URSABench: Comprehensive Benchmarking of Approximate Bayesian Inference Methods for Deep Neural Networks

Meet P. Vadera [*1] **Adam D. Cobb** [*2] **Brian Jalaian** [2] **Benjamin M. Marlin** [1]

Abstract

While deep learning methods continue to improve in predictive accuracy on a wide range of application domains, significant issues remain with other aspects of their performance including their ability to quantify uncertainty and their robustness. Recent advances in approximate Bayesian inference hold significant promise for addressing these concerns, but the computational scalability of these methods can be problematic when applied to large-scale models. In this paper, we describe initial work on the development of *URSABench* (the Uncertainty, Robustness, Scalability, and Accuracy Benchmark), an open-source suite of benchmarking tools for comprehensive assessment of approximate Bayesian inference methods with a focus on deep learning-based classification tasks.[1]

1. Introduction

As deep learning models continue to improve their predictive accuracy across many application domains, significant issues remain with respect to other highly important aspects of performance including their ability to robustly quantify uncertainty (Guo et al., 2017) and their ability to provide robust predictions in the presence of adversarial manipulations (Goodfellow et al., 2015) and out-of-distribution examples (Ovadia et al., 2019).

Approximate Bayesian inference methods (Neal, 1996; Jaakkola & Jordan, 2000) hold considerable promise for addressing such issues, and recent advances have significantly improved the feasibility of deploying approximate Bayesian inference methods to increasingly larger deep learning models (Welling & Teh, 2011; Zhang et al., 2020).

[*]Equal contribution [1]University of Massachusetts Amherst [2]US Army Research Laboratory. Correspondence to: Meet P. Vadera <mvadera@cs.umass.edu>, Adam D. Cobb <cobb.derek.adam@gmail.com>.

Presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning.

[1]Our PyTorch implementation is available at https://github.com/reml-lab/URSABench.

This paper describes initial work on URSABench, an open source suite of benchmarking tools for assessment of approximate Bayesian inference methods applied to deep neural network classification tasks. URSABench includes benchmark models, data sets, tasks and evaluation metrics focused on simultaneously assessing the uncertainty quantification performance, robustness, computational scalability and accuracy of learning and inference methods. We begin by briefly reviewing approximate Bayesian supervised learning. We then discuss principles for evaluation of such methods, followed by a description of the initial URSABench infrastructure and initial benchmarking results.

2. Bayesian Supervised Learning

In supervised learning, the data set $\mathcal{D}$ consists of a set of labeled instances $\{(\mathbf{x}_i, y_i)|1 \leq i \leq N\}$. $\mathbf{x}_i \in \mathbb{R}^D$ is the feature vector and $y_i \in \mathcal{Y}$ is the prediction target. We let $\mathcal{D}^x$ be the set of feature vectors and $\mathcal{D}^y$ be the set of targets. A probabilistic supervised learning model provides a conditional probability model of the form $p(y|\mathbf{x}, \theta)$ where $\theta \in \mathbb{R}^K$ are the model parameters. The conditional likelihood of the targets given the feature vectors and parameters is given by $p(\mathcal{D}^y|\mathcal{D}^x, \theta)$. The standard assumption that the data cases are independent and identically distributed leads to $p(\mathcal{D}^y|\mathcal{D}^x, \theta) = \prod_{i=1}^{N} p(y_i|\mathbf{x}_i, \theta)$ (Neal, 1996). Bayesian inference also requires asserting a prior distribution over the model parameters $p(\theta|\theta^0)$ that itself depends on prior parameters θ^0 (Neal, 1996).

The two key problems in Bayesian inference applied to supervised learning are the computation of the posterior distribution over the unknown parameters given a training data set $\mathcal{D}_{tr}$ as shown in Equation (1), and the computation of the posterior predictive distribution over the target variable y given a feature vector $\mathbf{x}$ and a data set $\mathcal{D}_{tr}$ as shown in Equation (2) (Neal, 1996).

$$p(\theta|\mathcal{D}_{tr}, \theta^0) = \frac{p(\mathcal{D}^y_{tr}|\mathcal{D}^x_{tr}, \theta)p(\theta|\theta^0)}{\int p(\mathcal{D}^y_{tr}|\mathcal{D}^x_{tr}, \theta)p(\theta|\theta^0)d\theta} \quad (1)$$

$$p(y|\mathbf{x}, \mathcal{D}_{tr}, \theta^0) = \mathbb{E}_{p(\theta|\mathcal{D}_{tr}, \theta^0)}[p(y|\mathbf{x}, \theta)] \quad (2)$$

It is well known that for neural network models the integrals required in Equations (1) and (2) are intractable. Approximate Bayesian inference methods thus aim to approximate

either the parameter posterior or expectations taken with respect to the parameter posterior such as Equation (2). Below, we briefly review three categories of Approximate Bayesian inference methods: Monte Carlo methods, surrogate density methods, and posterior distillation methods.

Monte Carlo Methods: Monte Carlo methods are a classical approach to Bayesian computation that approximate the intractable parameter posterior $p(\theta|\mathcal{D}, \theta^0)$ via a distribution constructed from a finite set of samples θ_s drawn from the true posterior $p(\theta|\mathcal{D}, \theta^0)$ (Smith & Roberts, 1993). This leads to the following approximate posterior predictive distribution: $p(y|\mathbf{x}, \mathcal{D}, \theta^0) \approx \frac{1}{S}\sum_{s=1}^{S} p(y|\mathbf{x}, \theta_s)$.

Of course, for complex models the problem of drawing samples from the true parameter posterior is also often computationally intractable. Markov chain Monte Carlo (MCMC) methods solve this problem by constructing a Markov chain with the true posterior $p(\theta|\mathcal{D}, \theta^0)$ as its equilibrium distribution. While classical MCMC methods are typically to slow to apply to large models (Casella & George, 1992; Chib & Greenberg, 1995; Duane et al., 1987; Neal, 2003; Girolami & Calderhead, 2011), a number of recent approaches have addressed this problem either by enabling sampling based on mini-batches of data (Welling & Teh, 2011; Chen et al., 2014; Zhang et al., 2020), or by sampling in reduced-dimensional parameter spaces (Izmailov et al., 2019).

Surrogate Density Methods: Another major family of methods are approaches based on approximating the true posterior density via an analytically tractable surrogate distribution $q(\theta|\mathcal{D}, \theta^0, \phi)$ where ϕ are auxiliary parameters of the surrogate distribution (Jordan et al., 1999; Jaakkola & Jordan, 2000; Ghosh et al., 2016; Minka, 2001). The most commonly used approaches in this family learn the parameters ϕ by minimizing the Kullback-Leibler (KL) divergence $\mathrm{KL}(p||p') = E_p[\log(p/p')]$ (MacKay, 2003). When the surrogate posterior is used as the first argument, the result is the variational inference (VI) framework (Jaakkola & Jordan, 2000). When it is used as the second argument, it yields the expectation propagation (EP) framework (Minka, 2001). Advances in the past decade have led to significantly more scalable methods in this family (Hoffman et al., 2013; Gal & Ghahramani, 2016; Dusenberry et al., 2020).

Distillation-Based Methods: The final class of methods that we review are posterior distillation-based methods including Bayesian Dark Knowledge (BDK) (Balan et al., 2015) and Generalized Posterior Expectation Distillation (GPED) (Vadera et al., 2020a). These methods directly approximate statistics of the posterior distribution by learning an auxiliary neural network model to mimic the output of corresponding Monte Carlo approximations. Importantly, their goal is not to improve over the Monte Carlo approximation, but rather to reduce the computation time required to compute the Monte Carlo average at deployment time.

3. URSABench Evaluation Principles

While advances in supervised deep learning methods have focused heavily on accuracy over the last decade, there are multiple aspects of models and inference algorithms that are of great interest. URSABench focuses on simultaneously assessing the uncertainty quantification performance, robustness, computational scalability and accuracy of learning and inference methods. In this section, we describe the evaluation principles that underlie URSABench. In the next section, we describe their current implementation.

Accuracy: Predictive performance is by far the most widely considered property of supervised machine learning models. In the classification setting, evaluation measures that only require that the true label y be correctly predicted provide the coarsest measures of the predictive performance. Accuracy is the most common such measure.

Uncertainty Quantification: A number of metrics are helpful for assessing the degree to which a method results in properly quantified predictive uncertainty. Predictive log likelihood can provide more insight into the predictive distribution than accuracy as it is sensitive to the predicted value of $p(y|\mathbf{x}, \mathcal{D}_{tr}, \theta^0)$. Both high-confidence errors and low-confidence correct predictions will result in lower log likelihood values.

Calibration is also an important property of predictive models and recent evaluations of deep learning methods have shown that their calibration properties can be quite poor (Guo et al., 2017). In the binary case, a classifier is said to be perfectly calibrated if exactly p percent of instances are predicted to be positive with p percent probability. The degree of calibration of a binary classifier can be quantified using the expected calibration error (ECE) (Guo et al., 2017). In the case of multi-class classification, a one-vs-all formulation of calibration error can be used. The Brier score provides an alternate measure of calibration (Brier, 1950) that can be interpreted as mixing together aspects of calibration and accuracy. Finally, misclassification detection performance (Hendrycks & Gimpel, 2017) is also helpful in assessing the utility of various uncertainty metrics.

Robustness: Another key property of models and inference methods is their robustness. Both predictive performance and uncertainty quantification metrics are typically computed on a test data set $\mathcal{D}_{te}$ that is assumed to be sampled from the same distribution as the training data set $\mathcal{D}_{tr}$. Out-of-distribution (OOD) detection tasks instead assess the ability of methods to detect examples from a set $\mathcal{D}_{ood}$ drawn from a different distribution than $\mathcal{D}_{tr}$ (Ovadia et al., 2019). The ability of methods to resist adversarial input perturbations as measured by the success rate of different adversarial attacks is also an important property (Goodfellow et al., 2015; Madry et al., 2018; Carlini & Wagner,

Table 1. URSABench **small-scale** benchmark performance. Results presented as mean ± std. dev. across 5 trials.)

Inference	Accuracy ↑	NLL ↓	Robustness ↑	Uncertainty ↑	Scalability ↓
HMC	0.9819 ± 0.0010	0.0593 ± 0.0016	0.9570 ± 0.0075	0.9734 ± 0.0012	0.72 ± 0.01
SGLD	0.9839 ± 0.0004	0.0492 ± 0.0022	0.9065 ± 0.0377	0.9679 ± 0.0233	2.02 ± 0.02
SGHMC	0.9862 ± 0.0003	0.0446 ± 0.0003	0.9426 ± 0.0048	0.9807 ± 0.0003	2.03 ± 0.02
cSGLD	0.9857 ± 0.0003	0.0476 ± 0.0011	0.9521 ± 0.0022	0.9795 ± 0.0007	14.08 ± 0.05
cSGHMC	0.9836 ± 0.0009	0.0533 ± 0.0016	0.9276 ± 0.0094	0.9759 ± 0.0015	14.77 ± 0.03
PCA + ESS (SI)	0.9840 ± 0.0007	0.0520 ± 0.0016	0.9360 ± 0.0038	0.9695 ± 0.0012	70.67 ± 0.20
MC dropout	0.9858 ± 0.0007	0.0501 ± 0.0031	0.9429 ± 0.0059	0.9769 ± 0.0019	2.02 ± 0.03
SGD	0.9860 ± 0.0002	0.0452 ± 0.0012	-	-	2.03 ± 0.02

Table 2. URSABench **medium-scale** benchmark performance.

Inference	Accuracy ↑	NLL ↓	Robustness ↑	Uncertainty ↑	Scalability ↓
SGLD	0.869	0.524	0.803	0.916	129.3
SGHMC	0.868	0.539	0.808	0.916	129.3
cSGLD	0.892	0.396	0.810	0.912	2103.3
cSGHMC	0.886	0.443	0.798	0.898	2114.9
SWAG	0.824	0.735	0.759	0.885	1351.7
PCA + ESS (SI)	0.869	0.482	0.804	0.901	1940.0
MC dropout	0.872	0.554	0.775	0.914	127.6
SGD	0.861	0.625	-	-	127.7

2017). We note that in the Bayesian supervised learning context, these attack methods require access to the posterior predictive distribution function and in many cases its gradients (Vadera et al., 2020b).

Scalability: Of primary interest in this work are how the accuracy, uncertainty quantification and robustness properties of methods trade off against their computational scalability properties with the goal of better understanding which methods offer the best trade-offs in different computational contexts (e.g., cluster, embedded system, etc.). The storage cost can be estimated via the number of parameters and the size of stored models (if variable bit depth is considered). The run-time of methods can be assessed in different ways including wall clock time as well as more portable statistics such as the number of floating point operations (flops) or multiply-accumulate operations (MACs).

4. URSABench Implementation Framework

In this section, we describe the current URSABench implementation framework, which leverages multiple datasets, models and tasks to implement the evaluation principles described in the previous section. The current framework includes small-scale and medium-scale benchmarks.

Models and Data Sets: The small-scale benchmark uses a basic, fully connected MLP with two hidden layers containing 200 units each as the benchmark model, with MNIST providing the benchmark in-domain data set (LeCun, 1998). At the medium-scale, we use ResNet50 (He et al., 2016) and WideResNet as the benchmark models (Zagoruyko & Komodakis, 2016), with CIFAR10 and CIFAR100 as the benchmark in-domain data sets (Krizhevsky et al., 2009).

Tasks and Metrics: The approximate parameter posterior and posterior predictive distribution are produced using each benchmark in-domain training set. Accuracy is assessed using the corresponding in-domain test sets. To assess uncertainty quantification, we compute negative log likelihood, Brier score, and performance on a misclassification detection task, all using the in-domain test sets. We also consider a decision-making task that focuses on assessing the quality of the tail of the predictive distribution using imbalanced data sets and costs that strongly penalize errors on the rare classes (Cobb et al., 2018) (see Appendix A for details). We assess robustness using an out-of-distribution (OOD) classification task (Ovadia et al., 2019; Vadera et al., 2020a) leveraging knowledge uncertainty (see Appendix B for a review of uncertinaty decomposition). The small-scale benchmark uses FashionMNIST (Xiao et al., 2017) and KMNIST (Clanuwat et al., 2018) as OOD test sets, while the medium-scale benchmark uses SVHN (Netzer et al., 2011) and STL10 (Coates et al., 2011) as OOD test sets. Performance on OOD tasks is assessed using AUROC. Finally, the current version of the benchmark focuses on computation time as the measure of computational scalability, measured in seconds/sample.

Composite Scores: The simultaneous assessment of multiple aspects of performance is the focus of URSABench. However, this yields many individual results for each inference method. An important design choice in URSABench is thus to summarize performance in terms of key selected individual metrics along with composite scores that combine related individual metrics. For accuracy we include an average over all benchmark models and all in-domain test sets. For robustness, we use an average over all mod-

els and OOD data sets. For uncertainty quantification, we separately compute an average over models and data sets in terms of negative log likelihood (NLL) and misclassification task performance.

5. URSABench Benchmark Results

In this section we report the initial benchmark results obtained using URSABench.

Inference Methods: We focus on bencmarking Monte Carlo methods including HMC[2], SGLD (Welling & Teh, 2011), SGHMC (Chen et al., 2014), cSGLD, cSGHMC (Zhang et al., 2020), SWAG (Maddox et al., 2019) and PCA-based subspace inference + elliptical slice sampling (PCA+ ESS (SI)) (Izmailov et al., 2019). As baselines, we also provide MC dropout (Gal & Ghahramani, 2016) and an SGD-point estimated model. Implementation details for the inference schemes have been provided in Appendix D.

Small-Scale Benchmark Results: The small-scale results are displayed in Table 1. The detailed experimental results behind each composite score can be found in Appendix E. The small-scale results show how challenging it can be to distinguish between different approximate inference schemes using relatively simple models and data sets. SGD and SGHMC are both marginally ahead in accuracy and NLL; HMC appears to show the most robust performance in OOD and SGHMC does best for the uncertainty metric. However the minor relative difference between all the performance metrics points to focusing on the compute time which shows HMC to be significantly less time consuming. This is due to the ability to fit all the data and model parameters on the GPU.

Medium-Scale Benchmark Results: The medium-scale results are displayed in Table 2. The detailed experimental results behind each composite score can again be found in Appendix E. Overall, the medium-scale experiments indicate a slight improvement on the predictive performance and decision-making tasks from both cSGHMC, and cSGLD followed by PCA + ESS (SI). However, once again a user may prefer using MC dropout or SGLD/SGHMC as they provide respectable performance in significantly less time. This is due to the large proportion of time that the cyclic schemes spend exploring without sampling. Furthermore, if the goal is to compute uncertainty metrics and ultimately use them for misclassification detection or OOD detection, then SGHMC/SGLD provide better performance in a majority of the cases. Another important result that can be seen from the Tables 6, 9, 12 and 15 in Appendix E is the demonstrated utility of the decision-making task in its ability to highlight the top performing approximate inference schemes for each model and data set, via its correlation with low NLL and high accuracy.

[2]HMC is only implemented for tasks where the model and full data set can fit on the GPU. We use the `hamiltorch` Python package (Cobb et al., 2019).

6. Conclusion and Future Work

This paper describes initial work on URSABench, a framework for benchmarking the performance of approximate Bayesian inference methods for deep neural networks. We hope that the development of this benchamrking toolbox will help to accelerate research in the domain of approximate Bayesian inference by helping to expose the trade-offs achieved by methods in terms of uncertainty, robustness, scalability and accuracy. We believe the simultaneous assessment of these properties is critical to better understand which methods are most effective on different downstream tasks and in different deployment contexts.

A further clear challenge in the development of this toolbox is ensuring a fair comparison between approaches. However, this can be difficult for new model/method/data set combinations without established hyperparameters, requiring careful hyperparameter optimization. This requirement highlights the issue of how to benchmark the end-to-end process of hyperparameter optimization and inference in terms of computational resource use.

As a first line of future work, we plan to continue to implement tasks and metrics to fully reflect all of the evaluation principles described in this paper. Important tasks and metrics yet to be implemented include robustness to adversarial examples (Vadera et al., 2020b) and common corruptions (Hendrycks & Dietterich, 2019), and assessment of test-time computational scalability. We further plan to add a large-scale benchmark that current approximate inference schemes will find challenging. Finally, we aim to expand the scope of models and data sets to include architectures such as recurrent neural networks and graph convolutional networks to provide a broader assessment of approximate inference methods.

Acknowledgement

Research reported in this paper was sponsored in part by the CCDC Army Research Laboratory under Cooperative Agreement W911NF-17-2-0196 (ARL IoBT CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

Balan, A. K., Rathod, V., Murphy, K. P., and Welling, M. Bayesian dark knowledge. In *NeurIPS*, 2015.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. Botorch: Programmable bayesian optimization in pytorch. *arXiv preprint arXiv:1910.06403*, 2019.

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

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

Casella, G. and George, E. I. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

Chen, T., Fox, E. B., and Guestrin, C. Stochastic gradient hamiltonian monte carlo. In *ICML*, 2014.

Chib, S. and Greenberg, E. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4): 327–335, 1995.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. Deep learning for classical japanese literature, 2018.

Coates, A., Ng, A. Y., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.

Cobb, A. D. *The practicalities of scaling Bayesian neural networks to real-world applications*. PhD thesis, University of Oxford, 2020.

Cobb, A. D., Roberts, S. J., and Gal, Y. Loss-calibrated approximate inference in Bayesian neural networks. *Theory of deep learning workshop, ICML*, 2018.

Cobb, A. D., Baydin, A. G., Markham, A., and Roberts, S. J. Introducing an Explicit Symplectic Integration Scheme for Riemannian Manifold Hamiltonian Monte Carlo. *arXiv preprint arXiv:1910.06243*, 2019.

Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., and Udluft, S. Decomposition of uncertainty for active learning and reliable reinforcement learning in stochastic systems. *ArXiv*, abs/1710.07283, 2017.

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

Dusenberry, M. W., Jerfel, G., Wen, Y., Ma, Y.-A., Snoek, J., Heller, K. A., Lakshminarayanan, B., and Tran, D. Efficient and scalable bayesian neural nets with rank-1 factors. *ArXiv*, abs/2005.07186, 2020.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

Ghosh, S., Delle Fave, F. M., and Yedidia, J. Assumed density filtering methods for learning bayesian neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Girolami, M. and Calderhead, B. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *ICML*, pp. 1321–1330, 2017.

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.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *ArXiv*, abs/1610.02136, 2017.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Izmailov, P., Maddox, W. J., Kirichenko, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. Subspace inference for bayesian deep learning. In *UAI*, 2019.

Jaakkola, T. S. and Jordan, M. I. Bayesian parameter estimation via variational methods. *Statistics and Computing*, 10(1):25–37, 2000.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

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

LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

MacKay, D. J. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Maddox, W., Garipov, T., Izmailov, P., Vetrov, D. P., and Wilson, A. G. A simple baseline for bayesian uncertainty in deep learning. In *NeurIPS*, 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Malinin, A., Mlodozeniec, B., and Gales, M. Ensemble distribution distillation. In *ICLR*, 2020.

Minka, T. P. Expectation propagation for approximate bayesian inference. In *UAI*, 2001.

Murray, I., Adams, R. P., and MacKay, D. J. C. Elliptical slice sampling. In *AISTATS*, 2010.

Neal, R. M. *Bayesian Learning for Neural Networks*. Springer-Verlag, 1996.

Neal, R. M. Slice sampling. *Annals of statistics*, pp. 705–741, 2003.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, Sebastian Dillon, J., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *NeurIPS*, pp. 13969–13980, 2019.

Smith, A. F. and Roberts, G. O. Bayesian computation via the gibbs sampler and related markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Methodological)*, 55(1):3–23, 1993.

Vadera, M. P., Jalaian, B., and Marlin, B. M. Generalized bayesian posterior expectation distillation for deep neural networks. In *UAI*, 2020a.

Vadera, M. P., Shukla, S. N., Jalaian, B., and Marlin, B. M. Assessing the adversarial robustness of monte carlo and distillation methods for deep bayesian neural network classification. In *AAAI SafeAI Workshop*, 2020b.

Wang, K.-C., Vicol, P., Lucas, J., Gu, L., Grosse, R., and Zemel, R. Adversarial distillation of bayesian neural network posteriors. *arXiv:1806.10317*, 2018.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, pp. 681–688, 2011.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *ArXiv*, abs/1708.07747, 2017.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *ArXiv*, abs/1605.07146, 2016.

Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. Cyclical stochastic gradient mcmc for bayesian deep learning. In *ICLR*, 2020.

A. Decision-making task

Bayesian decision theory takes Monte Carlo samples and averages them over a predetermined cost function, $\mathcal{C}(h, y)$ to result in an expected risk:

$$R(h|\mathbf{x}) \approx \frac{1}{S}\sum_{s=1}^{S} p(y|\mathbf{x}, \theta_s)\mathcal{C}(h, y).$$

The expected risk is a function of the decision, h, whereby the Bayesian optimal decision, h^* minimises the risk:

$$h^* = \underset{h}{\operatorname{argmin}}\, R(h|\mathbf{x}).$$

Once we have applied Bayesian decision theory to find an h^* for every input $\mathbf{x}$, for supervised classification, we can then determine the true cost of the decision taken by averaging over the test data (i.e. $\frac{1}{N}\sum_{n=1}^{N}\mathcal{C}(y_n^{\text{True}}, h_n^*)$), where the arguments have been reversed such that we compare the true label y_n^{True}, with the Bayesian optimal decision h_n^* (i.e. what cost did we actually have to pay when we took decision h_n^* for labelling y_n, when it was in fact y_n^{True}). A more detailed discussion can be seen in Ch. 4 of Cobb (2020).

The purpose of the decision-making task is to penalise inference schemes that provide poor calibrated uncertainty over the rarer (and hence more uncertain) classes. In particular, for MNIST, we retrain our models over a highly imbalanced data set, where 99% of the labels corresponding to classes 3 and 7 are removed. However, we then use the predictive distribution to with a predefined cost function to select the Bayes optimal decision for each predicted label. We then calculate the expected decision cost by averaging over the costs attributed to each decision compared to the true label. False negatives of the less frequent classes are penalised 1000 times more than false positives for the rest of the classes.

The small-scale setting for the decision-making task requires retraining over an imbalanced training set. However, for the medium-scale task we limit ourselves to using the same materialised samples from the balanced data set (although we expect to extend this to imbalanced training data in future work). We define our cost matrix to penalise false negatives 10 times as much as false positives. In particular, the task for the CIFAR10 penalises planes, automobiles, ships and trucks with a cost of 1.0 for false negatives and 0.1 for false positives. All other errors are penalised with 0.1 and correct decisions accrue zero cost. The same cost structure applies

to CIFAR100, where tanks, rockets and pick-up trucks are deemed the critical classes.

The decision costs in Tables 6, 9, 12 and 15 demonstrate the utility of this task as they show a correlation with the NLL and the accuracy across all models and data sets.

B. Uncertainty Decomposition for downstream tasks

The posterior predictive distribution is not the only statistic of the posterior distribution that is of interest. The decomposition of posterior uncertainty has also received recent attention in the literature. For example, Depeweg et al. (2017) and Malinin et al. (2020) describe the decomposition of the entropy of the posterior predictive distribution (the *total uncertainty*) into *expected data uncertainty* and *knowledge uncertainty*. These three forms of uncertainty are related by the equation shown below:

$$\underbrace{\mathcal{I}\left[y, \theta | \boldsymbol{x}, \mathcal{D}\right]}_{\text{Knowledge Uncertainty}} = \underbrace{\mathcal{H}\left[\mathbb{E}_{p(\theta|\mathcal{D})}\left[p\left(y|\boldsymbol{x}, \theta\right)\right]\right]}_{\text{Total Uncertainty}} - \underbrace{\mathbb{E}_{p(\theta|\mathcal{D})}\left[\mathcal{H}\left[p\left(y|\boldsymbol{x}, \theta\right)\right]\right]}_{\text{Expected Data Uncertainty}} \tag{3}$$

Total uncertainty, as the name suggests, measures the total uncertainty in a prediction. Expected data uncertainty measures the uncertainty arising from class overlap. Knowledge uncertainty corresponds to the conditional mutual information between labels and model parameters and measures the disagreement between different models in the posterior. However, it can be efficiently computed as the difference between total uncertainty and expected data uncertainty, both of which are (functions) of posterior expectations. In recent work, Wang et al. (2018), Malinin et al. (2020) and Vadera et al. (2020a) have leveraged this decomposition to explore a range of down-stream tasks that rely on uncertainty quantification and decomposition.

C. Composite Score Breakdown

As alluded to in the main text, we build composite scores for robustness and uncertainty. The robustness relies on averaging both the total uncertainty AUROC and the model uncertainty AUROC over the OOD data sets. We then average once again over the mean total uncertainty and model uncertainty. The uncertainty composite score is built from the average misclassification AUROCs (e.g. the first three columns of Tables 5, 8, 11, 14, 17). For the medium-scale experiment the uncertainty score is then averaged across CIFAR10 and CIFAR100 as well as ResNet50 and WideResNet28x10.

D. Implementation Details

In this section, we describe the implementation details for the different inference methods used in our benchmark. It must be noted that for all inference methods using ResNet50 and WideResNet28x10 models, we use a pretrained SGD solution to warm-start our samplers. This is a standard pre-training procedure followed to make the methods more competitive (Maddox et al., 2019). Further, the ensemble size is set to 50 for CIFAR datasets, and 100 for MNIST dataset. The difference in ensemble size is due to the large amounts of computational requirements for training ResNet50 and WideResNet28x10 on CIFAR datasets. While tuning hyperparameters for MNIST, we apply Bayesian optimization with a limit of 200 evaluations for each approach (Balandat et al., 2019). On the other hand, for CIFAR datasets, we refer to existing literature and use the same hyperparameters if directly applicable, or search around the hyperparameters obtained for similar models and datasets.

SGLD: For CIFAR datasets, we use a burn-in of 100 epochs and initial learning rates of 0.1 for WideResNet28x10 model and 0.05 for ResNet50. The prior std. dev. is set to 1 for both the cases. We decay the learning rate using cosine annealing schedule to its half value by the end of sampling. For MNIST, the optimal hyperparameter values obtained are: initial learning rate of 0.099, prior std. dev. of 0.16 and 50 burn in epochs.

SGHMC: We use the same hyperparameters and learning rate schedule as described for SGLD for the CIFAR datasets. Additionally, we set the friction term to 0.5 (Chen et al., 2014). This is equivalent to the α term shown in Zhang et al. (2020). For MNIST, the optimal hyperparameter values obtained are: initial learning rate of 0.03, prior std. dev. of 0.14, 50 burn in epochs, and friction term set to 0.1.

cSGHMC: We use the same hyperparameters given in Zhang et al. (2020) for CIFAR datasets. For MNIST, the optimal hyperparameter values obtained are: initial learning rate of 0.06, prior std. dev. of 0.33, cycle length of 22 epochs, of which 17 epochs are used for SGD-exploration phase, and samples are collected from the last 4 epochs, and friction term set to 0.21.

cSGLD: We use the same hyperparameters given in Zhang et al. (2020) for CIFAR datasets. For MNIST, the optimal hyperparameter values obtained are: initial learning rate of 0.06, prior std. dev. of 0.33, cycle length of 22 epochs, of which 17 epochs are used for SGD-exploration phase, and samples are collected from the last 4 epochs, and friction term set to 0.21.

SWAG: We use the same hyperparameters given in Izmailov et al. (2019) for CIFAR models, except that we set the weight decay for ResNet models to 10^{-4} and borrow its remaining hyperparameters from WideResNet28x10. This

means that we utilize last 20 SGD iterates to find parameters for the gaussian approximation to the mode. For MNIST, we start with an initial learning rate of 0.018 and decay it 0.0006 over 50 epochs. Further, we run SGD at the same learning rate for another 30 epochs and collect the final 20 iterates to construct our SWAG approximation. The momentum for our SGD optimizer is set to 0.7 through the entire run. Furthermore, the variant of SWAG used in our benchmark is SWAG-diagonal.

PCA + ESS (SI): We use the same hyperparameters given in Izmailov et al. (2019) for CIFAR models, except that we set the weight decay for ResNet models to 10^{-4} and borrow its remaining hyperparameters from WideResNet28x10. We construct a subspace of rank 20 for all models and datasets. For MNIST, we start with an initial learning rate of 0.04 and decay it 0.002 over 50 epochs. Further, we run SGD at the same learning rate for another 50 epochs and collect the iterates from each of the final 20 epochs to construct our PCA subspace. The momentum for our SGD optimizer is set to 0.54 through the entire run. For all the dataset and model combinations, we use elliptical slice sampling (Murray et al., 2010) on the low rank PCA subspace with a prior of 2. and a temperature of 5000.

MC Dropout: For all the models on CIFAR datasets, we use a dropout of 0.2 before the final linear layer while we use dropout after both hidden layers for MNIST-MLP200 with a dropout rate of 0.04.

E. Additional Experimental Results

Additional experimental results are provided in Tables 3 - 17.

Table 3. Comparison of predictive performance and decision making cost while using an MLP $[784, 200, 200, 10]$ on MNIST. Results presented as mean $\pm$ std. dev. across 5 trials.

Inference	Accuracy ↑	NLL ↓	BS ↓	ECE ↓	Decision Cost ↓	Samples/second ↓
HMC	$98.19 \pm 0.10\%$	0.0593 ± 0.0016	0.0280 ± 0.0008	0.0079 ± 0.0008	7101 ± 346	0.72 ± 0.01
SGLD	$98.39 \pm 0.04\%$	0.0492 ± 0.0022	0.0236 ± 0.0005	0.0041 ± 0.0024	5410 ± 778	2.02 ± 0.02
SGHMC	$98.62 \pm 0.03\%$	0.0446 ± 0.0003	0.0210 ± 0.0002	0.0073 ± 0.0004	5408 ± 240	2.03 ± 0.02
cSGLD	$98.57 \pm 0.03\%$	0.0476 ± 0.0011	0.0223 ± 0.0003	0.0056 ± 0.0003	6526 ± 2241	14.08 ± 0.05
cSGHMC	$98.36 \pm 0.09\%$	0.0533 ± 0.0016	0.0256 ± 0.0010	0.0033 ± 0.0003	4824 ± 1855	14.77 ± 0.03
PCA + ESS (SI)	$98.40 \pm 0.07\%$	0.0520 ± 0.0016	0.0251 ± 0.0007	0.0036 ± 0.0005	3809 ± 1150	70.67 ± 0.20
MC dropout	$98.58 \pm 0.07\%$	0.0501 ± 0.0031	0.0218 ± 0.0008	0.0042 ± 0.0006	15236 ± 1184	2.02 ± 0.07
SGD	$98.60 \pm 0.02\%$	0.0452 ± 0.0012	0.0213 ± 0.0003	0.0032 ± 0.0005	8613 ± 1428	2.03 ± 0.02

Table 4. Comparison of OOD detection performance while using an MLP $[784, 200, 200, 10]$ on MNIST. Results presented as mean $\pm$ std. dev. across 5 trials.

Inference	OOD Dataset	AUROC- Model Uncertainty ↑	AUROC - Total Uncertainty ↑
HMC	Fashion MNIST	0.966 ± 0.013	0.946 ± 0.017
	KMNIST	0.968 ± 0.013	0.948 ± 0.017
SGLD	Fashion MNIST	0.867 ± 0.110	0.944 ± 0.005
	KMNIST	0.871 ± 0.103	0.944 ± 0.005
SGHMC	Fashion MNIST	0.933 ± 0.009	0.953 ± 0.010
	KMNIST	0.932 ± 0.009	0.952 ± 0.010
cSGLD	Fashion MNIST	0.954 ± 0.004	0.950 ± 0.005
	KMNIST	0.954 ± 0.004	0.950 ± 0.005
cSGHMC	Fashion MNIST	0.923 ± 0.021	0.931 ± 0.017
	KMNIST	0.923 ± 0.020	0.933 ± 0.017
PCA + ESS (SI)	Fashion MNIST	0.933 ± 0.006	0.938 ± 0.009
	KMNIST	0.934 ± 0.006	0.940 ± 0.009
MC dropout	Fashion MNIST	0.942 ± 0.013	0.943 ± 0.010
	KMNIST	0.943 ± 0.013	0.944 ± 0.010
SGD	Fashion MNIST	N/A	0.945 ± 0.010
	KMNIST	N/A	0.943 ± 0.010

Table 5. Comparison of misclassification detection while using an MLP $[784, 200, 200, 10]$ on MNIST.

Inference	AUROC- Model Uncertainty ↑	AUROC - Total Uncertainty ↑	AUROC- Model Confidence ↑	AUCPR- Model Uncertainty ↑	AUCPR - Total Uncertainty ↑	AUCPR- Model Confidence ↑
HMC	0.9706	0.9734	0.9743	0.3429	0.3888	0.4145
SGLD	0.9739	0.9800	0.9800	0.3530	0.4502	0.4632
SGHMC	0.9786	0.9815	0.9823	0.3546	0.3929	0.4131
cSGLD	0.9769	0.9801	0.9798	0.3695	0.4477	0.4478
cSGHMC	0.9730	0.9786	0.9786	0.3260	0.4255	0.4404
PCA + ESS (SI)	0.9539	0.9774	0.9772	0.2059	0.4298	0.4248
MC dropout	0.9754	0.9763	0.976	0.4085	0.43	0.4199
SGD	N/A	0.9795	0.9794	N/A	0.4273	0.4389

Table 6. Comparison of predictive performance and decision making cost while using ResNet50 on CIFAR10.

Inference	Accuracy ↑	NLL ↓	BS ↓	ECE ↓	Decision Cost ↓
SGLD	0.954	0.144	0.069	0.009	139.500
SGHMC	0.954	0.144	0.068	0.011	138.100
cSGLD	0.966	0.128	0.053	0.020	112.100
cSGHMC	0.951	0.243	0.086	0.106	153.900
SWAG	0.931	0.311	0.114	0.047	200.900
PCA + ESS (SI)	0.949	0.174	0.080	0.027	166.600
MC dropout	0.948	0.208	0.083	0.032	159.500
SGD	0.943	0.274	0.095	0.040	171.700

Table 7. Comparison of OOD detection performance while using ResNet50 on CIFAR10.

Inference	OOD Dataset	AUROC- Model Uncertainty ↑	AUROC - Total Uncertainty ↑
SGLD	STL10	0.677	0.684
	SVHN	0.948	0.945
SGHMC	STL10	0.682	0.687
	SVHN	0.949	0.955
cSGLD	STL10	0.624	0.641
	SVHN	0.966	0.968
cSGHMC	STL10	0.631	0.657
	SVHN	0.920	0.945
SWAG	STL10	0.618	0.671
	SVHN	0.878	0.908
PCA + ESS (SI)	STL10	0.673	0.677
	SVHN	0.949	0.947
MC dropout	STL10	0.665	0.695
	SVHN	0.926	0.938
SGD	STL10	N/A	0.682
	SVHN	N/A	0.892

Table 8. Comparision of Misclassification detection while using ResNet50 on CIFAR10.

Inference	AUROC- Model Uncertainty ↑	AUROC - Total Uncertainty ↑	AUROC- Model Confidence ↑	AUCPR- Model Uncertainty ↑	AUCPR - Total Uncertainty ↑	AUCPR- Model Confidence ↑
SGLD	0.945	0.949	0.950	0.422	0.468	0.480
SGHMC	0.943	0.949	0.950	0.434	0.466	0.488
cSGLD	0.927	0.943	0.946	0.321	0.355	0.382
cSGHMC	0.885	0.935	0.943	0.311	0.390	0.444
SWAG	0.890	0.927	0.927	0.418	0.479	0.472
PCA + ESS (SI)	0.932	0.934	0.941	0.391	0.419	0.472
MC dropout	0.946	0.947	0.947	0.455	0.485	0.477
SGD	N/A	0.937	0.936	N/A	0.464	0.456

Table 9. Comparison of predictive performance and decision making cost while using ResNet50 on CIFAR100.

Inference	Accuracy ↑	NLL ↓	BS ↓	ECE ↓	Decision Cost ↓
SGLD	0.751	1.079	0.364	0.107	277.500
SGHMC	0.755	1.084	0.362	0.103	272.500
cSGLD	0.804	0.711	0.272	0.019	211.900
cSGHMC	0.814	0.667	0.261	0.012	198.500
SWAG	0.735	1.221	0.400	0.135	295.200
PCA + ESS (SI)	0.761	0.920	0.335	0.032	261.200
MC dropout	0.786	1.006	0.330	0.115	233.100
SGD	0.732	1.302	0.408	0.148	303.700

Table 10. Comparison of OOD detection performance while using ResNet50 on CIFAR100.

Inference	OOD Dataset	AUROC- Model Uncertainty ↑	AUROC - Total Uncertainty ↑
SGLD	STL10	0.769	0.782
	SVHN	0.772	0.802
SGHMC	STL10	0.773	0.784
	SVHN	0.809	0.823
cSGLD	STL10	0.806	0.827
	SVHN	0.809	0.816
cSGHMC	STL10	0.804	0.832
	SVHN	0.791	0.823
SWAG	STL10	0.748	0.778
	SVHN	0.732	0.771
PCA + ESS (SI)	STL10	0.779	0.797
	SVHN	0.816	0.807
MC dropout	STL10	0.785	0.801
	SVHN	0.755	0.752
SGD	STL10	N/A	0.765
	SVHN	N/A	0.763

Table 11. Comparision of Misclassification detection while using ResNet50 on CIFAR100.

Inference	AUROC- Model Uncertainty ↑	AUROC - Total Uncertainty ↑	AUROC- Model Confidence ↑	AUCPR- Model Uncertainty ↑	AUCPR - Total Uncertainty ↑	AUCPR- Model Confidence ↑
SGLD	0.870	0.882	0.879	0.648	0.683	0.672
SGHMC	0.863	0.873	0.871	0.635	0.659	0.651
cSGLD	0.872	0.880	0.891	0.564	0.623	0.654
cSGHMC	0.873	0.879	0.890	0.572	0.593	0.626
SWAG	0.855	0.870	0.869	0.625	0.671	0.667
PCA + ESS (SI)	0.858	0.863	0.877	0.618	0.634	0.667
MC dropout	0.875	0.880	0.877	0.613	0.639	0.624
SGD	N/A	0.873	0.870	N/A	0.687	0.680

Table 12. Comparison of predictive performance and decision making cost while using WideResNet28x10 on CIFAR10.

Inference	Accuracy ↑	NLL ↓	BS ↓	ECE ↓	Decision Cost ↓
SGLD	0.965	0.113	0.054	0.004	115.800
SGHMC	0.965	0.114	0.053	0.004	112.200
cSGLD	0.967	0.104	0.050	0.006	102.900
cSGHMC	0.957	0.196	0.072	0.079	140.800
SWAG	0.919	0.260	0.121	0.028	270.000
PCA + ESS (SI)	0.951	0.177	0.082	0.054	163.100
MC dropout	0.957	0.158	0.067	0.019	149.000
SGD	0.963	0.138	0.060	0.018	117.100

Table 13. Comparison of OOD detection performance while using WideResNet28x10 on CIFAR10.

Inference	OOD Dataset	AUROC- Model Uncertainty ↑	AUROC - Total Uncertainty ↑
SGLD	STL10	0.680	0.680
	SVHN	0.951	0.963
SGHMC	STL10	0.678	0.683
	SVHN	0.956	0.967
cSGLD	STL10	0.685	0.686
	SVHN	0.968	0.974
cSGHMC	STL10	0.614	0.648
	SVHN	0.864	0.952
SWAG	STL10	0.649	0.667
	SVHN	0.914	0.943
PCA + ESS (SI)	STL10	0.663	0.673
	SVHN	0.897	0.970
MC dropout	STL10	0.672	0.688
	SVHN	0.897	0.922
SGD	STL10	N/A	0.667
	SVHN	N/A	0.963

Table 14. Comparision of Misclassification detection while using WideResNet28x10 on CIFAR10.

Inference	AUROC- Model Uncertainty ↑	AUROC - Total Uncertainty ↑	AUROC- Model Confidence ↑	AUCPR- Model Uncertainty ↑	AUCPR - Total Uncertainty ↑	AUCPR- Model Confidence ↑
SGLD	0.952	0.954	0.955	0.402	0.414	0.435
SGHMC	0.954	0.956	0.958	0.380	0.415	0.439
cSGLD	0.949	0.952	0.953	0.354	0.383	0.406
cSGHMC	0.889	0.936	0.945	0.298	0.382	0.452
SWAG	0.900	0.915	0.916	0.375	0.468	0.468
PCA + ESS (SI)	0.918	0.931	0.948	0.337	0.387	0.478
MC dropout	0.946	0.947	0.947	0.432	0.467	0.467
SGD	N/A	0.941	0.942	N/A	0.390	0.387

Table 15. Comparison of predictive performance and decision making cost while using WideResNet28x10 on CIFAR100.

Inference	Accuracy ↑	NLL ↓	BS ↓	ECE ↓	Decision Cost ↓
SGLD	0.809	0.760	0.278	0.066	204.300
SGHMC	0.798	0.815	0.292	0.076	216.200
cSGLD	0.832	0.640	0.242	0.033	179.300
cSGHMC	0.821	0.666	0.258	0.059	191.800
SWAG	0.710	1.149	0.414	0.110	316.900
PCA + ESS (SI)	0.817	0.656	0.263	0.038	196.600
MC dropout	0.798	0.846	0.293	0.081	214.300
SGD	0.806	0.785	0.280	0.046	205.100

Table 16. Comparison of OOD detection performance while using WideResNet28x10 on CIFAR100.

Inference	OOD Dataset	AUROC- Model Uncertainty ↑	AUROC - Total Uncertainty ↑
SGLD	STL10	0.797	0.822
	SVHN	0.784	0.791
SGHMC	STL10	0.799	0.814
	SVHN	0.768	0.794
cSGLD	STL10	0.791	0.846
	SVHN	0.767	0.782
cSGHMC	STL10	0.816	0.845
	SVHN	0.786	0.837
SWAG	STL10	0.732	0.753
	SVHN	0.672	0.704
PCA + ESS (SI)	STL10	0.813	0.827
	SVHN	0.760	0.814
MC dropout	STL10	0.798	0.815
	SVHN	0.642	0.645
SGD	STL10	N/A	0.820
	SVHN	N/A	0.732

Table 17. Comparision of Misclassification detection while using WideResNet28x10 on CIFAR100.

Inference	AUROC- Model Uncertainty ↑	AUROC - Total Uncertainty ↑	AUROC- Model Confidence ↑	AUCPR- Model Uncertainty ↑	AUCPR - Total Uncertainty ↑	AUCPR- Model Confidence ↑
SGLD	0.881	0.888	0.892	0.579	0.616	0.629
SGHMC	0.884	0.893	0.894	0.625	0.650	0.654
cSGLD	0.870	0.874	0.892	0.499	0.543	0.595
cSGHMC	0.854	0.865	0.888	0.516	0.553	0.609
SWAG	0.837	0.857	0.860	0.601	0.675	0.686
PCA + ESS (SI)	0.853	0.868	0.888	0.520	0.559	0.603
MC dropout	0.883	0.887	0.887	0.617	0.638	0.637
SGD	N/A	0.869	0.879	N/A	0.586	0.622