# What Are Bayesian Neural Network Posteriors Really Like?

Pavel Izmailov Sharad Vikram Matthew D. Hoffman Andrew Gordon Wilson

NYU Google AI

# Quick intro into Bayesian neural networks

Bayes Rule: $$p(w|\text{Data}) = \frac{p(\text{Data}|w)p(w)}{\int p(\text{Data}|w')p(w')dw'} \propto p(\text{Data}|w)p(w)$$

Bayesian Model Average: $$p_{BMA}(y|x) = \int p(y|w,x)p(w|\text{Data})dw \approx \sum_i p(y|w_i,x)$$

$$w_i \sim p(w|\text{Data})$$

# Quick intro into Bayesian neural networks

Intractable

Bayes Rule: $$p(w|\text{Data}) = \frac{p(\text{Data}|w)p(w)}{\int p(\text{Data}|w')p(w')dw'} \propto p(\text{Data}|w)p(w)$$

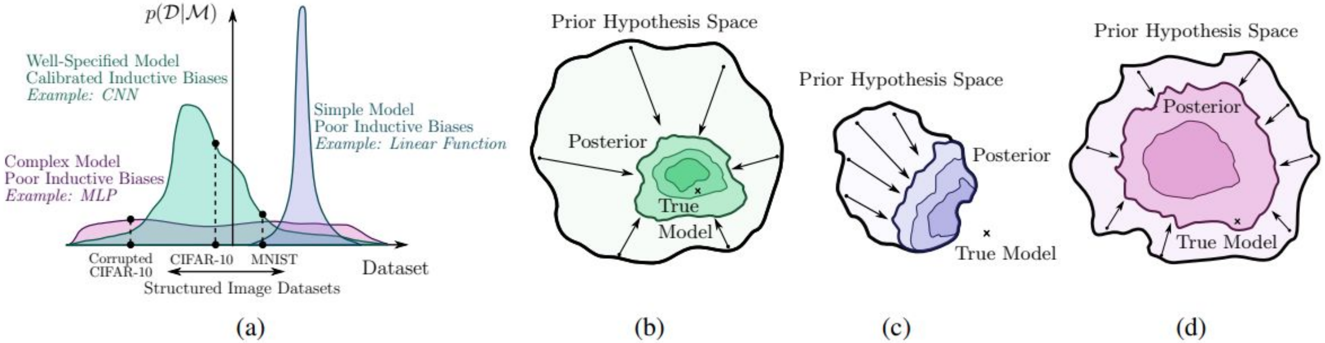
Bayesian Model Average: $$p_{BMA}(y|x) = \int p(y|w,x)p(w|\text{Data})dw \approx \sum_i p(y|w_i,x)$$

$$w_i \sim p(w|\text{Data})$$

# Bayesian deep learning literature overview

**Bayesian Deep Learning and a Probabilistic Perspective of Generalization**

**Andrew Gordon Wilson**  **Pavel Izmailov**
New York University



# Bayesian deep learning literature overview

## 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**

During the past five years the Bayesian deep learning community has developed increasingly accurate and efficient approximate inference procedures that allow for Bayesian inference in deep neural networks. However, despite this algorithmic progress and the promise of improved uncertainty quantification and sample efficiency there are—as of early 2020—no publicized deployments of Bayesian neural networks in industrial practice.

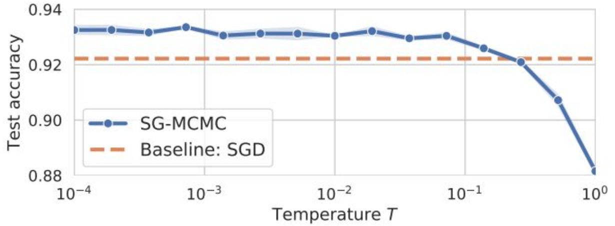



*Figure 1.* The "**cold posterior**" effect: for a ResNet-20 on CIFAR-10 we can improve the generalization performance significantly by cooling the posterior with a temperature $T \ll 1$, deviating from the Bayes posterior $p(\boldsymbol{\theta}|\mathcal{D}) \propto \exp(-U(\boldsymbol{\theta})/T)$ at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

# Bayesian deep learning literature overview

A STATISTICAL THEORY OF COLD POSTERIORS IN DEEP NEURAL NETWORKS

**Laurence Aitchison**
Department of Computer Science,
University of Bristol,
Bristol, UK, F94W 9Q
laurence.aitchison@bristol.ac.uk

All You Need is a Good Functional Prior for Bayesian Deep Learning

**Ba-Hien Tran** BA-HIEN.TRAN@EURECOM.FR
**Simone Rossi** SIMONE.ROSSI@EURECOM.FR
**Dimitrios Milios** DIMITRIOS.MILIOS@EURECOM.FR
**Maurizio Filippone** MAURIZIO.FILIPPONE@EURECOM.FR
*Data Science Department*
*EURECOM*
*Sophia Antipolis, FR*

BAYESIAN NEURAL NETWORK PRIORS REVISITED

**Vincent Fortuin***
ETH Zürich
fortuin@inf.ethz.ch

**Adrià Garriga-Alonso***
University of Cambridge
ag919@cam.ac.uk

**Florian Wenzel**
Humboldt University of Berlin

**Gunnar Rätsch**
ETH Zürich

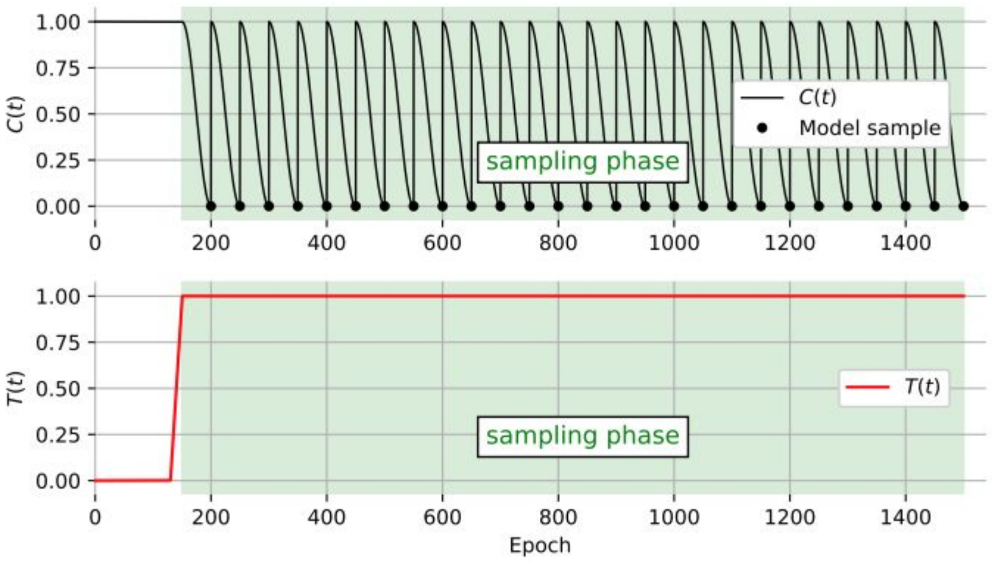
**Richard E. Turner**
University of Cambridge

**Mark van der Wilk**†
Imperial College London

**Laurence Aitchison**†
University of Bristol

# How do we know what is real?

We *assume* the results in these papers apply to *true BNNs*
But we are using simple and cheap approximate inference methods to show them

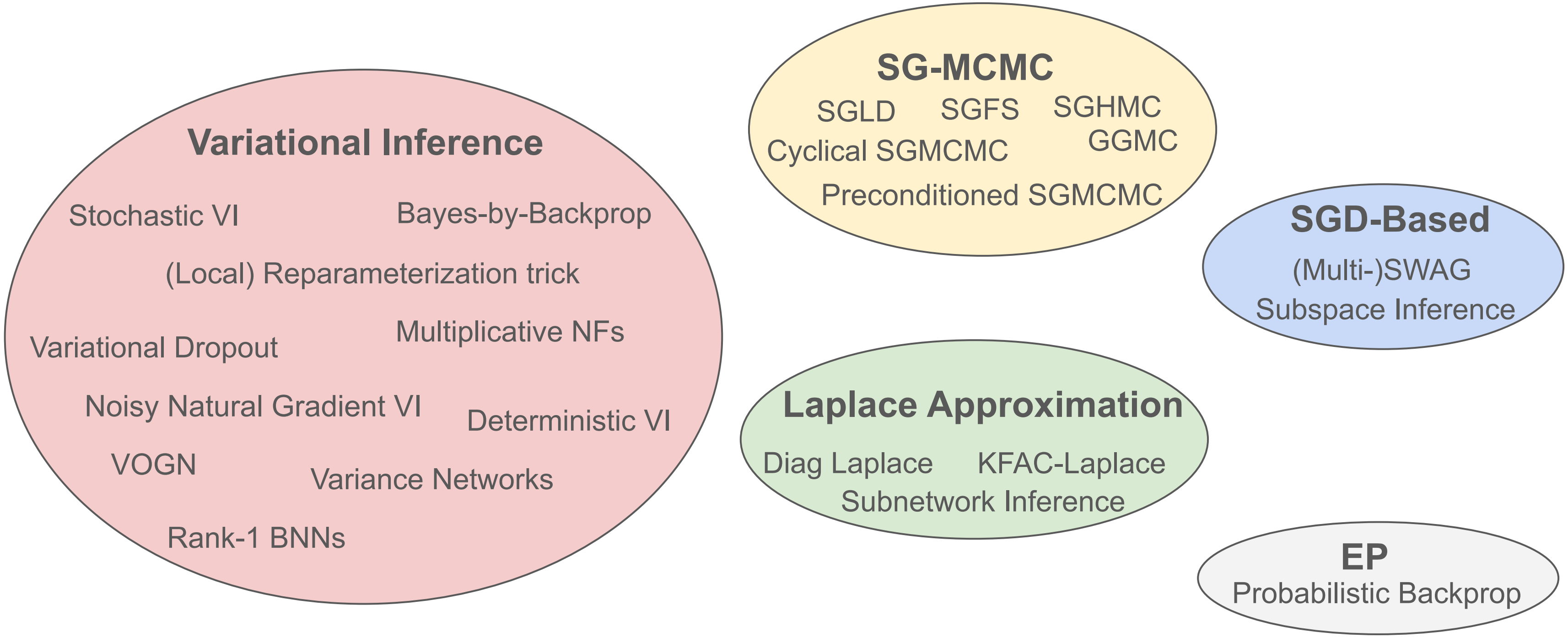



**Example: Cold Posteriors**

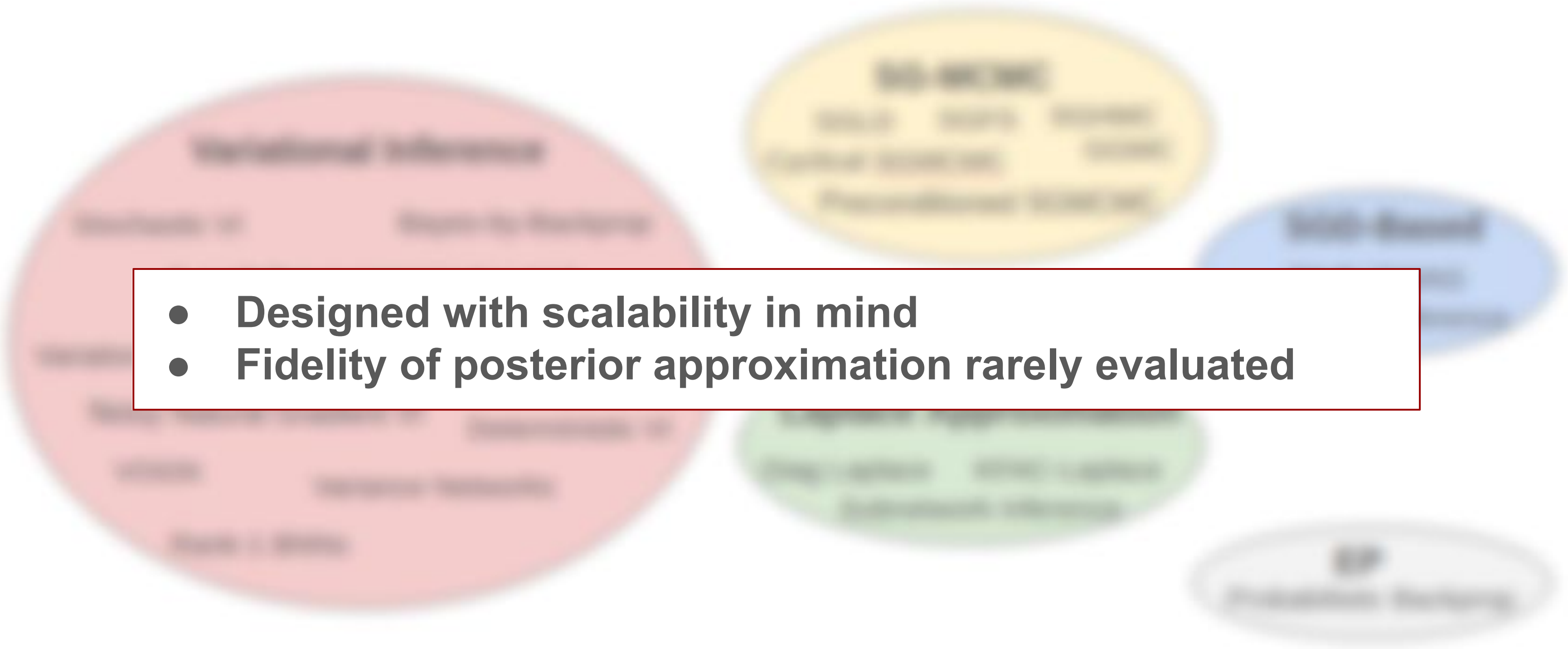
- SGHMC variation
- Only one MCMC chain
- 1500 epochs total
- No MH correction
- Minibatch noise

Do they really sample from the posterior?

# What tools do we have?



# What tools do we have?



- **Designed with scalability in mind**
- **Fidelity of posterior approximation rarely evaluated**

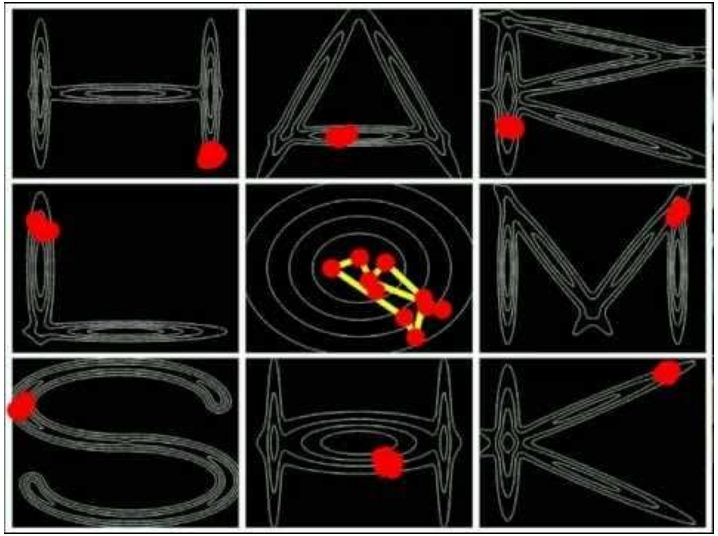
# What are we trying to achieve?

- Approximate inference method as exact as possible
- Ignore scalability and practicality
- Use it as a tool to *understand* Bayesian deep learning

# Hamiltonian Monte Carlo

+ Asymptotically exact
+ Well-studied and well-understood

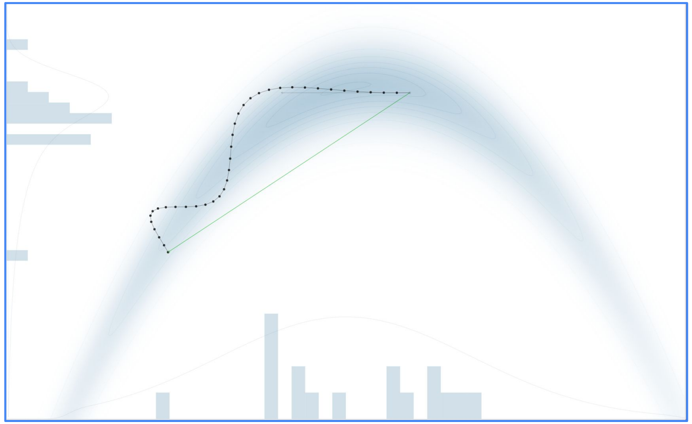
- Requires exact gradients
- Generally expensive



# Hamiltonian Monte Carlo

Demo; another demo

- Simulating the dynamics of a particle sliding on the plot of the density function that we are trying to sample from



**Algorithm 1** Hamiltonian Monte Carlo

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

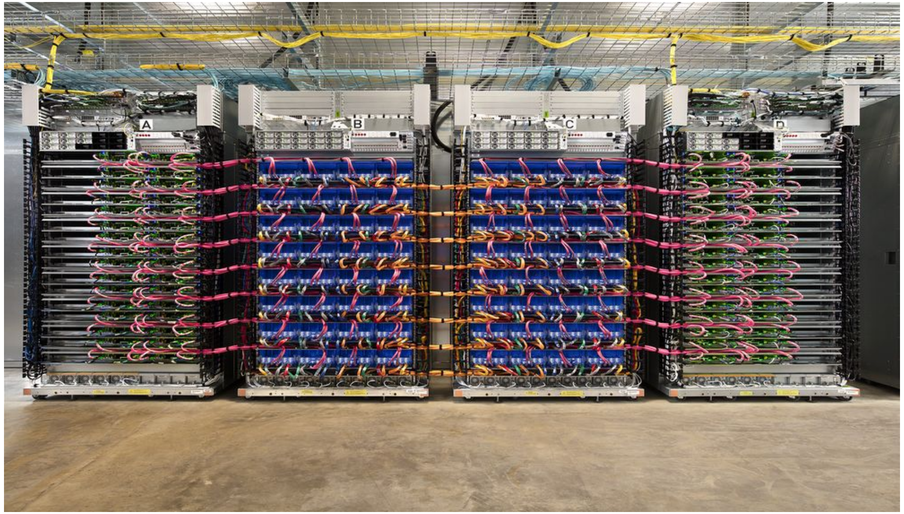
  \# Metropolis-Hastings correction
  $p_{\text{accept}} \leftarrow \min\left\{1, \frac{f(w')}{f(w)} \cdot \exp\left(\frac{1}{2}\|m\|^2 - \|m'\|^2\right)\right\};$
  $u \sim \text{Uniform}[0, 1];$
  **if** $u \leq p_{\text{accept}}$ **then**
    $w \leftarrow w';$
  **end if**
  $S \leftarrow S \cup \{w\};$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

**Algorithm 2** Leapfrog integration

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

# Hardware

- We run most of our HMC experiments on a TPU pod with 512 TPU-v3 devices



# HMC Hyper-Parameters

How to set the HMC hyper-parameters and what is their effect?

# Datasets and architectures

CIFAR-10, CIFAR-100

- No data augmentation

ResNet-20

- BatchNorm → Filter Response Norm
- ReLU → Swish

IMDB

- No data augmentation

CNN-LSTM

# Datasets and architectures

Same as in cold posteriors

CIFAR-10, CIFAR-100

- No data augmentation

ResNet-20

- BatchNorm → Filter Response Norm
- ReLU → Swish

IMDB

- No data augmentation

CNN-LSTM

- Can't use stochastic gradients
- Unclear how to do data augmentation in pure BNNs

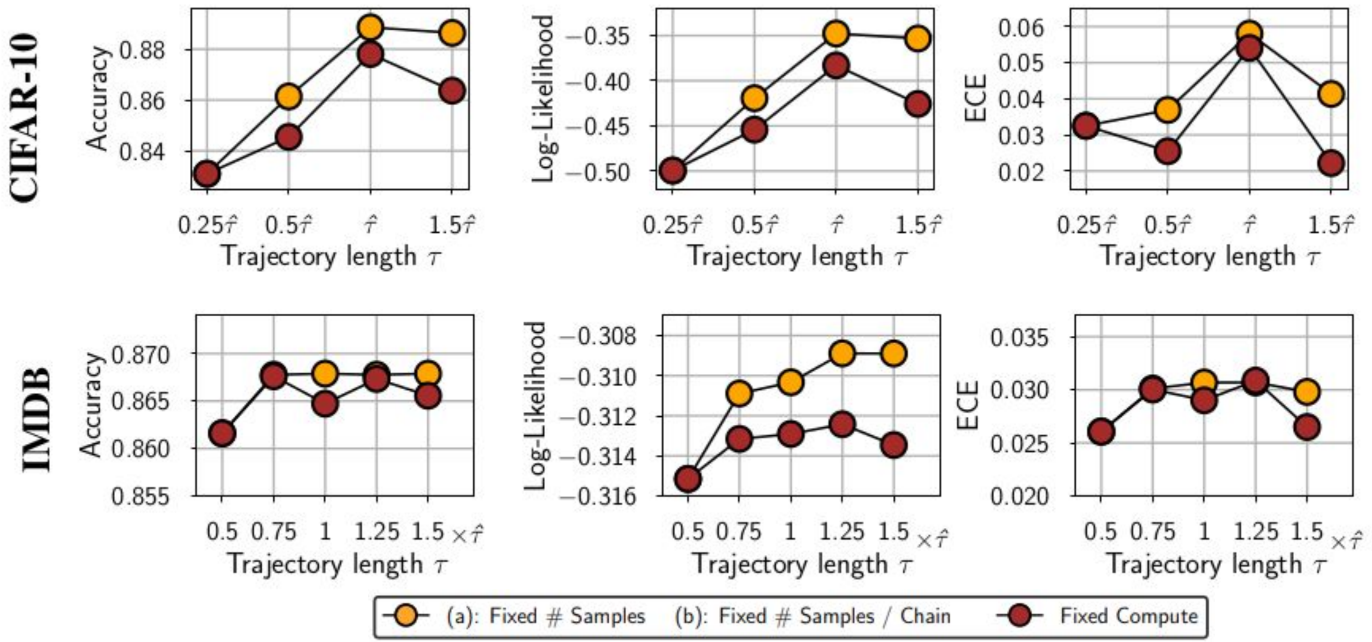
- Breaks train data independence

- Improves accept rates

# HMC hyper-parameters: trajectory length

- Longer trajectories → faster exploration (mixing)
- Longer trajectories → more expensive

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



# HMC hyper-parameters: step size

- Higher step-size → lower accept rates
- Lower step-size → more expensive

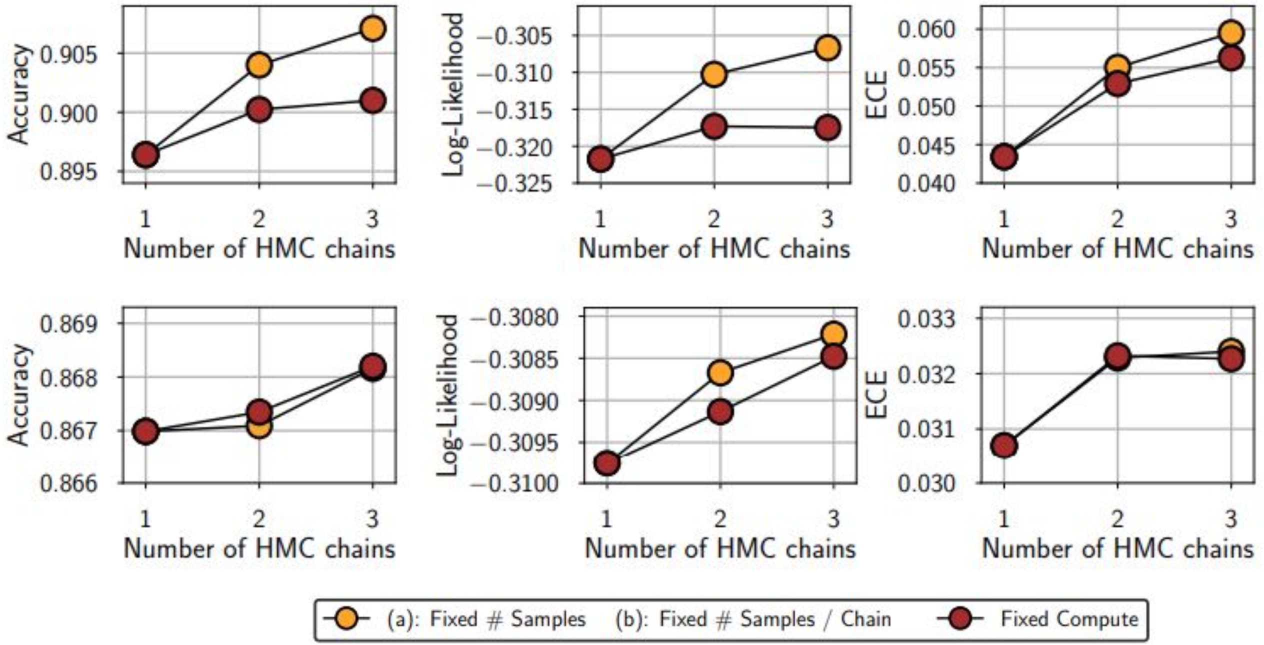
**Example: ResNet-20 FRN on CIFAR-10**

Prior std: *0.2*  Trajectory Length: *0.3*  Step size: $10^{-5}$

Gradient steps (epochs) to produce one sample: **30000**

# HMC hyper-parameters: number of chains

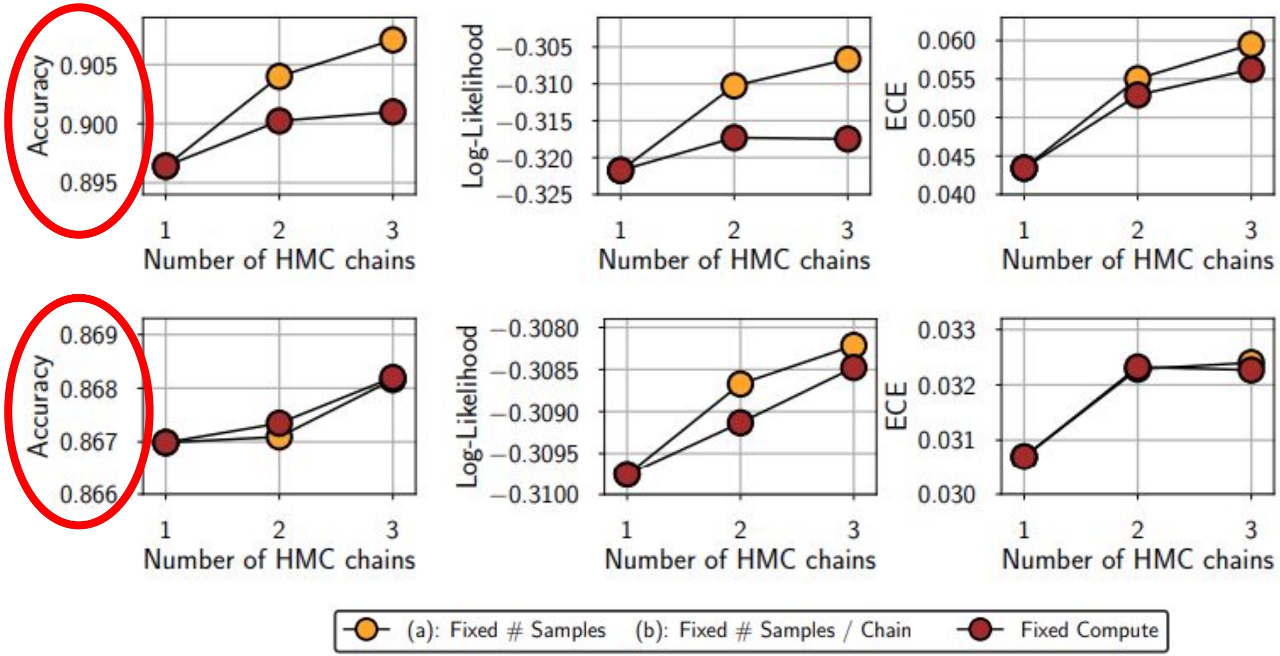
- More chains → better posterior approximation
- More chains → more expensive



# HMC hyper-parameters: number of chains

- More chains → better posterior approximation
- More chains → more expensive

Surprisingly small improvements



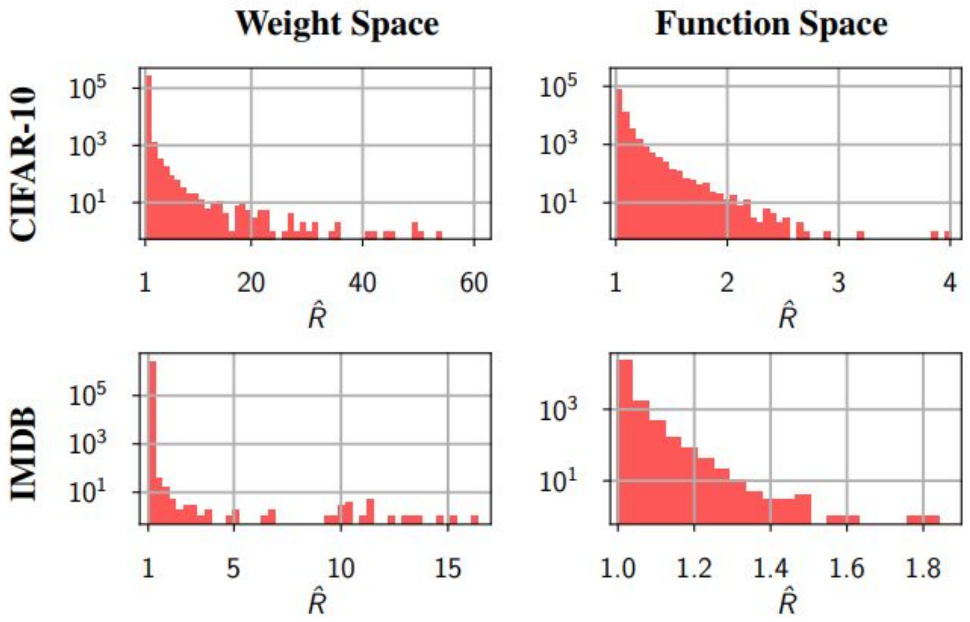
# Convergence and Mixing

Is HMC applied to BNNs mixing and converging?

# Mixing: R̂

$$\hat{R} \approx \frac{\text{between-chain variance}}{\text{avg within-chain variance}}$$

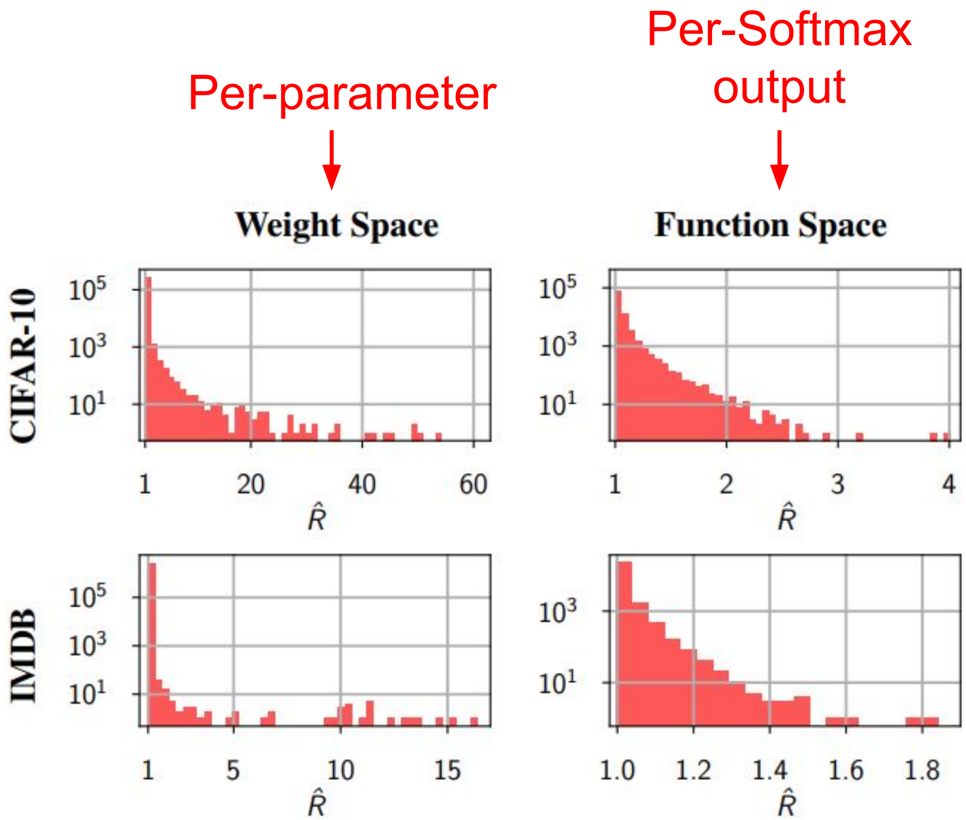
We want it close to 1



# Mixing: R̂

$$\hat{R} \approx \frac{\text{between-chain variance}}{\text{avg within-chain variance}}$$

We want it close to 1

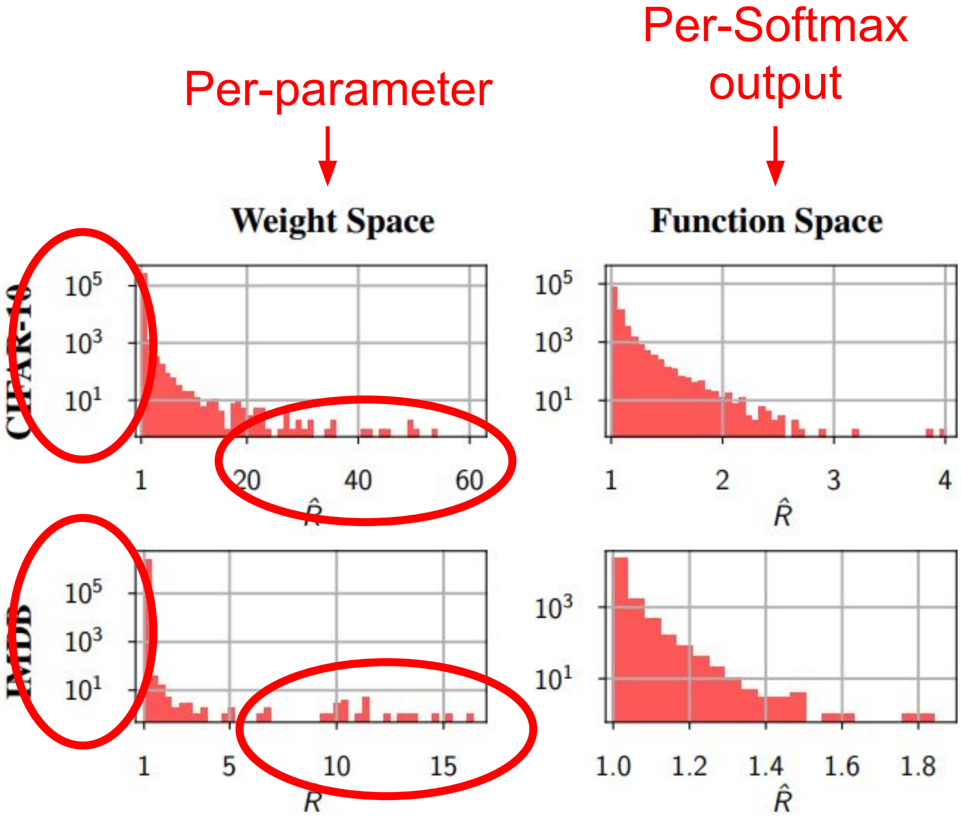


# Mixing: R̂

$$\hat{R} \approx \frac{\text{between-chain variance}}{\text{avg within-chain variance}}$$

We want it close to 1

Most R̂ are close to 1, especially in function space

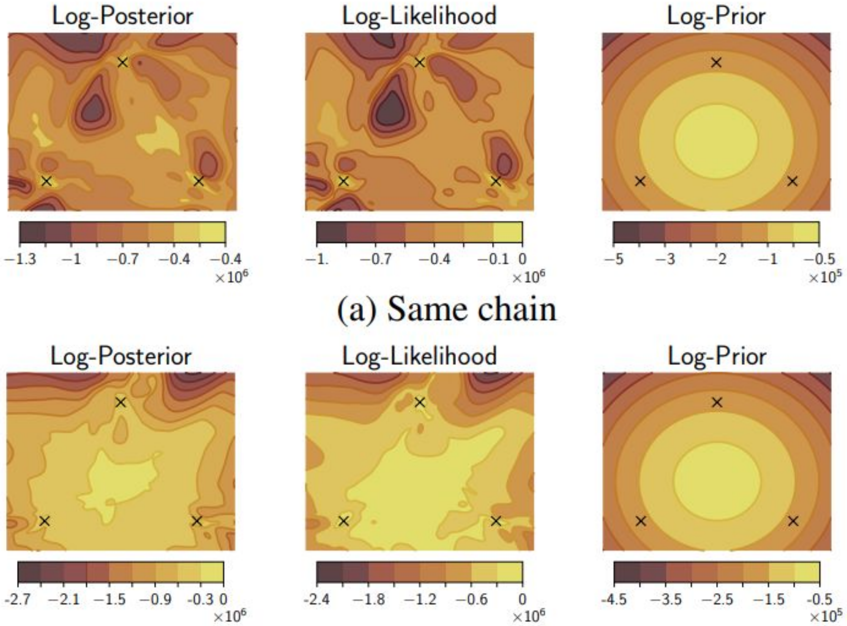


# Mixing: Posterior Geometry

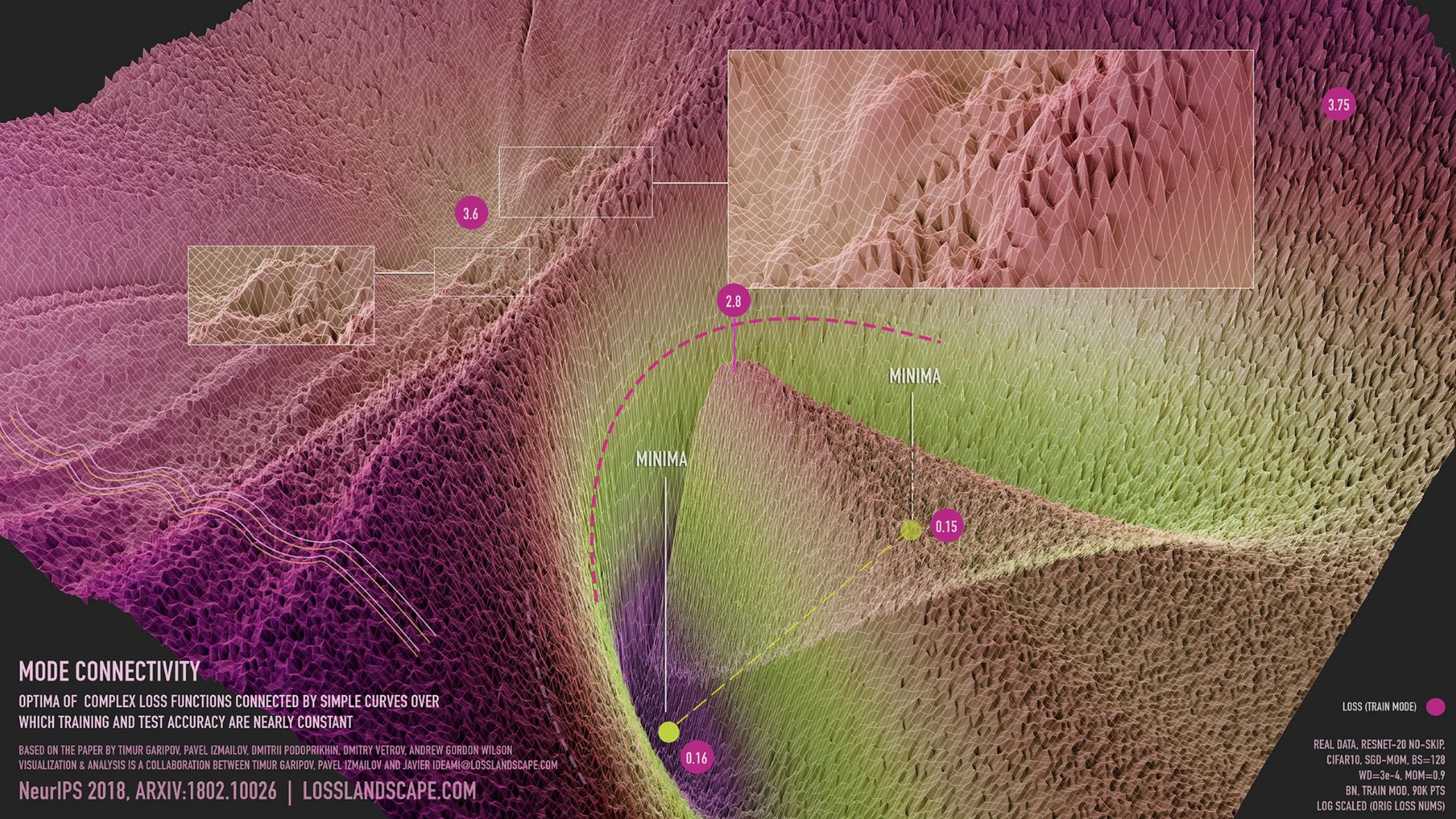
$\hat{R}$ results:

- We are not able to mix perfectly in parameter space
- A single HMC chain is able to explore functionally diverse connected regions of the posterior

**The posterior contains connected high-density regions that are functionally diverse and explorable by HMC!**

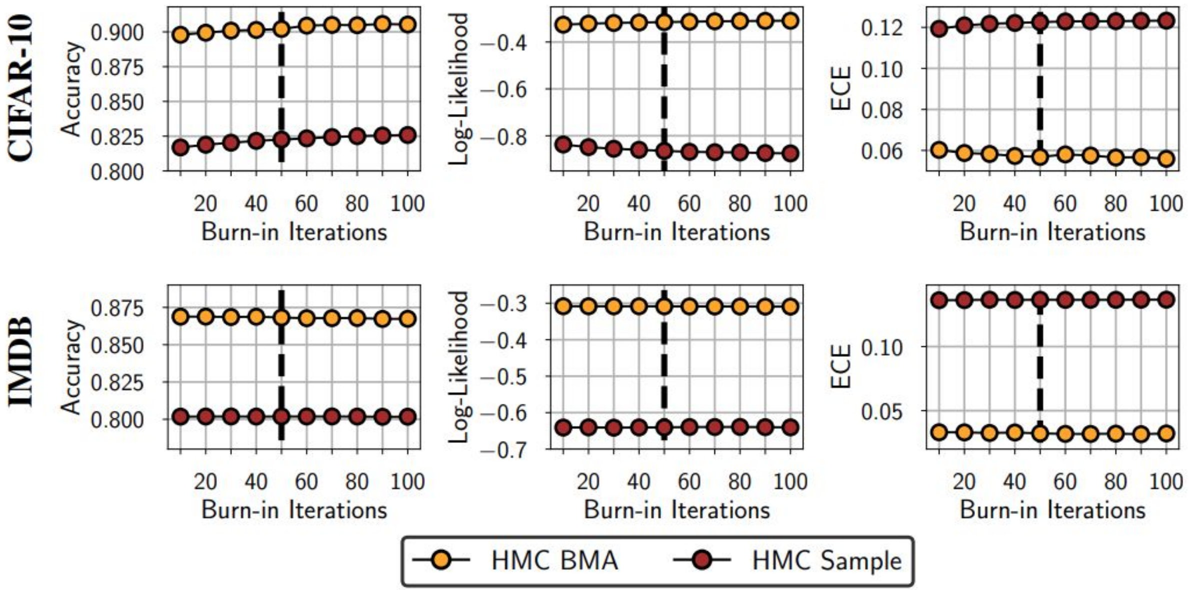



(a) Same chain


3.75
3.6
2.8
MINIMA
MINIMA
0.15
0.16
MODE CONNECTIVITY
OPTIMA OF COMPLEX LOSS FUNCTIONS CONNECTED BY SIMPLE CURVES OVER WHICH TRAINING AND TEST ACCURACY ARE NEARLY CONSTANT
BASED ON THE PAPER BY TIMUR GARIPOV, PAVEL IZMAILOV, DMITRII PODOPRIKHIN, DMITRY VETROV, ANDREW GORDON WILSON
VISUALIZATION & ANALYSIS IS A COLLABORATION BETWEEN TIMUR GARIPOV, PAVEL IZMAILOV AND JAVIER IDEAMI@LOSSLANDSCAPE.COM
NeurIPS 2018, ARXIV:1802.10026 | LOSSLANDSCAPE.COM
LOSS (TRAIN MODE)
REAL DATA, RESNET-20 NO-SKIP, CIFAR10, SGD-MOM, BS=128
WD=3e-4, MOM=0.9
BN, TRAIN MOD, 90K PTS
LOG SCALED (ORIG LOSS NUMS)

# HMC convergence

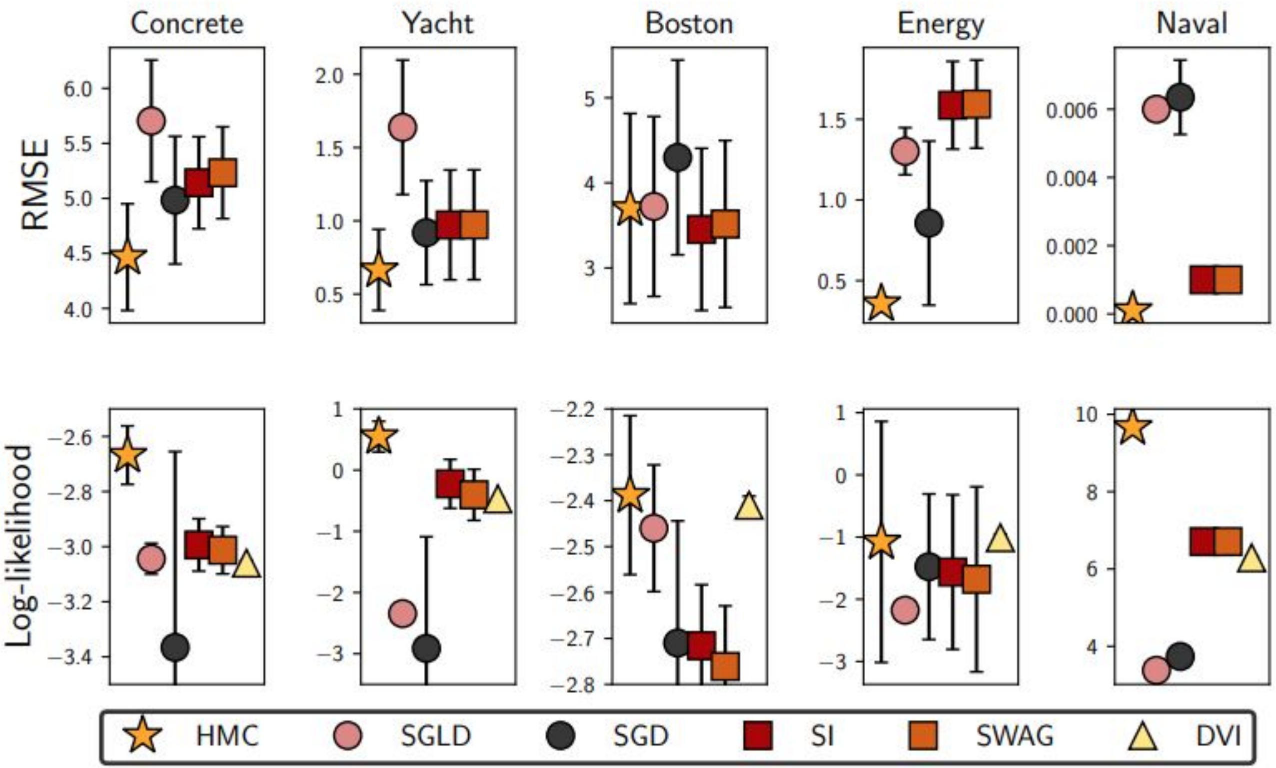
- HMC performance stabilizes fairly quickly, especially on IMDB
- We use a burn-in of 50 iterations



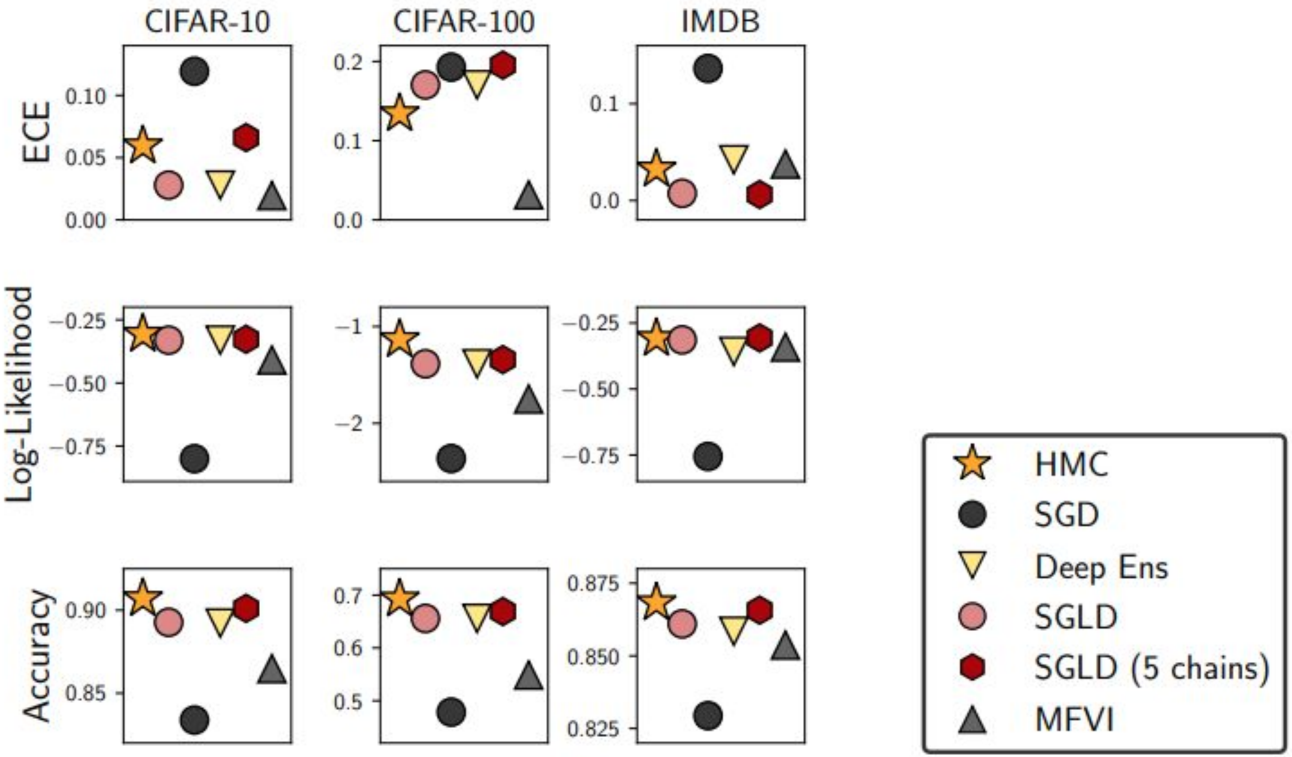
# BNN Evaluation

How do HMC BNNs perform in practice?

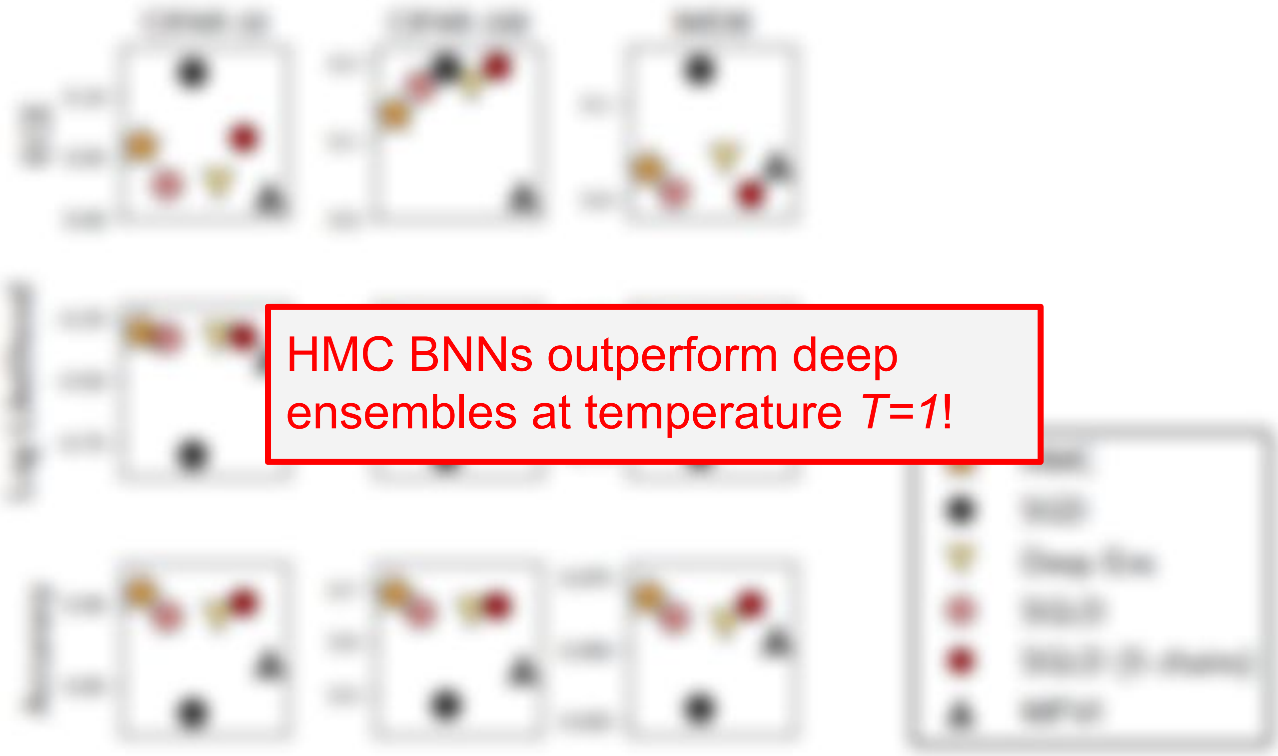
# BNN evaluation: UCI



# BNN evaluation: CIFAR and IMDB



# BNN evaluation: CIFAR and IMDB



HMC BNNs outperform deep ensembles at temperature *T=1*!

# BNN evaluation: OOD detection

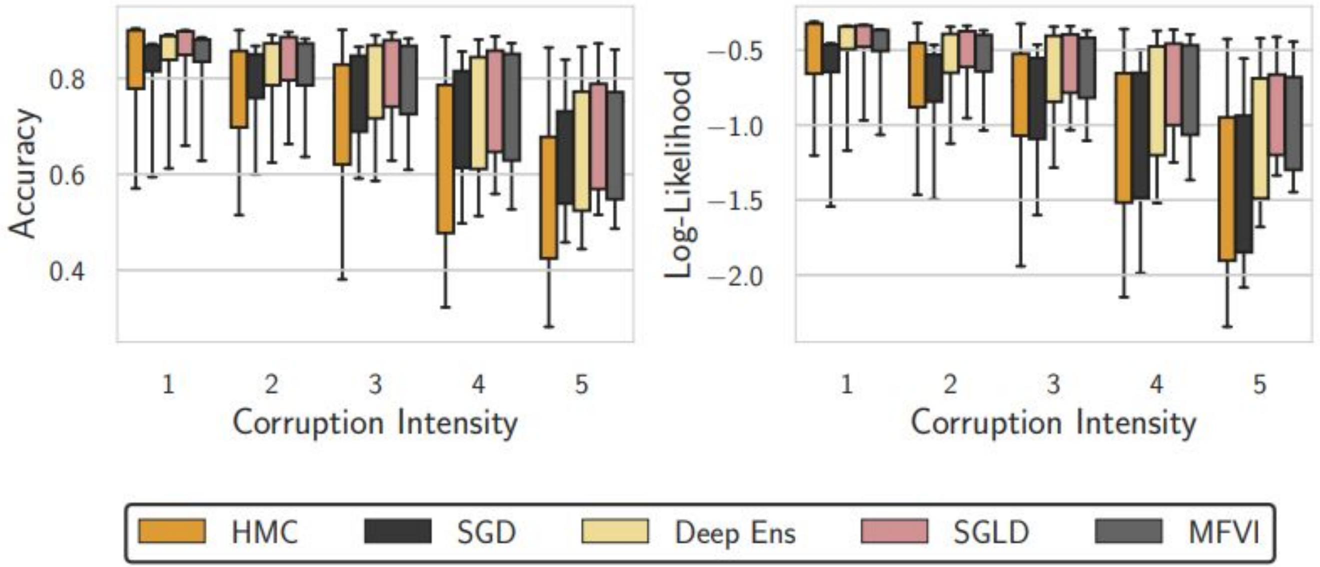
Train on CIFAR-10, detect OOD data by predictive uncertainty

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

HMC BNNs outperform deep ensembles in OOD detection

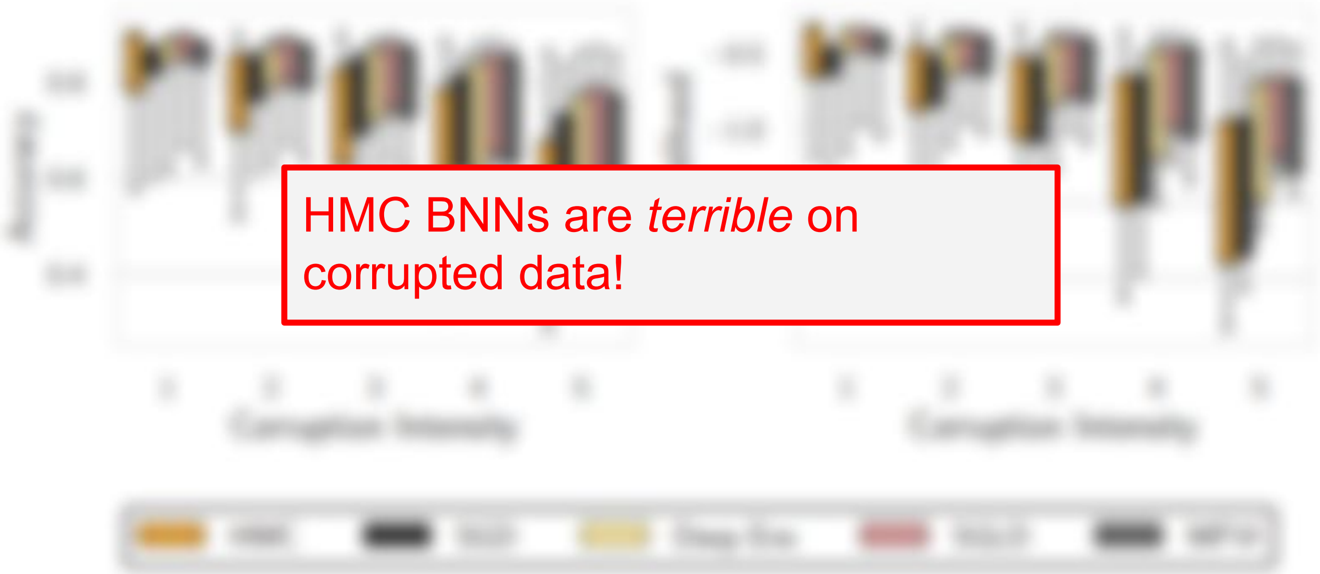
# BNN evaluation: OOD generalization

Train on CIFAR-10, test on CIFAR-10-C



# BNN evaluation: OOD generalization

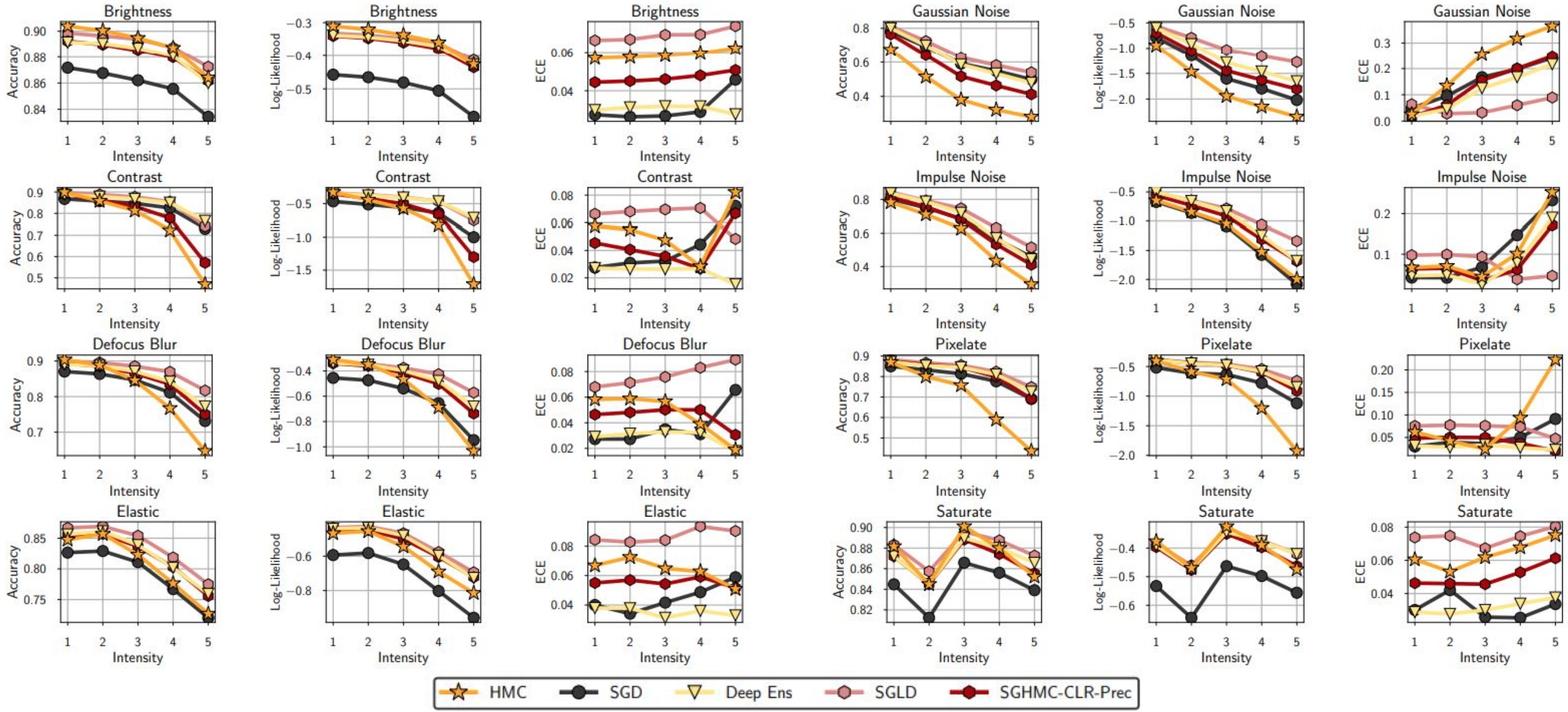
Train on CIFAR-10, test on CIFAR-10-C



HMC BNNs are *terrible* on corrupted data!

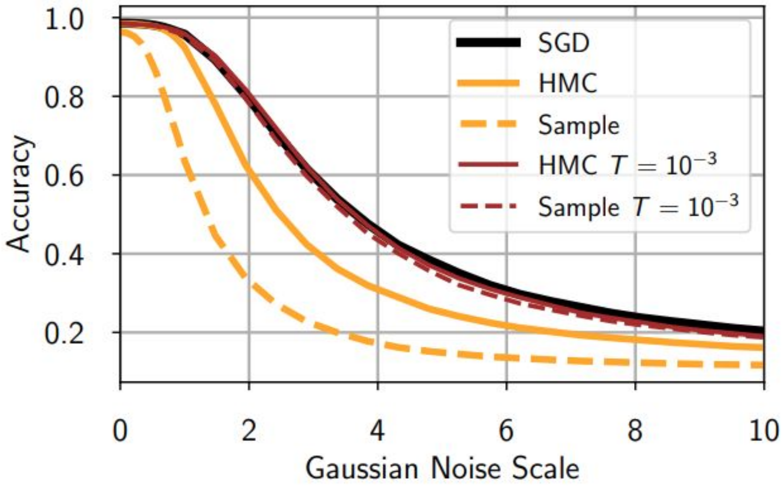
Surprising because BNNs are often evaluated on OOD generalization

# BNN evaluation: OOD generalization



# BNN evaluation: OOD generalization

Same behaviour on MNIST:



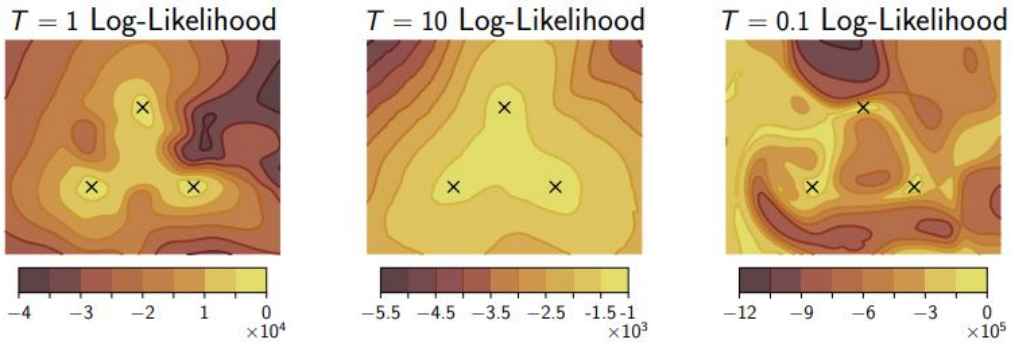
# Posterior Temperature Effect

What is the effect of posterior temperature? Do we need cold posteriors?

# Posterior temperature effect

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

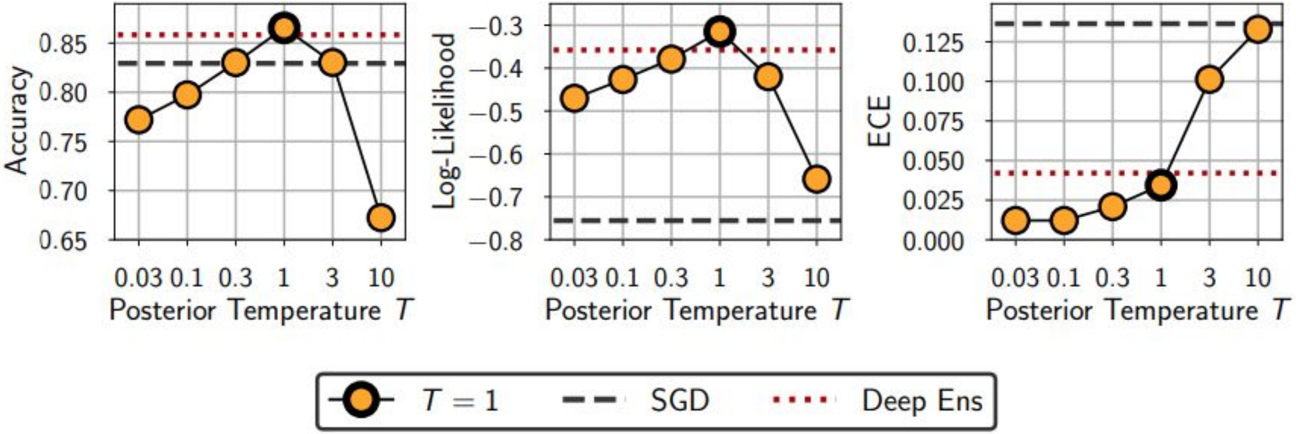
- Wenzel et al.: cold posteriors (temperatures T << 1) are needed to achieve good performance with BNNs
- Cold posteriors → sharper distribution, concentrated on high-density points




(c) IMDB, Log-Likelihood at different $T$

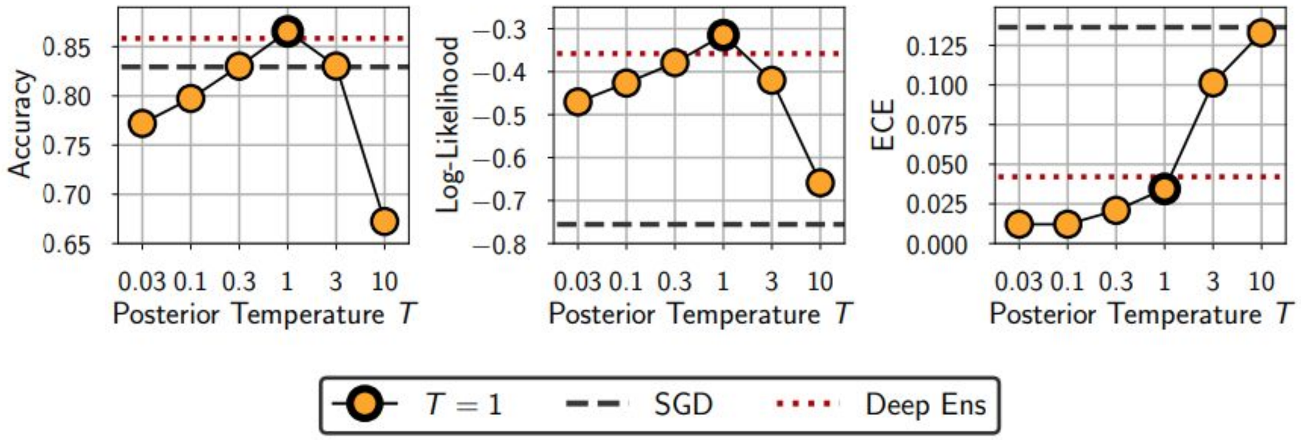
# Posterior temperature effect

- We have already seen that BNNs can do well at T=1
- What is the effect of T then?



# Posterior temperature effect

- We have already seen that BNNs can do well at T=1
- What is the effect of T then?




Cold posteriors are not required for good results and in fact can hurt performance!

# What's the difference with Wenzel et al.?

- Results using the original code of Wenzel et al. on CIFAR-10:

| | Acc, $T = 1$ | Acc, $T = 0.1$ | CE, $T = 1$ | CE, $T = 0.1$ |
|---|---|---|---|---|
| BN + Aug | 87.46 | 91.12 | 0.376 | 0.2818 |
| FRN + Aug | 85.47 | 89.63 | 0.4337 | 0.317 |
| BN + No Aug | 86.93 | 85.20 | 0.4006 | 0.4793 |
| FRN + No Aug | 84.27 | 80.84 | 0.4708 | 0.5739 |

# What's the difference with Wenzel et al.?

- Results reported by Wenzel et al.:

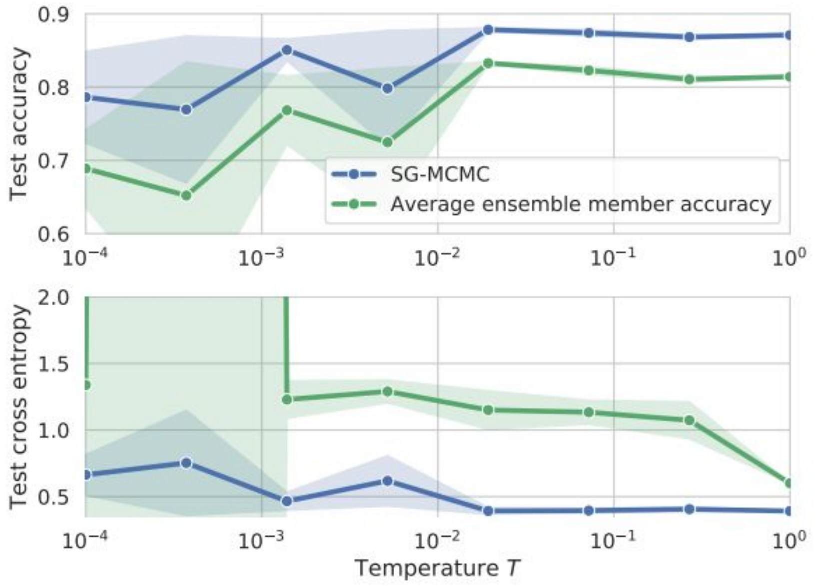



*Figure 28.* ResNet-20 with filter response normalization (FRN) instead of batch normalization and without any use of data augmentation.

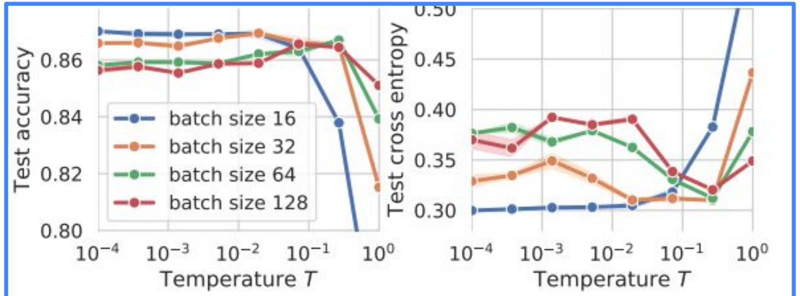



*Figure 6.* Batch size dependence of the CNN-LSTM/IMDB ensemble performance, reporting mean and standard error (3 runs): for all batch sizes, the optimal performance is achieved at $T < 1$.

# What's the difference with Wenzel et. al?

- Results reported by Fortuin et al. (concurrent):

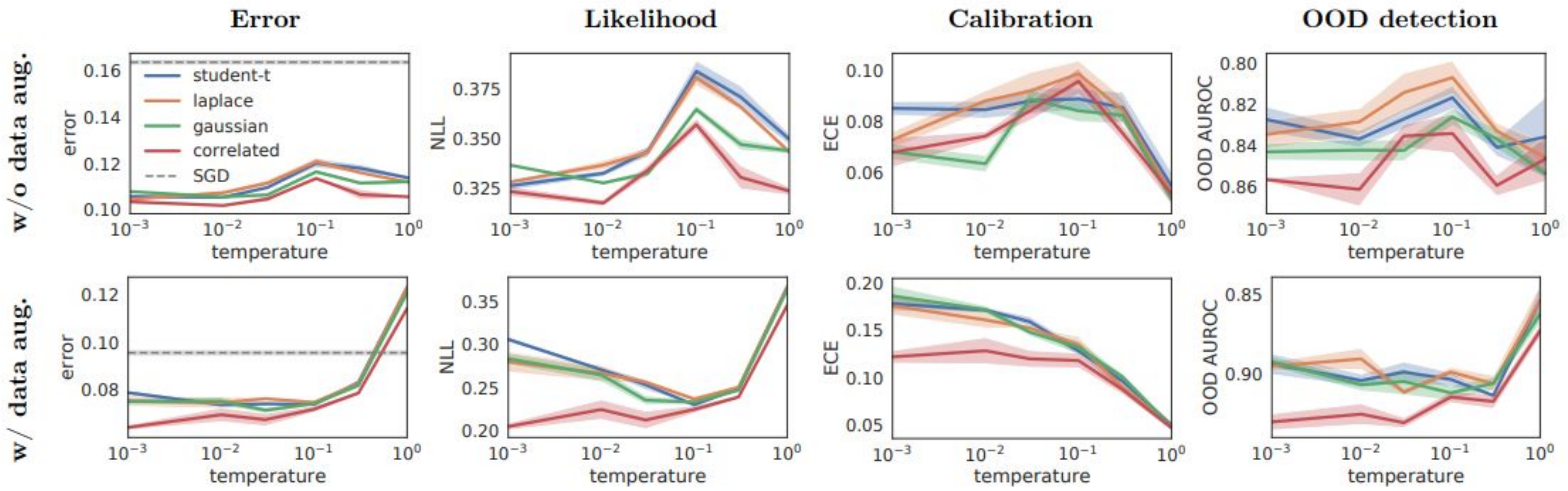



Figure A.11: Performances of Bayesian ResNets with different priors on CIFAR-10 with and without data augmentation in terms of different metrics. Data augmentation seems to increase the cold posterior effect.

# Sampling at low temperatures is hard

In fact, sampling at low temperatures is very hard:

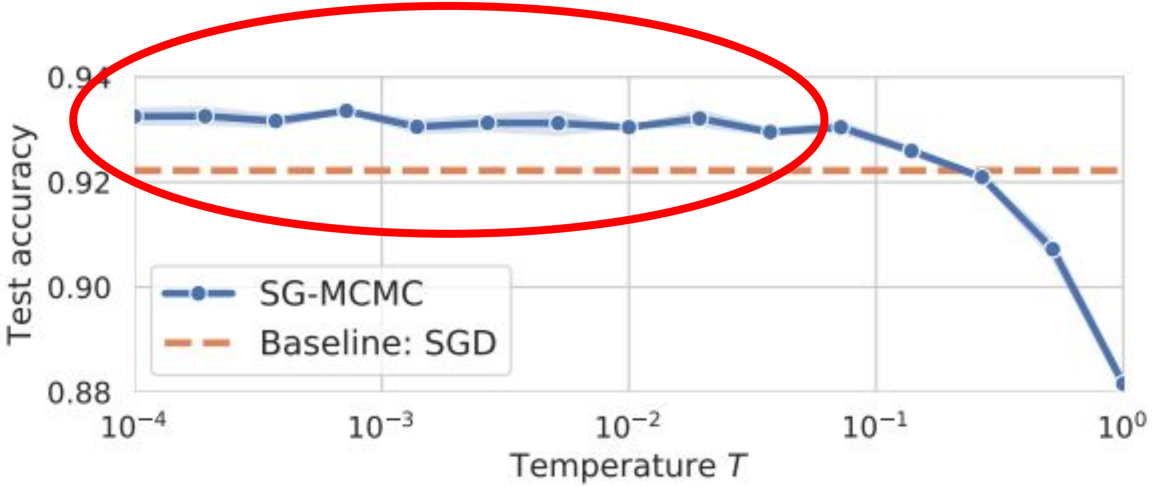
- We could only get high accept rates using 64-bit precision
- Low temperatures require very low step sizes

| | | | | | | |
|---|---|---|---|---|---|---|
| Temperature | 0.03 | 0.1 | 0.3 | 1 | 3 | 10 |
| Step Size | $3 \times 10^{-7}$ | $10^{-6}$ | $3 \times 10^{-6}$ | $10^{-5}$ | $3 \times 10^{-5}$ | $5 \times 10^{-5}$ |
| Epochs/Sample | 143K | 78K | 45K | 24K | 14K | 15K |

It is unlikely that other papers can truly sample the posterior at temperatures as low as $10^{-4}$ with SGMCMC.

# Sampling at low temperatures is hard

Possibly, this is why the curves never go down for low temperatures in [Wenzel et al](), [Fortuin et al.]()

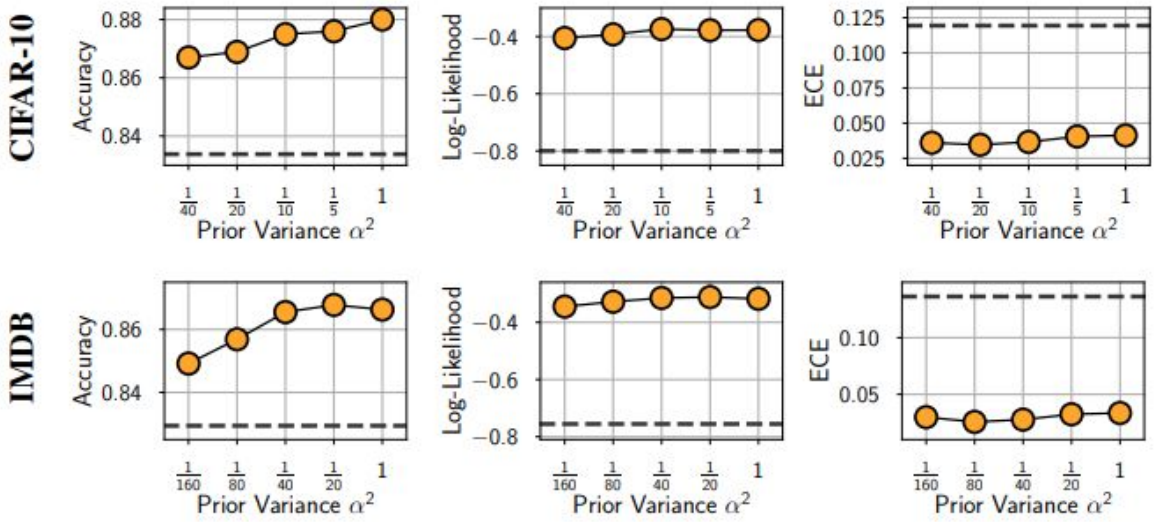



It is unlikely that other papers can truly sample the posterior at temperatures as low as $10^{-4}$ with SGMCMC.

# Effect of Priors

How robust are HMC BNNs to the choice of the prior?

# Effect of priors




HMC BNNs are fairly robust to Gaussian prior variance.

# Effect of priors

| 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** |

Results are fairly similar for different prior families.

# HMC as a reference

Do scalable BDL methods and HMC make similar predictions?

# HMC vs Scalable BDL

We compare the predictive distribution of HMC to that of scalable BDL methods using two metrics:

- Agreement $$\frac{1}{n}\sum_{i=1}^{n} I[\arg\max_j \hat{p}(y=j|x_i) = \arg\max_j p(y=j|x_i)]$$
- Total Variation $$\frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}\sum_j \left|\hat{p}(y=j|x_i) - p(y=j|x_i)\right|$$

# HMC vs Scalable BDL

| Metric | HMC (Reference) | SGD | Deep Ens | MFVI | SGMCMC: SGLD | SGMCMC: SGHMC | SGMCMC: SGHMC CLR | SGMCMC: 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 |

All scalable methods make predictions distinct from HMC.

# HMC vs Scalable BDL

| 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 |

Deep ensembles is closer to HMC than MFVI

# HMC vs Scalable BDL

| | | | | | SGMCMC | | | |
|---|---|---|---|---|---|---|---|---|
| METRIC | HMC (REFERENCE) | SGD | DEEP ENS | MFVI | 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 |

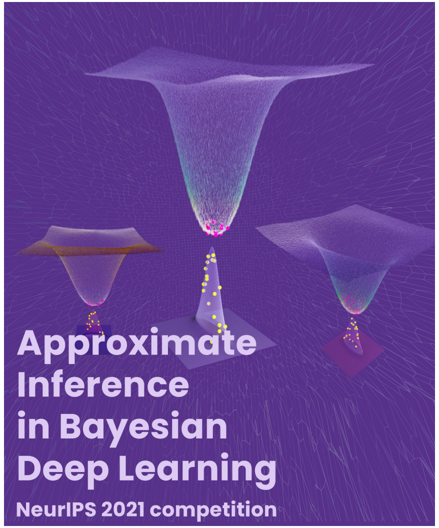
Advanced SGMCMC methods are closer to HMC and less accurate on CIFAR-10-C

# Links and resources

- Paper: arxiv
- Code: github/google-research/bnn_hmc
- Checkpoints: *coming very soon*
- NeurIPS competition: izmailovpavel.github.io/neurips_bdl_competition/

We hope that our HMC samples can be used by the BDL community to explore questions about BNNs and evaluate approximate inference methods.

We are also organizing a NeurIPS 2021 competition on approximate inference in BDL, more details soon!



# Takeaways

- **We can run full-batch HMC on Bayesian neural nets, although it is expensive**
- **HMC BNNs outperform SGD and Deep Ensembles and do not require cold posteriors**
- In fact cold posteriors can hurt performance
- Reliably sampling at low temperatures is very hard
- HMC BNNs are fairly robust to the choice of the prior
- **HMC BNNs are terrible when the test data is corrupted**
- We can use HMC as a reference to evaluate approximate inference methods
- **Deep Enembles are making more similar predictions to HMC BNNs compared to MFVI**
- Advanced SGMCMC methods provide the best approximation to HMC among the scalable BDL methods that we considered