Bayesian inference we need to calculate the posterior of weights $p(w|D)$. Then,

$$p(\hat{y}|\hat{x}) = \mathbb{E}_{p(w|D)}[p(\hat{y}|\hat{x}, w)]$$

Which is equivalent to taking an ensemble of uncountably infinite neural networks weighted by their likelihoods (posterior predictive distribution/bayes ensemble).

Or equivalently we approximate $p(w|D) \propto \exp(-U(w)/T)$ where $U(w)$ is the posterior energy function,

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

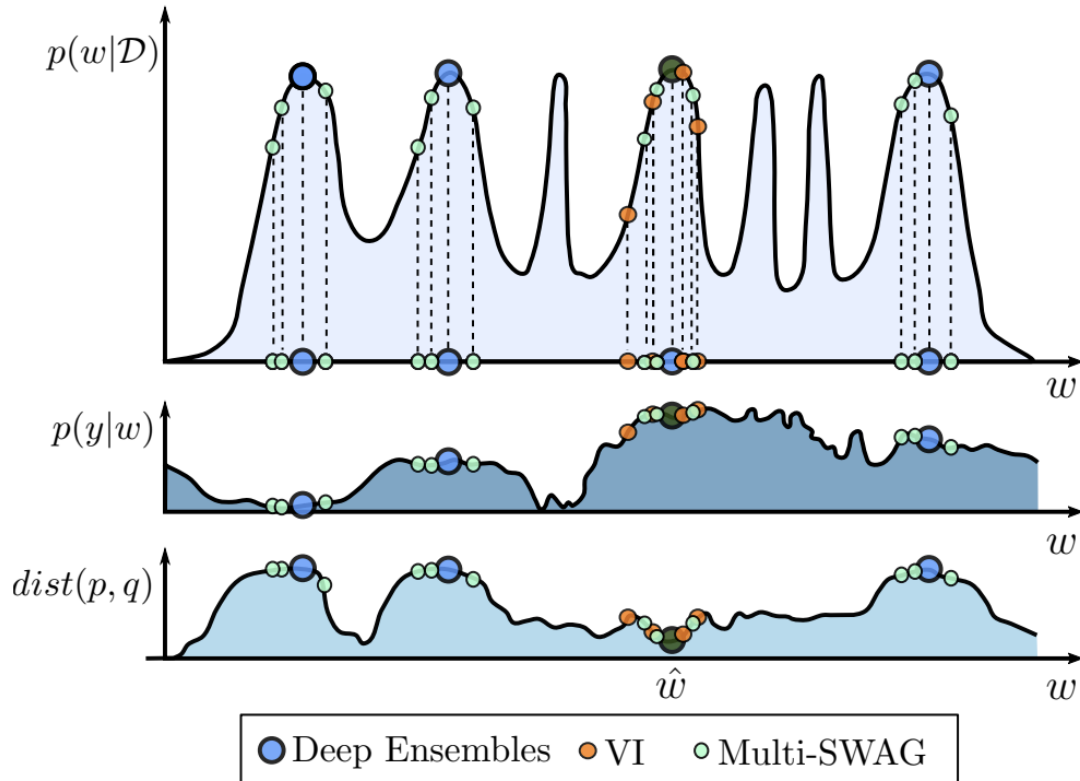
T is the temperature and $p(w)$ is a proper prior density function. We expect predictions made by this ensemble model to improve over predictions made at a single well-chosen parameter? Why?

- (a) (Komaki 96) On Asymptotic Properties of Predictive Distributions For many models it is shown that posterior predictive distribution dominates point-wise estimators based on likelihood, and (Fushiki 05) this is even true when the model is misspecified
- (b) (Geisser 93) Averaged predictions are more robust in practice for classical statistical models
- (c) (Lakshminarayanan 17) Ensembled models have better performance

But computing this expectation is intractable because of the requirement to sample $p(w|D)$ as I showed before.

8. MODERN BAYESIAN NNs ARE NOT QUITE RIGHT

To evaluate this expectation we must integrate over a multi-million dim multi-modal posterior. Most approaches are not quite right:



- (a) VI makes unimodal Gaussian approximations

- (b) Deep Ensembles (if we accept their Bayesian interpretation) only represent posterior modes
- (c) Stochastic MCMC makes biased estimates from omitting MH correction and subsampling noise
- (d) (Wenzel 20) "cold posteriors" improve performance
- (e) posterior quality is evaluated based on performance, but performance does not imply that the model is approximating bayesian model average well

Instead here we use multi-chain full-batch Hamiltonian Monte Carlo (HMC) as explained above (Neal 11).

9. PREVIOUS ATTEMPTS AT FULL-BATCH HMC IN BNNs

- (a) Cobb, Jalain 20 Scaling Hamiltonian Monte Carlo Inference for Bayesian Neural Networks with Symmetric Splitting

Symmetric integration scheme for split HMC that does not use stochastic gradients.

Shortcomings: (a) short trajectory lengths (leapfrog < 100), (b) small datasets

- (b) Wenzel 20 How Good is the Bayes Posterior in Deep Neural Networks Really?

This paper received a lot of attention for casting doubts on the quality of Bayes Posteriors. Recall that we optimize an energy, $p(\theta|D) \propto \exp(-U(\theta)/T)$. We call the Bayes Posterior the case where $T = 1$. This paper shows that for (a) ResNet-20 on CIFAR-10 and (b) CNN-LSTM on IMDB, accuracy and cross-entropy are improved for $T < 1/10$. Why is this a problem? T *sharpens the posterior*, which is like overcounting the data by a factor of $1/T$ and rescaling the prior as $p(\theta)^{1/T}$. This gives too strong evidence to individual models. For $T = 0$ all mass is concentrated on the MAP point estimates.

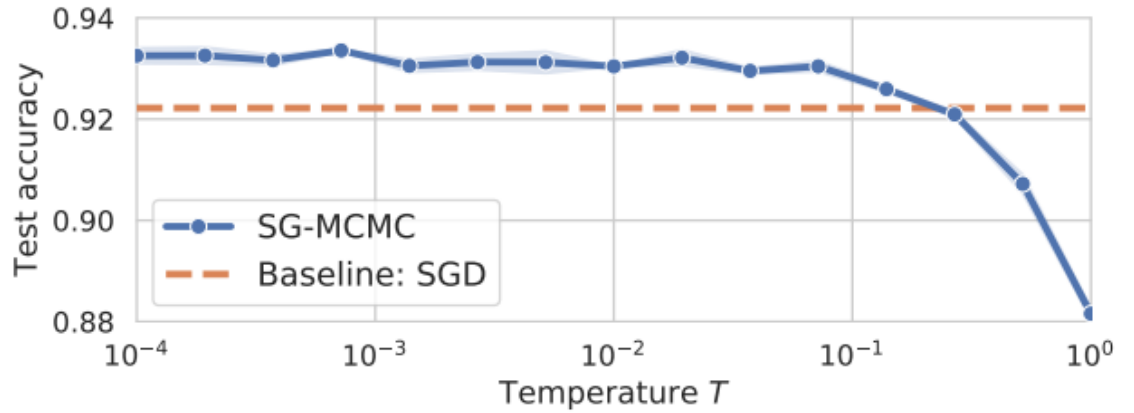


Figure 1. The “**cold posterior**” effect: for a ResNet-20 on CIFAR-10 we can improve the generalization performance significantly by cooling the posterior with a temperature $T \ll 1$, deviating from the Bayes posterior $p(\theta|\mathcal{D}) \propto \exp(-U(\theta)/T)$ at $T = 1$.

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).

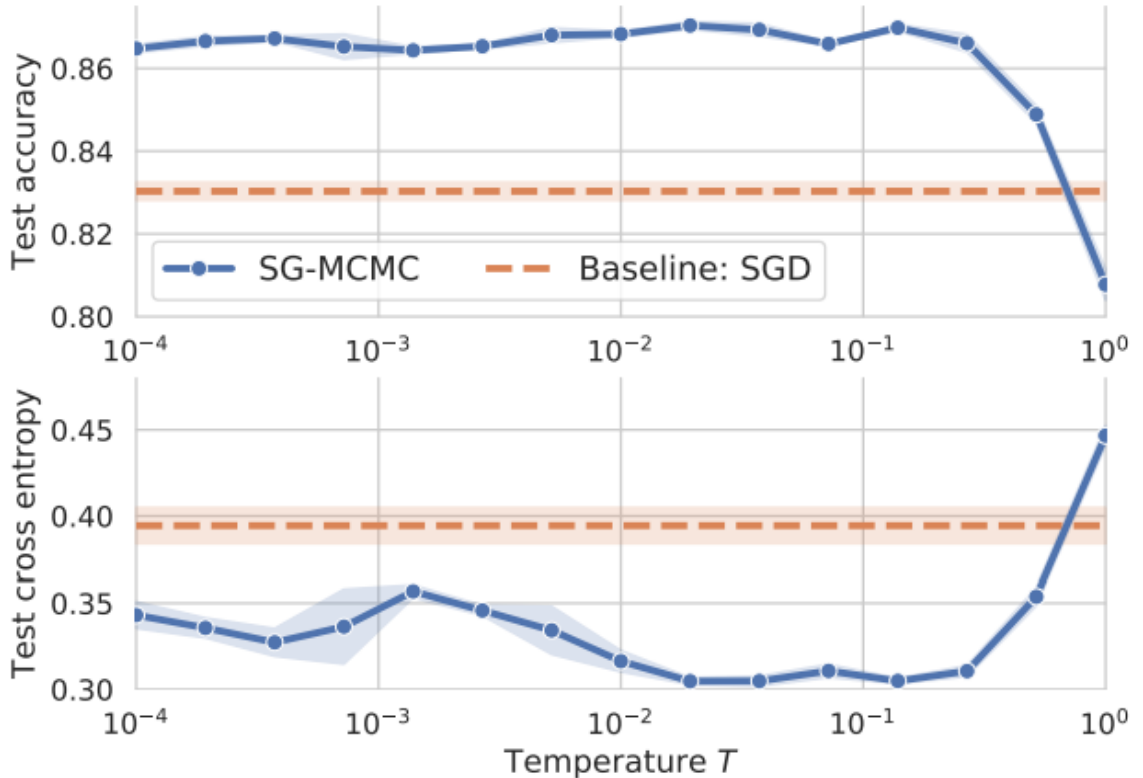


Figure 3. Predictive performance on the IMDB sentiment task test set for a tempered CNN-LSTM Bayes posterior. Error bars are \pm one standard error over three runs. See Appendix A.4.

10. EXPERIMENTAL SETUP (HOW THIS TIME IS DIFFERENT)

(a) Architecture:

ResNet-20-FRN and CNN-LSTM where batch norm is replaced by filter response normalization (FRN) because batch norm introduces dependencies between training examples.

Full batch Hamiltonian Monte Carlo

Parallelize over 512 TPUv3

Use single-program multiple-data configuration where a dataset is sharded evenly over devices and identical HMC is run on each device.

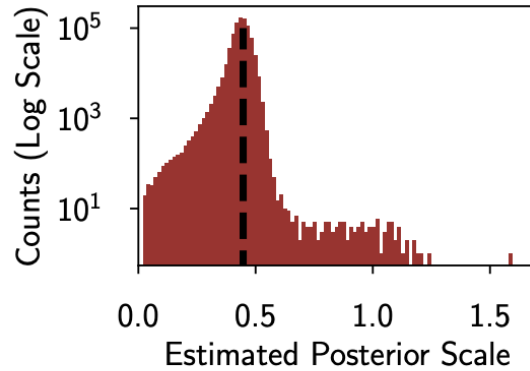
Devices synchronize the Markov chain state, full-batch gradients for leapfrog integration are computed using cross-device collection.

Swish activations improve acceptance rates of HMC proposals

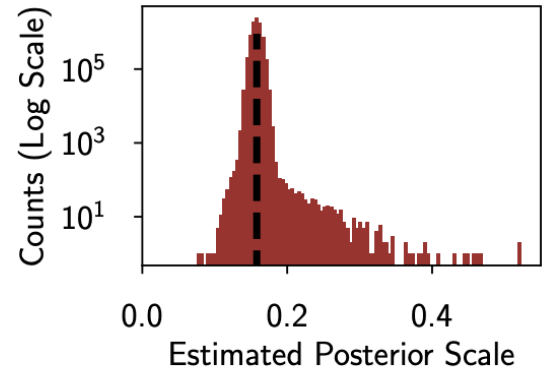
(b) Trajectory Length: This parameter determines the length of the dynamics simulation on each HMC iteration. Controls *the correlation of subsequent samples*. But it is also expensive to lengthen the trajectory because we evaluate the gradient a number of times proportional to trajectory length/step size. Propose to use,

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

where α_{prior} is the std deviation of the prior over parameters. If we were sampling a spherical gaussian prior, this would give us exact samples. Hand-wave over an argument that the prior determines the scale of the posterior and show empirically (line is prior) that this length should be good enough,

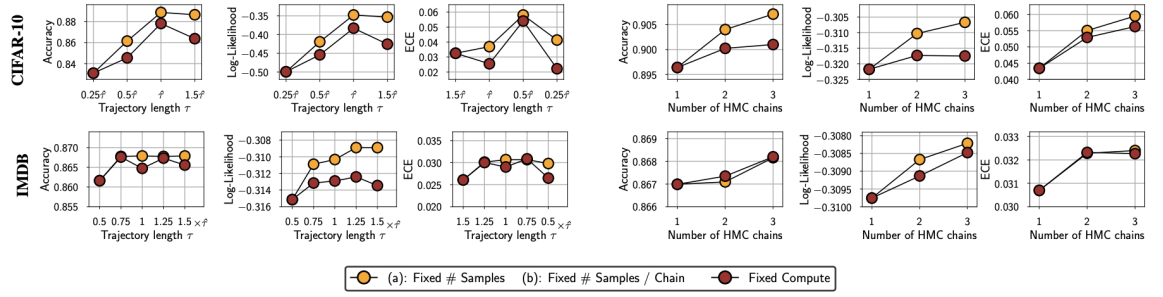


(a) ResNet-20-FRN



(b) CNN-LSTM

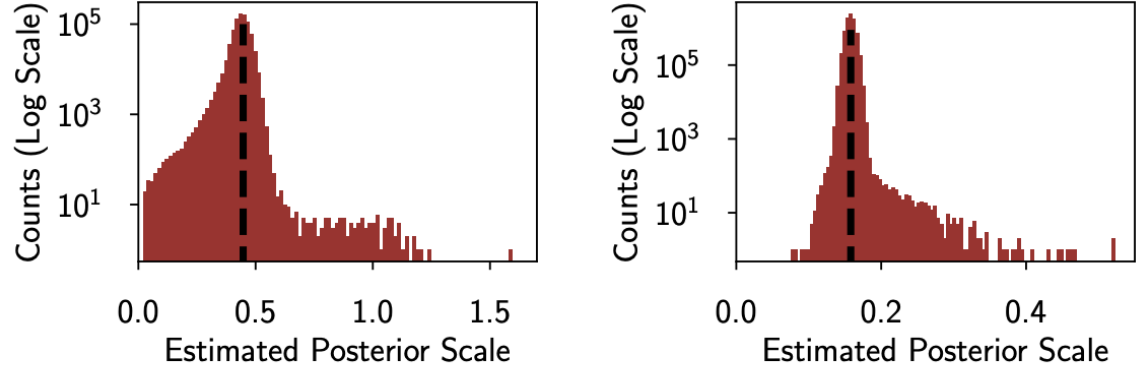
- (c) Step Size As we increase HMC step size from 1×10^{-5} , 5×10^{-5} , 1×10^{-4} , 5×10^{-4} the chains accept probability drops .722, .463, .222, .125 and B<A log-likelihoods degrade
- (d) Number of HMC chains Running multiple independent chains increases coverage of posterior.



11. EVALUATING POSTERIOR QUALITY

How do we know if the HMC sampler has converged? Consider weight space and function space mixing. Function space is just the space induced by NN architecture $f(x, w)$ and a distribution over weights w .

- (a) \hat{R} diagnostics (Gelman 92) If we have multiple chains and one chain is stuck, the diversity of samples from the other chains will be much greater.



(a) ResNet-20-FRN

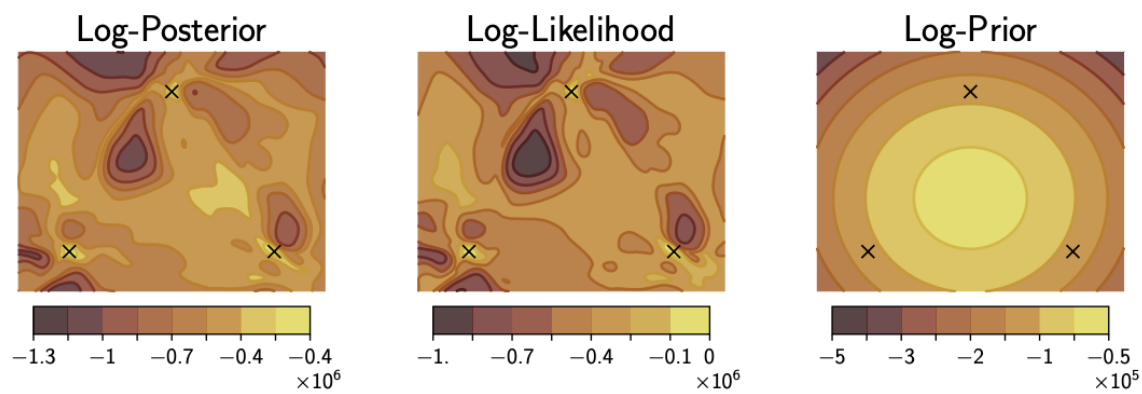
(b) CNN-LSTM

This is pretty promising, one chain seems to rarely get stuck

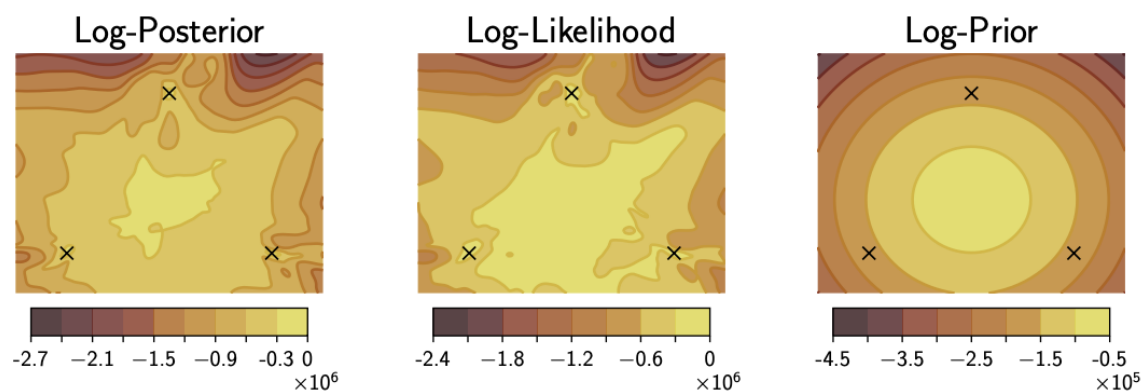
- (b) Posterior Density Visualization (Garipov 18) Study 2D subspaces of the parameter space,

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

the unique affine subspace of the parameter space that includes vectors w_1, w_2, w_3 . Here we show HMC at iterations 1, 51, and 101 after burn-in. Notice that the samples are diverse places in posterior,



(a) Same chain

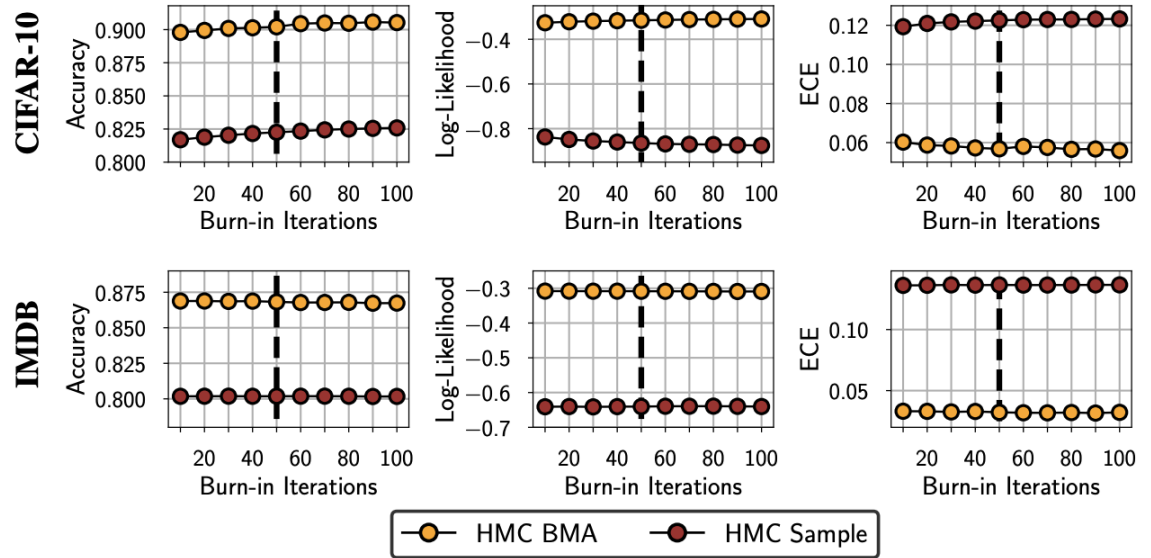


(b) Independent chains

Other approximate inference would miss some of these posterior modes!

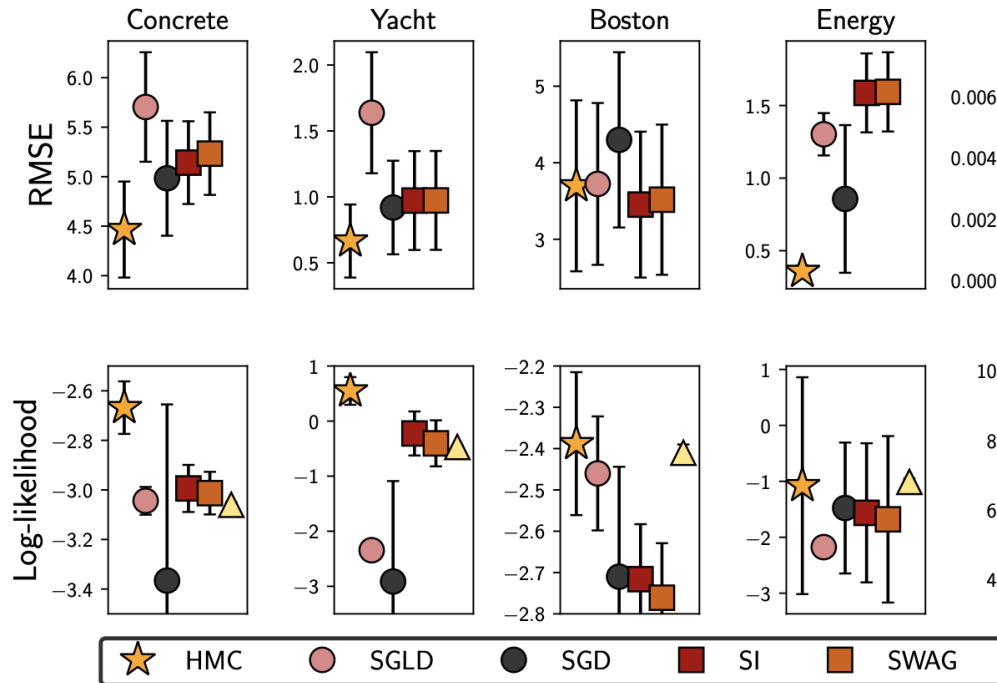
Also chains don't mix perfectly in weight space, notice the difference in smoothness and symmetry

- (c) Convergence of chains After sufficiently long burn-in, the performance of BMA and samples should be stationary. 10 HMC iterations should be sufficient on IMDB, 50 on CIFAR-10,

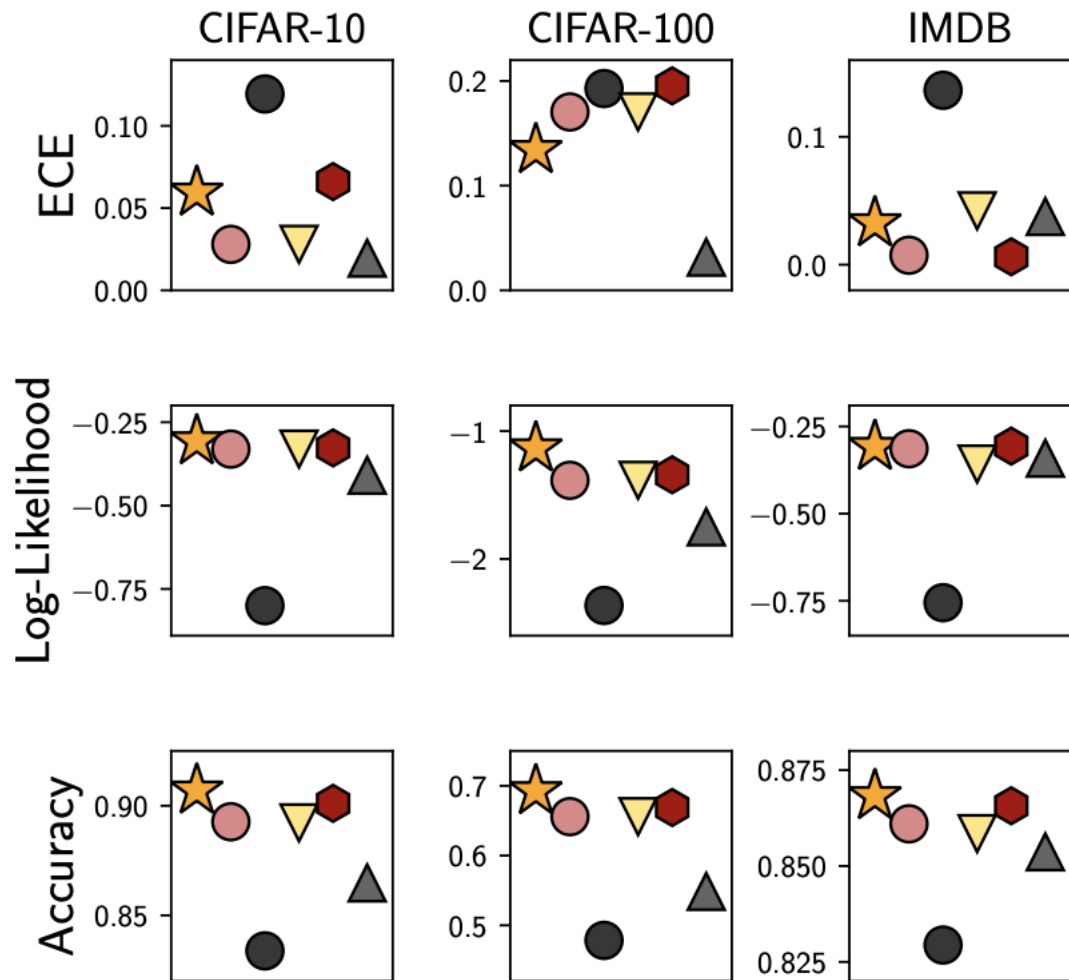


12. EVALUATING BNNs

- (a) Regression Tasks (UCI Regression Datasets) Fully connected NN with single 50 node hidden layer and 2 outputs. Single HMC chain with 10 burn-in and 90 sampling iterations (See Appendix of paper for detail)



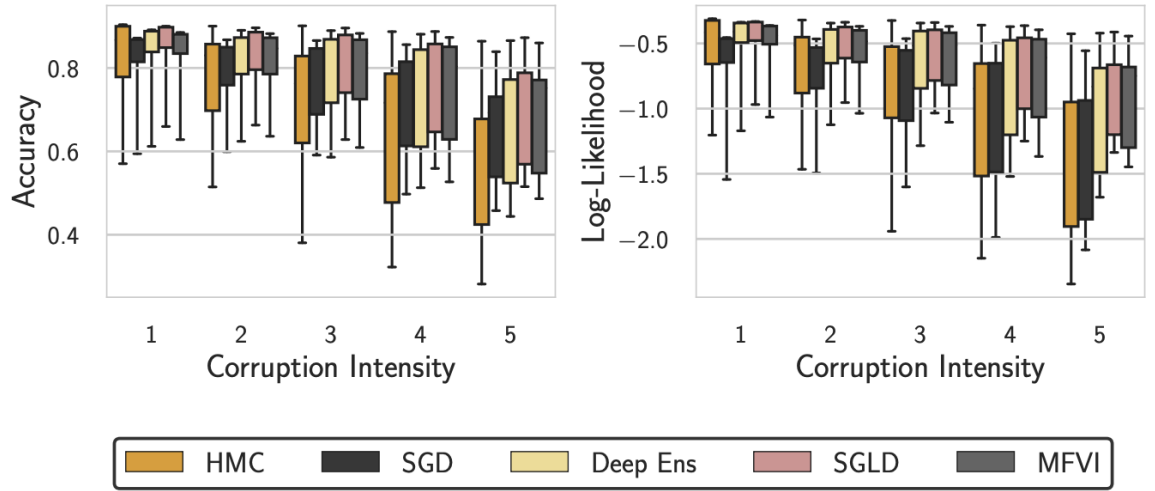
- (b) CIFAR-10 and CIFAR-100 3 HMC chains, 70,248 leapfrog steps per sample
Discard first 50 samples then draw 240 samples/chain



(c) OOD Detection

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

(d) Distribution Shift



13. DO WE NEED COLD POSTERiors?

$$p_T(w|D) \propto \left(p(D|w)p(w) \right)^{1/T}$$

Here they argue that cold posterior effect is a consequence of data augmentation.

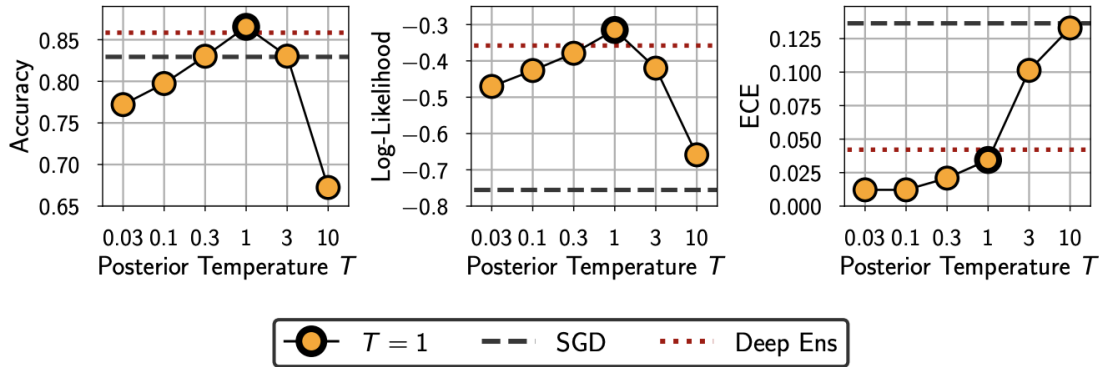


Figure 8. Effect of posterior temperature. The effect of posterior temperature T on the log-likelihood, accuracy and expected calibration error using the CNN-LSTM model on the IMDB dataset. For both the log-likelihood and accuracy $T = 1$ provides optimal performance, while for the ECE the colder posteriors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

Using code from Wenzel 20, turn off data augmentation and there is no cold posterior effect! Augmentation causes the posterior to be artificially narrow since data is repeatedly seen.

14. HOW IMPORTANT ARE PRIORS?

Wenzel 20 argues that $N(0, \alpha^2 I)$ priors are inadequate and others develop complicated priors based on this (Tran 20 GP). But previously we show good performance with vague Gaussian priors. Empirically, high-variance Gaussian priors have strong performance, but can be outperformed

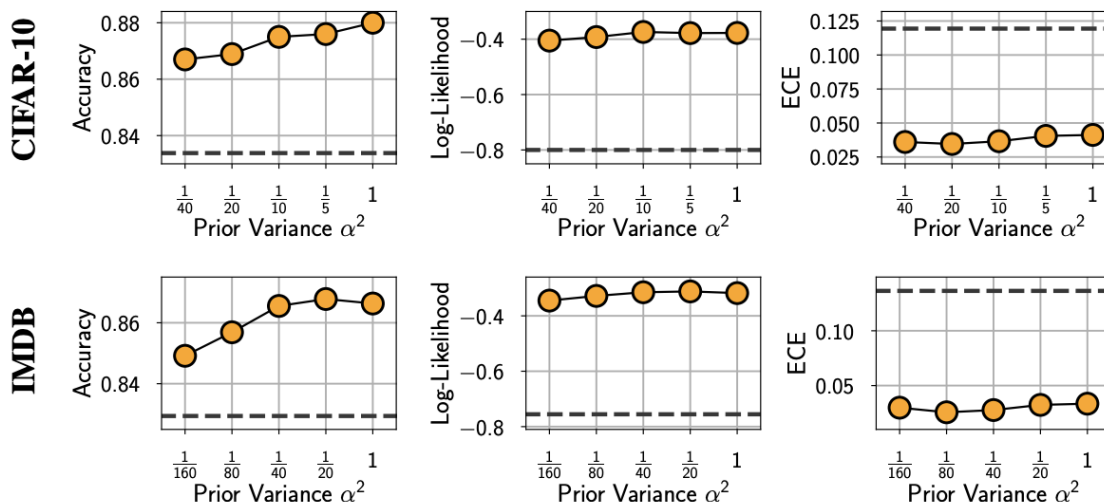


Figure 9. Effect of prior variance. The effect of prior variance on BNN performance. In each panel, the dashed line shows the performance of the SGD model from Section 6. While low prior variance may lead to over-regularization and hurt performance, all the considered prior scales lead to better results than the performance of an SGD-trained neural network of the same architecture.

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

Table 2. Non-Gaussian priors. BMA accuracy, ECE, and log-likelihood under different prior families using CNN-LSTM on IMDB. We produce 80 samples from a single HMC chain for each of the priors. The heavier-tailed logistic prior provides slightly better performance compared to the Gaussian and mixture of Gaussians (MoG) priors.

But why so robust to large prior scale? Large prior variance implies that the "true" classifier should make high-confidence predictions. The model can achieve any desired training accuracy so this is not overruled. Continuing to increase prior variance has no effect on already-saturated classifier probabilities. So every member of the BMA is overconfident. But the ensemble *overall* is calibrated

15. SO HOW GOOD ARE OUR APPROXIMATIONS EXACTLY?

Clearly we can't take this HMC approach in practice. We consider metrics: agreement and total variation. Agreement is fraction of test points for which top-1 predictions are the same,

$$\frac{1}{n} \sum_{i=1}^n I[\arg \max_j \hat{p}(y = j|x_i) = \arg \max_j p(y = j|x_i)]$$

Total variation is the total variation distance between predictive distributions averaged over test points,

$$\frac{1}{n} \sum_{i=1}^n \frac{1}{2} \sum_j \left| \hat{p}(y = j|x_i) - p(y = j|x_i) \right|$$

METRIC	HMC (REFERENCE)	SGD	DEEP ENS	MFVI	SGMCMC			
					SGLD	SGHMC	SGHMC CLR	SGHMC CLR-PREC
CIFAR-10								
ACCURACY	89.64 ±0.25	83.44 ±1.14	88.49 ±0.10	86.45 ±0.27	89.32 ±0.23	89.38 ±0.32	89.63 ±0.37	87.46 ±0.21
AGREEMENT	94.01 ±0.25	85.48 ±1.00	91.52 ±0.06	88.75 ±0.24	91.54 ±0.15	91.98 ±0.35	92.67 ±0.52	90.96 ±0.24
TOTAL VAR	0.074 ±0.003	0.190 ±0.005	0.115 ±0.000	0.136 ±0.000	0.110 ±0.001	0.109 ±0.001	0.099 ±0.006	0.111 ±0.002
CIFAR-10-C								
ACCURACY	70.91 ±0.93	71.04 ±1.80	76.99 ±0.39	75.40 ±0.34	78.80 ±0.17	78.20 ±0.25	76.43 ±0.39	73.42 ±0.39
AGREEMENT	86.00 ±0.44	72.01 ±0.82	79.29 ±0.18	75.47 ±0.27	77.99 ±0.22	78.98 ±0.22	80.93 ±0.73	79.65 ±0.35
TOTAL VAR	0.133 ±0.004	0.334 ±0.007	0.220 ±0.003	0.245 ±0.002	0.214 ±0.002	0.203 ±0.002	0.194 ±0.010	0.205 ±0.005

Table 3. Evaluation of cheaper alternatives to HMC. Agreement and total variation between predictive distributions of HMC and approximate inference methods: deep ensembles, mean field variational inference (MFVI), and stochastic gradient Monte Carlo (SGMCMC) variations. For all methods we use ResNet-20-FRN trained on CIFAR-10 and evaluate predictions on the CIFAR-10 and CIFAR-10-C test sets. For CIFAR-10-C we report the average results across all corruptions and corruption intensities. We additionally report the results for HMC for reference: we compute the agreement and total variation between one of the chains and the ensemble of the other two chains. For each method we report the mean and standard deviation of the results over three independent runs. MFVI provides the worst approximation of the predictive distribution. Deep ensembles despite often being considered non-Bayesian, significantly outperform MFVI. SG-MCMC methods provide the best results with SGHMC-CLR showing the best overall performance.

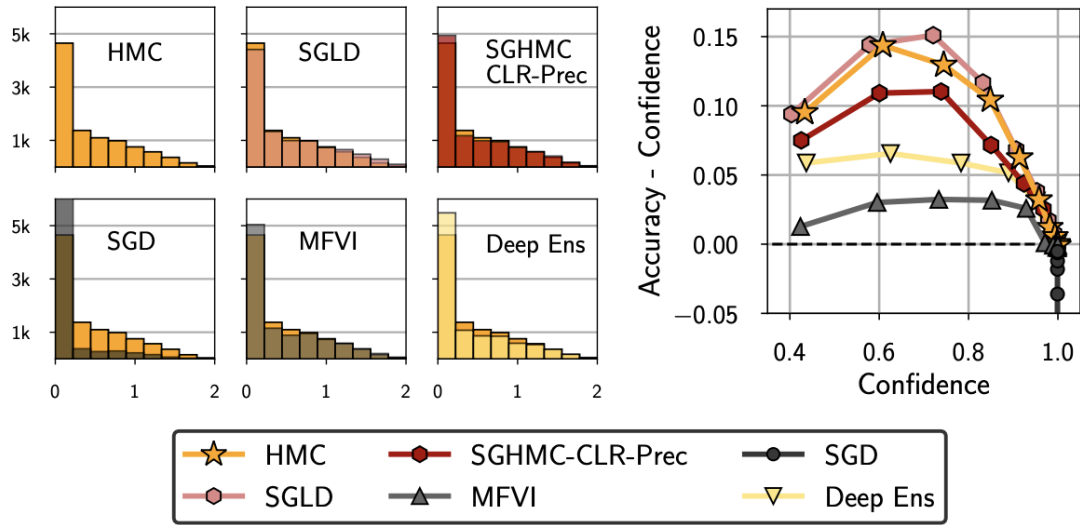


Figure 10. Distribution of predictive entropies (**left**) and calibration curve (**right**) of posterior predictive distributions for HMC, SGD, deep ensembles, MFVI, SGLD and SGHMC-CLR-Prec for ResNet20-FRN on CIFAR-10. On the left, for all methods, except HMC we plot a pair of histograms: for HMC and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

16. APPENDIX A: A HISTORY OF BAYESIAN NEURAL NETWORKS

Reference 16.1. *MacKay 92: A practical bayesian framework for backpropagation networks*

Neural network parameters are optimized online and settings are evaluated by test error or cross validation. Attempts to find "objective criteria" for setting parameters and comparing alternative solutions that depend only on the training set. Early example of doing bayesian things like putting priors over weights, selecting model architecture, choosing decay rate, comparing regularizers, penalizing complicated models.

Compares different ex. parameter settings by comparing evidence,

$$p(D|A, R) = \int p(D|\alpha, \beta, A, R)p(\alpha, \beta)d\alpha d\beta$$

Reference 16.2. *MacKay 95 Probable networks and plausible predictions? A review of practical Bayesian methods for supervised networks*

Why a probabilistic approach to modeling? (1) make explicit all modelling assumptions then probability theory gives a unique answer (2) satisfies the likelihood principle (Berger 85) (3) handles uncertainty (4) occam's razor through Bayesian Model selection

Reference 16.3. *Neal 96 (PhD Thesis w/ Hinton) Bayesian Learning for Neural Networks*

Addresses the question of prior distributions over network weights. It isn't clear what priors over weights should look like so Neal chooses based on limiting cases (network size $\rightarrow \infty$) where some priors converge to Gaussian process, others to non-Gaussian stable processes.

Introduces MCMC for posterior estimation.

Reference 16.4. *Hinton and Van Camp 93 Keeping the neural networks simple by minimizing the description length of the weights*

Variational approximation to posterior distribution on the weights.

$$\begin{aligned}\theta^* &= \arg \min_{\theta} KL[q(w|\theta) || p(w|D)] \\ &= \arg \min_{\theta} \int q(w|\theta) \log \frac{q(w|\theta)}{p(w)p(D|w)} dw \\ &= \arg \min_{\theta} KL[q(w|\theta) || p(w)] - \mathbb{E}_{q(w|\theta)} [\log p(D|w)]\end{aligned}$$

Reference 16.5. *Graves 99 Sequential MCMC for Bayesian model selection*

Negative results for online parameter optimization

Reference 16.6. *Graves 11 Practical Variational Inference for Neural Networks*

Reference 16.7. *Welling 11 Bayesian Learning via Stochastic Gradient Langevin Dynamics*

Robbins-Monro (1951) style algorithms (mini-batch algorithms) have been very successful, but were not yet adopted for Bayesian methods. Solution is Robbins-Monro + Langevin dynamics (inject noise into mini-batch parameter updates st trajectory of parameters converges to full posterior rather than mode)

To generate parameter samples $\theta \approx p(\theta|D)$ we consider Langevin dynamics over $\theta \in \mathbb{R}^d$ and momenta $m \in \mathbb{R}^d$ defined by the Langevin stochastic differential equation,

$$\begin{aligned}d\theta &= M^{-1}m dt \\ dm &= -\nabla_{\theta} U(\theta) dt - \gamma m dt + \sqrt{2\gamma T} M^{1/2} dW\end{aligned}$$

Sampling the Langevin SDE produces the Bayes posterior for $T = 1$. The stochastic version further approximates $\nabla_{\theta} U(\theta)$ with a minibatch gradient. The SDE is defined in continuous time, discretize time and solve the dynamics numerically by first-order symplectic Euler discretization. There is a whole body of work developing numerical solvers for Hamiltonian dynamics (Chen 15, Shang 15, Heber 19, Heek 20).

This method leaves two sources of error (a) minibatch noise from the gradient estimate, (b) discretization error

Reference 16.8. *Blundell 15 Weight Uncertainty in Neural Networks aka Bayes by Backprop*

Recall that making predictions in a bayesian neural net involves calculating an expectation,

$$p\left(\frac{\hat{y}}{\hat{x}}\right) = \mathbb{E}_{p(w|D)} [p(\hat{y}|\hat{x}, w)]$$

Can we differentiate this? Only under some conditions.

Proposition 16.9. *Let ϵ a rv with pdf $q(\epsilon)$ and let $w = t(\theta, \epsilon)$ where t deterministic. Suppose the marginal pdf of w , $q(w|\theta)$ is st $q(\epsilon)d\epsilon = q(w|\theta)dw$. Then for a function with derivatives in w ,*

$$\frac{\partial}{\partial \theta} \mathbb{E}_{q(w|\theta)} [f(w, \theta)] = \mathbb{E}_{q(\epsilon)} \left[\frac{\partial f(w, \theta)}{\partial w} \frac{\partial w}{\partial \theta} + \frac{\partial f(w, \theta)}{\partial \theta} \right]$$

This lets you move the derivative outside of the expectation, generalizing the reparameterization trick used for Gaussians (ex in VAE). And we get an approximation to the loss,

$$F(D, \theta) \approx \sum_{i=1}^n \log q(w^i | \theta) - \log p(w^i) - \log p(D | w^i)$$

by monte carlo sampling from the variational posterior q