Post-hoc Loss-Calibration for Bayesian Neural Networks

Meet P. Vadera[1]   **Soumya Ghosh**[2]   **Kenney Ng**[2]   **Benjamin M. Marlin**[1]

[1]College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, MA, USA
[2]IBM Research, Cambridge, MA, USA

Abstract

Bayesian decision theory provides an elegant framework for acting optimally under uncertainty when tractable posterior distributions are available. Modern Bayesian models, however, typically involve intractable posteriors that are approximated with, potentially crude, surrogates. This difficulty has engendered loss-calibrated techniques that aim to learn posterior approximations that favor high-utility decisions. In this paper, focusing on Bayesian neural networks, we develop methods for correcting approximate posterior predictive distributions encouraging them to prefer high-utility decisions. In contrast to previous work, our approach is agnostic to the choice of the approximate inference algorithm, allows for efficient test time decision making through amortization, and empirically produces higher quality decisions. We demonstrate the effectiveness of our approach through controlled experiments spanning a diversity of tasks and datasets.

1 INTRODUCTION

Decision-making under uncertainty is a frequently encountered challenge across diverse applications. Examples include, medical diagnosis [Leibig et al., 2017], hazard alarms [Economou et al., 2016], and autonomous driving [McAllister et al., 2017]. Bayesian decision theory (BDT) provides an elegant framework for decision-making under uncertainty. Given the Bayes posterior, a set of decisions, and a utility function encoding user preferences, BDT dictates that the decision that maximizes the expected utility, where the expectation is with respect to the Bayes posterior, is optimal. While promising, it is worth noting that the optimality guarantees provided by BDT only hold when the true Bayes posterior is available.

Inspired in part by the success of deep learning, modern Bayesian models are often high dimensional, not restricted to conjugate prior-likelihood families, and almost always have intractable posterior distributions. Bayesian neural networks (BNN), the primary focus of this paper, are a prime example of such models. In BNNs, lacking tractable posterior distributions, various, at times crude surrogates are employed to approximate the posterior. Decisions that maximize the expected utility, with the expectation computed with respect to the surrogate rather than the true posterior, are not guaranteed to be optimal. This observation has engendered research into loss-calibrated inference [Lacoste-Julien et al., 2011, Cobb et al., 2018, Kuśmierczyk et al., 2019b] techniques that modify the approximate inference procedures to prefer regions of the posterior, most relevant to the decision-making task at hand. These methods, however, intricately couple posterior inference with decision-making. Such coupling poses several difficulties. Any change in the utility function necessitates recomputing the loss-calibrated posterior. Approximate inference algorithms are typically computationally expensive, and such re-computations can be computationally onerous. Moreover, updating the posterior on account of a modified utility function rather than updated prior beliefs or data is conceptually unappealing. Furthermore, under the loss-calibrated inference framework, every posterior inference algorithm requires a bespoke loss-calibrated counterpart to be developed. Such loss-calibrated variants may be challenging to develop, and no obvious counterpart may exist for a practitioner's algorithm of choice.

Inspired by these difficulties, we focus on post-hoc corrections to the posterior predictive distributions. By choosing to correct predictions, we are able to decouple posterior inference from the process of correcting decisions. As a result, our proposed method is agnostic to the choice of the inference algorithm; any off-the-shelf procedure may be used. Indeed, given a posterior approximation and an *unlabeled* calibration dataset, we do not even need to access the original training data. Together these properties signifi-

cantly expand the applicability of loss-calibrated inference approaches. Further, we use a single neural network to parameterize the posterior predictive corrections. At test time, decision-making involves a single forward pass through the network. This provides significant speed-ups over alternate loss-calibrated approaches [Cobb et al., 2018] which require expensive Monte Carlo approximations. We empirically demonstrate that our approach can support applications that require real-time decisions. We also find that the post-hoc corrections can be efficiently learned and, since they do not involve any posterior inference inexpensively adapted to changing utility functions. Finally, through careful experiments across diverse applications, we demonstrate that the aforementioned conceptual and computational benefits do not come at the expense of empirical performance. Our post-hoc correction procedure performs as well or better than competing approaches.

2 RELATED WORK

Bayesian Neural Networks Let $p(y|\mathbf{x}, \theta)$ represent the probability distribution induced by a deep neural network classifier over classes $y \in \mathcal{Y} = \{1, .., C\}$ given feature vectors $\mathbf{x} \in \mathbb{R}^D$. Given training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)|1 \leq i \leq N\}$ they are commonly learned through maximum conditional likelihood maximization. Instead of attempting to find the single (locally) optimal set of parameters θ_*, Bayesian neural networks are learned by inferring a posterior distribution $p(\theta \mid \mathcal{D}, \theta^0)$ over the unknown parameters θ given the prior $p(\theta \mid \theta^0)$ with hyper-parameters θ^0. Predictions on unseen data points $\mathbf{x}_*$ are made via the *posterior predictive distribution* which involves averaging over the uncertainty in the posterior distribution,

$$p(y_* \mid \mathbf{x}_*, \mathcal{D}, \theta^0) = \int p(y_* \mid \mathbf{x}_*, \theta) p(\theta \mid \mathcal{D}, \theta^0) d\theta. \quad (1)$$

Applying Bayesian inference to neural networks is challenging because both the posterior and the posterior-predictive distributions are intractable to compute, and require approximations. We now briefly review various approximate inference algorithms that have been used to approximate the intractable posterior, including variational inference (VI) [Jordan et al., 1999] and Markov Chain Monte Carlo (MCMC) methods [Neal, 1996, Welling and Teh, 2011].

In VI, an auxiliary distribution $q_\phi(\theta)$ is defined to approximate the true parameter posterior $p(\theta|\mathcal{D}, \theta^0)$. The variational parameters ϕ are selected to minimize the Kullback-Leibler (KL) divergence between $q_\phi(\theta)$ and $p(\theta \mid \mathcal{D}, \theta^0)$. Hinton and Van Camp [1993] did early work applying VI to neural networks. Graves [2011] and Blundell et al. [2015] later developed stochastic variants of VI that scale to modern networks. In other related work, authors Hernández-Lobato and Adams [2015], Soudry et al. [2014], Ghosh et al. [2016] explored assumed density filtering (ADF) and expectation

propagation [Minka, 2001] based approaches [Li et al., 2015, Hernandez-Lobato et al., 2016] for learning BNNs. These approaches however result in biased posterior estimates for complex posterior distributions. MCMC methods on the other hand provide sampling-based posterior approximations that are unbiased, but are often computationally more expensive to use. MCMC methods allow for drawing a correlated sequence of samples $\theta_t \sim p(\theta|\mathcal{D}, \theta^0)$ from the parameter posterior. The samples can then be used to approximate the posterior predictive distribution as a Monte Carlo average as shown in equation 2,

$$p(y|\mathbf{x}, \mathcal{D}, \theta^0) \approx \frac{1}{T} \sum_{t=1}^{T} p(y|\mathbf{x}, \theta_t), \quad \theta_t \sim p(\theta|\mathcal{D}, \theta^0). \quad (2)$$

While Hamiltonian Monte Carlo [Neal, 1996] remains the gold standard for inference in BNNs, its stochastic gradient variants [Welling and Teh, 2011, Chen et al., 2014] are popular for large networks, and we will use them extensively in this paper.

Bayesian decision theory and loss-calibrated inference Bayesian decision theory provides a framework for decision-making under uncertainty [Berger, 1988]. Under the framework, we elicit a utility function $\mathbf{u}(h, y)$, where h denotes a decision within a set of possible actions $\mathcal{A}$ and y denotes model predictions. Next, given a data point $\mathbf{x}_*$, we evaluate the expected utility (also known as the *conditional gain*), $\mathcal{G}(h \mid \mathbf{x}_*)$, for all $h \in \mathcal{A}$ using the utility function $u(.)$ and the posterior predictive distribution $p(y_*|\mathbf{x}_*, \theta, \mathcal{D})$,

$$\mathcal{G}(h \mid \mathbf{x}_*) = \int_{y_*} u(h, y_*) p(y_* \mid \mathbf{x}_*, \mathcal{D}, \theta^0) d\mathbf{y}_* \quad (3)$$

Finally, we select the optimal decision c_*, such that it maximizes the conditional gain, $c_* = \arg\max_{h \in \mathcal{A}} \mathcal{G}(h \mid \mathbf{x}_*)$. However, an important assumption in this frameworks is that we have access to the *true* posterior predictive distribution. As noted earlier, the true posterior predictive distribution is intractable for Bayesian neural networks. Rather, in practice, we only have access to a crude approximation $\tilde{p}(y_* \mid \mathbf{x}_*, \mathcal{D}, \theta^0)$. Using this approximation as an drop-in replacement to $p(y_* \mid \mathbf{x}_*, \mathcal{D}, \theta^0)$ in Eq. (3) no longer guarantees optimality of decisions, c_*.

This observation has inspired research in loss-calibrated inference. Lacoste-Julien et al. [2011] presents a variational approach for Gaussian process classification that derives from lower-bounding the log-conditional gain. To train the variational distribution, it presents an EM algorithm with closed form updates which alternates between sampling from the variational posterior and making optimal decisions under the variational posterior. Cobb et al. [2018] extends the work done by Lacoste-Julien et al. [2011] to Bayesian neural networks, and derives an objective that is a cost-penalized version of the standard evidence lower-bound

(ELBO). Both Lacoste-Julien et al. [2011] and Cobb et al. [2018] deal with only discrete values for the decisions h, Kuśmierczyk et al. [2019b] generalizes these methods to continuous decisions. Beyond variational approximations, Abbasnejad et al. [2015] present an importance sampling-based approach that encourages high utility decisions.

Finally, this paper was in part inspired by the work of Kuśmierczyk et al. [2019a]. Similarly to us, they propose corrections to model predictions instead of the posterior approximations. However, unlike us, their primary focus is on problems with low-dimensional posteriors. As a result, their methods are challenging to apply to the large BNN models considered here.

3 POST-HOC CORRECTIONS FOR POSTERIOR PREDICTIVE DISTRIBUTIONS

Going forward, we will assume that we have access to a calibration dataset $\mathcal{D}' := \{\mathbf{x}_n\}_{n=1}^N$ and that we can evaluate the posterior predictive distribution, under some approximation to the posterior, at all $\mathbf{x}_n \in \mathcal{D}'$. The log conditional gain on $\mathcal{D}'$ is,

$$\log \mathcal{G}(\mathbf{h} = \mathbf{c} \mid \mathcal{D}') =$$
$$\sum_{n=1}^N \log \int_y u(h = c_n, y_n = y) p(y_n = y \mid \mathbf{x}_n, \mathcal{D}, \theta^0) dy,$$
(4)

where $\mathbf{c} = \{c_n\}_{n=1}^N$, and $c_n = \arg\max_{h \in \mathcal{A}} \mathcal{G}(h \mid \mathbf{x_n})$.

If we had access to the *true* posterior predictive distribution, guarantees from Bayesian decision theory ensure that the decisions c_n are optimal. However, for BNNs we only have access to potentially crude approximations to the posterior and c_n are no longer guaranteed to be optimal. To address this, we introduce an utility aware correction, $q(y_n \mid \mathbf{x}_n, \lambda)$ to the (approximate) posterior predictive distribution $p(y_n \mid \mathbf{x}_n, \mathcal{D}, \theta^0)$ evaluated at $\mathbf{x}_n \in \mathcal{D}'$. The correction is parameterized by a set of learnable parameters, λ. In our experiments, we use a neural network to parameterize q and λ corresponds to the weights of that network. We observe that the log conditional gain can be expressed as a function of λ,

$$\log \mathcal{G}(\mathbf{h} = \mathbf{c} \mid \mathcal{D}'; \lambda) =$$
$$\sum_{n=1}^N \log \mathbb{E}_{q(y_n = y \mid \mathbf{x}_n, \lambda)} \left[\frac{p(y_n = y \mid \mathbf{x}_n, \mathcal{D}, \theta^0) u(h = c_n, y_n = y)}{q(y_n = y \mid \mathbf{x}_n, \lambda)} \right],$$
(5)

and is lower bounded by,

$$\mathcal{U}(\lambda, \mathbf{c}, \mathcal{D}') = \sum_{n=1}^N \mathbb{E}_{q(y_n \mid \mathbf{x}_n, \lambda)} \left[\log u(c_n, y_n) \right]$$
$$- \mathrm{KL} \left[q(y_n \mid \mathbf{x}_n, \lambda) || p(y_n \mid \mathbf{x}_n, \mathcal{D}, \theta^0) \right],$$
(6)

where the bound $\log \mathcal{G}(\mathbf{h} = \mathbf{c} \mid \mathcal{D}'; \lambda) \geq \mathcal{U}(\lambda, \mathbf{c}; \mathcal{D}')$ follows from Jensen's inequality. See Appendix A.2 for a detailed derivation. We learn $q(\cdot \mid \cdot, \lambda)$ by maximizing $\mathcal{U}(\lambda, \mathbf{c}; \mathcal{D}')$ with respect to λ and $\mathbf{c}$. Our algorithm proceeds in an coordinate ascent fashion by alternating between fixing $\mathbf{c}$ and taking a gradient step in the direction maximizing $\mathcal{U}(\lambda, \mathbf{c}; \mathcal{D}')$ with respect to λ and then fixing λ and maximizing $\mathbf{c}$. We limit our attention to finite discrete-valued decision problems prevalent in classification settings. For these problems, we are able to trivially maximize $\mathbf{c}$ given λ by enumerating the expected utility of all decisions and selecting the highest utility decision.

The variational gap between the log conditional gain and the lower bound,

$$\log \mathcal{G}(\mathbf{h} = \mathbf{c} \mid \mathcal{D}'; \lambda) - \mathcal{U}(\lambda, \mathbf{c}, \mathcal{D}')$$
$$= \sum_{\mathbf{x}_n \in \mathcal{D}'} \mathrm{KL} \left[q(y_n \mid \mathbf{x}_n, \lambda) || \frac{p(y_n \mid \mathbf{x}_n, \mathcal{D}, \theta^0) u(c_n, y_n)}{Z_n} \right],$$
(7)

where $Z_n = \mathbb{E}_{p(y_n \mid \mathbf{x}_n, \mathcal{D}, \theta^0)}[u(c_n, y_n)]$, lends further insights into the optimization problem. For a fixed $\mathbf{c}$, maximizing Eq. (6) is equivalent to minimizing the KL divergence between q and the original posterior predictive distribution scaled by the utility function, pointwise over the calibration dataset. This further highlights a key aspect of the proposed approach, it corrects the (*typically*) low-dimensional posterior predictive distribution rather than the unwieldy, high-dimensional BNN posterior. The lower bound Eq. (6) also lends itself to an intuitive interpretation. The first term guides $q(\cdot \mid \cdot, \lambda)$ to higher utility decisions while the second Kullback-Leibler divergence term encourages $q(\cdot \mid \cdot, \lambda)$ to be close to the approximate posterior predictive distribution in the KL sense.

Although nearly operational, two key challenges remain in applying the developed framework. The first stems from computational considerations necessary when working with large Bayesian models like BNNs. Posterior predictive distributions for BNNs need to be approximated via Monte Carlo simulations. Computation and storage cost of Monte Carlo approximations grow linearly with the number of samples and can be prohibitive for large networks. The other challenge stems from user preferences typically being expressed as cost functions [Berger, 1988, Kuśmierczyk et al., 2019b] rather than utility functions, and yet our development thus far has dealt exclusively with utility functions. We next describe strategies effective at alleviating both these concerns.

3.1 PRACTICAL CONSIDERATIONS

Amortized posterior predictive distribution We tackle the computational concerns associated with Monte Carlo approximations to the posterior predictive distribution by

learning an amortized approximation [Balan et al., 2015, Vadera et al., 2020]. We use the online distillation algorithm proposed by Balan et al. [2015], a special case of the general framework of Vadera et al. [2020], and distill the posterior predictive distribution into a single "student" neural network model. This algorithm aims to minimize the Kullback-Leibler (KL) divergence between $p(y_n \mid \mathbf{x}_n, \mathcal{D}, \theta^0)$ and a student network $S(y_n \mid \mathbf{x}_n, \omega)$, parameterized by ω for $\mathbf{x}_n \in \mathcal{D}'$. The online nature of this algorithm allows us to amortize the computation of posterior predictive distribution, without having to instantiate a large number of posterior samples. Once we have trained the student model, we can use it as a drop-in replacement for the posterior predictive distribution in Eq. (6),

$$\mathcal{U}^s(\lambda, \mathbf{c}, \mathcal{D}') = \sum_{n=1}^{N} \mathbb{E}_{q(y_n|\mathbf{x_n},\lambda)} \left[\log u(c_n, y_n)\right] \\ - \mathrm{KL}(q(y_n \mid \mathbf{x_n}, \lambda)||S(y_n \mid \mathbf{x_n}, \omega)). \tag{8}$$

Decision cost v/s utilities: In practical applications it is common to have user preferences encoded as decision costs rather than utilities. We follow Kuśmierczyk et al. [2019b],to translate between costs and utilities. Let us denote the decision cost function as $\ell(h, y)$, where h again denotes the decision and y denotes the predicted class. W can re-write the utility function as $u(h, y) = M - \ell(h, y)$, where $M \geq \sup_{h,y} \ell(h, y)$. By substituting this in Eq. (6) we obtain,

$$\mathcal{L}(\lambda, \mathbf{c}; \mathcal{D}') = \sum_{n=1}^{N} \mathbb{E}_{q(y_n|\mathbf{x_n},\lambda)} \left[\log\left(M - \ell(c_n, y)\right)\right] \\ - \mathrm{KL}(q(y_n|\mathbf{x_n}, \lambda)||p(y_n|\mathbf{x_n}, \mathcal{D}, \theta^0)), \tag{9}$$

and the analogous amortized variant is given by,

$$\mathcal{L}^s(\lambda, \mathbf{c}; \mathcal{D}') = \sum_{n=1}^{N} \mathbb{E}_{q(y_n|\mathbf{x_n},\lambda)} \left[\log\left(M - \ell(c_n, y)\right)\right] \\ - \mathrm{KL}(q(y_n|\mathbf{x_n}, \lambda)||S(y_n|\mathbf{x_n}, \omega)). \tag{10}$$

Further performing a first order Taylor series expansion about M [Kuśmierczyk et al., 2019b, Lacoste-Julien et al., 2011] we obtain,

$$\mathcal{L}^s(\lambda, \mathbf{c}; \mathcal{D}') \approx \sum_{n=1}^{N} \mathbb{E}_{q(y_n|\mathbf{x_n},\lambda)} \left[\log M - \frac{\ell(c_n, y)}{M}\right] \\ - \mathrm{KL}(q(y_n|\mathbf{x_n}, \lambda)||S(y_n|\mathbf{x_n}, \omega)), \tag{11}$$

Noting that $\mathbb{E}_{q(y_n|\mathbf{x_n},\lambda)}[\log M]$ is constant with respect to λ and $\mathbf{c}$, we arrive at,

$$\tilde{\mathcal{L}}^s(\lambda, \mathbf{c}; \mathcal{D}') = -\sum_{n=1}^{N} \mathbb{E}_{q(y_n|\mathbf{x_n},\lambda)} \left[\frac{\ell(c_n, y)}{M}\right] \\ - \mathrm{KL}(q(y_n|\mathbf{x_n}, \lambda)||S(y_n|\mathbf{x_n}, \omega)). \tag{12}$$

If we have access to the original posterior predictive distribution $p(y_n|\mathbf{x_n}, \mathcal{D}, \theta^0))$, the analogous objective is,

$$\tilde{\mathcal{L}}(\lambda, \mathbf{c}; \mathcal{D}') = -\sum_{n=1}^{N} \mathbb{E}_{q(y_n|\mathbf{x_n},\lambda)} \left[\frac{\ell(c_n, y)}{M}\right] \\ - \mathrm{KL}(q(y_n|\mathbf{x_n}, \lambda)||p(y_n|\mathbf{x_n}, \mathcal{D}, \theta^0))). \tag{13}$$

Our experiments maximize either $\tilde{\mathcal{L}}^s(\lambda, \mathbf{c}; \mathcal{D}')$ or $\tilde{\mathcal{L}}(\lambda, \mathbf{c}; \mathcal{D}')$ depending on the experimental setup. An algorithm giving an overview of our approach is presented in Appendix A.1.

With the description of our method complete, we reemphasize the distinct advantages provided by it. Observe that we only require that we are either able to evaluate the posterior predictive distribution or an amortized approximation to it on $\mathcal{D}'$. We remain agnostic and make no assumptions about how the posterior or the posterior predictive distributions were computed. Moreover learning the corrections, $q(\cdot \mid \cdot, \lambda)$, involves optimizing Eq. (12) or Eq. (13) and is no more expensive than training standard deep neural networks. Finally, at a test point $\mathbf{x}_*$ the expected cost associated with a decision h is $\sum_{k=1}^{C} \ell(h, y = k)q(y = k \mid x_*, \lambda)$. Computing this expected cost involves a *single* forward pass through q. Our framework, thus, amortizes test time decision-making. This leads to significant speed-ups over existing loss-calibrated inference approaches, which must first compute the posterior predictive distribution by performing an expensive Monte Carlo integration over the corrected posterior before making decisions. Test time amortization allows our method to be used in applications that demand real-time decision-making. We summarize our approach's similarities and differences to relevant work in Table 1.

Table 1: **Overview of related loss-calibrated methods**. *Inference Agnostic*: Method does not modify or make assumptions about the posterior inference algorithm. *Scalable*: Method scales to modern Bayesian neural networks learned from large data. *Amortized Decisions*: Method does not require multiple forward passes for test time decision-making.

	Inference Agnostic	Scalable	Amortized Decisions
Lacoste-Julien et al. [2011]	✗	✗	✗
Cobb et al. [2018]	✗	✓	✗
Kuśmierczyk et al. [2019b]	✗	✓	✗
Kuśmierczyk et al. [2019a]	✓	✗	✓
Ours	✓	✓	✓

4 EXPERIMENTS

In this section, we carefully vet our proposed method against relevant baselines across a diverse range of applications.

Broadly, we divide our experiments into three major categories to target three practical scenarios — 1) Decision-making in poor data quality regimes, 2) Decision-making with a reject option (also known as *selective classification*), 3) Decision-making under real-time constraints.

Throughout this section, we experiment with fully-connected and ResNet18 [Krizhevsky et al., 2012] architectures. We test our methods on data from MNIST [LeCun, 1998], CIFAR10 [Krizhevsky et al., 2009], and the challenging CamVid [Brostow et al., 2008b] dataset. We demonstrate that our posterior correction consistently improves the quality of decisions when used in conjunction with popular BNN inference algorithms — black-box variational inference (BBVI) [Blundell et al., 2015], stochastic gradient Hamiltonian Monte-Carlo (SGHMC) [Chen et al., 2014], and Kronecker-factored Laplace approximation (KFAC-Laplace) [Ritter et al., 2018]. Noting that SGHMC [Yao et al., 2019] typically provides a more faithful approximation to the BNN posterior, we restrict ourselves to SGHMC for real data experiments. To compare against a loss-calibrated inference procedure, we develop a loss-calibrated variant of SGHMC. Following Lacoste-Julien et al. [2011] we define the following utility scaled posterior,

$$\tilde{p}(\theta|\mathcal{D}, u) \propto p(\theta|\mathcal{D})\mathcal{G}(\mathbf{h}^*|\mathcal{D}). \qquad (14)$$

The loss-calibrated stochastic gradient HMC (LC-SGHMC) algorithm then proceeds by sampling from this scaled posterior using SGHMC. Given that SGHMC is typically more accurate than competing variational methods, we view LC-SGHMC as a strong loss-calibrated inference baseline. We also provide a brief overview of SGHMC in Appendix A.4.

4.1 SYNTHETIC DATA EXPERIMENTS

We begin with experiments on synthetic data, employing fully connected architectures and three popular inference techniques, BBVI with local reparameterizations [Kingma et al., 2015], SGHMC, and KFAC-Laplace.

Experimental setup: We construct a two-dimensional, two class data set with class imbalance. We generate data from the two classes by sampling isotropic Gaussian distributions with means $[-1, -1]$ and $[+1, +1]$. For training, we use 90 data instances from the negative class and 10 data instances from the positive class. We resample the Gaussian distributions to create a test set, which again contains 90 negative examples and 10 positive ones. For calibration data, we uniformly sample the two dimensional space to generate 500 unlabeled data instances. We repeat this procedure ten times and generate ten training and calibration datasets. Fig. 1 visualizes one of these ten datasets. For each dataset, we learn a 50 unit, single hidden layer, multi-layer perceptron with ReLU activations using BBVI, SGHMC, and KFAC-Laplace and use a 100 sample Monte Carlo approximation
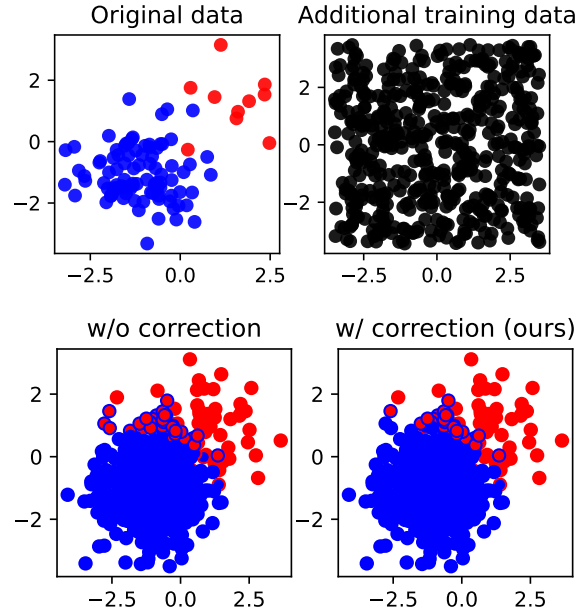

Figure 1: **Synthetic Data (Top)**. Labeled training data ($\mathcal{D}$) is shown in the left plot and unlabeled calibration data ($\mathcal{D}'$) is shown in the right plot. Blue markers represent the negative class and red markers represent the positive class. **Decision Visualization (Bottom)**. A visual comparison of decision-making using BBVI without correction, and with our correctional approach for a single trial, on test data drawn from the same distribution as the training data $\mathcal{D}'$. The edge colors indicate ground truth classes, while the face colors indicate the predicted classes. Consistency between edge and face colors indicate correct predictions.

to compute the corresponding posterior predictive distributions. We employ the following decision-cost function,

$$\ell(c, y) = \begin{cases} 0, & \text{for } y = c \\ 1, & \text{for } y \neq c, y = \text{positive} \\ 0.1, & \text{for } y \neq c, y = \text{negative} \end{cases},$$

which encourages decisions that minimize false negative errors for the minority class — often a desirable property in practice. In this experiment, we learn the corrections by maximizing Eq. (13). Additional experimental details can be found in Appendix A.5.

Results: We compute the average decision cost under our cost-function for each of the three inference algorithms with and without our post-hoc correction on the test set. Table 2 summarizes our results, where the error bars stem from having repeated the experiment on the ten randomly generated training and calibration datasets. Our post-hoc correction results in test decisions with lower decision costs compared to the decisions produced by the uncorrected variants. The costs are only marginally lower in this synthetic example, where the approximations to the posterior are likely already

good. In the following, we will see that the decision costs can be substantially lower in more challenging scenarios. In subsequent experiments, we solely rely on stochastic gradient HMC algorithms for approximating the Bayesian neural network posterior. A full panel of results is presented in Appendix A.6.

Table 2: **Results on synthetic data**. Test decision costs with and without post-hoc correction over 10 replicates. Post-hoc correction consistently provides lower cost decisions. Results presented as mean $\pm$ std. dev.

	W/O post-hoc correction	W/ post-hoc correction (ours)
VI	0.019 ± 0.011	0.016 ± 0.010
SGHMC	0.018 ± 0.008	0.017 ± 0.009
KFAC-Laplace	0.021 ± 0.007	0.018 ± 0.008

4.2 SELECTIVE CLASSIFICATION

Next, we consider the problem of selective classification, wherein the goal is to classify a data instance into one of C classes or choose not to classify and instead refer it to an oracle. The corresponding decision problem thus involves selecting one of $C + 1$ decisions for each data instance. By adjusting the cost of a referral the decision making system can trade erroneous decisions for potentially expensive oracle feedback. Different users of such a system will likely prefer different trade-offs and as a result choose different referral costs. We however have no reason to believe that the different users would have different posterior beliefs. Since our method does not involve relearning the posterior beliefs when faced with changing cost functions, it is well suited for such selective classification problems.

Experimental setup: We use the CIFAR10 [Krizhevsky et al., 2009] dataset along with SGHMC trained Bayesian ResNet18 [He et al., 2016] networks. To make the problem more challenging and encourage referrals, we contaminate the data via an additional data transformation. Under this contamination, we subject each image in the dataset by angle sampled uniformly at random from $[-30°, 30°]$. Next, following Murphy [2012] (section 5.7.1.2), we define our selective classification decision cost function as,

$$\ell(c, y) = \begin{cases} 0, & \text{for } y = c \\ 1, & \text{for } y \neq c \\ r, & \text{for } c = \text{referral.} \end{cases}$$

Here, r denotes the cost of a referral. With this setup we examine a) the effectiveness of our method as a function of r, b) whether using the amortized posterior predictive distribution S (maximizing Eq. (12)) adversely affects performance when compared to the non-amortized version (maximizing Eq. (13)), and c) the computational cost of learning

post-hoc corrections under multiple decision-cost functions. Additional details around models, training procedures, hyperparameters, baselines, and cost function is given in the Appendix A.6.

Results: The results for the experiment are presented in Fig. 2. As we would expect, lower values of the referral cost r lead to more referrals and as a result, the models tend to make a decision only when they are very confident, leading to higher values of accuracy. Lower values of r also result in lower average decision cost, as the models tend to refer to the oracle more, and due to the lower value of referral cost, the average decision cost reduces. As we can see from the comparison, the decision cost as well as accuracy of our post-hoc corrections outperforms those obtained using SGHMC and LC-SGHMC across all values of referral cost r. Furthermore, we observe that the accuracy and decision cost values are not adversely affected by using S in place of the non-amortized posterior predictive distribution. We also compare the training time of LC-SGHMC and our post-hoc corrections as a function of the number of decision-cost functions. The wall-clock times required by the two approaches are shown in Fig. 3. All the experiments were run using the same GPU hardware (Nvidia Tesla V100) and under identical conditions for consistency in wall-clock time comparisons. We observe that our approach is significantly faster to train, and its computational cost grows at a slower rate with increasing number of cost functions.

Finally, the fact that the amortized posterior predictive distribution S does not adversely affect performance, is significant, in that it suggests that our approach would likely continue to scale with increasing network size, when storing and averaging over multiple Monte Carlo samples to compute the posterior predictive distribution is the bottleneck. Moving forward, we only consider experiments with the amortized posterior predictive S.

4.3 DECISION MAKING UNDER POOR DATA QUALITY

In the current era of big-data, it is not uncommon to have data sets of poor quality. In many settings, the data sets are labeled by crowdsourcing, as well as other automated techniques. These labeling techniques can often lead to noisy labels, and affecting downstream performance. Thus, it is important to understand how our method performs in this practical scenario. In practical scenarios, it is common to have asymmetric decision cost functions. This means that for making certain incorrect decisions, the cost can be higher or lower than the rest to encourage or discourage making those decisions. In this experiment, we also incorporate an asymmetric cost function similar to the one introduced earlier in the synthetic data experiments.

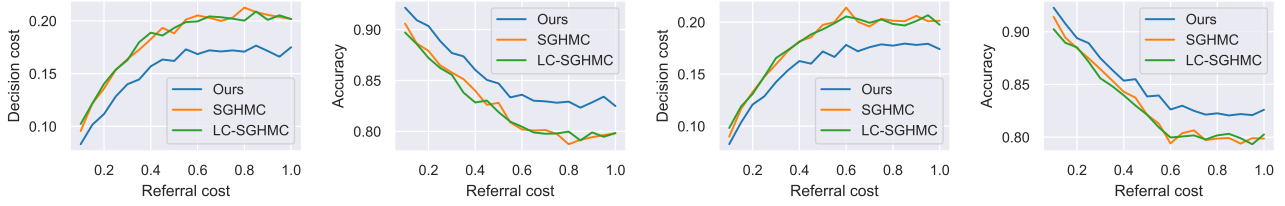**Experimental setup:** We simulate label corruption on

Figure 2: **Selective classification**. Average test decision costs and test accuracy as a function of referral cost on CIFAR10 using a Bayesian ResNet18 model. The two left plots show results without using an amortized posterior predictive distribution, S, and the two plots on the right display results when using S. Accuracy is determined on those test data points that are not referred to the oracle. We note that introducing the amortized approximation S does not adversely affect performance. Additional results involving negative log likelihood are presented in Appendix A.6.
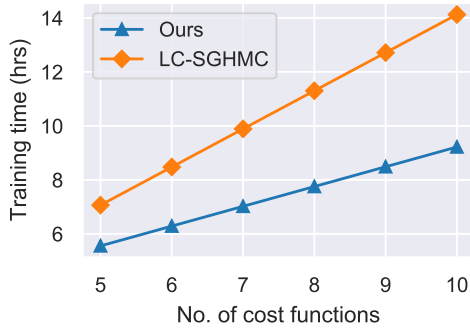
Figure 3: **Training cost comparison**. Wall-clock training time of our method and LC-SGHMC as a function of the number of decision cost functions.

MNIST [LeCun, 1998] and CIFAR10 [Krizhevsky et al., 2009]. For each dataset, we switch the true labels of a proportion of the training set to labels sampled uniformly at random. For MNIST, we use a simple multi-layer perceptron architecture with one hidden layer of 200 units, while for CI-FAR10, we use the ResNet18 architecture [He et al., 2016]. It is worth noting that SGLD can be derived special case for the SGHMC (refer to Appendix A.4). For each data set, we pick two classes to which we assign higher importance, and thus assigning a lower cost to the mistakes which involve choosing these classes as decisions. In MNIST, we assign a higher importance to classes 3 and 8, while in CIFAR10, we assign a higher importance to classes automobile and trucks. We use SGLD [Welling and Teh, 2011] for sampling from the posterior as well as the utility scaled posterior distribution. For a point-estimated model baseline, we introduce the class-weighted SGD (CW-SGD) baseline. In CW-SGD, we use our standard log loss on the neural network model, but assign a higher weight to the classes of interest. This encourages the model to make lesser mistakes on classes of higher importance. For additional details around models, training procedures, hyperparameters, and baselines, please refer to Appendix A.7.

Results: We present the results of this experiment in Fig-

ure 4. For a comprehensive evaluation of performance, we vary the label corruption proportion between 0.3 and 0.7. For performance assessment, we look at the decision cost ($\downarrow$), and accuracy ($\uparrow$) on the standard test sets for each data set. While looking across the set of performance metrics, similar trends emerge. For lower levels of corruption, we notice that our post-hoc correction method performs similarly to LC-SGHMC and LC-SGLD, and marginally better than the uncorrected posterior predictive distribution and CW-SGD. However, as we increase the label corruption proportion to moderate levels, we observe that our method outperforms the baselines. Finally, with increasing corruption proportion, all methods are overwhelmed by the label noise, and we see a sharp dip in performance across all methods. We note that beyond achieving similar or lower decision costs, our approach also achieves higher accuracy and negative log-likelihoods than the competing methods. With these encouraging results in mind, we move towards our final experiment which looks at a real-world data set and demonstrates the benefits of amortized decision making.

4.4 SEMANTIC SCENE SEGMENTATION

In this experiment, we consider the problem of semantic scene segmentation. Here, our goal is to segment an image into its components. This is achieved by labeling each pixel of an image with one of C known categories. Semantic segmentation can be useful for a variety of applications including aiding autonomous vehicles navigate the world. In such an application, it is crucial that the underlying decision problem of labeling pixels be solvable in near real-time. Existing loss-calibrated approaches struggle with such real-time requirements. through this experiment, we demonstrate both real time performance and improved decisions provided by our post-hoc loss correction framework.

Experimental setup: For this experiment, we use the Camvid data set [Brostow et al., 2008a,b] which contains per-pixel labeled images captured using a camera on the dashboard of a car moving on the streets. An illustration of the data is provided in Figure Fig. 5. The version of the data
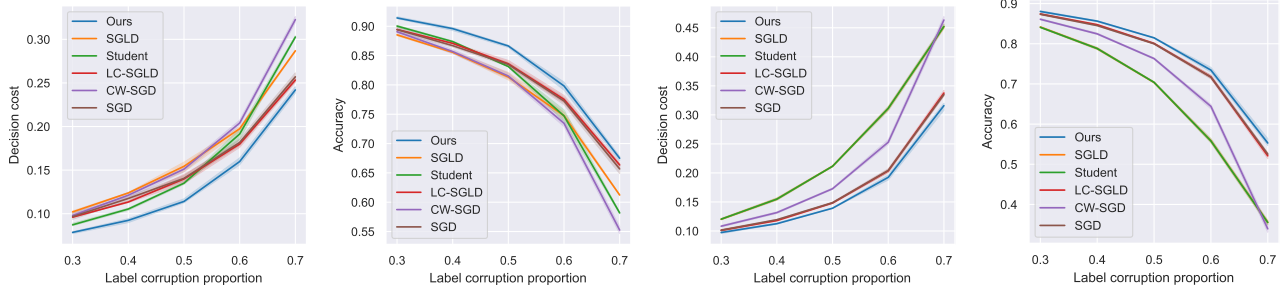
Figure 4: **Decision-making under label corruption.** The two left plots illustrate performance comparison for different levels of label corruption on CIFAR10 using a Bayesian ResNet18 model. The right two plots illustrate performance comparison for different levels of label corruption on MNIST using an MLP with a single hidden layer of 200 hidden units. The results are shown as mean $\pm$ std. dev. over 5 trials. Additional results involing negative log likelihood are presented in Appendix A.7.
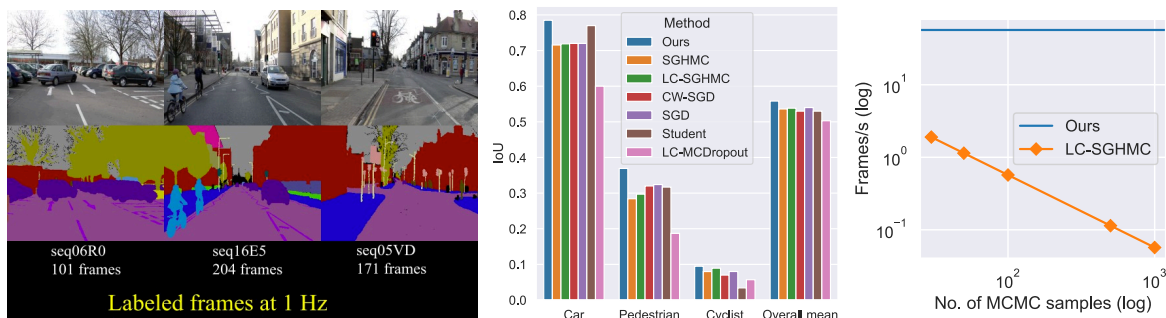

Figure 5: **Semantic Segmentation** (Left) Sample input images (top) and ground truth segmentations (bottom) from the CamVid dataset. (Middle) IoU scores acheived by different methods for classes deemed important by the cost function as well as overall mean IoU across all classes and all images. (Right) We compare the number of frames processed per second by our method and LC-SGHMC. Owing to amortization our approach is independent of the number of samples used to compute the Monte Carlo approximation to the posterior predictive distribution. In contrast, the number of frames processed per second by LC-SGHMC decreases dramatically (note the log scaling of the axes) with increasing number of samples.

set that we use, contains a total of 12 class labels. Of these class labels, we assign a lower cost to false decisions that involve picking either of the pedestrian, cyclist or car class. This decision cost structure is inspired from the experiments of Cobb et al. [2018], as the goal of an autonomous car is to avoid these obstacles for safety reasons. By assigning a lower penalty to an incorrect decision of classifying a pixel to one of these three classes, we encourage our model to be more risk averse. For our model, we use the *SegNet* model architecture [Badrinarayanan et al., 2017]. SegNet is an auto-encoder style model which has previously been used for semantic scene segmentation. For additional comparison, we add the loss calibrated MC dropout baseline based on the algorithm presented in Cobb et al. [2018]. We defer additional details around training procedures, hyperparameters, baselines and decision cost matrix to Appendix A.8.

Results: The aim of this experiment is two-fold. First, we want to evaluate the performance of our approach for the task, and compare it with relevant baselines. Secondly, we want to assess the test-time efficiency of our approach

against alternates.

For assessing performance, we use the intersection over union (IoU) ($\uparrow$) metric that is commonly used to assess semantic segmentation performance. This metric evaluates the ratio of the area of overlap and the area of union while comparing the ground truth segmentation and model output segmentation for each class. In Fig. 5, we present a performance comparison between our method and the baselines using the IoU metrics on test set. For the comparison, we look at each of the high utility classes separately, as well as we evaluate the overall mean IoU over all classes combined. We observe that our method performs consistently better on all the high utility classes, and thus doing a better job at capturing the preferences embedded in our cost function. Moreover, the mean IoU across all the classes indicates that our method does a better job overall for this task.

Next, we look at assessing test-time efficiency for our current task. The metric used for quantifying the test time efficiency is the number of frames that can be processed

1410

per second (FPS, ↑). For application such as autonomous driving, a slower processing pipeline can create a bottleneck when it comes to the efficacy for deployments. However, since we use a point-estimated model for our approach, it gives us inherent time savings when looking at test-time processing capabilities. In Figure 5, we present a comparison of no. of frames/sec that can be processed between our approach and LC-SGHMC. Each frame (image) in the data set has a resolution of 360×480. We compute the time to process an image after loading both the image and model (or model ensemble for LC-SGHMC) on a Nvidia Tesla V100 GPU. While we expect monte carlo based approximations to perform much slower than our point-estimated model, it is increasingly evident looking at Figure 5 (middle) that the number of frames/s can be inhibitively low for practical applications, for even a smaller number of MCMC samples. While the performance for MCMC methods in Figure 5 is computed using 30 samples, we compute the frames/s metric over larger ensemble sizes to give a sense of why Monte Carlo integration at test time can be impractical.

To summarize, there are two key findings of this experiment. First, our approach leads to improved decision making by capturing class preferences better, and improves on the overall performance averaging across all twelve classes. Second, our method is better positioned for deployments requiring real-time performance as at decision time it requires a single forward pass and no Monte Carlo approximations.

5 DISCUSSION AND FUTURE WORK

In this paper we introduced a novel framework for post-hoc loss calibration of Bayesian neural networks for decision-making. Through comprehensive empirical evaluations ranging from synthetic data sets to practical applications involving real world data, we have demonstrated that our approach consistently produces lower cost, higher utility decisions than competing approaches. We also demonstrated that the framework by decoupling posterior inference from decision-making provides computational advantages at training time, and through amortization provides fast test-time decisions. Future directions include extensions to continuous decisions, more carefully exploring the effect of the choice of an inference algorithm on the quality of the correction, exploring post-hoc corrections under distribution shift, and studying the connections to generalized Bayesian inference [Bissiri et al., 2016, Knoblauch et al., 2019].

References

Ehsan Abbasnejad, Justin Domke, and Scott Sanner. Loss-calibrated monte carlo action selection. In *AAAI*, 2015.

Vijay Badrinarayanan, Alex Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:2481–2495, 2017.

Anoop Korattikara Balan, Vivek Rathod, Kevin P Murphy, and Max Welling. Bayesian dark knowledge. In *NeurIPS*, 2015.

J. O. Berger. Statistical decision theory and bayesian analysis. 1988.

Pier Giovanni Bissiri, Chris C Holmes, and Stephen G Walker. A general framework for updating belief distributions. *Journal of the Royal Statistical Society. Series B, Statistical methodology*, 78(5):1103, 2016.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv:1505.05424*, 2015.

Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, xx (x):xx–xx, 2008a.

Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*, pages 44–57, 2008b.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *ICML*, pages 1683–1691, 2014.

Adam D. Cobb, Stephen J. Roberts, and Yarin Gal. Loss-calibrated approximate inference in bayesian neural networks. In *ICML theory of deep learning workshop*, 2018.

T. Economou, D. Stephenson, J. Rougier, R. Neal, and K. Mylne. On the use of bayesian decision theory for issuing natural hazard warnings. *Proceedings. Mathematical, Physical, and Engineering Sciences*, 472, 2016.

Soumya Ghosh, Francesco Delle Fave, and Jonathan Yedidia. Assumed density filtering methods for learning bayesian neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Alex Graves. Practical variational inference for neural networks. In *NeurIPS*, 2011.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, pages 770–778, 2016.

Jose Hernandez-Lobato, Yingzhen Li, Mark Rowland, Thang Bui, Daniel Hernández-Lobato, and Richard Turner. Black-box alpha divergence minimization. In *International Conference on Machine Learning*, pages 1511–1520. PMLR, 2016.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *ICML*, 2015.

Geoffrey Hinton and Drew Van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *COLT*, 1993.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2): 183–233, 1999.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *NeurIPS*, 2015.

Jeremias Knoblauch, Jack Jewson, and Theodoros Damoulas. Generalized variational inference: Three arguments for deriving new posteriors, 2019.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

Tomasz Kuśmierczyk, Joseph Sakaya, and Arto Klami. Correcting predictions for approximate bayesian inference. *arXiv preprint arXiv:1909.04919*, 2019a.

Tomasz Kuśmierczyk, Joseph Sakaya, and Arto Klami. Variational bayesian decision-making for continuous utilities. In *NeurIPS*, pages 6395–6405, 2019b.

Simon Lacoste-Julien, Ferenc Huszár, and Zoubin Ghahramani. Approximate inference for the loss-calibrated bayesian. In *AISTATS*, pages 416–424, 2011.

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

Christian Leibig, Vaneeda Allken, M. Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific Reports*, 7, 2017.

Yingzhen Li, Jose Miguel Hernández-Lobato, and Richard E Turner. Stochastic expectation propagation. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 2323–2331, 2015.

Rowan McAllister, Yarin Gal, Alex Kendall, M. V. D. Wilk, A. Shah, R. Cipolla, and Adrian Weller. Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. In *IJCAI*, 2017.

Thomas P Minka. Expectation propagation for approximate Bayesian inference. In *UAI*, 2001.

K. Murphy. Machine learning - a probabilistic perspective. In *Adaptive computation and machine learning series*, 2012.

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

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *ICLR*, volume 6, 2018.

Daniel Soudry, Itay Hubara, and Ron Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *NeurIPS*, 2014.

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

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

Jiayu Yao, Weiwei Pan, S. Ghosh, and Finale Doshi-Velez. Quality of uncertainty quantification for bayesian neural network inference. In *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2019.