Collapsed Inference for Bayesian Deep Learning

Zhe Zeng [1] **Guy Van den Broeck** [1]

Abstract

Bayesian neural networks (BNNs) provide a formalism to quantify and calibrate uncertainty in deep learning. Current inference approaches for BNNs often resort to few-sample estimation for scalability, which can harm predictive performance, while its alternatives tend to be computationally prohibitively expensive. We tackle this challenge by revealing a previously unseen connection between inference on BNNs and *volume computation* problems. With this observation, we introduce a novel collapsed inference scheme that performs Bayesian model averaging using *collapsed samples*. It improves over a Monte-Carlo sample by limiting sampling to a subset of the network weights while pairing it with some closed-form conditional distribution over the rest. A collapsed sample represents uncountably many models drawn from the approximate posterior and thus yields higher sample efficiency. Further, we show that the marginalization of a collapsed sample can be solved analytically and efficiently despite the non-linearity of neural networks by leveraging existing volume computation solvers. Our proposed use of collapsed samples achieves a balance between scalability and accuracy. On various regression and classification tasks, our collapsed Bayesian deep learning approach demonstrates significant improvements over existing methods and sets a new state of the art in terms of uncertainty estimation and predictive performance.

1. Introduction

Uncertainty estimation is crucial for decision making. Deep learning models, including those in safety-critical domains, tend to poorly estimate uncertainty. To overcome this issue, Bayesian deep learning obtains a posterior distribution over the model parameters hoping to improve predictions and provide reliable uncertainty estimates. Among Bayesian inference procedures with neural networks, Bayesian model averaging (BMA) is particularly compelling (Wasserman, 2000; Fragoso et al., 2018; Maddox et al., 2019). However, computing BMAs is distinctly challenging since it involves marginalizing over posterior parameters, which possess some unusual topological properties such as mode-connectivity (Izmailov et al., 2021). We show that even with simple low-dimensional approximate parameter posteriors as uniform distributions, doing BMA requires integrating over highly *non-convex* and *multi-modal* distributions with discontinuities arising from non-linear activations (cf. Figure 1a). Accurately approximating the BMA can achieve significant performance gains (Izmailov et al., 2021). Existing methods mainly focus on general-purpose MCMC, which can fail to converge, or provides inaccurate few-sample predictions (Kristiadi et al., 2022), because running longer sampling chains is computationally expensive, and variational approaches that typically assume mean-field and ignore correlations induced by activations (Jospin et al., 2022).

In this work, we are interested in developing *collapsed samplers*, also known as *cutset* or *Rao-Blackwellised* samplers for BMA. A collapsed sampler improves over classical particle-based methods by limiting sampling to a subset of variables and further pairing each sample with a closed-form representation of a conditional distribution over the rest whose marginalization is often tractable. Collapsed samplers are effective at variance reduction in graphical models (Koller & Friedman, 2009), however no collapsed samplers are known for Bayesian deep learning. We believe that this is due to the lack of a closed-form marginalization technique congruous with the non-linearity in deep neural networks. Our aim is to overcome this issue and improve BMA estimation by incorporating exact marginalization over (close approximate) conditional distributions into the inference scheme. Nevertheless, scalability and efficiency is guaranteed by the sampling part of our proposed algorithm.

Marginalization is made possible by our observation that BMA reduces to weighted volume computation. Certain classes of such problems can be solved exactly by so-called weighted model integration (WMI) solvers (Belle et al., 2015a). By closely approximating BMA with WMI, these solvers can provide accurate approximations to marginal-

---

[1]Computer Science Department, University of California, Los Angeles. Correspondence to: Zhe Zeng <zhezeng@cs.ucla.edu>.
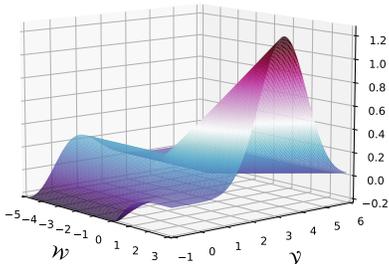
(a) $p(y \mid \boldsymbol{x}, \boldsymbol{w})$ being Gaussian.
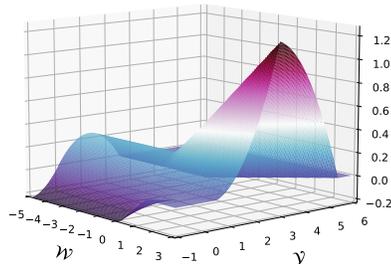
(b) $p(y \mid \boldsymbol{x}, \boldsymbol{w})$ being triangular.

Figure 1: The integral surface of (a) the expected prediction in BMA, and (b) our proposed approximation. Both are highly non-convex and multi-modal. The z-axis is the weighted prediction $y\, p(y \mid \boldsymbol{x}, \boldsymbol{w})\, p(\boldsymbol{w} \mid \mathcal{D})$. Integration of (a) does not admit a closed-form solution, yet integration of (b) is a close approximation that can be solved exactly by WMI solvers.

ization in BMA (cf. Figure 1b). With this observation, we propose CIBER, a collapsed sampler that uses WMI for computing conditional distributions. In the few-sample setting, CIBER delivers more accurate uncertainty estimates than the gold-standard Hamiltonian Monte Carlo (HMC) method (cf. Figure 2). We further evaluate the effectiveness of CIBER on regression and classification tasks and show significant improvements over other Bayesian deep learning approaches in terms of uncertainty estimation and accuracy.

2. BMA as WVC

In **Bayesian Neural Networks (BNN)**, given a neural network $f_{\boldsymbol{w}}$ parameterized by weights $\boldsymbol{w}$, instead of doing inference with deterministic $\boldsymbol{w}$ that optimize objectives such as cross-entropy or mean squared error, Bayesian learning infers a posterior $p(\boldsymbol{w} \mid \mathcal{D})$ over parameters $\boldsymbol{w}$ after observing data $\mathcal{D}$. During inference, this posterior distribution is then marginalized over to produce final predictions. This process is called **Bayesian Model Averaging (BMA)**. It can be seen as learning an ensemble of an infinite number of neural nets and aggregating results. Formally, the posterior predictive and the expected prediction for regression are

$$p(y \mid \boldsymbol{x}) = \int p(y \mid \boldsymbol{x}, \boldsymbol{w})\, p(\boldsymbol{w} \mid \mathcal{D})\, d\boldsymbol{w},$$

$$\text{and} \ \ \mathbb{E}_{p(y \mid \boldsymbol{x})}[y] = \int y\, p(y \mid \boldsymbol{x})\, dy. \tag{1}$$

For classification, the (most likely) prediction is the class $\arg\max_y p(y \mid \boldsymbol{x})$. BMA is intuitively attractive because it can be risky to base inference on a single neural network model. The marginalization in BMA gets around this issue by averaging over models according to a Bayesian posterior.

BMA requires approximations to compute posterior predictive distributions and expected predictions, as the integrals

in Equation 1 are intractable in general. Deriving efficient and accurate approximations remains an active research topic (Izmailov et al., 2021). We approach this problem by observing that the marginalization in BMA with ReLU neural networks can be cast as weighted volume computation (WVC). Later we show that it can be generalized to any neural networks when combined with sampling. In WVC, various tools exist for solving certain WVC problem classes (Baldoni et al., 2014; Kolb et al., 2019; Zeng et al., 2020c). This section reveals the connection between BMA and WVC. It opens up a new perspective for developing BMA approximations by leveraging WVC tools.

Definition 1 (WVC). *A weighted volume computation problem is a pair $(\boxdot, \phi)$ where a region $\boxdot$ is a conjunction of arithmetic constraints and weight $\phi : \boxdot \to \mathbb{R}$ is an integrable function assigning weights to elements in $\boxdot$. The task of WVC is to compute the integral $\int_{\boxdot} \phi(\boldsymbol{x})\, d\boldsymbol{x}$.*

2.1. A General Reduction of BMA to WVC

Let model $f_{\boldsymbol{w}}$ be a ReLU neural net. Denote the set of inputs to its ReLU activations by $\mathcal{R} = \{r_i\}_{i=1}^{R}$, where each r_i is a linear combination of weights. For a given input $\boldsymbol{x}$, the parameter space is partitioned by whether each ReLU activation outputs zero or not. This gives the WVC reduction

$$p(y \mid \boldsymbol{x}) = \sum_{\boldsymbol{B} \in \{0,1\}^R} \int_{\boxdot_{\boldsymbol{B}}} p(y \mid \boldsymbol{x}, \boldsymbol{w})\, p(\boldsymbol{w} \mid \mathcal{D})\, d\boldsymbol{w},$$

where $\boldsymbol{B}$ is a binary vector. The region $\boxdot_{\boldsymbol{B}}$ is defined as $\wedge_{i=1}^{R} \ell_i$ where arithmetic constraint ℓ_i is $r_i \geq 0$ if $\boldsymbol{B}_i = 1$ and $r_i \leq 0$ otherwise. The expected prediction $\mathbb{E}_{p(y \mid \boldsymbol{x})}[y]$ is analogous but includes an additional factor and variable of integration y in each WVC problem.

This general reduction, however, is undesirable since it

(a) HMC with 10 samples

(b) CIBER with 10 collapsed samples

(c) HMC with $2k$ samples
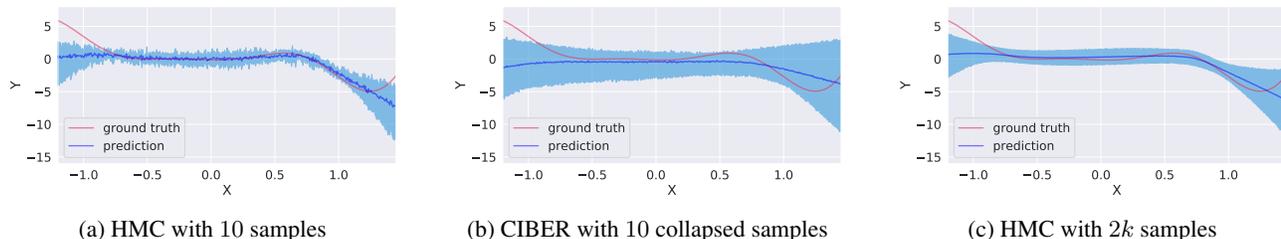
Figure 2: Uncertainty estimates for regression. Red line is the ground truth. Dark blue line shows predictive mean. Shaded region is the 90% confidence interval of the predictive distribution. For the same number of samples, (b) our proposed CIBER is closer than (a) small-sample HMC to a highly accurate but slow (c) HMC with a large number of samples.

amounts to a brute-force enumeration that implies a complexity exponential in the number of ReLU activations. Moreover, not all WVC problems resulting from this reduction are amenable to existing solvers. We therefore appeal to a framework called weighted model integration (WMI) that allows for a compact representation and a characterization of their tractability for WMI solvers.

2.2. Approximating BMA by WMI

WMI is a modeling and inference framework that supports integration in the presence of logical and arithmetic constraints (Belle et al., 2015a;b) and various WMI solvers have been proposed in recent years (Kolb et al., 2019). However, even with the reduction from BMA to WVC shown above, WMI solvers are not directly applicable due to two main limitations of existing solvers: (i) feasible regions need to be defined by Boolean combinations of linear arithmetic constraints, and (ii) weight functions need to be polynomials. Next, we show how these issues can be bypassed.

In WMI, the feasible region is defined by *satisfiability modulo theories* (SMT) constraints (Barrett et al., 2010): an SMT formula is a (typically quantifier-free) expression containing both propositional and theory literals connected with logical connectives; the theory literals are often restricted to *linear real arithmetic*, where literals are of the form $(\mathbf{c}^T \mathbf{X} \leq b)$ with variable $\mathbf{X}$ and constants $\mathbf{c}^T$ and b.

Definition 2. *(WMI) Let $\mathbf{X}$ be a set of continuous random variables. A weighted model integration problem is a pair $\mathcal{M} = (\Delta, \Phi)$, where Δ is an SMT formula over $\mathbf{X}$ and Φ is a set of per-literal weights defined as $\Phi = \{\phi_\ell\}_{\ell \in \mathcal{L}}$, where $\mathcal{L}$ is a set of SMT literals and each ϕ_ℓ is a function defined over variables in literal ℓ. The task of weighted model integration is to compute*

$$\mathsf{WMI}(\Delta, \Phi) = \int_{\boldsymbol{x} \models \Delta} \prod_{\ell \in \mathcal{L}} \phi_\ell(\boldsymbol{x})^{[\![\boldsymbol{x} \models \ell]\!]} \, d\boldsymbol{x}.$$

where $\boldsymbol{x} \models \Delta$ denotes the satisfaction of an SMT formula Δ by $\boldsymbol{x}$, and $[\![\boldsymbol{x} \models \Delta]\!]$ be its corresponding indicator function.

To bypass issue (i), we propose to use the encoding of ReLU

neural networks into SMT formulas to define the feasible region of WMI problems. This encoding has been explored in existing work to enable verification of the behaviour of neural networks and provide formal guarantees (Katz et al., 2017; Huang et al., 2017; Sivaraman et al., 2020). *To bypass issue (ii),* we propose to encode both the posterior distribution and the predictive distribution using polynomial densities. Next, we show how this process can be generalized to a scalable and accurate approximation of BMA.

3. CIBER: Collapsed Inference for Bayesian Deep Learning via WMI

Given a BNN with a large number of weights, naively approximating it by WMI problems can lead to computational issues, since it involves doing integration over polytopes in arbitrarily high dimensions and this is known to be #P-hard (Valiant, 1979; De Loera et al., 2012; Zeng et al., 2020c). Further, weights involved with non-ReLU activation might not be amenable to the WMI encoding. To tackle these issues, we propose to use collapsed samples to combine the strengths from two worlds: the scalability and flexibility from sampling and the accuracy from WMI solvers.

Definition 3. *(Collapsed BMA) Let $(\boldsymbol{W}_s, \boldsymbol{W}_c)$ be a partition of parameters $\boldsymbol{W}$. A collapsed sample is a tuple $(\boldsymbol{w}_s, q)$, where $\boldsymbol{w}_s$ is an assignment to the sampled parameters $\boldsymbol{W}_s$ and q is a representation of the conditional posterior $p(\boldsymbol{W}_c \mid \boldsymbol{w}_s, \mathcal{D})$ over the collapsed parameter set $\boldsymbol{W}_c$. Given collapsed samples $\mathcal{S}$, collapsed BMA estimates the predictive posterior and expected prediction as*

$$p(y \mid \boldsymbol{x}) \approx \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{w}_s, q) \in \mathcal{S}} \left[\int p(y \mid \boldsymbol{x}, \boldsymbol{w}) \, q(\boldsymbol{w}_c) \, d\boldsymbol{w}_c \right], \text{ and}$$

$$\mathbb{E}_{p(y|\boldsymbol{x})}[y] \approx \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{w}_s, q) \in \mathcal{S}} \left[\int y \, p(y \mid \boldsymbol{x}, \boldsymbol{w}) \, q(\boldsymbol{w}_c) \, d\boldsymbol{w}_c \, dy \right]. \tag{2}$$

The size of the collapsed set $\boldsymbol{W}_c$ determines the trade-off between scalability and accuracy. The more parameters in the collapsed set, the more accurate the approximation to

Table 1: Average test log likelihood for the large UCI regression task.

	ELEVATORS	KEGGD	KEGGU	PROTEIN	SKILLCRAFT	POL
CIBER (SECOND)	-0.378 ± 0.026	$\mathbf{1.245 \pm 0.090}$	$\mathbf{1.125 \pm 0.269}$	-0.720 ± 0.036	-1.003 ± 0.035	$\mathbf{2.555 \pm 0.115}$
CIBER (LAST)	-0.371 ± 0.023	1.178 ± 0.088	0.964 ± 0.231	-0.720 ± 0.036	$\mathbf{-1.001 \pm 0.032}$	2.506 ± 0.150
SWAG	-0.374 ± 0.021	1.080 ± 0.035	0.749 ± 0.029	$\mathbf{-0.700 \pm 0.051}$	-1.180 ± 0.033	1.533 ± 1.084
PCA+ESS (SI)	-0.351 ± 0.030	1.074 ± 0.034	0.752 ± 0.025	-0.734 ± 0.063	-1.181 ± 0.033	-0.185 ± 2.779
PCA+VI (SI)	$\mathbf{-0.325 \pm 0.019}$	1.085 ± 0.031	0.757 ± 0.028	-0.712 ± 0.057	-1.179 ± 0.033	1.764 ± 0.271

BMA is. The fewer parameters in $\boldsymbol{W}_c$, the more efficient the computations of the integrals are since the integration is performed in a lower-dimensional space.

To develop an algorithm to compute collapsed BMA, we are faced with two main design choice questions: **(Q1)** how to sample $\boldsymbol{w}_s$ from the posterior? **(Q2)** what should be the representation of the conditional posterior q such that the integrals in Equation 2 can be computed exactly?

3.1. Approximation to Posteriors

For **(Q1)**, we follow Maddox et al. (2019) and sample from the stochastic gradient descent (SGD) trajectory after convergence and use the information contained in SGD trajectories to efficiently approximate the posterior distribution, leveraging the interpretation of SGD as approximate Bayesian inference (Mandt et al., 2017; Chen et al., 2020). Given a set of parameter samples $\mathcal{W}$ from the SGD trajectory, the sample set is defined as $\mathcal{W}_s = \{\boldsymbol{w}_s \mid \boldsymbol{w} \in \mathcal{W}\}$. For each assignment $\boldsymbol{w}_s$, an approximation $q(\boldsymbol{W}_c)$ to the conditional posterior $p(\boldsymbol{W}_c \mid \boldsymbol{w}_s, \mathcal{D})$ is necessary since the posteriors induced by SGD trajectories are implicit.

3.2. Encoding into WMI Problems

When a BNN can be encoded as a WMI problem, the posterior predictive distribution and the expected prediction, which involve marginalization over the parameter space, can be computed exactly using WMI solvers. This inspires us to use the WMI framework as the closed-form representations for the conditional posteriors of parameters. The challenge is how to approximate the integrand in Equation 2 using SMT formulas and polynomial weight functions in order to obtain a WMI problem amenable to existing solvers.

For the conditional posterior approximation $q(\boldsymbol{W}_c)$, we choose it to be a uniform distribution which can be encoded into a WMI problem as $\mathcal{M}_{pos} = (\Delta_{pos}, \Phi_{pos})$ with the SMT formula being $\Delta_{pos} = \wedge_{i \in c} (l_i \leq W_i \leq u_i)$ and weights being $\Phi_{pos} = \{\phi_\ell(\boldsymbol{W}_c) = 1 \mid \ell = \texttt{true}\}$, where l_i and u_i are domain lower and upper bounds for the uniform distribution respectively. While seemingly over-simplistic, this choice of approximation to the conditional posterior is sufficient to robustly deliver surprisingly strong empirical performance. The intuition is that uniform distributions are

better than a few samples, which is further illustrated by comparing the predictive distributions of CIBER and HMC in a few-sample setting as shown in Figure 2.

For the choice of predictive distribution $p(y \mid \boldsymbol{x}, \boldsymbol{w})$, we propose to use piecewise polynomial densities. Common predictive distributions can be approximated by polynomials up to arbitrary precision in theory by the Stone–Weierstrass theorem (De Branges, 1959). Take regression as an example, the de facto choice is Gaussian and we propose to use triangular distribution as the approximation, i.e., $p(y \mid \boldsymbol{x}, \boldsymbol{w}) = \frac{1}{r} - \frac{1}{r^2}|y - f_{\boldsymbol{w}}(\boldsymbol{x})|$, with domain $|y - f_{\boldsymbol{w}}(\boldsymbol{x})| \leq r$, and $r := \alpha\sqrt{\sigma^2(\boldsymbol{x})}$ where the constant α parameterizes the triangular distribution and $\sigma^2(\boldsymbol{x})$ the variance estimate. Then $p(y \mid \boldsymbol{x}, \boldsymbol{w})$ can be encoded into a WMI problem as:

$$
\Delta_{pred} = \begin{cases} Y - f_{\boldsymbol{w}}(\boldsymbol{x}) \leq r \\ Y - f_{\boldsymbol{w}}(\boldsymbol{x}) \geq -r \end{cases}
$$
$$
\Phi_{pred} = \left\{ \begin{array}{l} \phi_{\ell_1}(Y, \boldsymbol{W}_c) = \frac{1}{r} - \frac{Y - f_{\boldsymbol{w}}(\boldsymbol{x})}{r^2} \;\; with \;\; \ell_1 = (Y > f_{\boldsymbol{w}}(\boldsymbol{x})) \\ \phi_{\ell_2}(Y, \boldsymbol{W}_c) = \frac{1}{r} - \frac{f_{\boldsymbol{w}}(\boldsymbol{x}) - Y}{r^2} \;\; with \;\; \ell_2 = (f_{\boldsymbol{w}}(\boldsymbol{x}) > Y) \end{array} \right\}
$$

3.3. Exact Integration in Collapsed BMA

By encoding the collapsed BMA into WMI problems, we are ready to answer **(Q2)**, i.e., how to perform exact computation of the integrals shown in Equation 2.

Proposition 4. *Let the SMT formula $\Delta = \Delta_{\mathsf{ReLU}} \wedge \Delta_{pos} \wedge \Delta_{pred}$, and the set of weights $\Phi = \Phi_{pos} \cup \Phi_{pred}$ as defined in Section 3.2. Let the set of weights $\Phi^* = \Phi \cup \{\phi_\ell(Y) = Y \text{ with } \ell = \texttt{true}\}$. The integrals in collapsed BMA (Equation 2) can be computed by WMI solvers as*

$$
\int p(y \mid \boldsymbol{x}, \boldsymbol{w}) \, q(\boldsymbol{w}_c) \, d\boldsymbol{w}_c = \frac{\mathsf{WMI}(\Delta \wedge (\boldsymbol{Y} = y), \Phi)}{\mathsf{WMI}(\Delta, \Phi)},
$$
and $\int y \, p(y \mid \boldsymbol{x}, \boldsymbol{w}) \, q(\boldsymbol{w}_c) \, d\boldsymbol{w}_c \, dy = \frac{\mathsf{WMI}(\Delta, \Phi^*)}{\mathsf{WMI}(\Delta, \Phi)}.$

With both question **(Q1)** and **(Q2)** answered, we summarize our proposed algorithm CIBER as Algorithm 1 in Appendix. To quantitatively analyze how close the approximation delivered by CIBER is to the ground-truth BMA, we consider the following experiments with closed-form BMA.

Regression. We consider a Bayesian linear regression setting where exact sampling from the posterior is available.

Table 2: Average test performance for image classification tasks on CIFAR-10 and CIFAR-100.

METRIC	NLL		ACC		ECE	
DATASET	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
CIBER	**0.1927 ± 0.0029**	**0.9193 ± 0.0027**	**93.64 ± 0.09**	**74.71 ± 0.18**	0.0130 ± 0.0011	**0.0168 ± 0.0025**
SWAG	0.2503 ± 0.0081	1.2785 ± 0.0031	93.59 ± 0.14	73.85 ± 0.25	0.0391 ± 0.0020	0.1535 ± 0.0015
SGD	0.3285 ± 0.0139	1.7308 ± 0.0137	93.17 ± 0.14	73.15 ± 0.11	0.0483 ± 0.0022	0.1870 ± 0.0014
SWA	0.2621 ± 0.0104	1.2780 ± 0.0051	93.61 ± 0.11	74.30 ± 0.22	0.0408 ± 0.0019	0.1514 ± 0.0032
SGLD	0.2001 ± 0.0059	0.9699 ± 0.0057	93.55 ± 0.15	74.02 ± 0.30	**0.0082 ± 0.0012**	0.0424 ± 0.0029
KFAC	0.2252 ± 0.0032	1.1915 ± 0.0199	92.65 ± 0.20	72.38 ± 0.23	0.0094 ± 0.0005	0.0778 ± 0.0054
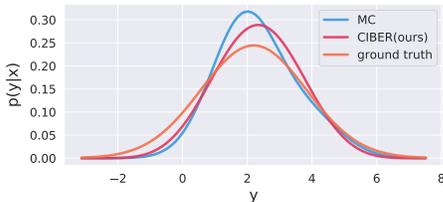
Figure 3: Exact and approximate posterior predictive distributions in Bayesian linear regression.

Both the likelihood and the weight posterior are Gaussian such that the ground-truth posterior predictive distribution is also Gaussian. We evaluate the posterior predictive estimated by CIBER and Monte Carlo (MC), both using the same five samples drawn from the weight posterior, in Figure 3. CIBER approximates the samples with a uniform distribution as posterior $p(\boldsymbol{w}|\mathcal{D})$ and further approximates the likelihood with a triangular distribution such that the integral in $p(y|\boldsymbol{x}, \mathcal{D})$ can be computed exactly by WMI. The CIBER approximation is closer to the ground truth than the MC estimate. The KL divergence between the ground truth and CIBER is 0.030 while the one for MC estimation is 0.085, indicating that CIBER yields a better approximation.

Classification. For analyzing classification performance, Kristiadi et al. (2022) propose to compute the integral $I = \int \sigma(f_*) p_{\mathcal{N}}(f_*) \, df_*$ with σ being the sigmoid function and $f_* = f(\boldsymbol{x}^*; \boldsymbol{w})$ that amounts to the posterior predictive distribution. We consider a simple case with $f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{x}$ such that the ground-truth integral can be obtained. With a randomly chosen $\boldsymbol{x}$, the ground-truth integral is $I = 0.823$. The integral estimated by CIBER is $I_C = 0.826$ while the MC estimate is $I_{MC} = 0.732$. That is, CIBER gives an estimate with a much lower error than the MC estimation error, indicating that CIBER is able to deliver high-quality approximations in classification tasks.

4. Experiments

Regression on UCI Datasets We experiment on both small and large UCI datasets following the setup of Izmailov et al.

(2020). We run CIBER with two different ways of choosing the collapsed parameter set: *CIBER (last)* chooses all the weights at the last layer to be the collapsed set; *CIBER (second)* chooses three out of all the weights at the second-to-last layer to be the collapsed set. The baselines we choose include SWAG (Maddox et al., 2019), PCA+ESS (SI) and PCA+VI (SI) (Izmailov et al., 2020). We present the test log likelihoods for large UCI datasets in Table 1 and other results for more dataset with more baselines in Appendix. CIBER either outperforms or delivers highly comparable results on both likelihood and accuracy, which illustrates that exact marginalization over conditional approximate posteriors enabled by WMI solvers achieves accurate estimation of the true BMA and boosts predictive performance.

Classification on CIFAR Datasets We experiment with two image datasets: CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009) and evaluate the test performance using three metrics: 1) negative log likelihood (NLL), 2) classification accuracy (ACC), and 3) expected calibration errors (ECE) (Naeini et al., 2015). We run CIBER by choosing the collapsed parameter set to be weights at the last layer of the neural network models. We compare CIBER with strong baselines including SWAG (Maddox et al., 2019) reproduced by their open-source implementation, standard SGD, SWA (Izmailov et al., 2018), SGLD (Welling & Teh, 2011) and KFAC (Ritter et al., 2018). We present the test performance on dataset CIFAR-10 and CIFAR-100 using VGG-16 networks (Simonyan & Zisserman, 2014) in Table 2. We present more results with various network architectures in Appendix. CIBER outperforms all baselines in most evaluations and comparable otherwise, demonstrating the effectiveness of using collapsed samples in improving uncertainty estimation and classification performance.

5. Conclusions

We reveal the connection between BMA, a way to perform Bayesian deep learning, and WVC, which inspires us to approximate BMA using WMI. To further make this approximation scalable and flexible, we combine it with collapsed samples which gives our algorithm CIBER. It compares favorably to baselines on regression and classification tasks.

Acknowledgments

References

Baldoni, V., Berline, N., De Loera, J. A., Dutra, B., Köppe, M., Moreinis, S., Pinto, G., Vergne, M., and Wu, J. A user's guide for latte integrale v1. 7.2. *Optimization*, 22 (2), 2014.

Barrett, C., de Moura, L., Ranise, S., Stump, A., and Tinelli, C. The SMT-LIB initiative and the rise of SMT. In *Proceedings of the 6th international conference on Hardware and software: verification and testing*, pp. 3–3. Springer-Verlag, 2010.

Belle, V., Passerini, A., and Van den Broeck, G. Probabilistic inference in hybrid domains by weighted model integration. In *Proceedings of IJCAI*, pp. 2770–2776, 2015a.

Belle, V., Van den Broeck, G., and Passerini, A. Hashing-based approximate probabilistic inference in hybrid domains. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI)*, 2015b.

Bui, T., Hernández-Lobato, D., Hernandez-Lobato, J., Li, Y., and Turner, R. Deep gaussian processes for regression using approximate expectation propagation. In *International conference on machine learning*, pp. 1472–1481. PMLR, 2016.

Chen, X., Lee, J. D., Tong, X. T., and Zhang, Y. Statistical inference for model parameters in stochastic gradient descent. *The Annals of Statistics*, 48(1):251–273, 2020.

Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

De Branges, L. The stone-weierstrass theorem. *Proceedings of the American Mathematical Society*, 10(5):822–824, 1959.

De Loera, J. A., Dutra, B., Koeppe, M., Moreinis, S., Pinto, G., and Wu, J. Software for exact integration of polynomials over polyhedra. *ACM Communications in Computer Algebra*, 45(3/4):169–172, 2012.

de Salvo Braz, R., O'Reilly, C., Gogate, V., and Dechter, R. Probabilistic inference modulo theories. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pp. 3591–3599. AAAI Press, 2016.

Fragoso, T. M., Bertoli, W., and Louzada, F. Bayesian model averaging: A systematic review and conceptual classification. *International Statistical Review*, 86(1): 1–28, 2018.

Gal, Y. and Ghahramani, Z. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Hein, M., Andriushchenko, M., and Bitterwolf, J. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 41–50, 2019.

Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. Safety verification of deep neural networks. In *International conference on computer aided verification*, pp. 3–29. Springer, 2017.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Izmailov, P., Maddox, W. J., Kirichenko, P., Garipov, T., Vetrov, D., and Wilson, A. G. Subspace inference for bayesian deep learning. In *Uncertainty in Artificial Intelligence*, pp. 1169–1179. PMLR, 2020.

Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. G. What are bayesian neural network posteriors really like? In *International Conference on Machine Learning*, pp. 4629–4640. PMLR, 2021.

Jospin, L. V., Laga, H., Boussaid, F., Buntine, W., and Bennamoun, M. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In *International conference on computer aided verification*, pp. 97–117. Springer, 2017.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kolb, S., Mladenov, M., Sanner, S., Belle, V., and Kersting, K. Efficient symbolic integration for probabilistic inference. In *IJCAI*, pp. 5031–5037, 2018.

Kolb, S., Morettin, P., Zuidberg Dos Martires, P., Sommavilla, F., Passerini, A., Sebastiani, R., and De Raedt, L. The pywmi framework and toolbox for probabilistic inference using weighted model integration. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, pp. 6530–6532, 7 2019. doi: 10.24963/ijcai.2019/946.

Koller, D. and Friedman, N. Probabilistic graphical models. 2009.

Kristiadi, A., Eschenhagen, R., and Hennig, P. Posterior refinement improves sample efficiency in bayesian neural networks. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30333–30346. Curran Associates, Inc., 2022.

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

MacKay, D. J. The evidence framework applied to classification networks. *Neural computation*, 4(5):720–736, 1992.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32:13153–13164, 2019.

Mandt, S., Hoffman, M. D., and Blei, D. M. Stochastic gradient descent as approximate bayesian inference. *arXiv preprint arXiv:1704.04289*, 2017.

Morettin, P., Passerini, A., and Sebastiani, R. Efficient weighted model integration via SMT-based predicate abstraction. In *Proceedings of IJCAI*, pp. 720–728, 2017.

Morettin, P., Passerini, A., and Sebastiani, R. Advanced smt techniques for weighted model integration. *Artificial Intelligence*, 275:1–27, 2019.

Naeini, M. P., Cooper, G., and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

Nguyen, A., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

Nguyen, S., Nguyen, D., Nguyen, K., Than, K., Bui, H., and Ho, N. Structured dropout variational inference for bayesian neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Riquelme, C., Tucker, G., and Snoek, J. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

Ritter, H., Botev, A., and Barber, D. A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, volume 6. International Conference on Representation Learning, 2018.

Salimbeni, H., Cheng, C.-A., Boots, B., and Deisenroth, M. Orthogonally decoupled variational gaussian processes. *Advances in neural information processing systems*, 31, 2018.

Sang, T., Beame, P., and Kautz, H. A. Performing bayesian inference by weighted model counting. In *AAAI*, volume 5, pp. 475–481, 2005.

Sanner, S. and Abbasnejad, E. Symbolic variable elimination for discrete and continuous graphical models. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

Sanner, S., Delgado, K. V., and De Barros, L. N. Symbolic dynamic programming for discrete and continuous state mdps. *arXiv preprint arXiv:1202.3762*, 2012.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sivaraman, A., Farnadi, G., Millstein, T., and Van den Broeck, G. Counterexample-guided learning of monotonic neural networks. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, dec 2020.

Valiant, L. G. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

Wasserman, L. Bayesian model selection and model averaging. *Journal of mathematical psychology*, 44(1):92–107, 2000.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016.

Wu, A., Nowozin, S., Meeds, E., Turner, R. E., Hernandez-Lobato, J. M., and Gaunt, A. L. Deterministic variational inference for robust bayesian neural networks. In *International Conference on Learning Representations*, 2019.

Yang, Z., Wilson, A., Smola, A., and Song, L. A la carte–learning fast kernels. In *Artificial Intelligence and Statistics*, pp. 1098–1106. PMLR, 2015.

Zeng, Z. and Van den Broeck, G. Efficient search-based weighted model integration. *Proceedings of UAI*, 2019.

Zeng, Z., Morettin, P., Yan, F., Vergari, A., and Broeck, G. V. d. Scaling up hybrid probabilistic inference with logical and arithmetic constraints via message passing. In *Proceedings of the International Conference of Machine Learning (ICML)*, 2020a.

Zeng, Z., Morettin, P., Yan, F., Vergari, A., and Van den Broeck, G. Relax, compensate and then integrate. In *Proceedings of the ECML-PKDD Workshop on Deep Continuous-Discrete Machine Learning (DeCoDeML)*, 2020b.

Zeng, Z., Morettin, P., Yan, F., Vergari, A., and Van den Broeck, G. Probabilistic inference with algebraic constraints: Theoretical limits and practical approximations. *Advances in Neural Information Processing Systems*, 33: 11564–11575, 2020c.

Zeng, Z., Morettin, P., Yan, F., Vergari, A., and Van den Broeck, G. Is parameter learning via weighted model integration tractable? In *Proceedings of the UAI Workshop on Tractable Probabilistic Modeling (TPM)*, jul 2021.

Zuidberg Dos Martires, P. M., Dries, A., and De Raedt, L. Exact and approximate weighted model integration with probability density functions using knowledge compilation. In *Proceedings of the 30th Conference on Artificial Intelligence*. AAAI Press, 2019.

A. Proofs

Proposition 4 *Let the SMT formula $\Delta = \Delta_{\mathsf{ReLU}} \wedge \Delta_{pos} \wedge \Delta_{pred}$, and the set of weights $\Phi = \Phi_{pos} \cup \Phi_{pred}$ as defined in Section 3.2. Let the set of weights $\Phi^* = \Phi \cup \{\phi_\ell(Y) = Y$ with $\ell = \texttt{true}\}$. The integrals in collapsed BMA (Equation 2) can be computed by WMI solvers as*

$$\int p(y \mid \boldsymbol{x}, \boldsymbol{w}) \, q(\boldsymbol{w}_c) \, d\boldsymbol{w}_c = \frac{\mathsf{WMI}(\Delta \wedge (\boldsymbol{Y} = y), \Phi)}{\mathsf{WMI}(\Delta, \Phi)},$$

and $\displaystyle \int y \, p(y \mid \boldsymbol{x}, \boldsymbol{w}) \, q(\boldsymbol{w}_c) \, d\boldsymbol{w}_c \, dy = \frac{\mathsf{WMI}(\Delta, \Phi^*)}{\mathsf{WMI}(\Delta, \Phi)}.$

Proof. By construction, it holds that

$$p(y \mid \boldsymbol{x}, \boldsymbol{w}) \propto \mathsf{WMI}(\Delta_{pred} \wedge (\boldsymbol{Y} = y), \Phi_{pred})$$
$$\propto \prod_{\ell \in \mathcal{L}_{pred}} \phi_\ell(y, \boldsymbol{w}_c)^{[\![y, \boldsymbol{w}_c \models \ell]\!]},$$
$$\text{with } (y, \boldsymbol{w}_c) \models \Delta_{pred} \wedge \Delta_{\mathsf{ReLU}}$$
$$q(\boldsymbol{w}_c) \propto \mathsf{WMI}(\Delta_{pos} \wedge (\boldsymbol{W}_c = \boldsymbol{w}_c), \Phi_{pos})$$
$$\propto \prod_{\ell \in \mathcal{L}_{pos}} \phi_\ell(\boldsymbol{w}_c)^{[\![\boldsymbol{w}_c \models \ell]\!]}, \text{ with } \boldsymbol{w}_c \models \Delta_{pos}$$

Thus, we have that the likelihood weighted by the approximate posterior would be

$$p(y \mid \boldsymbol{x}, \boldsymbol{w}) \, q(\boldsymbol{w}_c) \propto \prod_{\ell \in \mathcal{L}_{pred} \wedge \mathcal{L}_{pos}} \phi_\ell(y, \boldsymbol{w}_c)^{[\![y, \boldsymbol{w}_c \models \ell]\!]}$$
$$\text{with } (y, \boldsymbol{w}_c) \models \Delta_{\mathsf{ReLU}} \wedge \Delta_{pred} \wedge \Delta_{pos},$$

or equivalently,

$$p(y \mid \boldsymbol{x}, \boldsymbol{w}) \, q(\boldsymbol{w}_c) = \frac{\prod_{\ell \in \mathcal{L}_{pred} \wedge \mathcal{L}_{pos}} \phi_\ell(y, \boldsymbol{w}_c)^{[\![y, \boldsymbol{w}_c \models \ell]\!]}}{\mathsf{WMI}(\Delta, \Phi)},$$
$$\text{with } (y, \boldsymbol{w}_c) \models \Delta.$$

By integrating over the collapsed set $\boldsymbol{W}_c$, it further holds that

$$\int p(y \mid \boldsymbol{x}, \boldsymbol{w}) \, q(\boldsymbol{w}_c) \, d\boldsymbol{w}_c$$
$$= \frac{\int \prod_{\ell \in \mathcal{L}_{pred} \wedge \mathcal{L}_{pos}} \phi_\ell(y, \boldsymbol{w}_c)^{[\![y, \boldsymbol{w}_c \models \ell]\!]} \, d\boldsymbol{w}_c}{\mathsf{WMI}(\Delta, \Phi)},$$
$$\text{with } (y, \boldsymbol{w}_c) \models \Delta$$
$$= \frac{\mathsf{WMI}(\Delta \wedge (\boldsymbol{Y} = y), \Phi)}{\mathsf{WMI}(\Delta, \Phi)}$$

which proves the first equation.

Similarly, we have that

$$y \, p(y \mid \boldsymbol{x}, \boldsymbol{w}) \, q(\boldsymbol{w}_c)$$
$$\propto \prod_{\ell \in \mathcal{L}_{pred}} y \, \phi_\ell(y, \boldsymbol{w}_c)^{[\![y, \boldsymbol{w}_c \models \ell]\!]} \prod_{\ell \in \mathcal{L}_{pos}} \phi_\ell(\boldsymbol{w}_c)^{[\![\boldsymbol{w}_c \models \ell]\!]},$$
$$\text{with } (y, \boldsymbol{w}_c) \models \Delta$$

Algorithm 1 CIBER

Input: input $\boldsymbol{x}$, sampled weights $\mathcal{W}$, neural network model $f_{\boldsymbol{w}}$, prediction ground truth y^*
Ouput: predictions and likelihoods

1: Choose a partition $(\boldsymbol{W}_s, \boldsymbol{W}_c)$ for network parameters
2: Derive approximate posterior $q(\boldsymbol{w}_c)$ from sampled weights $\{\boldsymbol{w}_c \mid \boldsymbol{w} \in \mathcal{W}\}$ *// cf. Section 3.2*
3: Encode posterior $q(\boldsymbol{w}_c)$ into WMI problem $\mathcal{M}_{pos} = (\Delta_{pos}, \Phi_{pos})$ *// cf. Section 3.2*
4: $\mathcal{Y} \leftarrow \emptyset, \mathcal{P} \leftarrow \emptyset$ *// Initialization*
5: **for** sample $\boldsymbol{w}_s$ in $\{\boldsymbol{w}_s \mid \boldsymbol{w} \in \mathcal{W}\}$ **do**
6:     Encode neural network model f_{ReLU} parameterized by $(\boldsymbol{w}_s, \boldsymbol{W}_c)$ into an SMT formula $\Delta_{f_{\boldsymbol{w}}}$
7:     Encode predictive $p(Y \mid \boldsymbol{x}, \boldsymbol{w}_s, \boldsymbol{W}_c)$ into a WMI problem $\mathcal{M}_{pred} = (\Delta_{pred}, \Phi_{pred})$
8:     SMT formula $\Delta \leftarrow \Delta_{\mathsf{ReLU}} \wedge \Delta_{pos} \wedge \Delta_{pred}$
9:     Weights $\Phi \leftarrow \Phi_{pos} \cup \Phi_{pred}$
10:    Weights $\Phi^* \leftarrow \Phi \cup \{\phi_\ell(Y) = Y$ with $\ell = \texttt{true}\}$
11:    Add prediction $y = \mathsf{WMI}(\Delta, \Phi^*)/\mathsf{WMI}(\Delta, \Phi)$ to prediction set $\mathcal{Y}$ *// cf. Section 3.3*
12:    Add likelihood $p = \mathsf{WMI}(\Delta \wedge (Y = y^*), \Phi)/\mathsf{WMI}(\Delta, \Phi)$ to set $\mathcal{P}$ *// cf. Section 3.3*
13: **end for**
14: **return** $y = \textsc{Mean}(\mathcal{Y}), p(y^* \mid \boldsymbol{x}) = \textsc{Mean}(\mathcal{P})$

By integrating over the collapsed set $\boldsymbol{W}_c$ and prediction y, it holds that

$$\int y \, p(y \mid \boldsymbol{x}, \boldsymbol{w}) \, q(\boldsymbol{w}_c) \, d\boldsymbol{w}_c \, dy$$
$$= \frac{\int y \prod_{\ell \in \mathcal{L}_{pred}} \phi_\ell(y, \boldsymbol{w}_c)^{[\![y, \boldsymbol{w}_c \models \ell]\!]} \prod_{\ell \in \mathcal{L}_{pos}} \phi_\ell(\boldsymbol{w}_c)^{[\![\boldsymbol{w}_c \models \ell]\!]} \, d\boldsymbol{w}_c \, dy}{\mathsf{WMI}(\Delta, \Phi)},$$
$$\text{with } (y, \boldsymbol{w}_c) \models \Delta$$
$$= \frac{\mathsf{WMI}(\Delta, \Phi^*)}{\mathsf{WMI}(\Delta, \Phi)}$$

which finishes our proof. □

B. Pseudo Code for CIBER

We summarize our proposed algorithm **CIBER**, **C**ollapsed **I**nference **B**ayesian D**E**ep Lea**R**ning, for regression tasks, in Algorithm 1. For the classification task, the algorithm is basically the same except the encoding of the predictive of the distribution. Specifically, for a given class y, the predictive distribution $p(y \mid \boldsymbol{x}, \boldsymbol{w})$ can be encoded into a WMI problem as shown below:

$$\Delta_{pred} = f_{\boldsymbol{w}}(\boldsymbol{x}) \geq -d$$
$$\Phi_{pred} = \left\{ \begin{array}{ll} \phi_{\ell_1}(\boldsymbol{W}_c) & \text{with } \ell_1 = (f_{\boldsymbol{w}}(\boldsymbol{x}) \leq d) \\ \phi_{\ell_2}(\boldsymbol{W}_c) = 1 & \text{with } \ell_2 = (f_{\boldsymbol{w}}(\boldsymbol{x}) > d) \end{array} \right\}$$

Table 3: Average test RMSE for the small UCI regression task.

	BOSTON	CONCRETE	YACHT	NAVAL	ENERGY
CIBER (SECOND)	3.488 ± 1.123	4.880 ± 0.506	0.828 ± 0.241	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.447 \pm 0.081}$
CIBER (LAST)	3.478 ± 1.128	$\mathbf{4.854 \pm 0.503}$	$\mathbf{0.752 \pm 0.294}$	$\underline{\mathbf{0.000 \pm 0.000}}$	$\mathbf{0.447 \pm 0.081}$
SWAG	3.517 ± 0.981	5.233 ± 0.417	0.973 ± 0.375	0.001 ± 0.000	1.594 ± 0.273
PCA+ESS (SI)	3.453 ± 0.953	5.194 ± 0.448	0.972 ± 0.375	0.001 ± 0.000	1.598 ± 0.274
PCA+VI (SI)	$\mathbf{3.457 \pm 0.951}$	5.142 ± 0.418	0.973 ± 0.375	0.001 ± 0.000	1.587 ± 0.272
SGD	3.504 ± 0.975	5.194 ± 0.446	0.973 ± 0.374	0.001 ± 0.000	1.602 ± 0.275
MCD	2.830 ± 0.170	4.930 ± 0.140	0.720 ± 0.050	$\underline{0.000 \pm 0.000}$	1.080 ± 0.030
VSD	$\underline{2.640 \pm 0.170}$	$\underline{4.720 \pm 0.110}$	$\underline{0.690 \pm 0.060}$	$\underline{0.000 \pm 0.000}$	$\underline{0.470 \pm 0.010}$

Table 4: Average test RMSE for the large UCI regression task.

	ELEVATORS	KEGGD	KEGGU	PROTEIN	SKILLCRAFT	POL
CIBER (SECOND)	$\mathbf{0.088 \pm 0.002}$	0.142 ± 0.074	$\mathbf{0.115 \pm 0.007}$	0.438 ± 0.009	$\mathbf{0.251 \pm 0.010}$	2.212 ± 0.230
CIBER (LAST)	$\mathbf{0.088 \pm 0.002}$	0.142 ± 0.072	0.118 ± 0.012	0.438 ± 0.009	$\mathbf{0.251 \pm 0.010}$	$\underline{\mathbf{2.199 \pm 0.182}}$
SWAG	$\mathbf{0.088 \pm 0.001}$	0.129 ± 0.029	0.160 ± 0.043	$\mathbf{0.415 \pm 0.018}$	0.293 ± 0.015	3.110 ± 0.070
PCA+ESS (SI)	0.089 ± 0.002	0.129 ± 0.028	0.160 ± 0.043	0.425 ± 0.017	0.293 ± 0.015	3.755 ± 6.107
PCA+VI (SI)	$\mathbf{0.088 \pm 0.001}$	$\mathbf{0.128 \pm 0.028}$	0.160 ± 0.043	0.418 ± 0.021	0.293 ± 0.015	2.499 ± 0.684
SGD	0.103 ± 0.035	0.132 ± 0.017	0.186 ± 0.034	0.436 ± 0.011	0.288 ± 0.014	3.900 ± 6.003
NL	0.101 ± 0.002	0.134 ± 0.036	0.120 ± 0.003	0.447 ± 0.012	0.253 ± 0.011	4.380 ± 0.853
DKL	$\underline{0.084 \pm 0.020}$	$\underline{0.100 \pm 0.010}$	$\underline{0.110 \pm 0.000}$	0.460 ± 0.010	$\underline{0.250 \pm 0.000}$	6.617
ORTHVGP	0.095	0.120	0.117	0.461	—	4.300 ± 0.200
FF	0.089 ± 0.002	0.120 ± 0.000	0.120 ± 0.000	0.470 ± 0.010	$\underline{0.250 \pm 0.020}$	—

where ϕ_{ℓ_1} is a cubic polynomial that approximates the sigmoid function such that the posterior predictive distribution $p(y \mid \boldsymbol{x})$ can be solved by WMI solvers by $p(y \mid \boldsymbol{x}) = \mathsf{WMI}(\Delta, \Phi)$. Further, the prediction of BMA for classification tasks is made by $y^* = \arg\max_y p(y \mid \boldsymbol{x})$.

C. Additional Experiments

C.1. Toy Regression in Figure 2

We evaluate the predictive distributions obtained by our CIBER and HMC respectively, in a toy dataset generated by sampling 10 input x uniformly distributed in the interval $[-1, -0.5]$ and interval $[0.5, 1]$. For each input x, the corresponding target y is computed from a cubic polynomial with Gaussian noises. We apply to these data a Bayesian neural network which is a ReLU neural network with two hidden layers, where both parameter priors and likelihood are Gaussian distributions. We compare HMC and our CIBER in a few-sample setting which is common in most Bayesian deep learning applications, with 10 samples from the posterior distribution. An estimation generated by HMC with a sufficiently large number of samples of size $2,000$ is further

presented as a ground truth.

The results are shown in Figure 2. Even with the same 10 samples drawn from the posterior distribution, since CIBER further approximates the 10 samples with a uniform distribution as $q(\boldsymbol{w})$, it yields a predictive distribution $p(y \mid \boldsymbol{x})$ closer to the ground truth than HMC. The intuition behind is that using a uniform distribution instead of a few samples forms a better approximation to the true posterior since the uniform distribution in a collapsed sample represents uncountably many models.

C.2. Regression on Small and Large Datasets

Sampling from SGD Trajectories. During training, we use Gaussian log likelihood as the objective for obtaining smooth gradients and use early stopping to prevent overfitting. At convergence, we start the sampling process by keeping running SGD and collecting the weights. At deployment time, we approximate the Gaussian predictive distribution with the triangular distributions.

Hyperparameters. The hyperparameters including learning rates and weight decay are tuned by performing a grid

Table 5: Average test log likelihood for the small UCI regression task.

	BOSTON	CONCRETE	YACHT	NAVAL	ENERGY
CIBER (SECOND)	**-2.471 ± 0.140**	-2.975 ± 0.102	-0.678 ± 0.301	7.276 ± 0.532	**-0.716 ± 0.211**
CIBER (LAST)	**-2.471 ± 0.140**	**-2.959 ± 0.109**	-0.687 ± 0.301	**7.482 ± 0.188**	**-0.716 ± 0.211**
SWAG	-2.761 ± 0.132	-3.013 ± 0.086	-0.404 ± 0.418	6.708 ± 0.105	-1.679 ± 1.488
PCA+ESS (SI)	-2.719 ± 0.132	-3.007 ± 0.086	**-0.225 ± 0.400**	6.541 ± 0.095	-1.563 ± 1.243
PCA+VI (SI)	-2.716 ± 0.133	-2.994 ± 0.095	-0.396 ± 0.419	6.708 ± 0.105	-1.715 ± 1.588
SGD	-2.752 ± 0.132	-3.178 ± 0.198	-0.418 ± 0.426	6.567 ± 0.185	-1.736 ± 1.613
DVI	-2.410 ± 0.020	-3.060 ± 0.010	-0.470 ± 0.030	6.290 ± 0.040	-1.010 ± 0.060
DGP	<u>-2.330 ± 0.060</u>	-3.130 ± 0.030	-1.390 ± 0.140	3.600 ± 0.330	-1.320 ± 0.030
VI	-2.430 ± 0.030	-3.040 ± 0.020	-1.680 ± 0.040	5.870 ± 0.290	-2.380 ± 0.020
MCD	-2.400 ± 0.040	-2.970 ± 0.020	-1.380 ± 0.010	4.760 ± 0.010	-1.720 ± 0.010
VSD	-2.350 ± 0.050	-2.970 ± 0.020	-1.140 ± 0.020	4.830 ± 0.010	-1.060 ± 0.010

Table 6: Average test log likelihood for the large UCI regression task.

	ELEVATORS	KEGGD	KEGGU	PROTEIN	SKILLCRAFT	POL
CIBER (SECOND)	-0.378 ± 0.026	**1.245 ± 0.090**	**1.125 ± 0.269**	-0.720 ± 0.036	-1.003 ± 0.035	**2.555 ± 0.115**
CIBER (LAST)	-0.371 ± 0.023	1.178 ± 0.088	0.964 ± 0.231	-0.720 ± 0.036	**-1.001 ± 0.032**	2.506 ± 0.150
SWAG	-0.374 ± 0.021	1.080 ± 0.035	0.749 ± 0.029	**-0.700 ± 0.051**	-1.180 ± 0.033	1.533 ± 1.084
PCA+ESS (SI)	-0.351 ± 0.030	1.074 ± 0.034	0.752 ± 0.025	-0.734 ± 0.063	-1.181 ± 0.033	-0.185 ± 2.779
PCA+VI (SI)	**-0.325 ± 0.019**	1.085 ± 0.031	0.757 ± 0.028	-0.712 ± 0.057	-1.179 ± 0.033	1.764 ± 0.271
SGD	-0.538 ± 0.108	1.012 ± 0.154	0.602 ± 0.224	-0.854 ± 0.085	-1.162 ± 0.032	1.073 ± 0.858
ORTHVGP	-0.448	1.022	0.701	-0.914	—	0.159
NL	-0.698 ± 0.039	0.935 ± 0.265	0.670 ± 0.038	-0.884 ± 0.025	-1.002 ± 0.050	-2.840 ± 0.226

search to maximize the Gaussian log likelihood using a validation split.

We experiment on 5 small UCI datasets: *boston*, *concrete*, *yacht*, *naval* and *energy*. We follow the setup of Izmailov et al. (2020) and use a fully-connected network with a single hidden layer and 50 units with ReLU activations. We further experiment on 6 large UCI datasets: *elevators*, *keggdirected*, *keggundirected*, *pol*, *protein* and *skillcraft*. We use a feedforward network with five hidden layers of sizes [1000, 1000, 500, 50, 2] and ReLU activations on all datasets except *skillcraft*. For *skillcraft*, a smaller architecture is adopted with four hidden layers of size [1000, 500, 50, 2]. All models have two outputs for the prediction and the heteroscedastic variance respectively.

We run CIBER with two different ways of choosing the collapsed parameter set: *CIBER (last)* chooses all the weights at the last layer to be the collapsed set; *CIBER (second)* chooses three out of all the weights at the second-to-last layer to be the collapsed set. The heuristic we use for choosing the weights is to look into the sampled weights from

SGD trajectories to see which ones have the greatest variance. The intuition is that a greater variance indicates that the weight is prone to have greater uncertainty and thus one might want to perform more accurate inference over it.

Baselines. We compare CIBER to the state-of-the-art approximate BNN inference methods. We separate these methods into two categories: those sampling from SGD trajectories as approximate posteriors, which includes SWAG (Maddox et al., 2019), PCA+ESS (SI) and PCA+VI (SI) (Izmailov et al., 2020), vs. those who do not, which includes the SGD baseline, deterministic variational inference (DVI) (Wu et al., 2019), Deep Gaussian Processes (DGP) (Bui et al., 2016), variational inference (VI) (Kingma & Welling, 2013), MC Dropout (MCD) (Gal & Ghahramani, 2015; 2016), and variational structured dropout (VSD) (Nguyen et al., 2021). These methods achieved state-of-the-art performance on the small UCI datasets. We also compare to baselines Bayesian final layers (NL) (Riquelme et al., 2018), deep kernel learning (DKL) (Wilson et al., 2016), orthogonally decoupled variational GPs (OrthVGP) (Salimbeni et al., 2018) and Fast-

Table 7: Average test log likelihoods for image classification tasks on CIFAR-10 and CIFAR-100.

	CIFAR-10			CIFAR-100		
MODEL	VGG-16	PRERESNET-164	WIDERESNET	VGG-16	PRERESNET-164	WIDERESNET
CIBER	**0.1927 ± 0.0029**	**0.1352 ± 0.0014**	0.1913 ± 0.0029	**0.9193 ± 0.0027**	0.8144 ± 0.0065	0.7930 ± 0.0065
SWAG	0.2503 ± 0.0081	0.1459 ± 0.0013	0.1076 ± 0.0009	1.2785 ± 0.0031	1.0703 ± 0.4861	0.6719 ± 0.0035
SGD	0.3285 ± 0.0139	0.1814 ± 0.0025	0.1294 ± 0.0022	1.7308 ± 0.0137	0.9465 ± 0.0191	0.7958 ± 0.0089
SWA	0.2621 ± 0.0104	0.1450 ± 0.0042	**0.1075 ± 0.0004**	1.2780 ± 0.0051	0.7370 ± 0.0265	**0.6684 ± 0.0034**
SGLD	0.2001 ± 0.0059	0.1418 ± 0.0005	0.1289 ± 0.0009	0.9699 ± 0.0057	**0.6981 ± 0.0052**	0.6780 ± 0.0022
KFAC	0.2252 ± 0.0032	0.1471 ± 0.0012	0.1210 ± 0.0020	1.1915 ± 0.0199	0.7881 ± 0.0025	0.7692 ± 0.0092

Table 8: Average test accuracy for image classification tasks on CIFAR-10 and CIFAR-100.

	CIFAR-10			CIFAR-100		
MODEL	VGG-16	PRERESNET-164	WIDERESNET	VGG-16	PRERESNET-164	WIDERESNET
CIBER	**93.64 ± 0.09**	95.95 ± 0.06	95.63 ± 0.16	**74.71 ± 0.18**	79.23 ± 0.25	81.25 ± 0.35
SWAG	93.59 ± 0.14	**96.09 ± 0.08**	96.38 ± 0.08	73.85 ± 0.25	73.02 ± 10.30	82.27 ± 0.07
SGD	93.17 ± 0.14	95.49 ± 0.06	96.41 ± 0.10	73.15 ± 0.11	78.50 ± 0.32	80.76 ± 0.29
SWA	93.61 ± 0.11	**96.09 ± 0.08**	**96.46 ± 0.04**	74.30 ± 0.22	**80.19 ± 0.52**	**82.40 ± 0.16**
SGLD	93.55 ± 0.15	95.55 ± 0.04	95.89 ± 0.02	74.02 ± 0.30	80.09 ± 0.05	80.94 ± 0.17
KFAC	92.65 ± 0.20	95.49 ± 0.06	96.17 ± 0.00	72.38 ± 0.23	78.51 ± 0.05	80.94 ± 0.41

food approximate kernels (FF) (Yang et al., 2015), which have achieved state-of-the-art performance on the large UCI datasets.

Results. We present the test log likelihoods for small UCI datasets in Table 5 and those for large UCI datasets in Table 6. In both tables, the first block summarizes SGD-trajectory sampling based approaches and the second summarizes the rest. Underlined results are the best among all and bold results are the best among SGD-trajectory sampling based approaches. From the results, our CIBER has substantially better performance than all others on three out of the five small UCI datasets four out of six large UCI datasets, with comparable performance on the rest, demonstrating that CIBER provides accurate uncertainty estimation. We also present the test root-mean-squared-error (RMSE) results for small UCI datasets in Table 3 and RMSE results for large UCI datasets in Table 4. where CIBER outperforms all other SGD-trajectory sampling based baselines on four out of five small UCI datasets and four out of six large UCI datasets; it outperforms all baselines on two small UCI datasets and one large UCI datasets, and has comparable performance on the rest. This further illustrates that exact marginalization over conditional approximate posteriors enabled by WMI solvers achieves accurate estimation of the true BMA and boosts predictive performance.

C.3. Image Classification

Sampling from SGD Trajectories. All the network models are trained for 300 epochs using SGD. We start the weight collection after epoch 160 with step size 5. We follow exactly the same hyperparameters as Maddox et al. (2019) including learning rates and weight decay parameters.

CIFAR datasets. We experiment with two image datasets: CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009). We run CIBER by choosing the collapsed parameter set to be 10 weights and 100 weights at the last layer of the neural network models for CIFAR-10 and CIFAR-100 respectively. The weights are chosen using the same heuristic as the one for regression tasks, i.e., to choose the weights whose samples from the SGD trajectories have large variances. We compare CIBER with strong baselines including SWAG (Maddox et al., 2019) reproduced by their open-source implementation, standard SGD, SWA (Izmailov et al., 2018), SGLD (Welling & Teh, 2011) and KFAC (Ritter et al., 2018).

Transfer from CIFAR-10 to STL-10. We further consider a transfer learning task where we use the model trained on CIFAR-10 to be evaluated on dataset STL-10 (Coates et al., 2011). STL-10 shares nine out of ten classes with the CIFAR-10 dataset but has a different image distribution. It is a common benchmark in transfer learning to adapt models

Table 9: Average test ECE for image classification tasks on CIFAR-10 and CIFAR-100.

	CIFAR-10			CIFAR-100		
MODEL	VGG-16	PRERESNET-164	WIDERESNET	VGG-16	PRERESNET-164	WIDERESNET
CIBER	0.0130 ± 0.0011	0.0250 ± 0.0005	0.0760 ± 0.0011	$\mathbf{0.0168 \pm 0.0025}$	0.1423 ± 0.0029	0.1650 ± 0.0046
SWAG	0.0391 ± 0.0020	0.0214 ± 0.0005	0.0096 ± 0.0006	0.1535 ± 0.0015	0.1031 ± 0.0471	0.0678 ± 0.0006
SGD	0.0483 ± 0.0022	0.0255 ± 0.0009	0.0166 ± 0.0007	0.1870 ± 0.0014	0.1012 ± 0.0009	0.0479 ± 0.0010
SWA	0.0408 ± 0.0019	0.0203 ± 0.0010	0.0087 ± 0.0002	0.1514 ± 0.0032	0.0700 ± 0.0056	0.0684 ± 0.0022
SGLD	$\mathbf{0.0082 \pm 0.0012}$	0.0251 ± 0.0012	0.0192 ± 0.0007	0.0424 ± 0.0029	0.0363 ± 0.0008	$\mathbf{0.0296 \pm 0.0008}$
KFAC	0.0094 ± 0.0005	$\mathbf{0.0092 \pm 0.0018}$	$\mathbf{0.0060 \pm 0.0003}$	0.0778 ± 0.0054	$\mathbf{0.0158 \pm 0.0014}$	0.0379 ± 0.0047

Table 10: Average test log likelihoods for image transfer learning task.

MODEL	VGG-16	PRERESNET-164	WIDERESNET
CIBER	$\mathbf{0.9869 \pm 0.0102}$	$\mathbf{0.9684 \pm 0.0075}$	$\mathbf{0.8259 \pm 0.0148}$
SWAG	1.3425 ± 0.0015	1.3842 ± 0.0122	1.0142 ± 0.0032
SGD	1.6528 ± 0.0390	1.4790 ± 0.0000	1.1308 ± 0.0000
SWA	1.3993 ± 0.0502	1.3552 ± 0.0000	1.0047 ± 0.0000

Table 11: Average test accuracy for image transfer learning task.

MODEL	VGG-16	PRERESNET-164	WIDERESNET
CIBER	$\mathbf{72.56 \pm 0.23}$	75.70 ± 0.17	75.02 ± 0.31
SWAG	72.30 ± 0.11	$\mathbf{76.30 \pm 0.06}$	76.96 ± 0.08
SGD	72.42 ± 0.07	75.56 ± 0.00	76.75 ± 0.00
SWA	71.92 ± 0.01	76.02 ± 0.00	$\mathbf{77.50 \pm 0.00}$

trained on CIFAR-10 to STL-10.

Following the set-up of Maddox et al. (2019), we run experiments with VGG-16, PreResNet-164 and WideResNet network models on both the image classification task and the transfer learning task. For the image classification task on CIFAR datasets, we present the log likelihood results in Table 7, the accuracy results in Table 8, and ECE results in Table 9. For the transfer learning task from dataset CIFAR-10 to dataset STL-10, we present the log likelihood results in Table 10, the accuracy results in Table 11, and ECE results in Table 12.

D. Related Work

Bayesian Deep Learning. Bayesian inference over deep neural networks (MacKay, 1992) is proposed to fix the issue that deep learning models give poor uncertainty estimations (Nguyen et al., 2015; Hein et al., 2019). Some methods use samples from SGD trajectories to approximate

Table 12: Average test ECE for image transfer learning task.

MODEL	VGG-16	PRERESNET-164	WIDERESNET
CIBER	$\mathbf{0.0925 \pm 0.0028}$	$\mathbf{0.0704 \pm 0.0031}$	$\mathbf{0.0336 \pm 0.0009}$
SWAG	0.1988 ± 0.0028	0.1668 ± 0.0006	0.1303 ± 0.0008
SGD	0.2149 ± 0.0027	0.1758 ± 0.0000	0.1561 ± 0.0000
SWA	0.2082 ± 0.0056	0.1739 ± 0.0000	0.1413 ± 0.0000

the implicit true posteriors similar to us: Izmailov et al. (2020) (SI) proposes to perform Bayesian inference in a subspace of the parameter space spanned by a few vectors derived from principle component analysis (PCA+ESS(SI)) or variational inference (PCA+VI(SI)); SWAG (Maddox et al., 2019) proposes to approximate the full parameter space using an approximate Gaussian posterior whose mean and covariance are from a partial SGD trajectory with a modified learning rate scheduler.

Some other approaches using approximate posteriors include MC Dropout (MCD) (Gal & Ghahramani, 2015; 2016) which is one of the Bayesian dropout methods and recently, one of its modification called Variational Structured Dropout (VSD) (Nguyen et al., 2021) using variational inference is proposed. Other state-of-the-art approximate BNN inference methods including deterministic variational inference (DVI) (Wu et al., 2019), deep Gaussian processes (DGP) (Bui et al., 2016) with Gaussian process layers and variational inference (VI) (Kingma & Welling, 2013).

WMI Solvers. WMI generalizes weighted model counting (WMC) (Sang et al., 2005) from discrete to continuous domains (Belle et al., 2015a;b) with WMC being a state-of-the-art inference approach in many discrete probabilistic models. The tractability of WMI is extensively studied (Zeng et al., 2020c; 2021) and is leveraged for building WMI solvers for structured problems (Zeng & Van den Broeck, 2019; Zeng et al., 2020a;c;b). Existing exact WMI solvers for arbitrarily structured problems include DPLL-based search with numerical (Belle et al., 2015a; Morettin et al., 2017; 2019) or symbolic integration (de Salvo Braz et al., 2016) and

compilation-based algorithms (Kolb et al., 2018; Zuidberg Dos Martires et al., 2019) that use extended algebraic decision diagrams (Sanner & Abbasnejad, 2012; Sanner et al., 2012) as a compilation target. Recent WMI efforts converge in the *pywmi python package* (Kolb et al., 2019).