Collapsed Inference for Bayesian Deep Learning

Zhe Zeng  
Computer Science Department  
University of California, Los Angeles  
zhezeng@cs.ucla.edu

Guy Van den Broeck  
Computer Science Department  
University of California, Los Angeles  
guyvdb@cs.ucla.edu

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 as well as 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. However, computing BMAs is distinctly challenging since it involves marginalizing over posterior parameters, which possess some unusual topological properties such as mode-connectivity. 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.

Existing methods mainly focus on general-purpose MCMC, which can fail to converge, or provides inaccurate few-sample predictions, because running longer sampling chains is computationally expensive, and variational approaches that typically use a mean-field approximation that ignores correlations induced by activations.

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
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 $y p(y | x, w) p(w | D)$. Integration of (a) does not admit a closed-form solution, yet integration of (b) is a close approximation that can be solved exactly and efficiently by WMI solvers.

representation of a conditional distribution over the rest whose marginalization is often tractable. Collapsed samplers are effective at variance reduction in graphical models (Koller and 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 marginalization 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 benchmarks and show significant improvements over other Bayesian deep learning approaches in terms of both uncertainty estimation and accuracy.

2 Bayesian Model Averaging as Weighted Volume Computation

In Bayesian Neural Networks (BNN), given a neural network $f_w$ parameterized by weights $w$, instead of doing inference with deterministic $w$ that optimize objectives such as cross-entropy or mean squared error, Bayesian learning infers a posterior $p(w | D)$ over parameters $w$ after observing data $D$. 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 their results. Formally, given input $x$, the posterior predictive distribution and the expected prediction for a regression task are

$$p(y | x) = \int p(y | x, w) p(w | D) dw, \quad \text{and} \quad E_{p(y|x)}[y] = \int y p(y | x) dy.$$  \hspace{1cm} (1)

For classification, the (most likely) prediction is the class $\arg \max_y p(y | x)$. 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., 2015a).
Example 2. Assume a model \( f_w(x) = \text{ReLU}(w \cdot x) \) with a uniform posterior over the parameter: \( p(w \mid D) = \frac{1}{6} \) with \( w \in [-3, 3] \). Let the input be \( x = 1 \). For parameter \( w \in [-3, 0] \), the model \( f_w \) always predicts 0, and otherwise \( w \in (0, 3) \), it predicts \( w \). Thus, the expected prediction \( \mathbb{E}_{p(y \mid x)}[y] = \frac{1}{2} \int_{w \leq 0} y \cdot p_N(y; w, 1) \, dw + \frac{1}{2} \int_{w > 0} y \cdot p_N(y; w, 1) \, dw \). That is, a summation of two WVC problems \( \mathbb{P}(w \leq 0, 0) \) and \( \mathbb{P}(w > 0, 6) \) with \( \mathbb{P}_\perp = (-3 \leq w \leq 0) \) and \( \mathbb{P}_\top = (0 \leq w \leq 3) \). The BMA integral decomposes into WVC problems with different weights due to the ReLU activation.

These WVC problems have easy closed-form solutions. This is no longer the case in the following.

Example 3. Assume a model \( f_w(x) \) and posterior distribution \( p(w \mid D) \) as in Example 2. Let the predictive distribution \( p(y \mid x, w) \) be a Gaussian distribution \( p_N(y; f_w(x), 1) \) with mean \( f_w(x) \) and variance 1. Given input \( x = 1 \), the expected prediction (Equation 7) is

\[
\mathbb{E}_{p(y \mid x = 1)}[y] = \int_{\mathbb{P}_\perp} y \cdot p_N(y; 0, 1) \cdot \frac{1}{6} \, dy \, dw + \int_{\mathbb{P}_\top} y \cdot p_N(y; w, 1) \cdot \frac{1}{6} \, dy \, dw.
\]

It is a summation of two WVC problems with \( \mathbb{P}_\perp = (-3 \leq w \leq 0) \) and \( \mathbb{P}_\top = (0 \leq w \leq 3) \) \( \land (y \in \mathbb{R}) \), whose joint integral surface is shown in Figure 1a.

These WVC problems do not admit closed-form solutions since they involve truncated Gaussian distributions. Moreover, Figure 1a shows that computing BMA, even in such a low-dimensional parameter space, requires integration over non-convex and multi-modal functions.

2.2 General Reduction of BMA to WVC

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

\[
p(y \mid x) = \sum_{B \in \{0, 1\}^N} \int_{\mathbb{P}_B} p(y \mid x, w) p(w \mid D) \, dw,
\]

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

This general reduction, however, is undesirable since it amounts to a brute-force enumeration that implies a complexity exponential in the number of ReLU activations. Moreover, not all WVC...
WMI is a modeling and inference framework that supports integration in the presence of logical with polynomial densities, we simply use a triangular distribution encoded as a WMI problem pair:

\[ \Delta_{\text{pred}} = \left\{ \begin{array}{ll}
Y \leq Z + \alpha & \Phi_{\text{pred}} = \left\{ \phi_{\ell_1}(Y, Z) = \frac{1-Y+Z}{\alpha} \text{ with } \ell_1 = Y \geq Z \\
Y \geq Z - \alpha & \phi_{\ell_2}(Y, Z) = \frac{1-Y-Z}{\alpha} \text{ with } \ell_2 = Y < Z
\end{array} \right. \]

problems resulting from this reduction are amenable to existing solvers. We will therefore appeal to a framework called weighted model integration (WMI) that allows for a compact representation of these WVC problems, and a characterization of their tractability for WMI solvers (Kolb et al., 2019). This inspires us to approximate BMA by first reducing it to WVC problems and further closely approximating those with tractable WMI problems.

## 3 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). Various WMI solvers have been proposed in recent years (Kolb et al., 2019), ranging from general-purpose ones to others that assume some problem structures to gain scalability. However, even with the reduction from BMA to WVC from the previous section, WMI solvers are not directly applicable. Existing solvers have two main limitations: (i) feasible regions need to be defined by Boolean combinations of linear arithmetic constraints, and (ii) weight functions need to be polynomials. In this section, we show that these issues can be bypassed using a motivating example of how to form a close approximation to BMA using WMI.

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 \((c^T X \leq b)\) with variable \(X\) and constants \(c^T\) and \(b\).

**Example 4.** The ReLU model \(f_w(x)\) of Example 2 can be encoded as an SMT formula (see box).

The encoding of ReLU neural networks into SMT formulas is 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). We propose to use this encoding to define the feasible region of WMI problems. Let \(\models x \models \Delta\) denote the satisfaction of an SMT formula \(\Delta\) by an assignment \(x\), and \(\models x \models \Delta\) be its corresponding indicator function. We formally introduce WMI next.

**Definition 5.** (WMI) Let \(X\) be a set of continuous random variables. A weighted model integration problem is a pair \(M = (\Delta, \Phi)\), where \(\Delta\) is an SMT formula over \(X\) and \(\Phi\) is a set of per-literal weights defined as \(\Phi = \{\phi_{\ell}\}_{\ell \in L}\), where \(L\) is a set of SMT literals and each \(\phi_{\ell}\) is a function defined over variables in literal \(\ell\). The task of weighted model integration is to compute

\[ \text{WMI}(\Delta, \Phi) = \int_{x: x \models \Delta} \prod_{\ell \in L} \phi_{\ell}(x) [x \models \ell] \, dx. \]

The task is to integrate over the weighted assignments of \(X\) that satisfy the SMT formula \(\Delta\).

An approximation to the BMA of Example 3 can be achieved with WMI using the following four steps:

**Step 1. Encoding model** \(f_w(x)\). This has been shown as the SMT formula \(\Delta_{\text{ReLU}}\) in Example 4.

**Step 2. Encoding posterior distribution** \(p(w \mid D)\). The uniform distribution \(p(w \mid D) = \frac{1}{b}\) with \(w \in [-3, 3]\) can be encoded as a WMI problem pair \((\Delta_{\text{pos}}, \Phi_{\text{pos}})\) as follows:

\[ \Delta_{\text{pos}} = -3 \leq W \leq 3 \quad \Phi_{\text{pos}} = \left\{ \phi_{\ell}(W) = \frac{1}{6} \text{ with } \ell = \text{true} \right\} \]

**Step 3. Approximate encoding of predictive distribution** \(p(y \mid w, x)\). Recall that \(p(y \mid w, x) = p_{\mathcal{N}}(y; f_w(x), 1)\) is Gaussian, which cannot be handled by existing WMI solvers. To approximate it with polynomial densities, we simply use a triangular distribution encoded as a WMI problem pair:

\[ \Delta_{\text{pred}} = \left\{ \begin{array}{ll}
Y \leq Z + \alpha & \Phi_{\text{pred}} = \left\{ \phi_{\ell_1}(Y, Z) = \frac{1-Y+Z}{\alpha} \text{ with } \ell_1 = Y \geq Z \\
Y \geq Z - \alpha & \phi_{\ell_2}(Y, Z) = \frac{1-Y-Z}{\alpha} \text{ with } \ell_2 = Y < Z
\end{array} \right. \]

\(^1\)In the literature, WMI is defined on mixed discrete-continuous domains. Since we only work with WMI problems over continuous variables, we ignore the discrete ones in the definition for succinctness.
In this encoding, $\alpha$ is a constant that defines the shape of the triangular distribution. It is obtained by minimizing the $L2$ distance between a standard normal distribution and the symmetric triangular distribution. We visualize this approximation in the right figure.

**Step 4. Approximating BMA by calling WMI solvers.** With the above encodings, the predictive posterior $p(y \mid x)$ (Equation 1) can be computed using two calls to a WMI solver. For example, the uncertainty of a prediction $y = 1$ for input $x = 1$ is

$$p(y = 1 \mid x = 1) = \frac{\text{WMI}(\Delta \cup (Y = 1), \Phi)}{\text{WMI}(\Delta, \Phi)} = 0.164 / 1,$$

where $\Delta = \Delta_{\text{ReLU}} \cup \Delta_{\text{pos}} \cup \Delta_{\text{pred}}$ and $\Phi = \Phi_{\text{pos}} \cup \Phi_{\text{pred}}$. Similarly, the expected prediction $E_p(y \mid x = 1)$ (Equation 1) can be computed using two calls to a WMI solver:

$$E_p(y \mid x = 1) = \frac{\text{WMI}(\Delta, \Phi^*)}{\text{WMI}(\Delta, \Phi)} = 0.752 / 1,$$

where $\Phi^* = \Phi \cup \{ \phi_\ell(Y) = Y \text{ with } \ell = \text{true} \}$. The above formulations also work for unnormalized distributions since the WMI in the denominator serves to compute the partition function.

We visualize the integral surface of the resulting approximate BMA problem in Figure 1b. It is very close to the integral surface of the original BMA problem in Figure 1a. However, it can be exactly integrated using existing WMI solvers while the original one does not admit such solutions. Next, we show how this process can be generalized to a scalable and accurate approximation of BMA.

4 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., 2020b). 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 6. (Collapsed BMA)** Let $(W_s, W_c)$ be a partition of parameters $W$. A collapsed sample is a tuple $(w_s, q)$, where $w_s$ is an assignment to the sampled parameters $W_s$ and $q$ is a representation of the conditional posterior $p(W_c \mid w_s, D)$ over the collapsed parameter set $W_c$. Given collapsed samples $S$, collapsed BMA estimates the predictive posterior and expected prediction as

$$p(y \mid x) \approx \frac{1}{|S|} \sum_{(w_s, q) \in S} \left[ \int p(y \mid x, w) q(w_c) \, dw_c \right], \text{ and}$$

$$E_p(y \mid x) \approx \frac{1}{|S|} \sum_{(w_s, q) \in S} \left[ \int y p(y \mid x, w) q(w_c) \, dw_c \, dy \right].$$

(2)

The size of the collapsed set $W_c$ determines the trade-off between scalability and accuracy. The more parameters in the collapsed set, the more accurate the approximation to BMA is. The fewer parameters in $W_c$, the more efficient the computations of the integrals are since the integration is performed in a lower-dimensional space. Later in our experiments, we choose a subset of weights at the last or second-to-last hidden layer of the neural networks to be the collapsed set. This choice is known to be effective in capturing uncertainty as shown in (Kristiadi et al., 2020; Snoek et al., 2015).

To develop an algorithm to compute collapsed BMA, we are faced with two main design choice questions: (Q1) how to sample $w_s$ 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? Next we provide our answers to these two questions that together give our proposed algorithm CIBER.

4.1 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 over the parameters of the neural network, leveraging the interpretation of SGD as approximate Bayesian inference (Mandt et al., 2017; Chen et al., 2020).
Given a set of parameter samples $W$ from the SGD trajectory, the sample set is defined as $W_s = \{w_s \mid w \in W\}$. For each assignment $w_s$, an approximation $q(W_c)$ to the conditional posterior $p(W_c \mid w_s, D)$ is necessary since the posteriors induced by SGD trajectories are implicit. Next we discuss the choice of approximation to the conditional posterior that are amenable to WMI.

### 4.2 Encoding into WMI Problems

As shown in Section 4 if 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 main challenge is how to approximate the integrand in Equation 2 using an SMT formula and a polynomial weight function in order to obtain a WMI problem amenable to existing solvers.

For the conditional posterior approximation $q(W_c)$, we choose it to be a uniform distribution which can be encoded into a WMI problem as $M_{\text{pos}} = (\Delta_{\text{pos}}, \Phi_{\text{pos}})$ with the SMT formula being $\Delta_{\text{pos}} = \bigwedge_{i \in C} (l_i \leq W_i \leq u_i)$ and weights being $\Phi_{\text{pos}} = \{\phi_{\ell}(W_c) = 1 \mid \ell = \text{true}\}$, 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 as shown in Section 6. The intuition is that uniform distributions are better than a few samples. We further illustrate this point by comparing the predictive distributions of CIBER and HMC in a few-sample setting. Figure 2 shows that even with the same 10 samples drawn from the posterior distribution, since CIBER further approximates the 10 samples with a uniform distribution, it yields a predictive distribution closer to the ground truth than HMC, indicating that using a uniform distribution instead of a few samples forms a better approximation.

For the choice of predictive distribution $p(y \mid x, w)$, 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). For regression, the de facto choice is Gaussian and we propose to use triangular distribution as the approximation, i.e., $p(y \mid x, w) = \frac{1}{r} - \frac{1}{2r}[y - f_w(x)]$, with domain $|y - f_w(x)| \leq r$, and $r := \alpha \sqrt{\sigma^2(x)}$ where the constant $\alpha$ parameterizes the triangular distribution as described in Section 3. Here, $\sigma^2(x)$ is the variance estimate, which can be a function of input $x$ depending on whether the BNN is homoscedastic or heteroscedastic. Then $p(y \mid x, w)$ can be encoded into WMI as:

\[
\Delta_{\text{pred}} = \begin{cases} 
Y - f_w(x) \leq r & \Phi_{\text{pred}} = \left\{ \phi_{\ell_1}(Y, W_c) = \frac{1}{r} - \frac{Y - f_w(x)}{r}, \text{ with } \ell_1 = (Y > f_w(x)) \right\} \\
Y - f_w(x) \geq -r & \phi_{\ell_2}(Y, W_c) = \frac{1}{r} - \frac{f_w(x) - Y}{r}, \text{ with } \ell_2 = (f_w(x) > Y) \end{cases}
\]

Similar piecewise polynomial approximations are adopted for classification tasks when the predictive distributions are induced by softmax functions. Those details are presented in Appendix.

### 4.3 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 7.** Let the SMT formula $\Delta = \Delta_{\text{ReLU}} \land \Delta_{\text{pos}} \land \Delta_{\text{pred}}$, and the set of weights $\Phi = \Phi_{\text{pos}} \cup \Phi_{\text{pred}}$ as defined in Section 4.2. Let the set of weights $\Phi^* = \Phi \cup \{\phi_y(Y) = Y \mid \ell = \text{true}\}$. The integrals in collapsed BMA (Equation 2) can be computed by WMI solvers as

\[
\int p(y \mid x, w) q(w_c) \, dw_c = \text{WMI}(\Delta \land (Y = y), \Phi) / \text{WMI}(\Delta, \Phi), \\
\int y \, p(y \mid x, w) q(w_c) \, dw_c \, dy = \text{WMI}(\Delta, \Phi^*) / \text{WMI}(\Delta, \Phi).
\]

With both question (Q1) and (Q2) answered, we summarize our proposed algorithm CIBER in 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 distribution is available. Both the likelihood and the weight posterior are Gaussian such that the ground-truth posterior predictive distribution is Gaussian as well. We evaluate the posterior predictive distribution estimated by CIBER and Monte Carlo (MC) sampling, both using the same five samples drawn from the weight posterior. Results are show in Figure 3. CIBER approximates the samples with a uniform distribution as posterior \(p(w|\mathcal{D})\) and further approximates the likelihood with a triangular distribution such that the integral \(p(y|x, \mathcal{D}) = \int p(y|x, w)p(w|\mathcal{D})\,dw\) 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 \(I = \int \sigma(f_*)p_N(f_*)\,df_*\) with \(\sigma\) being the sigmoid function and \(f_* = f(x^*; w)\) that amounts to the posterior predictive distribution. We consider a simple case with \(f(x; w) = w \cdot x\) such that the ground-truth integral can be obtained. With a randomly chosen \(x\), 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.

5 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; Heim et al., 2019). Some methods use samples from SGD trajectories to approximate 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 and 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 and Welling, 2013). The SMT formulas considered in this work can be seen as distributional constraints on continuous domain. There are also plenty of work exploring the integration of discrete constraints into neural networks models including the architectures (Ahmed et al., 2022b; 2023) and the loss (Xu et al., 2018; Ahmed et al., 2022a).

WMI Solvers. WMI generalizes weighted model counting (WMC) (Sang et al., 2005), a state-of-the-art inference approach in many discrete probabilistic models, from discrete to mixed discrete-continuous domains (Belle et al., 2015a,b). Recent research on WMI includes its tractability (Zeng et al., 2020a; 2021) and the advancements in WMI solver development. 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; Derkinderen et al., 2020) that use extended algebraic decision diagrams (XADDs) (Sanner et al., 2012) as a compilation target which is a powerful tool for inference on mixed domains (Sanner and Abbasnejad, 2012; Zamani et al., 2012). Some exact WMI solvers aiming to improve efficiency for a certain class of models are proposed such as SMI (Zeng and Van den Broeck, 2019) and MP-WMI (Zeng et al., 2020a) which are greatly scalable for WMI problems that satisfy certain structural constraints. Approximate solvers are also
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

---

1 Code and experiments are available at https://github.com/UCLA-StarAI/CIBER.
posterior, 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 and Welling, 2013), MC Dropout (MCD) (Gal and 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 inference (VI) (Kingma and Welling, 2013), MC Dropout (MCD) (Gal and Ghahramani, 2015), Deep Gaussian Processes (DGP) (Bui et al., 2016), variational structured dropout (VSD) (Nguyen et al., 2021), and Fastfood 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 1 and those for large UCI datasets in Table 2. 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 errors results in Appendix, 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.

6.2 Image Classification

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) that reflects the quality of both uncertainty estimation and prediction accuracy, 2) classification accuracy (ACC), and 3) expected calibration errors (ECE) (Naeini et al., 2015) that show the difference between predictive confidence and accuracy and should be close to zero for a well-calibrated approach.

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 and 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 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 trained on CIFAR-10 to STL-10.
Results. We present the test classification performance on CIFAR-10 and CIFAR-100 in Table 3 and that of transfer learning in Table 4, both of which use VGG-16 networks (Simonyan and Zisserman, 2014). More results using different network architectures are presented in Appendix. With the same number of samples as SWAG, CIBER outperforms SWAG and other baselines in most evaluations and deliver comparable performance otherwise, demonstrating the effectiveness of using collapsed samples in improving uncertainty estimation as well as classification performance.

7 Conclusions

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

Acknowledgments

We would like to thank Yuhang Fan and Kareem Ahmed for helpful discussions. We would also like to thank Wesley Maddox for answering queries on the implementation of baseline SWAG. This work was funded in part by the DARPA Perceptually-enabled Task Guidance (PTG) Program under contract number HR00112220005, NSF grants #IIS-1943641, #IIS-1956441, #CCF-1837129, Samsung, CISCO, a Sloan Fellowship, and a gift from RelationalAI. GVdB discloses a financial interest in RelationalAI. ZZ is supported by an Amazon Doctoral Student Fellowship.

References


A Proofs

Proposition 7 Let the SMT formula $\Delta = \Delta_{\text{ReLU}} \land \Delta_{\text{pos}} \land \Delta_{\text{pred}}$, and the set of weights $\Phi = \Phi_{\text{pos}} \cup \Phi_{\text{pred}}$ as defined in Section 4.2. Let the set of weights $\Phi^* = \Phi \cup \{ \phi_{\ell}(Y) = Y \text{ with } \ell = \text{true} \}$. The integrals in collapsed BMA (Equation 2) can be computed by WMI solvers as

$$\int p(y \mid x, w) q(w_c) \, dw_c = \frac{\text{WMI}(\Delta \land (Y = y), \Phi)}{\text{WMI}(\Delta, \Phi)}$$

and

$$\int y \, p(y \mid x, w) \, q(w_c) \, dw_c \, dy = \frac{\text{WMI}(\Delta, \Phi^*)}{\text{WMI}(\Delta, \Phi)}.$$ 

Proof. By construction, it holds that

$$p(y \mid x, w) \propto \text{WMI}(\Delta_{\text{pred}} \land (Y = y), \Phi_{\text{pred}}) \propto \prod_{\ell \in \mathcal{L}_{\text{pred}}} \phi_{\ell}(y, w_c)^{[y,w_c] \models [\ell]}, \text{ with } (y, w_c) \models \Delta_{\text{pred}} \land \Delta_{\text{ReLU}}$$

and

$$q(w_c) \propto \text{WMI}(\Delta_{\text{pos}} \land (W_c = w_c), \Phi_{\text{pos}}) \propto \prod_{\ell \in \mathcal{L}_{\text{pos}}} \phi_{\ell}(w_c)^{[w_c] \models [\ell]}, \text{ with } w_c \models \Delta_{\text{pos}}$$

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

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

or equivalently,

$$p(y \mid x, w) q(w_c) = \frac{\prod_{\ell \in \mathcal{L}_{\text{pred}} \land \mathcal{L}_{\text{pos}}} \phi_{\ell}(y, w_c)^{[y,w_c] \models [\ell]} \text{ with } (y, w_c) \models \Delta_{\text{ReLU}} \land \Delta_{\text{pred}} \land \Delta_{\text{pos}}}{\text{WMI}(\Delta, \Phi)}.$$ 

By integrating over the collapsed set $W_c$, it further holds that

$$\int p(y \mid x, w) q(w_c) \, dw_c = \frac{\prod_{\ell \in \mathcal{L}_{\text{pred}} \land \mathcal{L}_{\text{pos}}} \phi_{\ell}(y, w_c)^{[y,w_c] \models [\ell]} \, dw_c}{\text{WMI}(\Delta, \Phi)} \text{ with } (y, w_c) \models \Delta$$

which proves the first equation.

Similarly, we have that

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

By integrating over the collapsed set $W_c$ and prediction $y$, it holds that

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

which finishes our proof. \qed
We summarize our proposed algorithm 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.

### Algorithm 1: CIBER

**Input:** input $x$, sampled weights $W$, neural network model $f_w$, prediction ground truth $y^*$

**Output:** predictions and likelihoods

1. Choose a partition $(W_s, W_c)$ for network parameters
2. Derive approximate posterior $q(w_c)$ from sampled weights $\{w_c \mid w \in W\}$. // cf. Section 4.2
3. Encode posterior $q(w_c)$ into WMI problem $M_{pos} = (\Delta_{pos}, \Phi_{pos})$ // cf. Section 4.2
4. $Y \leftarrow \emptyset$, $P \leftarrow \emptyset$ // Initialization
5. for sample $w_s$ in $\{w_s \mid w \in W\}$ do
6. Encode neural network model $f_{ReLU}$ parameterized by $(w_s, W_c)$ into an SMT formula $\Delta_f_w$
7. Encode predictive $p(Y \mid x, w_s, W_c)$ into a WMI problem $M_{pred} = (\Delta_{pred}, \Phi_{pred})$
8. SMT formula $\Delta \leftarrow \Delta_{ReLU} \wedge \Delta_{pos} \wedge \Delta_{pred}$
9. Weights $\Phi \leftarrow \Phi_{pos} \cup \Phi_{pred}$
10. Weights $\Phi^* \leftarrow \Phi \cup \{\phi(Y) = Y \text{ with } \ell = \text{true}\}$
11. Add prediction $y = \text{WMI}(\Delta, \Phi^*) / \text{WMI}(\Delta, \Phi)$ to prediction set $Y$ // cf. Section 4.3
12. Add likelihood $p = \text{WMI}(\Delta \wedge (Y = y^*), \Phi) / \text{WMI}(\Delta, \Phi)$ to set $P$ // cf. Section 4.3
13. return $y = \text{MEAN}(Y)$, $p(y^* \mid x) = \text{MEAN}(P)$

<table>
<thead>
<tr>
<th>Boston</th>
<th>Concrete</th>
<th>Yacht</th>
<th>Naval</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIBER (SECOND)</td>
<td>3.488 ± 1.123</td>
<td>4.880 ± 0.506</td>
<td>0.828 ± 0.241</td>
<td><strong>0.000 ± 0.000</strong></td>
</tr>
<tr>
<td>CIBER (LAST)</td>
<td>3.478 ± 1.128</td>
<td><strong>4.854 ± 0.503</strong></td>
<td><strong>0.752 ± 0.294</strong></td>
<td><strong>0.000 ± 0.000</strong></td>
</tr>
<tr>
<td>SWAG</td>
<td>3.517 ± 0.981</td>
<td>5.233 ± 0.417</td>
<td>0.973 ± 0.375</td>
<td>0.001 ± 0.000</td>
</tr>
<tr>
<td>PCA+ESS (SI)</td>
<td>3.453 ± 0.953</td>
<td>5.194 ± 0.448</td>
<td>0.972 ± 0.375</td>
<td>0.001 ± 0.000</td>
</tr>
<tr>
<td>PCA+VI (SI)</td>
<td><strong>3.457 ± 0.951</strong></td>
<td>5.142 ± 0.418</td>
<td>0.973 ± 0.375</td>
<td>0.001 ± 0.000</td>
</tr>
<tr>
<td>SGD</td>
<td>3.504 ± 0.975</td>
<td>5.194 ± 0.446</td>
<td>0.973 ± 0.374</td>
<td>0.001 ± 0.000</td>
</tr>
<tr>
<td>MCD</td>
<td>2.830 ± 0.170</td>
<td>4.930 ± 0.140</td>
<td>0.720 ± 0.050</td>
<td>0.000 ± 0.000</td>
</tr>
<tr>
<td>VSD</td>
<td>2.640 ± 0.170</td>
<td>4.720 ± 0.110</td>
<td>0.690 ± 0.060</td>
<td>0.000 ± 0.000</td>
</tr>
</tbody>
</table>

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

## B Pseudo Code for CIBER

We summarize our proposed algorithm CIBER, Collapsed Inference Bayesian DEep LeaRning, 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 $p(y \mid x, w)$ can be encoded into a WMI problem as shown below:

$$
\Delta_{pred} = f_w(x) \geq -d \quad \Phi_{pred} = \left\{ \begin{array}{ll}
\phi_{\ell_1}(W_c) & \text{with } \ell_1 = (f_w(x) \leq d) \\
\phi_{\ell_2}(W_c) = 1 & \text{with } \ell_2 = (f_w(x) > d)
\end{array} \right.
$$

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

## 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.
Additional Results. 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.

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.

Additional Results. 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.

## Table 7: Average test RMSE for the large UCI regression task.

<table>
<thead>
<tr>
<th>MODEL</th>
<th>VGG-16</th>
<th>PreResNet-164</th>
<th>WideResNet</th>
<th>VGG-16</th>
<th>PreResNet-164</th>
<th>WideResNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIBER</td>
<td>0.1927 ± 0.0029</td>
<td>0.1352 ± 0.0014</td>
<td>0.1913 ± 0.0029</td>
<td>0.9193 ± 0.0027</td>
<td>0.8144 ± 0.0065</td>
<td>0.7930 ± 0.0065</td>
</tr>
<tr>
<td>SWAG</td>
<td>0.2503 ± 0.0081</td>
<td>0.1459 ± 0.0013</td>
<td>0.1076 ± 0.0009</td>
<td>1.2785 ± 0.0031</td>
<td>1.0703 ± 0.4861</td>
<td>0.6719 ± 0.0035</td>
</tr>
<tr>
<td>SGD</td>
<td>0.3285 ± 0.0139</td>
<td>0.1814 ± 0.0025</td>
<td>0.1294 ± 0.0022</td>
<td>1.7308 ± 0.0137</td>
<td>0.9465 ± 0.0191</td>
<td>0.7958 ± 0.0089</td>
</tr>
<tr>
<td>DKL</td>
<td>0.2621 ± 0.0104</td>
<td>0.1450 ± 0.0042</td>
<td>0.1075 ± 0.0004</td>
<td>1.2780 ± 0.0051</td>
<td>0.7370 ± 0.0265</td>
<td>0.6684 ± 0.0034</td>
</tr>
<tr>
<td>ORTH</td>
<td>0.095</td>
<td>0.120</td>
<td>0.117</td>
<td>0.461</td>
<td>—</td>
<td>4.300 ± 0.200</td>
</tr>
<tr>
<td>FF</td>
<td>0.089 ± 0.002</td>
<td>0.120 ± 0.000</td>
<td>0.120 ± 0.000</td>
<td>0.470 ± 0.010</td>
<td>0.250 ± 0.020</td>
<td>—</td>
</tr>
</tbody>
</table>

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 \( q(w) \), it yields a predictive distribution \( p(y \mid x) \) 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 over-fitting. 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 search to maximize the Gaussian log likelihood using a validation split.

Additional Results. Following the set-up of Izmailov et al. (2020), the test performance on small UCI datasets are averaged over 20 trials and the test performance of large UCI datasets averaged over 10 trials. The test log likelihood results for small UCI datasets are unnormalized while those for large UCI datasets are normalized. Besides the log likelihood results as shown in Section 6.1 root-mean-squared-error (RMSE) results for small UCI datasets are further presented in Table 5 and RMSE results for large UCI datasets are presented in Table 6.

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 \( q(w) \), it yields a predictive distribution \( p(y \mid x) \) 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.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.

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

<table>
<thead>
<tr>
<th>Model</th>
<th>CIFAR-10 VGG-16</th>
<th>CIFAR-10 WideResNet</th>
<th>CIFAR-100 VGG-16</th>
<th>CIFAR-100 WideResNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIBER</td>
<td>93.64 ± 0.09</td>
<td>95.95 ± 0.06</td>
<td>74.71 ± 0.18</td>
<td>79.23 ± 0.25</td>
</tr>
<tr>
<td>SWAG</td>
<td>93.59 ± 0.14</td>
<td>96.09 ± 0.08</td>
<td>76.41 ± 0.10</td>
<td>80.19 ± 0.52</td>
</tr>
<tr>
<td>SGD</td>
<td>93.17 ± 0.14</td>
<td>96.41 ± 0.10</td>
<td>71.95 ± 0.14</td>
<td>82.40 ± 0.16</td>
</tr>
<tr>
<td>SWA</td>
<td>93.61 ± 0.11</td>
<td>96.09 ± 0.08</td>
<td>74.70 ± 0.22</td>
<td>83.65 ± 0.25</td>
</tr>
<tr>
<td>SGLD</td>
<td>95.55 ± 0.15</td>
<td>95.55 ± 0.04</td>
<td>74.02 ± 0.30</td>
<td>80.69 ± 0.05</td>
</tr>
<tr>
<td>KFAC</td>
<td>92.65 ± 0.20</td>
<td>95.49 ± 0.06</td>
<td>72.38 ± 0.23</td>
<td>80.94 ± 0.41</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Model</th>
<th>CIFAR-10 VGG-16</th>
<th>CIFAR-10 WideResNet</th>
<th>CIFAR-100 VGG-16</th>
<th>CIFAR-100 WideResNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIBER</td>
<td>0.0130 ± 0.0011</td>
<td>0.0250 ± 0.0005</td>
<td>0.0168 ± 0.0025</td>
<td>0.1423 ± 0.0029</td>
</tr>
<tr>
<td>SWAG</td>
<td>0.0391 ± 0.0020</td>
<td>0.0214 ± 0.0005</td>
<td>0.0096 ± 0.0006</td>
<td>0.1031 ± 0.0047</td>
</tr>
<tr>
<td>SGD</td>
<td>0.0483 ± 0.0022</td>
<td>0.0255 ± 0.0009</td>
<td>0.0166 ± 0.0007</td>
<td>0.1012 ± 0.0009</td>
</tr>
<tr>
<td>SWA</td>
<td>0.0408 ± 0.0019</td>
<td>0.0032 ± 0.0010</td>
<td>0.0087 ± 0.0002</td>
<td>0.0700 ± 0.0056</td>
</tr>
<tr>
<td>SGLD</td>
<td>0.0082 ± 0.0012</td>
<td>0.0251 ± 0.0012</td>
<td>0.0192 ± 0.0007</td>
<td>0.0363 ± 0.0008</td>
</tr>
<tr>
<td>KFAC</td>
<td>0.0094 ± 0.0005</td>
<td>0.0092 ± 0.0018</td>
<td>0.0060 ± 0.0003</td>
<td>0.0158 ± 0.0014</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Model</th>
<th>VGG-16</th>
<th>PreResNet-164</th>
<th>WideResNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIBER</td>
<td>0.9869 ± 0.0102</td>
<td>0.9684 ± 0.0075</td>
<td>0.8259 ± 0.0148</td>
</tr>
<tr>
<td>SWAG</td>
<td>1.3425 ± 0.0015</td>
<td>1.3842 ± 0.0122</td>
<td>1.0142 ± 0.0032</td>
</tr>
<tr>
<td>SGD</td>
<td>1.6528 ± 0.0390</td>
<td>1.4790 ± 0.0000</td>
<td>1.1308 ± 0.0000</td>
</tr>
<tr>
<td>SWA</td>
<td>1.3993 ± 0.0502</td>
<td>1.3552 ± 0.0000</td>
<td>1.0047 ± 0.0000</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Model</th>
<th>VGG-16</th>
<th>PreResNet-164</th>
<th>WideResNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIBER</td>
<td>72.56 ± 0.23</td>
<td>75.70 ± 0.17</td>
<td>75.02 ± 0.31</td>
</tr>
<tr>
<td>SWAG</td>
<td>72.30 ± 0.11</td>
<td>76.30 ± 0.06</td>
<td>76.96 ± 0.08</td>
</tr>
<tr>
<td>SGD</td>
<td>72.42 ± 0.07</td>
<td>75.56 ± 0.00</td>
<td>76.75 ± 0.00</td>
</tr>
<tr>
<td>SWA</td>
<td>71.92 ± 0.01</td>
<td>76.02 ± 0.00</td>
<td>77.50 ± 0.00</td>
</tr>
</tbody>
</table>

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