Posterior Refinement Improves Sample Efficiency in Bayesian Neural Networks

Agustinus Kristiadi
University of Tübingen
agustinus.kristiadi@uni-tuebingen.de

Runa Eschenhagen
University of Tübingen
runa.eschenhagen@uni-tuebingen.de

Philipp Hennig
University of Tübingen and MPI for Intelligent Systems, Tübingen
philipp.hennig@uni-tuebingen.de

Abstract

Monte Carlo (MC) integration is the *de facto* method for approximating the predictive distribution of Bayesian neural networks (BNNs). But, even with many MC samples, Gaussian-based BNNs could still yield bad predictive performance due to the posterior approximation's error. Meanwhile, alternatives to MC integration tend to be more expensive and biased. In this work, we experimentally show that the key to good MC-approximated predictive distributions is the quality of the approximate posterior itself. However, previous methods for obtaining accurate posterior approximations are expensive and non-trivial to implement. We, therefore, propose to refine Gaussian approximate posteriors with normalizing flows. When applied to last-layer BNNs, it yields a simple *post hoc* method for improving pre-existing parametric approximations. We show that the resulting posterior approximation is competitive with even the gold-standard full-batch Hamiltonian Monte Carlo.

1 Introduction

Predictive uncertainty is crucial in safety-critical systems [1]. Yet, commonly-used neural network (NN) predictive systems are overconfident [2, 3]. Approximate Bayesian inference of NNs, resulting in Bayesian neural networks (BNNs), is a principled way of mitigating this issue [4]. Indeed, even crude approximations of BNNs' posteriors, such as the Laplace approximation [5], can lead to good predictive uncertainty quantification (UQ) performance [6], provided the predictive distributions are correctly computed [7].

A prediction in a BNN amounts to an integration of the likelihood w.r.t. the (approximate) posterior measure. Due to the non-linearity of NNs, no analytic solution to the integral exists, even when the likelihood and the approximate posterior are both Gaussian. A low-cost, unbiased, stochastic approximation can be obtained via Monte Carlo (MC) integration: obtain S samples from the approximate posterior and then compute the empirical expectation of the likelihood w.r.t. these samples. While MC integration is accurate for large S, because of the sheer size of modern (B)NNs, virtually all BNNs use small S (typically 10 to 30, see [8–14, etc.]). Due to its well-known error scaling of $\Theta(1/\sqrt{S})$, intuitively, MC integration with a small S is inadequate for an accurate prediction—yet, this has not been studied in depth for BNNs. Furthermore, while linearization of an NN around a point estimate in the parameter space is an alternative to MC integration [7, 15–17], it is generally costly due to the computation of *per-example* Jacobian matrices.

In this work, we study the quality of MC integration for making predictions in BNNs. We show that few-sample MC integration is inaccurate for even an "easy" integral such as when the domain of

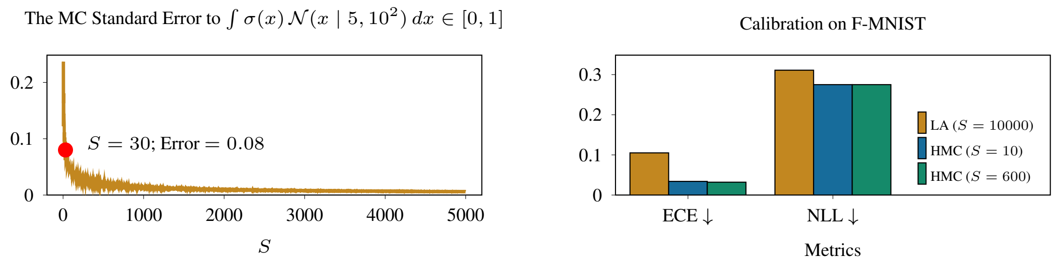


Figure 1: **Left:** Few-sample MC integration is inaccurate for computing classification predictive distribution: the standard error of the MC integration of the logistic-Gaussian integral is large for the commonly used (few) numbers of samples. **Right:** But one can still obtain good predictive performance with a small S if a high-quality posterior approximation, e.g. HMC, is used.

integration is the output space of a binary-classification BNNs, cf. Fig. 1 (left). Further, we show that its alternative, the network linearization, disagrees with large-sample MC integration, making it unsuitable as a general-purpose predictive approximation method, i.e. when the Gauss-Newton matrix is *not* employed.[1] Meanwhile, as indicated by full-batch Markov Chain Monte Carlo methods [18, 19] or even (the Bayesian interpretation of) ensemble methods [20], few-sample MC integration might still be useful for making predictions, provided that the approximate posterior measure is "close enough" to the true posterior—cf. Fig. 1 (right). This implies that one should focus on improving the accuracy of posterior approximations.

Nevertheless, prior methods for obtaining expressive posteriors [e.g., 10, 21] require either significant modification to the NN or storing many copies of the parameters. Moreover, they require training from scratch and thus introduce a significant overhead, which can be undesirable in practical applications. Therefore, we propose a *post hoc* method for "refining" a Gaussian approximate posterior by leveraging normalizing flows [22]. Contrary to the existing normalizing flow methods with *a priori* base distribution (e.g. $\mathcal{N}(0, I)$), the proposed refinement method converges faster with shorter flows, making it cheaper than the naïve application of normalizing flows in BNNs. When used in conjunction with last-layer BNNs, which have been shown to be competitive to their all-layer counterparts [6], the proposed method is simple, cheap, yet competitive to even the gold-standard Hamiltonian Monte Carlo in terms of predictive performance.

To summarize, our contributions are as follows:

(i) We highlight the deficiencies of both few- and many-sample MC integration for computing BNNs' predictive distributions.

(ii) We argue that one must be careful when applying analytic alternatives to MC integration, such as linearization and the probit approximation since they can yield unintended effects.

(iii) We propose a widely-applicable technique for refining parametric posterior approximations by leveraging normalizing flows, which yields a cost-efficient *post hoc* method when used in conjunction with last-layer BNNs.

(iv) We validate the method via extensive experiments and show that refined posteriors are competitive with the much more expensive full-batch Hamiltonian Monte Carlo.

2 Preliminaries

2.1 Bayesian neural networks

Let $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^m$ be an i.i.d. dataset, sampled from some distributions $p(x)$ on $\mathbb{R}^n$ and $p(y \mid x)$ on $\mathbb{R}^c$. A *Bayesian neural network (BNN)* is a neural network (NN) $f_\theta : \mathbb{R}^n \to \mathbb{R}^c$ with a random parameter $\theta \sim p(\theta)$ on $\mathbb{R}^d$ and a likelihood $p(y \mid f_\theta(x))$, along with the associated posterior $p(\theta \mid \mathcal{D}) \propto p(\theta) \prod_{i=1}^m p(y_i \mid f_\theta(x_i))$. However, the exact posterior $p(\theta \mid \mathcal{D})$ is intractable in general

[1]The generalized Gauss-Newton matrix is the *exact* Hessian matrix of a linearized network.

due to its normalization constant. *Approximate BNNs*, which approximate $p(\theta \mid \mathcal{D})$ with only its samples or with a simpler parametric distributions, must thus be employed in most cases.

2.1.1 Posterior approximations

The *Laplace approximation* [LA, 5] is one of the simplest approximation methods for BNNs. The main idea is to construct a local Gaussian approximation to $p(\theta \mid \mathcal{D})$, centered at a *maximum a posteriori (MAP)* estimate $\theta_{\text{MAP}} := \arg\max_\theta p(\theta \mid \mathcal{D})$, with the covariance matrix given by the negative inverse-Hessian $(-\nabla^2_\theta \log p(\theta \mid \mathcal{D})|_{\theta_{\text{MAP}}})^{-1}$. Since the MAP estimation is the standard training procedure for NNs, the LA can be efficiently applied on top of pre-trained NNs [6], especially due to recent efficient second-order optimization libraries [23, 24].

Variational Bayes [VB, 25] assumes a family of simple approximate distributions over the parameter space, e.g. $M := \{q(\theta) := \mathcal{N}(\theta \mid \mu, \Sigma) : (\mu, \Sigma) \in \mathbb{R}^{d+d^2}\}$, and finds a $q \in M$ that is the closest to $p(\theta \mid \mathcal{D})$ by minimizing the reverse Kullback-Leibler (KL) divergence $D_{\text{KL}}(q(\theta), p(\theta \mid \mathcal{D}))$ over M. Unlike the LA, VB is not constrained to capture just the local mode of the log-posterior landscape. However, VB is not *post hoc* and is generally not as straightforward to implement.

Unlike the previous two approximations, *Markov Chain Monte Carlo (MCMC)* methods do not obtain parametric approximations to the posterior. Instead, they collect samples from asymptotically the true posterior and use them to approximate integrals under the posterior measure. A popular MCMC method is the *Hamiltonian Monte Carlo (HMC)* method [18, 19], where sampling processes are casted as Hamiltonian dynamics. Nevertheless, MCMC methods are generally expensive since they require full batches of data and storage of many copies of the networks' parameters.

2.1.2 Predictive approximations

Let $q(\theta)$ be an approximate posterior of a NN f_θ under $\mathcal{D}$. Given a test point x_*, the prediction y_* is distributed as $p(y_* \mid x_*, \mathcal{D}) := \int_{\mathbb{R}^d} p(y_* \mid f_\theta(x_*))\, q(\theta)\, d\theta$. However, even when both $p(y_* \mid f_\theta(x))$ and $q(\theta)$ are Gaussians, this integral does not have an analytic solution due to the nonlinearity of f_θ. Further approximations on top of the posterior approximation are thus necessary.

Monte Carlo integration The *Monte Carlo (MC) integration* is a simple technique to approximate integrals under probability measures. Specifically, in our case,

$$p(y_* \mid x_*, \mathcal{D}) \approx \frac{1}{S} \sum_{s=1}^{S} p(y_* \mid f_{\theta_s}(x_*)); \qquad \text{with } \theta_s \sim q(\theta) \quad \forall s = 1, \dots, S, \tag{1}$$

where S is the number of samples from $q(\theta)$. It is easy to show that MC integration is unbiased with error given by the *standard error* around the mean that scales like $\Theta(1/\sqrt{S})$, see e.g. Murphy [26, Section 2.7.3]. That is, MC integration (slowly) becomes more accurate as the number of samples S increases. In the realm of BNNs, however, S is often chosen to be a small number, e.g. $S = 20$, due to the computational cost of evaluating f_θ.

Analytic approximations Since a Gaussian $\mathcal{N}(\theta \mid \mu, \Sigma)$ is the *de facto* choice for $q(\theta)$, an analytic approximation of the marginal output distribution[2] $p(f(x) \mid x, \mathcal{D})$ can be obtained by linearizing f_θ around μ. Specifically, we perform a first-order Taylor expansion $f_\theta(x) \approx f_\mu(x) + J_f(x) \cdot (\theta - \mu)$ where $J_f(x) \in \mathbb{R}^{c \times d}$ is the Jacobian matrix of the network output w.r.t. θ at μ and $\cdot$ denotes matrix multiplication. Under this scheme, denoting $f_* = f(x_*)$, we thus have for each test point x_*

$$p(f_* \mid x_*, \mathcal{D}) \approx \mathcal{N}(f_* \mid f_\mu(x_*), J_f(x_*) \Sigma J_f(x_*)^\top) =: \mathcal{N}(f_* \mid f_\mu(x_*), S(x_*)).$$

One can then use this distribution to obtain the predictive distribution $p(y_* \mid x_*, \mathcal{D}) \approx \int_{\mathbb{R}^c} p(y \mid f_*)\, p(f_* \mid x_*, \mathcal{D})\, df_*$, which again can be approximated via MC integration.

Nevertheless, there exist analytic approximations to this integral [27, 15, 28, 29]. Specifically for binary classification where $p(y \mid f_*) = \sigma(f_*)$ for the logistic function σ, one can use the so-called *probit approximation* [27, 15]: $p(y = 1 \mid x_*, \mathcal{D}) \approx \sigma(f_\mu(x_*)/\sqrt{1 + \pi/8\, S(x_*)})$.[3] Moreover, for

[2]Where θ has been marginalized out.

[3]Here, $S(x_*) \in \mathbb{R}_{>0}$ since $f(x_*)$ is a real-valued random variable.

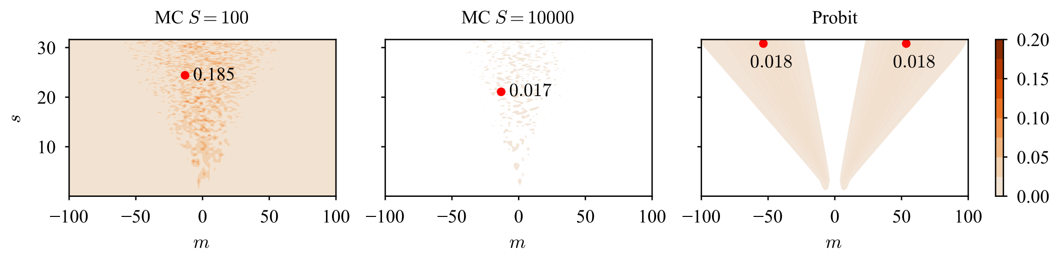


Figure 2: The (absolute) errors of the MC integration and the probit approximation for computing $I(m, s)$ across different values of m and s. A trapezoid method with 20000 evaluation points is used as a gold-standard baseline. Red dots indicate the maximum errors. White indicates (near) zero error.

the multi-class case, we can use the *multi-class probit approximation* [MPA, 28]:

$$p(y \mid x_*, \mathcal{D}) \approx \text{softmax}\left(f_\mu(x_*)/\sqrt{1 + \pi/8 \text{ diag}(S(x_*))}\right), \tag{2}$$

where the vector division is taken component-wise.

2.2 Normalizing flows for Bayesian inference

Let $F : \mathbb{R}^d \to \mathbb{R}^d$ be a diffeomorphism[4] in the sense of $C^k(\mathbb{R}^d)$ for a fixed $k \geq 1 \in \mathbb{N}$. If p is a density on $\mathbb{R}^d$,[5] then the density $\widetilde{p}$ of the random variable $\widetilde{\theta} = F(\theta)$ is given by $\widetilde{p}(\widetilde{\theta}) = p(F^{-1}(\widetilde{\theta}))\,|\det J_{F^{-1}}(\widetilde{\theta})|$, where $J_{F^{-1}} = J_F^{-1}$ is the Jacobian matrix of F^{-1}. A *normalizing flow (NF)* is a method exploiting this relation [22, 30, 31]. Specifically, it is a way of constructing a sophisticated diffeomorphism F, by composing several simple parametric ones. The resulting diffeomorphism thus transforms a simple density into a complicated one in a tractable manner.

Let $F_{\phi_l} : \mathbb{R}^d \to \mathbb{R}^d$ be a diffeomorphism parametrized by $\phi_l \in \mathbb{R}^k$ with a known inverse $F_{\phi_l}^{-1}$ and a known $d \times d$ Jacobian matrix $J_{F_{\phi_l}}$ for $l = 1, \dots, \ell$. Writing $F_\phi := F_{\phi_\ell} \circ \dots \circ F_{\phi_1}$ and $\phi := (\phi_l)_{l=1}^{\ell}$, then the change-of-density formula becomes

$$\widetilde{p}_\phi(\widetilde{\theta}) = p\left(F_\phi^{-1}(\widetilde{\theta})\right) \left|\prod_{l=1}^{\ell} \det J_{F_{\phi_l}^{-1}}(\widetilde{\theta})\right|.$$

Given a target posterior density p^*, the goal of a normalizing flow is then to estimate ϕ s.t. $\widetilde{p}_\phi$ is close to p^*. In Bayesian inference, the reverse KL-divergence $D_{\text{KL}}(\widetilde{p}_\phi, p^*)$ is often used.

3 Pitfalls of BNNs' Approximate Predictive Distributions

In this section, we study the failure modes of BNNs, especially last-layer ones, for predictive uncertainty calibration. We shall show that there are two parts contributing to the inaccuracy in the predictive distribution of a BNN: (i) the weight-space approximation and (ii) the integration method to do Bayesian model averaging. We shall observe that while the accuracy of the integration method is impactful (i.e., the number of samples in MC integration), it is less important than the quality of the posterior approximation itself. That is, higher-quality weight-space approximations—even in last-layer BNNs—yield samples that are more efficient: Few-sample MC integration can already yield good predictive performance in this case.

3.1 Accurate MC integration requires many samples

We begin our analysis with the simplest yet practically relevant case. Let $x_* \in \mathbb{R}^n$ and $f_* := f(x_*)$, with $p(f_* \mid x_*, \mathcal{D}) = \mathcal{N}(f_* \mid m, s^2)$ for some $m \in \mathbb{R}$ and $s^2 \in \mathbb{R}_{>0}$. We are interested in computing

[4]A smooth function with a smooth inverse.

[5]All densities considered in this paper are assumed to be w.r.t. the Lebesgue measure.

the integral (cf. Fig. 3)

$$I(m, s) := p(y \mid x_*, \mathcal{D}) = \int_{\mathbb{R}} \sigma(f_*) \, \mathcal{N}(f_* \mid m, s^2) \, df_*.$$

Note that this integral is prevalent in practice, e.g. in linearized or last-layer classification BNNs [4, 7, 17, 32].

We compare MC integration with $S = 100$ against the probit approximation, using a trapezoid quadrature with 20000 evaluation points to represent a gold-standard baseline. That is, we compute the discrepancy $|\widetilde{I}(m, s) - I_*(m, s)|$ where $\widetilde{I}(m, s)$ is $I(m, s)$ computed either with MC integration or the probit approximation, and $I_*(m, s)$ obtained via the trapezoid method.

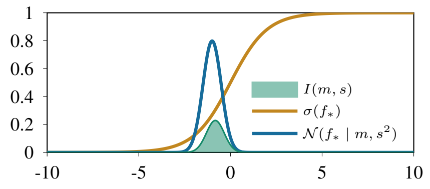


Figure 3: The intuition of $I(m, s)$.

The results are in Fig. 2: Even with $S = 100$ samples—larger than the usual $S = $ 10-30—MC integration is inaccurate. Its error can be as high as 0.18, which is substantial considering $I(m, s) \in [0, 1]$. As S increases beyond 10000, MC integration improves and eventually overtakes the probit approximation, as to be expected given its theoretical guarantees. This highlights the flaw of few-sample MC integration—accurate MC integration generally requires a large number of samples.

3.2 Many-sample MC integration is not sufficient

However, even with a large S, MC integration can still fail to yield good predictive performance in BNNs. This can happen when the approximation $q(\theta)$ used in (1) is an inaccurate approximation of $p(\theta \mid \mathcal{D})$—virtually the case for every parametric BNN. Thankfully, there is some evidence that the error of MC integration might be relatively small in comparison to the error generated by crude posterior approximations. As an extreme example, Deep Ensemble [20] and its variants, even though they perform MC integration with a small number of samples (usually $S = 5$), generally yield better approximations to the predictive distributions. This is perhaps due to their multimodality, i.e. due to their finer-grained posterior approximations.

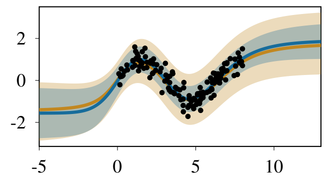
Table 1: The expected calibration error (ECE) and negative log-likelihood (NLL) of LA ($S = 10000$) and HMC ($S = 10$).

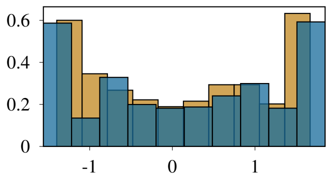
Methods	ECE ↓	NLL ↓
F-MNIST		
LA	10.5±0.4	0.311±0.005
HMC	**3.4**±0.2	**0.275**±0.004
CIFAR-10		
LA	4.9±0.2	0.161±0.001
HMC	**4.2**±0.2	**0.158**±0.001

To show this more concretely, consider the following experiment. We take Fashion-MNIST (F-MNIST) pre-trained LeNet and CIFAR-10 pre-trained WideResNet-16-4. For each case, we perform a last-layer Laplace approximation with the exact Hessian and a full-batch NUTS-HMC [19], under the same prior and likelihood. We find in Table 1 that HMC, even with few samples, yields better-calibrated predictive distribution than the LA with three orders of magnitude more samples. This finding validates the widely-believed wisdoms [9, 10, 14, 32, etc.] that highly accurate posterior approximations are most important for BNNs. Furthermore, this also shows that with a fine-grained posterior approximation, the predictive performance of a BNN is less sensitive to the number of MC samples, leading to better test-time efficiency.

3.3 Analytic alternatives to MC integration are not the definitive answers

Of course, fine-grained posterior approximations are expensive. So, a natural question is whether one can keep a cheap but crude posterior approximation by replacing MC integration. Network linearization seems to be a prime candidate to replace MC integration in the case of Gaussian approximations [7, 15]. But it poses several problems: First, it requires relatively expensive computation of the per-example Jacobian $J_f(x_*)$, where *for each* test point x_*, one must store the associated $c \times d$ matrix, where d could easily be tens of millions (e.g. in ResNets) and c could be in the order of thousands (e.g. ImageNet). Second, while Immer et al. [7] argued that network linearization is the correct way to make predictions in Gauss-Newton-based Gaussian approximations, it is not generally applicable. We show this in Fig. 4: Everything else being equal, MC integration ($S = 10000$) and linearization yield different results especially in terms of predictive uncertainty. Since one should


(a) Predictive distributions.


(b) Densities of pred. means.

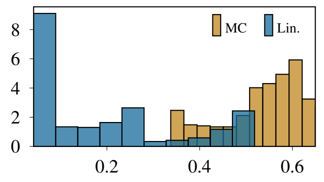

(c) Densities of pred. std. dev.

Figure 4: Predictive distributions, computed via MC integration ($S = 10000$) and network linearization, of a BNN with a weight-space Gaussian approximate posterior (an all-layer LA with the exact Hessian on a two-layer NN under a toy regression dataset).

prefer MC integration in this many-sample regime, linearization is thus not accurate for the general cases.

Moreover, we show that the multi-class probit approximation (MPA), which is often used on top of a linearized output distribution $p(f(x_*) \mid x_*, \mathcal{D})$ to obtain the predictive distribution $p(y_* \mid x_*, \mathcal{D})$ [6, 17, 32], can also obscure the predictive performance, albeit often for the better [6, 32]. To that end, we use the same experiment setting as Section 3.2 and Table 1, and compare the MPA against MC integration. The results are in Table 2.

Table 2: Calibration of MC integration $S = 10000$ and the multi-class probit approximation.

Methods	ECE ↓	NLL ↓
F-MNIST		
MPA	**3.3**±0.2	**0.281**±0.002
MC	10.5±0.4	0.311±0.005
CIFAR-10		
MPA	**3.8**±0.1	**0.161**±0.001
MC	4.9±0.2	**0.161**±0.001

Indeed, we see that the MPA yields better-calibrated predictive distributions. However, this result should be taken with a grain of salt: The MPA ignores the off-diagonal elements of the covariance of $p(f(x_*) \mid x_*, \mathcal{D})$, as shown in (2)—further detail is in Appendix A. That is, the MPA "biases" the predictive distribution $p(y \mid x_*, \mathcal{D})$ since it generally assumes that f_* has lower uncertainty than it actually has. And since $p(f_* \mid x_*, \mathcal{D})$ is usually induced by the LA or VB which are often underconfident on large networks [7, 33, 34], the bias of MPA towards overconfidence thus counterbalances the underconfidence of $p(f_* \mid x_*, \mathcal{D})$. Therefore, in this case, the MPA can yield better-calibrated predictive distributions than MC integration [32, 6]. Nevertheless, it can also fail even in simple cases such as in Fig. 5,[6] where the MPA yields underconfident predictions even near the training data (notice the lighter shade in Fig. 5, bottom). Moreover, the structured nature of the error of the MPA (cf. Fig. 2 for the binary case) might also contribute to the cases where the MPA differs from MC integration. Both examples above thus highlights the need for careful consideration when analytic alternatives to MC integration are employed.

4 Refining Gaussian Approximate Posteriors

The previous analysis indicates the importance of accurate approximate posteriors $q(\theta)$ for BNNs' predictive distributions. However, the current *de facto* way of obtaining accurate posterior approximations, HMC [18, 19], are *very* expensive since they require a full-batch of data in their updates [35]. While mini-batch versions of HMC exist, they do not seem to yield as good of results as their full-batch counterparts. Indeed, Daxberger et al. [6] showed that a well-tuned *last-layer* LA can outperform a state-of-the-art *all-layer* stochastic-gradient MCMC method [21]. Furthermore, these sample-based methods—both the full- and mini-batch versions, along with deep ensembles and their variants—are costly in terms of storage since one effectively must store S copies of the network's parameters. We, therefore, propose a simple *post hoc* technique for "refining"

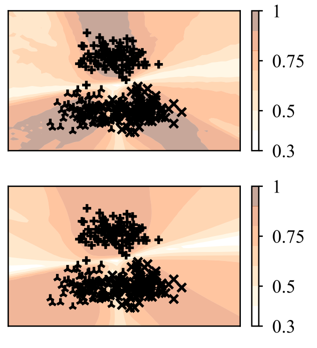


Figure 5: Confidence estimate of MC integration (**top**) and the MPA (**bottom**).

[6]An all-layer full-Hessian LA on a three-layer `tanh` network is used.

Gaussian approximations using normalizing flows. The resulting method is thus analytic yet can produce high-quality samples.

Let f_θ be a NN equipped with a Gaussian approximate posterior $q(\theta) = \mathcal{N}(\theta \mid \mu, \Sigma)$ (e.g., via a LA, VB, or SWAG [13]) on the parameter space under a dataset $\mathcal{D}$. Given a NF F_ϕ of length ℓ with parameter ϕ, we obtain the *refined posterior* by

$$\widetilde{q}_\phi(\widetilde{\theta}) = q(F_\phi^{-1}(\widetilde{\theta})) \left|\det J_{F_\phi}(\widetilde{\theta})\right|^{-1}. \tag{3}$$

Then, a *refinement* of $q(\theta)$ amounts to minimizing the reverse KL-divergence to the true posterior, using the evidence lower bound (ELBO) as a proxy ($\mathbb{H}$ below is the entropy functional):

$$\phi^* = \arg\max_\phi \mathop{\mathbb{E}}_{\widetilde{\theta} \sim \widetilde{q}_\phi} \left[\log p(\mathcal{D} \mid f_{\widetilde{\theta}}) + \log p(\widetilde{\theta})\right] + \mathbb{H}\left[\widetilde{q}_\phi\right], \tag{4}$$

Given a refined posterior $q_{\phi^*}(\widetilde{\theta})$ and a test point x^*, we can obtain the predictive distribution via MC integration:

$$p(y^* \mid x^*, \mathcal{D}) \approx \frac{1}{S} \sum_{s=1}^{S} p\left(y^* \mid f_{\widetilde{\theta}_s}(x^*)\right); \;\text{ where }\; \widetilde{\theta}_s = F_{\phi^*}(\theta_s); \;\; \theta_s \sim q(\theta) \quad \forall s = 1, \dots, S.$$

Due to the expressiveness of NFs [36], we can expect based on the previous analysis that a large S is not necessary here to obtain good predictive performance. We shall validate this in Section 6. Last but not least, this refinement technique is especially useful for last-layer BNNs since their parameter spaces typically have manageable dimensions, cf. Section 6.4.[7]

5 Related Work

Normalizing flows have previously been used for approximate Bayesian inference in BNNs. An obvious way to do so, based on the flexibility of NFs in approximating any density [36], would be to apply a NF on top of the standard normal distribution $\mathcal{N}(\theta \mid 0, I)$ to approximate $p(\theta \mid \mathcal{D})$, see e.g. Izmailov et al. [38] and the default implementation of variational approximation with NF in Pyro [39]. We show in Section 6.3 that the subtle difference that we make—using an approximate posterior instead of an *a priori* distribution—is more cost-effective. In a more sophisticated model, Louizos and Welling [10] combine VB with NF by assuming a compound distribution on each NN's weight matrix and using the NF to obtain an expressive mixing distribution. However, their method requires training both the BNN and the NF jointly from scratch. In an adjacent field, Maroñas et al. [40] use NFs to transform Gaussian process priors.

Posterior refinement in approximate Bayesian inference has recently been studied. Immer et al. [7] proposes to refine the LA using a Gaussian-based VB and Gaussian processes. However, this implies that they still assume a Gaussian posterior. To obtain a non-Gaussian posterior, Miller et al. [41] form a mixture-of-Gaussians approximation by iteratively adding component distributions. But, at *every* iteration, their methods require a full ELBO optimization, making it costly for BNNs. A lower-cost LA-based alternative to their work has also been proposed by Eschenhagen et al. [32]. These methods have high storage costs since they must store many high-dimensional Gaussians. By contrast, our method only does an ELBO optimization once and only requires the storage of the base Gaussian and the parameters of a NF. In a similar vein, Havasi et al. [42] perform an iterative ELBO optimization for refining a sample of a variational approximation. But, this inner-loop optimization needs to be conducted each time a sample is drawn, e.g., during MC integration at test time. Our method, on the other hand, only requires sampling from a Gaussian and evaluations of a NF, cf. (4).

While our results also reaffirm the widely held belief that single-mode Gaussian approximations are inferior to more fine-grained counterparts, our conclusion differs slightly from that of Wilson and Izmailov [43]: They argue "[...] Ultimately, the goal is to accurately compute the predictive distribution, rather than find a generally accurate representation of the posterior. [...]" and subsequently propose a multi-modal approximation to the true posterior. Here, we show that finding an accurate (non-Gaussian) weight-space posterior approximation can still be a worthy goal, even when only utilizing a single mode of the true posterior.

[7]E.g., WideResNets' last-layer features' dimensionality typically range from 256 to 2048 [37].

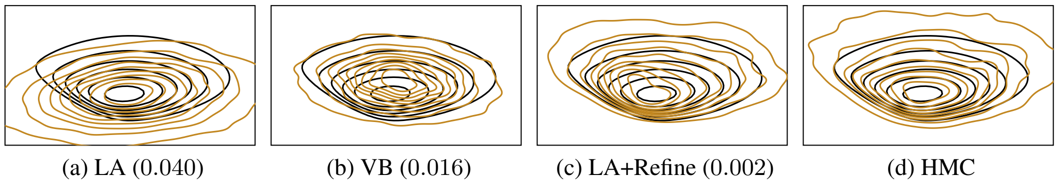


Figure 6: Comparison of approximate posterior densities, visualized via a kernel density estimation, on a 2D logistic regression problem. **Black contour:** The exact posterior contour, up to a normalizing constant. **Colored contour:** The kernel density estimate obtained from the posterior samples of each method. **Number:** The MMD distance to HMC's samples; lower is better.

6 Experiments

Code available at: `https://github.com/runame/laplace-refinement`. See Appendix B and Appendix C for additional details.

6.1 Setups

Datasets We validate our method using standard classification datasets: Fashion-MNIST (F-MNIST), CIFAR-10, and CIFAR-100.[8] For the out-of-distribution (OOD) detection task, we use three standard OOD test sets for each in-distribution dataset, see Appendix B.2 for the full list. Finally, for the toy logistic regression experiment, a dataset of size 50 is generated by sampling from a bivariate, bimodal Gaussian.

Network architectures For the F-MNIST experiments, we use the LeNet-5 architecture [44]. Meanwhile, for the CIFAR experiments, we use the WideResNet architecture with a depth of 16 and widen factor of 4 [WRN-16-4, 37]. For the NF, we use the radial flow [22] which is among the simplest and cheapest non-trivial NF architectures.

Baselines We focus on last-layer Bayesian methods to validate the refinement technique.[9] We use the LA to obtain the base distribution for our refinement method—following recent practice [6], we tune the prior precision via *post hoc* marginal likelihood maximization. The No-U-Turn-Sampler Hamiltonian Monte Carlo [HMC, 18, 19] with 600 samples is used as a gold-standard baseline—all HMC results presented in this paper are well-converged in term of the Gelman-Rubin diagnostic [45], see Appendix B. Furthermore, we compare our method against recent, all-layer BNN baselines: variational Bayes with the Flipout estimator [VB, 46] and cyclical stochastic-gradient HMC [CSGHMC, 21]. For all methods, we use MC integration with 20 samples to obtain the predictive distribution, except for HMC and CSGHMC where we use $S = 600$ and $S = 12$, respectively. We use prior precisions of 510 and 40 for the last-layer F-MNIST and CIFAR experiments, respectively. These prior precisions are obtained via grid search on the respective HMC baseline, maximizing validation log-likelihood. Additionally, we use MAP and its temperature-scaled version (MAP-Temp) as non-Bayesian baselines. More implementation details are in Appendix B.

Metrics We use standard metrics to measure both calibration and OOD-detection performance. For the former, we use the negative log-likelihood (NLL) and the expected calibration error [ECE, 47]. For the latter, we employ the false-positive rate at 95% true-positive rate (FPR95). Finally, to measure the closeness between an approximate posterior to the true posterior, we use the maximum-mean discrepancy [MMD, 48] distance between the said approximation's samples to the HMC samples, i.e. we use HMC as a proxy to the true posterior.

6.2 Toy example

We visualize different approximations to the posterior of the toy logistic regression problem in Fig. 6. Note that the true posterior density is non-symmetric and non-Gaussian.

[8]Additionally, see Appendix C for results on text classifications.

[9]Results for an all-layer network are in Table 6 (Appendix C).

Table 3: In-distribution calibration performance. All-layer baselines are marked with an asterisk.

	F-MNIST		CIFAR-10		CIFAR-100	
Methods	**MMD ↓**	**NLL ↓**	**MMD ↓**	**NLL ↓**	**MMD ↓**	**NLL ↓**
MAP	1.093±0.003	0.3116±0.0049	0.438±0.001	0.1698±0.0009	0.416±0.001	0.9365±0.0063
MAP-Temp	1.093±0.003	0.2694±0.0025	0.438±0.001	0.1545±0.0007	0.416±0.001	0.9155±0.0047
VB*	-	0.2673±0.0016	-	0.2823±0.0020	-	1.2638±0.0059
CSGHMC*	-	0.2854±0.0018	-	0.2101±0.0033	-	0.9892±0.0070
LA	0.418±0.002	0.3076±0.0046	0.299±0.001	0.1672±0.0009	0.063±0.000	0.9865±0.0057
LA-Refine-1	0.356±0.004	0.2752±0.0031	0.346±0.000	0.1616±0.0007	0.063±0.000	0.9548±0.0062
LA-Refine-5	0.022±0.002	0.2699±0.0028	0.290±0.000	0.1582±0.0007	0.018±0.000	0.9073±0.0062
LA-Refine-10	0.013±0.002	0.2701±0.0028	0.130±0.001	0.1577±0.0008	0.019±0.000	0.9037±0.0058
LA-Refine-30	0.012±0.002	0.2701±0.0028	0.002±0.000	0.1581±0.0008	0.020±0.000	0.9035±0.0055
HMC	0.000±0.000	0.2699±0.0028	0.000±0.000	0.1581±0.0008	0.000±0.000	0.8849±0.0047

The LA matches the weight-space posterior mean, but is inaccurate the further away from it. It even assigns probability mass on what are supposed to be low-density regions. While VB yields a more accurate result than the LA, it still assigns some probability mass on low-density regions due to the symmetry of the Gaussian approximation. Furthermore, it is unable to match the posterior's mode well. The proposed refinement method, on the other hand, is able to make the LA more accurate—it yields a skewed, non-Gaussian approximation, similar to HMC.

We further quantify the previous observation using the MMD distance between each approximation's samples and HMC's samples. The LA, as expected, obtain the worst weight-space MMD. While VB is better than the LA with an MMD, the refined LAs achieve even better MMDs. This quantifies the previous visual observation.

6.3 Image classification

We present the calibration results in Table 3 using the LA and HMC as baselines, which represent two "extremes" in the continuum of posterior approximations. As has previously shown by e.g. Guo et al. [49], the vanilla MAP approximation yields uncalibrated, low-quality predictive distributions in terms of NLL and ECE. While the LA is a cheap way to improve MAP predictive, its predictive performance is still lagging behind HMC. By refining it with a NF, the LA becomes even better and closer to the HMC predictive. We also note that one does not need a complicated or long (thus expensive) NF to achieve these improvements. Furthermore, we observe a positive correlation between posterior-approximation quality (measured via MMD) and predictive quality. Considering that $S = 20$ is used, this validates our hypothesis that one can "get away" with fewer MC samples when accurate weight-space posterior approximations are employed.

Table 4: OOD detection in terms of FPR95 (in percent, lower is better), averaged over three test sets and five seeds. All-layer baselines are asterisk-marked.

Methods	**CIFAR-10**	**CIFAR-100**
VB*	62.9±2.0	80.8±1.0
CSGHMC*	58.7±1.6	79.3±1.0
LA	49.2±2.4	79.6±1.0
LA-Refine-1	47.7±2.1	77.3±0.7
LA-Refine-5	46.8±2.2	77.8±0.7
LA-Refine-10	46.2±2.3	77.8±0.7
LA-Refine-30	46.1±2.3	77.9±0.8
HMC	46.0±2.3	77.8±0.9

Moreover, we present out-of-distribution (OOD) data detection in Table 4. We observe that while the last-layer LA baseline can already be better than all-layer baselines—as also observed by Daxberger et al. [6]—refining it can yield better results: Even with a small NF, e.g., $\ell = 5$, the OOD detection performance of the refined LA is close to that of the gold-standard HMC.

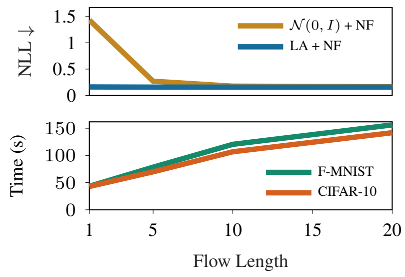


Figure 7: Calibration and wall-clock refinement time vs. flow length.

Finally, we show that it is indeed desirable to do *refinement*, i.e. using an approximate posterior as the base distribution of the NF, instead of starting from scratch, i.e. starting from a data-independent distribution such as $\mathcal{N}(0, I)$. As shown in Fig. 7, starting from an approximate posterior yields better predictive distributions faster than when $\mathcal{N}(0, I)$ is used as the base distribution of the NF. This is particularly important since the computational cost of a NF depends on its length: We see a 53% increase in training time from $\ell = 5$ to $\ell = 10$—the latter is required for the *a priori* NF approximation to yield similar predictive performance to the refined posterior.

6.4 Costs

The proposed refinement technique is *post hoc* and cheap when applied to last-layer BNNs. Suppose one already has a last-layer Gaussian approximate posterior. Using a standard consumer GPU (Nvidia RTX 2080Ti), each epoch a length-5 NF's optimization takes around 3.4 seconds. From our experiments, we found that a low number of epochs (we use 20) is already sufficient for improving a crude approximate posterior. Thus, the entire refinement process is quick, especially when compared to MAP estimation, ELBO optimization, or HMC sampling. Additional details in Appendix C.3.

7 Concluding Remarks

7.1 Limitations

A NF is a composition of diffeomorphisms—it is bijective and invertible. Since we use the NF to transform the density on the parameter space of a NN, which has tens of millions of dimensions, the proposed refinement framework is thus costly for *all-layer* BNNs or when the number of last-layer parameters is prohibitively large. Interesting future works to alleviate this limitation are to refine a posterior in the subspace of the parameter space [38] or in the "pruned" parameter space [50].

Moreover, by refining a Gaussian with a NF, one loses the convenient properties associated with Gaussians, making the resulting BNN unsuitable for cheaply addressing tasks such as continual learning [51] and model selection [52]. Nevertheless, this is not unique to our method—any sample-based method such as MCMC and ensembles are unsuitable for such tasks.

7.2 Conclusion

We have shown that MC integration tends to produce underperforming predictive distributions under a crude posterior approximation, even with a large number of samples. While recently analytic approximations such as network linearization and the probit approximation have been extensively used in BNNs, we show that one must be careful with them since they are biased and do not have theoretical guarantees like MC integration. Inspired by the high predictive performance of full-batch HMC—even when a low number of samples are used—we confirm the widely-held belief that a finer-grained posterior approximation is a key to achieving better predictive performance in BNNs. To that end, we proposed a *post hoc* method for last-layer parametric BNNs, based on normalizing flows, that can cheaply refine crude posterior approximations. In conjunction with the *post hoc* last-layer Laplace approximation, the proposed method can thus give practitioners similar predictive performance to full-batch, expensive HMC in a cost-efficient manner.

Acknowledgments and Disclosure of Funding

The authors gratefully acknowledge financial support by the European Research Council through ERC StG Action 757275 / PANAMA; the DFG Cluster of Excellence "Machine Learning - New Perspectives for Science", EXC 2064/1, project number 390727645; the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A); and funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg. The authors are also grateful to Felix Dangel for feedback. AK is also grateful to the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for support.

References

[1] Alexander A Alemi, Ian Fischer, and Joshua V Dillon. Uncertainty in the variational information bottleneck. In *UAI*, 2018.

[2] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, 2015.

[3] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *CVPR*, 2019.

[4] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being Bayesian, even just a bit, fixes overconfidence in ReLU networks. In *ICML*, 2020.

[5] David JC MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3), 1992.

[6] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux–effortless Bayesian deep learning. In *NeurIPS*, 2021.

[7] Alexander Immer, Maciej Korzepa, and Matthias Bauer. Improving predictions of Bayesian neural nets via local linearization. In *AISTATS*, 2021.

[8] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *ICML*, 2015.

[9] Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix Gaussian posteriors. In *ICML*, 2016.

[10] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational Bayesian neural networks. In *ICML*, 2017.

[11] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable Laplace approximation for neural networks. In *ICLR*, 2018.

[12] Kazuki Osawa, Siddharth Swaroop, Mohammad Emtiyaz E Khan, Anirudh Jain, Runa Eschenhagen, Richard E Turner, and Rio Yokota. Practical deep learning with Bayesian principles. In *NeurIPS*, 2019.

[13] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for Bayesian uncertainty in deep learning. In *NeurIPS*, 2019.

[14] Michael Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable Bayesian neural nets with rank-1 factors. In *ICML*, 2020.

[15] David JC MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(5), 1992.

[16] Andrew YK Foong, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. In-between uncertainty in Bayesian neural networks. *arXiv*, 2019.

[17] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. An infinite-feature extension for Bayesian ReLU nets that fixes their asymptotic overconfidence. In *NeurIPS*, 2021.

[18] Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2 (11), 2011.

[19] Matthew D Hoffman, Andrew Gelman, et al. The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *JMLR*, 15(1), 2014.

[20] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, 2017.

[21] Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for Bayesian deep learning. In *ICLR*, 2020.

[22] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

[23] Felix Dangel, Frederik Kunstner, and Philipp Hennig. BackPACK: Packing more into backprop. In *ICLR*, 2020.

[24] Kazuki Osawa. ASDL: Automatic second-order differentiation (for Fisher, gradient covariance, Hessian, Jacobian, and kernel) library. https://github.com/kazukiosawa/asdfghjkl, 2021.

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

[26] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[27] David J Spiegelhalter and Steffen L Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20(5), 1990.

[28] Mark Gibbs. *Bayesian Gaussian Processes for Regression and Classification*. PhD thesis, University of Cambridge, 1998.

[29] Marius Hobbhahn, Agustinus Kristiadi, and Philipp Hennig. Fast predictive uncertainty for classification with Bayesian deep networks. *arXiv preprint arXiv:2003.01227*, 2020.

[30] Oren Rippel and Ryan Prescott Adams. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, 2013.

[31] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. In *ICLR Workshop*, 2015.

[32] Runa Eschenhagen, Erik Daxberger, Philipp Hennig, and Agustinus Kristiadi. Mixtures of Laplace approximations for improved post-hoc uncertainty in deep learning. In *NeurIPS Workshop of Bayesian Deep Learning*, 2021.

[33] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational Bayesian neural networks. In *ICLR*, 2019.

[34] Sanae Lotfi, Pavel Izmailov, Gregory Benton, Micah Goldblum, and Andrew Gordon Wilson. Bayesian model selection, the marginal likelihood, and generalization. *arXiv preprint arXiv:2202.11678*, 2022.

[35] Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Wilson. What are Bayesian neural network posteriors really like? In *ICML*, 2021.

[36] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *JMLR*, 22(57), 2021.

[37] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[38] Pavel Izmailov, Wesley J Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Subspace inference for Bayesian deep learning. In *UAI*, 2019.

[39] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *JMLR*, 20(1), 2019.

[40] Juan Maroñas, Oliver Hamelijnck, Jeremias Knoblauch, and Theodoros Damoulas. Transforming Gaussian processes with normalizing flows. In *AISTATS*, 2021.

[41] Andrew C Miller, Nicholas J Foti, and Ryan P Adams. Variational boosting: Iteratively refining posterior approximations. In *ICML*, 2017.

[42] Marton Havasi, Jasper Snoek, Dustin Tran, Jonathan Gordon, and José Miguel Hernández-Lobato. Refining the variational posterior through iterative optimization. *Entropy*, 23(1475), 2021.

[43] Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *NeurIPS*, 2020.

[44] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.

[45] Andrew Gelman and Donald B Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4), 1992.

[46] Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. In *ICLR*, 2018.

[47] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using Bayesian binning. In *AAAI*, 2015.

[48] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *JMLR*, 13(1), 2012.

[49] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *ICML*, 2017.

[50] Erik Daxberger, Eric Nalisnick, James U Allingham, Javier Antorán, and José Miguel Hernández-Lobato. Bayesian deep learning via subnetwork inference. In *ICML*, 2021.

[51] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured Laplace approximations for overcoming catastrophic forgetting. In *NIPS*, 2018.

[52] Alexander Immer, Matthias Bauer, Vincent Fortuin, Gunnar Rätsch, and Mohammad Emtiyaz Khan. Scalable marginal likelihood estimation for model selection in deep learning. In *ICML*, 2021.

[53] Convolution of Gaussians and the Probit integral. `https://agustinus.kristia.de/techblog/2022/06/25/conv-probit/`. Accessed: 2022-09-26.

[54] Zhiyun Lu, Eugene Ie, and Fei Sha. Mean-field approximation to Gaussian-softmax integral with application to uncertainty estimation. *arXiv preprint arXiv:2006.07584*, 2020.

[55] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[56] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[57] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

[58] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *NeurIPS*, 2019.

[59] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *ICLR*, 2019.

[60] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being a bit frequentist improves Bayesian neural networks. In *AISTATS*, 2022.

[61] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

Checklist

1. For all authors...
    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
    (b) Did you describe the limitations of your work? [Yes] Section 7.1.
    (c) Did you discuss any potential negative societal impacts of your work? [N/A]
    (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
    (a) Did you state the full set of assumptions of all theoretical results? [N/A]
    (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
    (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
    (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Appendix B.
    (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
    (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Section 6.4.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
    (a) If your work uses existing assets, did you cite the creators? [Yes]
    (b) Did you mention the license of the assets? [N/A]
    (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
    (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
    (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
    (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
    (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
    (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Posterior Refinement Improves Sample Efficiency in Bayesian Neural Networks

Appendix A Derivation of the Multi-Class Probit Approximation

Remark. This derivation first appeared in the first author's blog post [53].

Let $p(z) := \mathcal{N}(z \mid \mu, \Sigma)$ be the distribution a random variable $z \in \mathbb{R}^c$. The multi-class probit approximation (MPA) is defined by

$$\int_{\mathbb{R}^c} \operatorname{softmax}(z)\, p(z)\, dz \approx \operatorname{softmax}\left(\frac{\mu}{\sqrt{1 + \pi/8 \operatorname{diag}(\Sigma)}}\right),$$

where the vector division is defined component-wise. Its derivation, based on Lu et al. [54] is as follows.

Notice that we can write the i-th component of $\operatorname{softmax}(z)$ as $1/(1 + \sum_{j \neq i} \exp(-(z_i - z_j)))$. So, for each $i = 1, \dots, c$, using $z_{ij} := z_i - z_j$, we can write

$$\begin{aligned}\frac{1}{1 + \sum_{j \neq i} \exp(-z_{ij})} &= \frac{1}{1 - (K-1) + \sum_{j \neq i} \frac{1}{\frac{1}{1+\exp(-z_{ij})}}} \\ &= \frac{1}{2 - K + \sum_{j \neq i} \frac{1}{\sigma(z_{ij})}}.\end{aligned}$$

Then, we use the following approximations:

1. $\mathbb{E}(f(x)) \approx f(\mathbb{E}(x))$ for a non-zero function f,
2. the mean-field approximation $\mathcal{N}(z \mid \mu, \Sigma) \approx \mathcal{N}(z \mid \mu, \operatorname{diag}(\Sigma))$, and thus we have $z_{ij} := z_i - z_j \sim \mathcal{N}(z_{ij} \mid \mu_i - \mu_j, \Sigma_{ii} + \Sigma_{jj})$, and
3. using the (binary) probit approximation (see Section 2.1.2), with a further approximation:

$$\begin{aligned}\int_{\mathbb{R}} \sigma(z_{ij})\, \mathcal{N}(z_{ij} \mid \mu_i - \mu_j, \Sigma_{ii} + \Sigma_{jj})\, dz_{ij} &\approx \sigma\left(\frac{\mu_i - \mu_j}{\sqrt{1 + \pi/8\, \Sigma_{ii} + \Sigma_{jj}}}\right) \\ &\approx \sigma\left(\frac{\mu_i}{\sqrt{1 + \pi/8\, \Sigma_{ii}}} - \frac{\mu_j}{\sqrt{1 + \pi/8\, \Sigma_{jj}}}\right),\end{aligned}$$

we obtain

$$\begin{aligned}\int_{\mathbb{R}^c} \operatorname{softmax}_i(z)\, \mathcal{N}(z \mid \mu, \Sigma) &\approx \frac{1}{2 - K + \sum_{j \neq i} \frac{1}{\mathbb{E}\, \sigma(z_{ij})}} \\ &\approx \frac{1}{2 - K + \sum_{j \neq i} \frac{1}{\sigma\left(\frac{\mu_i}{\sqrt{1+\pi/8\, \Sigma_{ii}}} - \frac{\mu_j}{\sqrt{1+\pi/8\, \Sigma_{jj}}}\right)}} \\ &= \frac{1}{1 + \sum_{j \neq i} \exp\left(-\left(\frac{\mu_i}{\sqrt{1+\pi/8\, \Sigma_{ii}}} - \frac{\mu_j}{\sqrt{1+\pi/8\, \Sigma_{jj}}}\right)\right)} \\ &= \frac{\exp\left(\mu_i / \sqrt{1 + \pi/8\, \Sigma_{ii}}\right)}{\sum_{j=1}^{k} \exp\left(\mu_j / \sqrt{1 + \pi/8\, \Sigma_{jj}}\right)}\end{aligned}$$

We identify the last equation above as the i-th component of $\operatorname{softmax}\left(\frac{\mu}{\sqrt{1+\pi/8 \operatorname{diag}(\Sigma)}}\right)$.

Appendix B Implementation Details

B.1 Training

We use the Pyro library [39] to implement the normalizing flow (NF) used for the refinement. The NF is trained by maximizing the evidence lower bound using the Adam optimizer [55] and the cosine learning rate decay [56] for 20 epochs, with an initial learning rate of 0.001. Following [35], we do not use data augmentation.

For the HMC baseline, we use the default implementation of NUTS in Pyro. We confirm that the HMC used in our experiments are well-converged: The average Gelman-Rubin $\widehat{R}$'s are 0.998, 0.999, 0.997, and 1.096—below the standard threshold of 1.1—for the last-layer F-MNIST, last-layer CIFAR-10, last-layer CIFAR-100, and all-layer F-MNIST experiments, respectively.

For the MAP, VB, and CSGHMC baselines, we use the same settings as Daxberger et al. [6]: We train them for 100 epochs with an initial learning rate of 0.1, annealed via the cosine decay method [56]. The minibatch size is 128, and data augmentation is employed. For MAP, we use weight decay of 5×10^{-4}. For VB and CSGHMC, we use the prior precision corresponding to the previous weight decay value.

For the LA baseline, we use the `laplace-torch` library [6]. The diagonal Hessian is used for CIFAR-100 and all-layer F-MNIST, while the full Hessian is used for other cases. Following the current best-practice in LA, we tune the prior precision with *post hoc* marginal likelihood maximization [6].

Finally, for methods which require validation data, e.g. HMC (for finding the optimal prior precision), we obtain a validation test set of size 2000 by randomly splitting a test set. Note that, these validation data are not used for testing.

B.2 Datasets

For the dataset-shift experiment, we use the following test sets: Rotated F-MNIST and Corrupted CIFAR-10 [57, 58]. Meanwhile, we use the following OOD test sets for each the in-distribution training set:

- **F-MNIST:** MNIST, K-MNIST, E-MNIST.
- **CIFAR-10:** LSUN, SVHN, CIFAR-100.
- **CIFAR-100:** LSUN, SVHN, CIFAR-10.

Appendix C Additional Results

C.1 Image classification

To complement Table 3 in the main text, we present results for additional metrics (accuracy, Brier score, and ECE) in Table 5. We see that the trend Table 3 is also observable here. We also show that the refinem In Table 6, we observe that refining an *all-layer* posterior improves its predictive quality further.[10]

In Table 7, we present the detailed, non-averaged results to complement Table 4. Moreover, we also present dataset-shift results on standard benchmark problems (Rotated F-MNIST and Corrupted CIFAR-10). In both cases, we observe that the performance of the refined posterior approaches HMC's.

C.2 Text classification

We further validate the proposed method on text classification problems. We the benchmark in Hendrycks et al. [59], Kristiadi et al. [60]: A GRU recurrent network architecture [61] is used and trained for 10 epochs under the 20NG, SST, and TREC datasets. The prior precision for HMC and the refinement method is 40. We use the same normalizing flow as in the image classification experiment and follow a similar training procedure.

[10]The network is a two-layer fully-connected ReLU network with 50 hidden units.

Table 5: In-distribution calibration performance.

	F-MNIST			CIFAR-10			CIFAR-100		
Methods	**Acc. ↑**	**Brier ↓**	**ECE ↓**	**Acc. ↑**	**Brier ↓**	**ECE ↓**	**Acc. ↑**	**Brier ↓**	**ECE ↓**
MAP	90.4±0.1	0.1445±0.0008	11.7±0.3	94.8±0.1	0.0790±0.0004	10.5±0.2	76.5±0.1	0.3396±0.0012	13.7±0.2
MAP-Temp	90.4±0.1	0.1377±0.0009	3.2±0.0	94.9±0.1	0.0767±0.0008	6.0±1.3	75.9±0.1	0.3394±0.0007	8.9±0.2
VB*	90.5±0.1	0.1377±0.0013	3.9±0.2	91.1±0.1	0.1333±0.0008	5.3±0.2	69.6±0.2	0.4144±0.0016	5.5±0.2
CSGHMC*	89.6±0.1	0.1492±0.0011	3.7±0.1	93.6±0.2	0.0975±0.0017	7.2±0.3	73.8±0.1	0.3647±0.0019	7.1±0.1
LA	90.4±0.0	0.1439±0.0008	11.1±0.2	94.8±0.0	0.0785±0.0004	9.8±0.3	75.6±0.1	0.3529±0.0009	9.7±0.1
LA-Refine-1	90.4±0.1	0.1386±0.0007	5.2±0.2	94.7±0.0	0.0776±0.0003	4.3±0.2	75.9±0.1	0.3445±0.0010	8.0±0.2
LA-Refine-5	90.4±0.1	0.1375±0.0009	3.2±0.1	94.8±0.1	0.0768±0.0004	4.3±0.2	76.2±0.1	0.3311±0.0007	4.5±0.2
LA-Refine-10	90.5±0.1	0.1376±0.0008	3.6±0.1	94.9±0.1	0.0765±0.0004	4.4±0.2	76.1±0.1	0.3312±0.0008	4.4±0.1
LA-Refine-30	90.4±0.1	0.1376±0.0009	3.5±0.1	94.9±0.1	0.0765±0.0004	4.4±0.1	76.1±0.1	0.3315±0.0007	4.2±0.2
HMC	90.4±0.1	0.1375±0.0009	3.4±0.0	94.9±0.1	0.0765±0.0004	4.3±0.1	76.4±0.1	0.3283±0.0007	4.6±0.1

Table 6: Calibration of all-layer BNNs on F-MNIST. The architecture is two-layer ReLU fully-connected network with 50 hidden units.

Methods	**MMD ↓**	**Acc. ↑**	**NLL ↓**	**Brier ↓**	**ECE ↓**
LA	0.278±0.003	88.0±0.1	0.3597±0.0009	0.18±0.0006	7.7±0.1
LA-Refine-1	0.194±0.006	87.6±0.1	0.3564±0.0015	0.1807±0.0006	6.1±0.1
LA-Refine-5	0.19±0.006	87.6±0.1	0.3483±0.0012	0.1781±0.0005	4.9±0.3
LA-Refine-10	0.186±0.006	87.7±0.1	0.3459±0.0008	0.1771±0.0004	4.7±0.3
LA-Refine-30	0.183±0.006	87.8±0.1	0.3432±0.0014	0.176±0.0007	4.6±0.3
HMC	-	89.7±0.0	0.2908±0.0002	0.1502±0.0001	4.5±0.1

The results are in Table 8. We observe similar results as in the previous experiment: Refining a LA posterior, even with a very simple and cheap NF, makes the weight-space posterior closer to that of HMC. Moreover, the calibration performance (in terms of NLL) also becomes better.

C.3 Costs

We compare the theoretical costs of the refinement method in Table 9. For comparisons, we include the costs of popular Bayesian baselines: deep ensemble [20], SWAG [13], and MultiSWAG [43]. Our refinement method introduces overheads over the very cheap Kronecker-factored last-layer LA [6, Fig. 5, Tab. 5] due to the need of training the normalizing flow and feeding posterior samples forward through it during prediction. Combined with our findings in Section 6.4, our method only introduces a small overall overhead on top of the standard MAP network. Finally, using synthetic datasets, we show the empirical costs of performing refinement on different number of classes in Fig. 9. We note that even with 4096 features (e.g. in WideResNet-50-2) and 1000 classes (e.g. in ImageNet), the refinement method is cost-efficient.

C.4 Weight-space distributions obtained by refinement

To validate that the refinement technique yields accurate posterior approximations, we plot the empirical marginal densities $q(w_i \mid \mathcal{D})$ in Figs. 10 to 12. We validate that the refinement method makes the crude, base LA posteriors closer to HMC in the weight space.

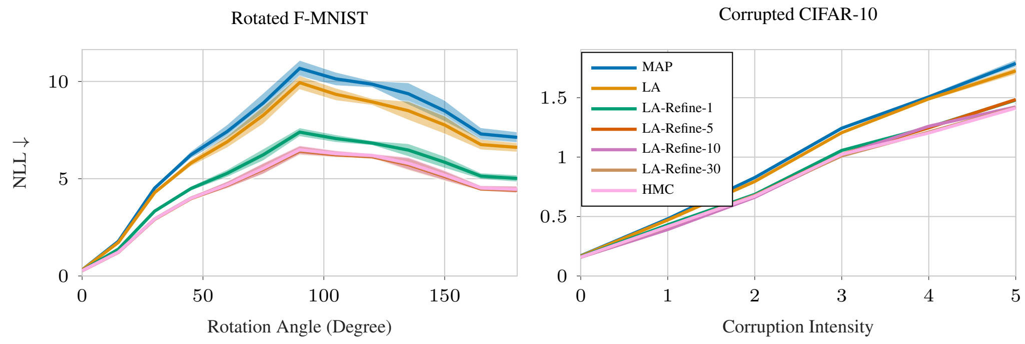


Figure 8: Calibration under dataset shifts in terms of NLL—lower is better.

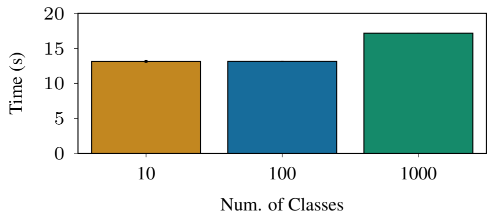


Figure 9: Empirical training costs of the refinement method, with 4096 features—this simulates the upper end of commonly-used networks' last-layer features, e.g. WideResNet-50-2. The dataset is synthetically generated by drawing 10000 points from a mixture of Gaussians, where number of modes equals number of classes. The training protocol for the NF is identical to the one used in the main text. A single RTX 2080Ti GPU is used to perform this experiment.

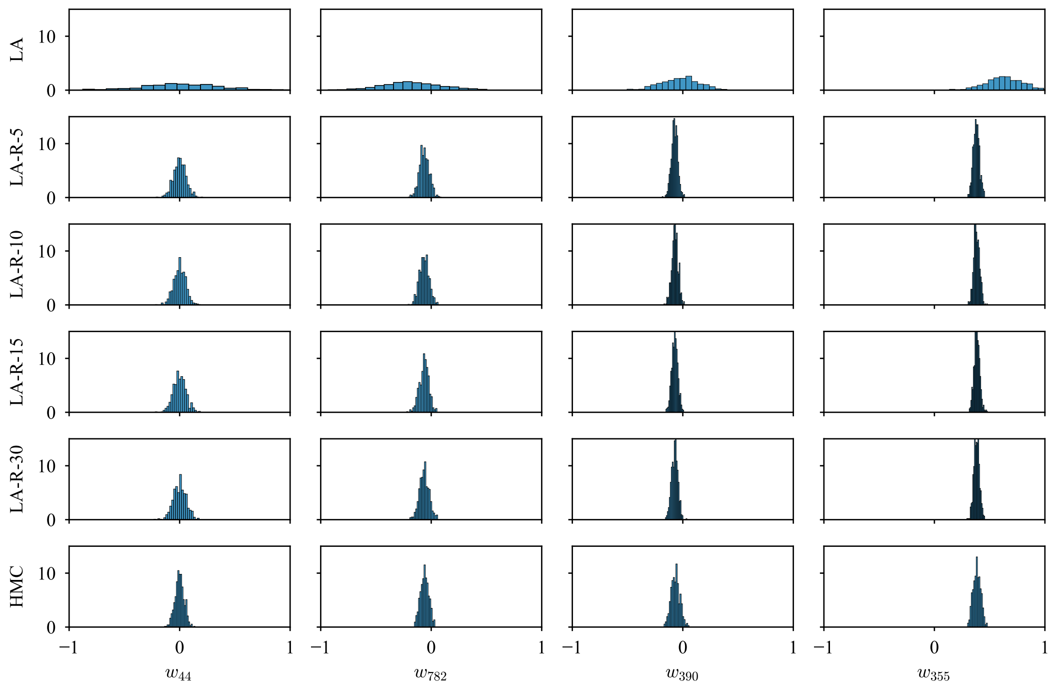


Figure 10: Empirical marginal posterior densities of some F-MNIST BNNs' random weights. "LA-R" is an abbreviation to "LA-Refine".

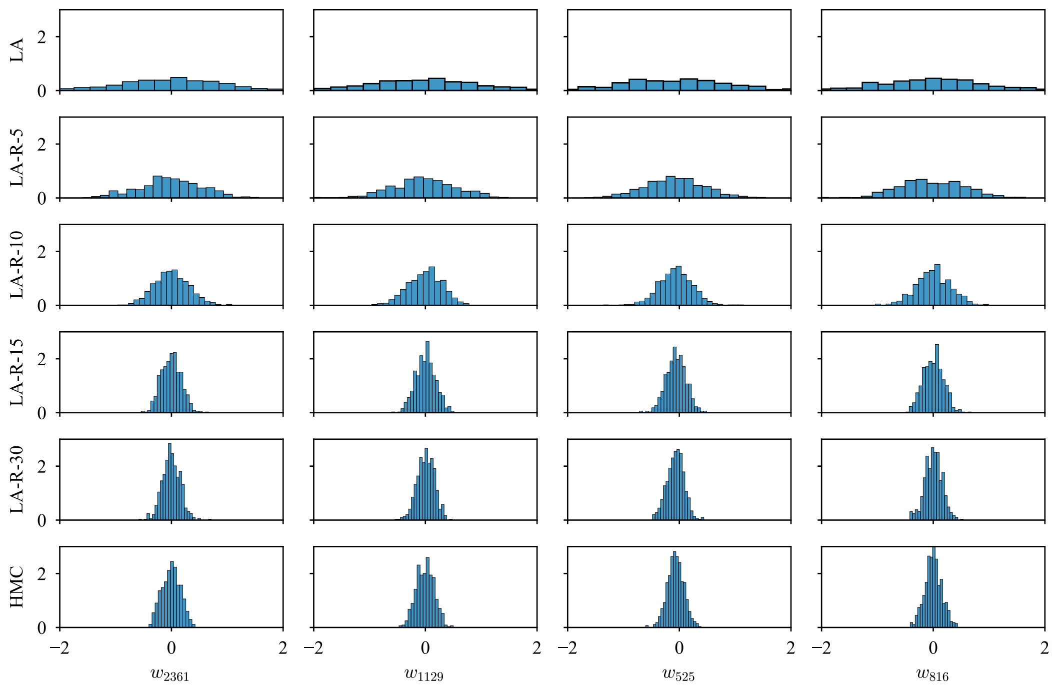


Figure 11: Empirical marginal posterior densities of some CIFAR-10 BNNs' random weights. "LA-R" is an abbreviation to "LA-Refine".

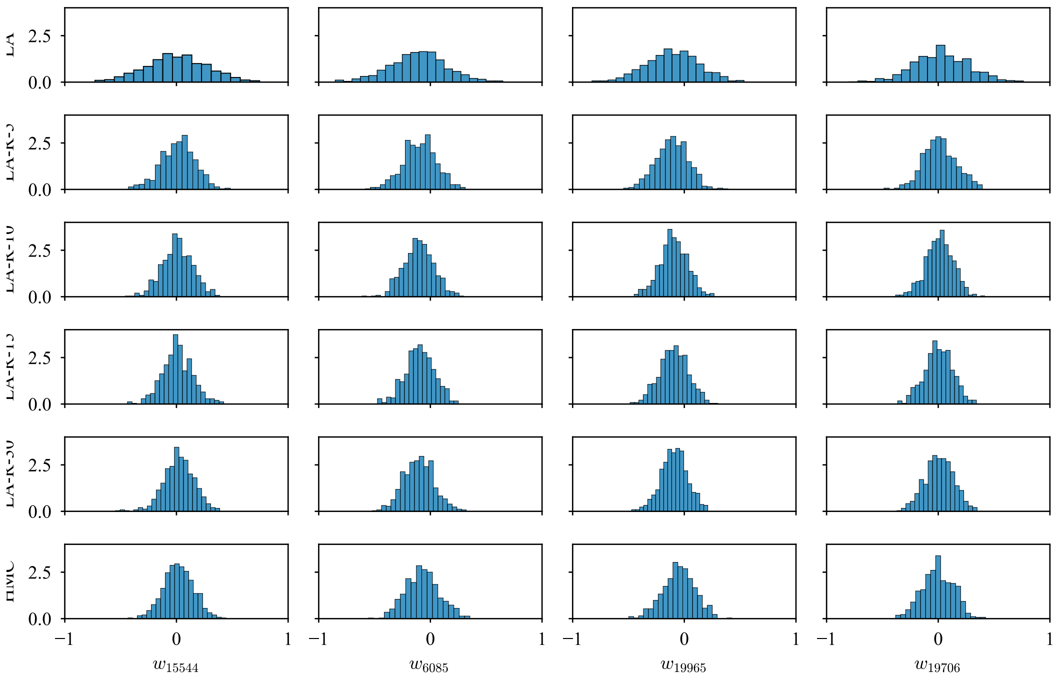


Figure 12: Empirical marginal posterior densities of some CIFAR-100 BNNs' random weights. "LA-R" is an abbreviation to "LA-Refine".

Table 7: Detailed OOD detection results. Values are FPR95. "LA-R" stands for "LA-Refine".

Datasets	**VB***	**CSGHMC***	**LA**	**LA-R-1**	**LA-R-5**	**LA-R-10**	**LA-R-30**	**HMC**
FMNIST								
EMNIST	83.4±0.6	86.5±0.5	84.7±0.7	85.4±0.8	87.6±0.6	87.6±0.6	87.6±0.6	87.2±0.6
MNIST	76.0±1.6	75.8±1.8	77.9±0.8	77.5±0.9	79.6±1.0	79.6±1.0	79.6±1.1	79.0±0.9
KMNIST	71.3±0.9	74.4±0.5	78.5±0.6	78.3±0.8	79.9±0.9	79.9±0.9	79.9±0.9	79.4±0.9
CIFAR-10								
SVHN	66.1±1.2	59.8±1.4	38.3±2.9	40.1±3.4	38.2±3.2	36.5±3.0	35.8±2.9	36.0±3.0
LSUN	53.3±2.5	51.7±1.5	51.1±1.1	46.9±0.5	46.7±0.6	46.9±0.7	47.1±0.5	46.7±0.8
CIFAR-100	69.3±0.2	64.6±0.3	58.2±0.8	56.1±0.6	55.7±0.8	55.3±0.6	55.2±0.5	55.3±0.4
CIFAR-100								
SVHN	81.7±0.7	75.9±1.5	82.2±0.8	77.7±1.2	78.1±1.3	77.9±1.5	78.3±1.6	78.2±1.6
LSUN	76.6±1.8	79.3±1.8	75.5±1.6	75.1±1.2	75.7±1.4	75.9±1.2	75.8±1.3	75.5±1.7
CIFAR-10	84.2±0.4	82.8±0.3	81.0±0.3	79.1±0.4	79.5±0.4	79.5±0.4	79.6±0.4	79.7±0.2

Table 8: Text classification performance.

	20NG		**SST**		**TREC**	
Methods	**MMD** $\downarrow$	**NLL** $\downarrow$	**MMD** $\downarrow$	**NLL** $\downarrow$	**MMD** $\downarrow$	**NLL** $\downarrow$
MAP	0.834	1.3038	0.515	1.3125	0.620	1.4289
LA	0.808	1.2772	0.474	1.2570	0.708	1.4206
LA-Refine-5	0.643	0.9904	0.126	1.0464	0.554	1.3375
HMC	-	0.9838	-	1.0787	-	1.1545

Table 9: Theoretical cost of our method. M is the number of parameters of the base neural network, N is the number of training data, C is the number of classes, P is the number of last-layer features, F is the length of the normalizing flow, S is the number of MC samples for approximating the predictive distribution, R is the number of model snapshots, K is the number ensemble components.

		Prediction	
Methods	**Overhead over MAP**	**Computation**	**Memory**
MAP	-	M	M
Last-Layer LA	$NM + C^3 + P^3$	SCP	$M + C^2 + P^2$
Last-Layer LA+Refine	$NM + C^3 + P^3 + NFCP$	$SFCP$	$M + C^2 + P^2 + FCP$
SWAG	RNM	SRM	RM
DE	-	KM	KM
MultiSWAG	$KRNM$	$KSRM$	KRM