Fast Predictive Uncertainty for Classification with Bayesian Deep Networks

Anonymous Author(s)
Affiliation
Address
email

Appendix

In Section 1 we give some background knowledge and a proof for the propositon of the main experiment. The Laplace approximation of the Dirichlet has already been done by David JC MacKay in [1]. Subsection 2 shows a concise summary of said approximation with some additional explanations. The Inversion of this Laplace approximation has already been done by Philipp Hennig in their PhD Thesis [2]. Section 3 shows a brief overview of the main steps. Note that we are not claiming any of their original ideas but rather show their work here as a look up for interested readers. In Section 4 we give a more detailed overview over the setup of some experiments that are beyond what is necessary to understand them but might be interesting for some readers.

1 Appendix A: Background and Proofs

Change of Variable for pdf

Let $\mathbf{x}$ be an n-dimensional continuous random variable with joint density function $p_{\mathbf{x}}$. If $\mathbf{y} = G(\mathbf{x})$, where G is a differentiable function, then $\mathbf{y}$ has density $p_{\mathbf{y}}$:

$$g(\mathbf{y}) = f\Big(G^{-1}(\mathbf{y})\Big) \left| \det \left[\frac{dG^{-1}(\mathbf{z})}{d\mathbf{z}} \bigg|_{\mathbf{z}=\mathbf{y}} \right] \right| \tag{1}$$

where the differential is the Jacobian of the inverse of G evaluated at $\mathbf{y}$. This procedure, also known as 'change of basis', is at the core of the Laplace bridge since it is used to transform the Dirichlet into the softmax basis.

Proof for Proposition

Proof. Considering that α_k is a decreasing function of $\mathbf{\Sigma}_{kk}$ by definition (19), it is sufficient to show that under the hypothesis, the derivative of $\frac{\partial}{\partial \alpha_k} \mathrm{Var}(\pi_k|\boldsymbol{\alpha})$ is negative.

By definition, the variance $\mathrm{Var}(\pi_k|\boldsymbol{\alpha})$ is

$$\mathrm{Var}(\pi_k|\boldsymbol{\alpha}) = \frac{\frac{\alpha_k}{\alpha_k+\alpha_{\neq k}} - \frac{\alpha_k^2}{(\alpha_k+\alpha_{\neq k})^2}}{\alpha_k + \alpha_{\neq k} + 1}\,.$$

The derivative is therefore

$$\frac{\partial}{\partial \alpha_k} \mathrm{Var}(\pi_k|\boldsymbol{\alpha}) = \frac{\alpha_{\neq k}(\alpha_{\neq k}^2 - \alpha_{\neq k}\alpha_k + \alpha_{\neq k} - \alpha_k(2\alpha_k+1))}{(\alpha_k+\alpha_{\neq k})^3(\alpha_k+\alpha_{\neq k}+1)^2}\,.$$

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

Solving $\frac{\partial}{\partial \alpha_k} \mathrm{Var}(\pi_k|\boldsymbol{\alpha}) < 0$ for α_k yields

$$\alpha_k > \frac{1}{4}\left(\sqrt{9\alpha_{\neq k}^2 + 10\alpha_{\neq k} + 1} - \alpha_{\neq k} - 1\right).$$

Therefore, under this hypothesis, $\mathrm{Var}(\pi_k|\boldsymbol{\alpha})$ is a decreasing function of α_k. □

Experimental Evaluation of the Proposition

[To test how often the condition is fulfilled we count its frequency. The fact that the condition is fulfilled implies a good approximation. The fact that the condition is not fulfilled does not automatically imply a bad approximation.]

		frequency
MNIST	MNIST	-
MNIST	FMNIST	-
MNIST	notMNIST	-
MNIST	KMNIST	-
CIFAR-10	CIFAR-10	0.998
CIFAR-10	CIFAR-100	0.925
CIFAR-10	SVHN	0.832
SVHN	SVHN	0.999
SVHN	CIFAR-100	0.668
SVHN	CIFAR-10	0.653
CIFAR-100	CIFAR-100	0.662
CIFAR-100	CIFAR-10	0.214
CIFAR-100	SVHN	0.166

Table 1

2 Appendix B: Laplace Approximation of the Dirichlet

Assume we have a Dirichlet in the standard basis with parameter vector $\boldsymbol{\alpha}$ and probability density function:

$$\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) := \frac{\Gamma\left(\sum_{k=1}^K \alpha_k\right)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \pi_k^{\alpha_k - 1}, \tag{2}$$

We aim to transform the basis of this distribution via the softmax transform to be in the new base π:

$$\pi_k(\mathbf{z}) := \frac{\exp(z_k)}{\sum_{l=1}^K \exp(z_l)}, \tag{3}$$

Usually, to transform the basis we would need the inverse transformation $H^{-1}(\mathbf{z})$ as described in the main paper. However, the softmax does not have an analytic inverse. Therefore David JC MacKay uses the following trick. Assume we know that the distribution in the transformed basis is:

$$\mathrm{Dir}_{\mathbf{z}}(\boldsymbol{\pi}(\mathbf{z})|\boldsymbol{\alpha}) := \frac{\Gamma\left(\sum_{k=1}^K \alpha_k\right)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \pi_k(\mathbf{z})^{\alpha_k}, \tag{4}$$

then we can show that the original distribution is the result of the basis transform by the softmax.

The Dirichlet in the softmax basis: We show that the density over $\boldsymbol{\pi}$ shown in Equation 4 transforms into the Dirichlet over $\mathbf{z}$. First, we consider the special case where $\boldsymbol{\pi}$ is confined to a $I-1$ dimensional subspace satisfying $\sum_i \boldsymbol{\pi}_i = c$. In this subspace we can represent $\boldsymbol{\pi}$ by an $I-1$ dimensional vector $\mathbf{a}$ such that

$$\pi_i = \mathbf{a}_i \quad i, ..., I-1 \tag{5}$$

$$\pi_I = c - \sum_i^{I-1} \mathbf{a}_i \tag{6}$$

and similarly we can represent $\mathbf{z}$ by an $I-1$ dimensional vector φ:

$$z_i = \varphi_i \quad i, ..., I-1 \tag{7}$$

$$z_I = 1 - \sum_i^{I-1} \varphi_i \tag{8}$$

then we can find the density over φ (which is proportional to the required density over $\mathbf{z}$) from the density over $\boldsymbol{\pi}$ (which is proportional to the given density over $\boldsymbol{\pi}$) by finding the determinant of the $(I-1) \times (I-1)$ Jacobian $\mathbf{J}$ given by

$$\begin{aligned} J_{ik} = \frac{\partial \xi_i}{\partial \tau_i} &= \sum_j^I \frac{\partial x_i}{\partial \pi_j} \frac{\partial \pi_j}{\partial \tau_k} \\ &= \delta_{ik}\mathbf{x}_i - \mathbf{x}_i\mathbf{x}_k + \mathbf{x}_i\mathbf{x}_I = \mathbf{x}_i(\delta_{ik} - (\mathbf{x}_k - \mathbf{x}_I)) \end{aligned} \tag{9}$$

We define two additional $I-1$ dimensional helper vectors $\mathbf{x}_k^+ := \mathbf{x}_k - \mathbf{x}_I$ and $n_k := 1$, and use $\det(I - xy^T) = 1 - x \cdot y$ from linear algebra. It follows that

$$\begin{aligned} \det J &= \prod_{i=1}^{I-1} \mathbf{x}_i \times \det[I - n\mathbf{x}^{+^T}] \\ &= \prod_{i=1}^{I-1} \mathbf{x}_i \times (1 - n \cdot \mathbf{x}^+) \\ &= \prod_{i=1}^{I-1} \mathbf{x}_i \times \left(1 - \sum_k \mathbf{x}_k^+\right) = I \prod_{i=1}^{I} \mathbf{x}_i \end{aligned} \tag{10}$$

Therefore, using Equation 4 we find that

$$P(\mathbf{z}) = \frac{P(\boldsymbol{\pi})}{|\det \mathbf{J}|} \propto \prod_{i=1}^{I} \mathbf{z}_i^{\alpha_i - 1} \tag{11}$$

This result is true for any constant c since it can be put into the normalizing constant. Thereby we make sure that the integral of the distribution is 1 and we have a valid probability distribution.

3 Appendix C: Inverting the Laplace Approximation of the Dirichlet

Through the figures of the 1D Dirichlet approximation in the main paper we have already established that the mode of the Dirichlet lies at the mean of the Gaussian distribution and therefore $\boldsymbol{\pi}(\mathbf{y}) = \frac{\alpha}{\sum_i \alpha_i}$. Additionally, the elements of $\mathbf{y}$ must sum to zero. These two constraints combined yield only one possible solution for $\boldsymbol{\mu}$.

$$\mu_k = \log \alpha_k - \frac{1}{K} \sum_{l=1}^{K} \log \alpha_l \tag{12}$$

Calculating the covariance matrix $\mathbf{\Sigma}$ is more complicated but layed out in the following. The logarithm of the Dirichlet is, up to additive constants

$$\log p_y(y|\alpha) = \sum_k \alpha_k \pi_k \tag{13}$$

Using π_k as the softmax of $\mathbf{y}$ as shown in Equation 3 we can find the elements of the Hessian $\mathbf{L}$

$$L_{kl} = \hat{\alpha}(\delta_{kl}\hat{\pi_k} - \hat{\pi_k}\hat{\pi_l}) \tag{14}$$

where $\hat{\boldsymbol{\alpha}} := \sum_k \alpha_k$ and $\hat{\boldsymbol{\pi}} = \frac{\alpha_k}{\hat{\alpha}}$ for the value of $\boldsymbol{\pi}$ at the mode. Analytically inverting $\mathbf{L}$ is done via a lengthy derivation using the fact that we can write $\mathbf{L} = \mathbf{A} + \mathbf{XBX}^\top$ and inverting it with the Schur-complement. This process results in the inverse of the Hessian

$$L^{-1}_{kl} = \delta_{kl}\frac{1}{\alpha_k} - \frac{1}{K}\left[\frac{1}{\alpha_k} + \frac{1}{\alpha_l} - \frac{1}{K}\left(\sum_u^K \frac{1}{\alpha_u}\right)\right] \tag{15}$$

We are mostly interested in the diagonal elements, since we desire a sparse encoding for computational reasons and we otherwise needed to map a $K \times K$ covariance matrix to a $K \times 1$ Dirichlet parameter vector which would be a very overdetermined mapping. Note that K is a scalar not a matrix. The diagonal elements of $\mathbf{\Sigma} = \mathbf{L}^{-1}$ can be calculated as

$$\Sigma_{kk} = \frac{1}{\alpha_k}\left(1 - \frac{2}{K}\right) + \frac{1}{K^2}\sum_l^k \frac{1}{\alpha_l}. \tag{16}$$

To invert this mapping we transform Equation 12 to

$$\alpha_k = e^{\mu_k}\prod_l^K \alpha_l^{1/K} \tag{17}$$

by applying the logarithm and re-ordering some parts. Inserting this into Equation 16 and re-arranging yields

$$\prod_l^K \alpha_l^{1/K} = \frac{1}{\mathbf{\Sigma}_{kk}}\left[e^{-\mu}\left(1 - \frac{2}{K}\right) + \frac{1}{K^2}\sum_u^K e^{-\mu_u}\right] \tag{18}$$

which can be re-inserted into Equation 17 to give

$$\alpha_k = \frac{1}{\Sigma_k k}\left(1 - \frac{2}{K} + \frac{e^{-\mu_k}}{K^2}\sum_l^K e^{-\mu_k}\right) \tag{19}$$

which is the final mapping. With Equations 12 and 16 we are able to map from Dirichlet to Gaussian and with Equation 19 we are able to map the inverse direction.

4 Appendix D: Experiments Details

The exact experimental setups, i.e. network architectures, learning rates, random seeds, etc. can be found in the accompanying github repository (not linked for anonymity. Code is attached as zip file). This section is mostly used to justify some of the decisions we made during the process in more detail and highlight some miscellaneous interesting things.

Mathematical description of the setup

In principle, the Gaussian over the weights required by the Laplace Bridge for BNNs can be constructed by any Gaussian approximate Bayesian method such as variational Bayes [3; 4] and Laplace approximations for NNs [5; 6]. We will focus on the Laplace approximation, which uses the same principle as the Laplace Bridge. However, in the Laplace approximation for neural networks, the posterior distribution over the weights of a network is the one that is approximated as a Gaussian, instead of a Dirichlet distribution over the outputs as in the Laplace Bridge.

Given a dataset $\mathcal{D} := \{(\mathbf{x}_i, t_i)\}_{i=1}^D$ and a prior $p(\boldsymbol{\theta})$, let

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta}) = p(\boldsymbol{\theta}) \prod_{(\mathbf{x},t)\in\mathcal{D}} p(y = t|\boldsymbol{\theta}, \mathbf{x}) \,, \tag{20}$$

be the posterior over the parameter $\boldsymbol{\theta}$ of an L-layer network $f_{\boldsymbol{\theta}}$. Then we can get an approximation of the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ by fitting a Gaussian $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ where

$$\begin{aligned}\boldsymbol{\mu}_{\boldsymbol{\theta}} &= \boldsymbol{\theta}_{\text{MAP}} \,, \\ \boldsymbol{\Sigma}_{\boldsymbol{\theta}} &= (-\nabla^2|_{\boldsymbol{\theta}_{\text{MAP}}} \log p(\boldsymbol{\theta}|\mathcal{D}))^{-1} =: \mathbf{H}_{\boldsymbol{\theta}}^{-1} \,.\end{aligned}$$

That is, we fit a Gaussian centered at the mode $\boldsymbol{\theta}_{\text{MAP}}$ of $p(\boldsymbol{\theta}|\mathcal{D})$ with the covariance determined by the curvature at that point. We assume that the prior $p(\boldsymbol{\theta})$ is a zero-mean isotropic Gaussian $\mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \sigma^2\mathbf{I})$ and the likelihood function is the Categorical density

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{(\mathbf{x},t)\in\mathcal{D}} \text{Cat}(y = t|\text{softmax}(f_{\boldsymbol{\theta}}(\mathbf{x}))) \,.$$

For various applications in Deep Learning, an approximation with full Hessian is often computationally too expensive. Indeed, for each input $\mathbf{x} \in \mathbb{R}^N$, one has to do K backward passes to compute the Jacobian $\mathbf{J}(\mathbf{x})$. Moreover, it requires an $\mathcal{O}(PK)$ storage which is also expensive since P is often in the order of millions. A cheaper alternative is to fix all but the last layer of $f_{\boldsymbol{\theta}}$ and only apply the Laplace approximation on $\mathbf{W}_L$, the last layer's weight matrix. This scheme has been used successfully by Snoek et al. [7]; Wilson et al. [8]; Brosse et al. [9], etc. and has been shown theoretically that it can mitigate overconfidence problems in ReLU networks [10]. In this case, given the approximate last-layer posterior

$$p(\mathbf{W}^L|\mathcal{D}) \approx \mathcal{N}(\text{vec}(\mathbf{W}^L)|\text{vec}(\mathbf{W}^L_{\text{MAP}}), \mathbf{H}^{-1}_{\mathbf{W}^L}) \,, \tag{21}$$

one can efficiently compute the distribution over the logits. That is, let $\boldsymbol{\phi} : \mathbb{R}^N \to \mathbb{R}^Q$ be the first $L-1$ layers of $f_{\boldsymbol{\theta}}$, seen as a feature map. Then, for each $\mathbf{x} \in \mathbb{R}^N$, the induced distribution over the logit $\mathbf{W}^L\boldsymbol{\phi}(\mathbf{x}) =: \mathbf{z}$ is given by

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mathbf{W}^L_{\text{MAP}}\boldsymbol{\phi}(\mathbf{x}), (\boldsymbol{\phi}(\mathbf{x})^\top \otimes \mathbf{I})\mathbf{H}^{-1}_{\mathbf{W}^L}(\boldsymbol{\phi}(\mathbf{x}) \otimes \mathbf{I})) \,, \tag{22}$$

where $\otimes$ denotes the Kronecker product.

An even more efficient last-layer approximation can be obtained using a Kronecker-factored matrix normal distribution [11; 12; 6]. That is, we assume the posterior distribution to be

$$p(\mathbf{W}^L|\mathcal{D}) \approx \mathcal{MN}(\mathbf{W}^L|\mathbf{W}^L_{\text{MAP}}, \mathbf{U}, \mathbf{V}) \,, \tag{23}$$

where $\mathbf{U} \in \mathbb{R}^{K\times K}$ and $\mathbf{V} \in \mathbb{R}^{Q\times Q}$ are the Kronecker factorization of the inverse Hessian matrix $\mathbf{H}^{-1}_{\mathbf{W}^L}$ [13]. In this case, for any $\mathbf{x} \in \mathbb{R}^N$, one can easily show that the distribution over logits is given by

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mathbf{W}^L_{\text{MAP}}\boldsymbol{\phi}(\mathbf{x}), (\boldsymbol{\phi}(\mathbf{x})^\top\mathbf{V}\boldsymbol{\phi}(\mathbf{x}))\mathbf{U}) \,, \tag{24}$$

which is easy to implement and computationally cheap. Finally, and even more efficient, is a last-layer approximation scheme with a diagonal Gaussian approximate posterior, i.e. the so-called mean-field approximation. In this case, we assume the posterior distribution to be

$$p(\mathbf{W}^L|\mathcal{D}) \approx \mathcal{N}(\text{vec}(\mathbf{W}^L)|\text{vec}(\mathbf{W}^L_{\text{MAP}}), \text{diag}(\boldsymbol{\sigma}^2)) \,, \tag{25}$$

where $\boldsymbol{\sigma}^2$ is obtained via the diagonal of the Hessian of the log-posterior w.r.t. $\text{vec}(\mathbf{W}^L)$ at $\text{vec}(\mathbf{W}^L_{\text{MAP}})$.

Uncertainty estimates on MNIST

Most of the experimental setup is already explained in the main paper. The exact details can be found in the accompanying code. Every experiment has been conducted with 5 different seeds.

OOD detection

Every experiment has been conducted with 5 different seeds. In the tables the mean and standard deviations are presented. The reason why the sampling procedure for the CIFAR-10 and CIFAR-100 case are similarly fast even though we draw from a 10- vs 100-dimensional Gaussian is because the sampling procedures were parallelized on a GPU. All prior uncertainties over the weights were chosen such that the MMC of the sampling averages was around 5% lower than the MAP estimate. In the following we show the results including a KFAC approximation of the last layer.

Table 2: OOD detection results. Same table as in the main text but a different GPU (GTX 1080 Ti) was used. Results for the all-layer Laplace approximation have been changed since a bug was fixed.

		Diag Sampling		Diag LB		KFAC Sampling		KFAC LB		Time in s ↓	
Train	**Test**	**MMC ↓**	**AUROC ↑**	**MMC ↓**	**AUROC ↑**	**MMC ↓**	**AUROC ↑**	**MMC↓**	**AUROC ↑**	**Sampling**	**LB**
MNIST	MNIST	0.942 ± 0.007	-	**0.987** ± 0.000	-	-	-	-	-	26.8	**0.062**
MNIST	FMNIST	0.397 ± 0.001	0.992 ± 0.000	**0.363** ± 0.000	**0.996** ± 0.000	-	-	-	-	26.8	**0.062**
MNIST	notMNIST	**0.543** ± 0.000	0.960 ± 0.000	0.649 ± 0.000	**0.961** ± 0.000	-	-	-	-	50.3	**0.117**
MNIST	KMNIST	**0.513** ± 0.001	**0.974** ± 0.000	0.637 ± 0.000	0.973 ± 0.000	-	-	-	-	26.9	**0.062**
CIFAR-10	CIFAR-10	0.948 ± 0.000	-	**0.966** ± 0.000	-	0.857 ± 0.003	-	**0.966** ± 0.000	-	6.58	**0.017**
CIFAR-10	CIFAR-100	**0.708** ± 0.000	**0.889** ± 0.000	0.742 ± 0.000	0.866 ± 0.000	**0.562** ± 0.003	**0.880** ± 0.012	0.741 ± 0.000	0.866 ± 0.000	6.59	**0.016**
CIFAR-10	SVHN	**0.643**± 0.000	0.933 ± 0.000	0.647 ± 0.000	**0.934** ± 0.000	**0.484** ± 0.004	**0.939** ± 0.001	0.648 ± 0.003	0.934 ± 0.001	17.0	**0.040**
SVHN	SVHN	0.986 ± 0.000	-	**0.993** ± 0.000	-	0.947 ± 0.002	-	**0.993** ± 0.000	-	17.1	**0.042**
SVHN	CIFAR-100	0.595 ± 0.000	0.984 ± 0.000	**0.526** ± 0.000	**0.985** ± 0.000	**0.460** ± 0.004	**0.986** ± 0.001	0.527 ± 0.002	0.985 ± 0.000	6.62	**0.017**
SVHN	CIFAR-10	0.593 ± 0.000	0.984 ± 0.000	**0.520** ± 0.000	**0.987** ± 0.000	**0.458** ± 0.004	0.986 ± 0.001	0.520 ± 0.002	**0.987** ± 0.000	6.62	**0.017**
CIFAR-100	CIFAR-100	**0.762** ± 0.000	-	0.590 ± 0.000	-	0.404 ± 0.000	-	**0.593** ± 0.000	-	6.76	**0.016**
CIFAR-100	CIFAR-10	0.467 ± 0.000	0.788 ± 0.000	**0.206** ± 0.000	**0.791** ± 0.000	0.213 ± 0.000	0.788 ± 0.000	**0.209** ± 0.000	**0.791** ± 0.000	6.71	**0.017**
CIFAR-100	SVHN	0.461 ± 0.000	0.795 ± 0.000	**0.170** ± 0.000	**0.815** ± 0.000	0.180 ± 0.001	**0.838** ± 0.001	**0.173** ± 0.000	0.815 ± 0.000	17.3	**0.041**

		Sampling (100)			Laplace Bridge		
Train	**Test**	**MMC ↓**	**AUROC ↑**	**Time in s ↓**	**MMC ↓**	**AUROC ↑**	**Time in s ↓**
MNIST	MNIST	0.989± 0.000	-	76.3	**0.987** ± 0.000	-	**0.062**
MNIST	FMNIST	0.465± 0.004	0.992 ± 0.000	76.3	**0.363** ± 0.000	**0.996** ± 0.000	**0.062**
MNIST	notMNIST	**0.645**± 0.001	0.955 ± 0.001	42.6	0.649 ± 0.000	**0.961** ± 0.000	**0.117**
MNIST	KMNIST	**0.632**± 0.001	0.967± 0.001	76.0	0.637 ± 0.000	**0.973** ± 0.000	**0.062**

Table 3: Results for sampling from all weights instead of the last layer. Number of samples was 100. Sampling all weights and doing a forward pass seems to be worse than using the Laplace Bridge even though it takes much longer.

Time comparison

Every experiment has been conducted with 5 different seeds. The presented curves are the averages over these 5 experiments with errorbars. The reason why taking one sample is slower than two is because of the way random numbers are generated for the normal distribution. For further information read up on the Box-Mueller Transform.

Uncertainty-aware output ranking on ImageNet

The prior covariances for the Laplace approximation of the Hessian over the weights were chosen such that uncertainty estimate of the Laplace bridge MMC over the outputs was not more than 5% lower than the MAP estimate. The length of list generated by our uncertainty aware method was chosen such that it contained at least one and maximally ten samples. Originally we wanted to choose the maximal length according to the size of the largest category (e.g. fishes or dogs) but the class tree hierarchy of ImageNet does not answer this question meaningfully. We chose ten because there are no reasonable bins larger than ten when looking at a histogram. Pseudo code for the uncertainty aware top-k ranking can be found in Algorithm 1.

Algorithm 1 Uncertainty-aware top-k

Input: A Dirichlet parameter $\boldsymbol{\alpha} \in \mathbb{R}^K$ obtained by applying the Laplace Bridge to the Gaussian over the logit of an input, a percentile threshold T e.g. 0.05, a function $\mathrm{class\_of}$ that returns the underlying class of a sorted index.

$\tilde{\boldsymbol{\alpha}} = \mathrm{sort\_descending}(\boldsymbol{\alpha})$ *// start with the highest confidence*
$\alpha_0 = \sum_i \alpha_i$
$\mathcal{C} = \{\mathrm{class\_of}(1)\}$ *// initialize top-k, must include at least one class*
for $i = 2, \ldots, K$ **do**
  $F_{i-1} = \mathrm{Beta}(\tilde{\alpha}_{i-1}, \alpha_0 - \tilde{\alpha}_{i-1})$ *// the previous marginal CDF*
  $F_i = \mathrm{Beta}(\tilde{\alpha}_i, \alpha_0 - \tilde{\alpha}_i)$ *// the current marginal CDF*
  $l_{i-1} = F_{i-1}^{-1}(T/2)$ *// left $\frac{T}{2}$ percentile of the previous marginal*
  $r_i = F_i^{-1}(1 - T/2)$ *// right $\frac{T}{2}$ percentile of the current marginal*
  if $r_i > l_{i-1}$ **then**
    $\mathcal{C} = \mathcal{C} \cup \{\mathrm{class\_of}(i)\}$ *// overlap detected, add the current class*
  else
    break *// No more overlap, end the algorithm*
  end if
end for

Output: $\mathcal{C}$ *// return the resulting top-k prediction*

References

[1] David J.C. MacKay. Choice of basis for laplace approximation. *Machine Learning*, 33(1): 77–86, Oct 1998. ISSN 1573-0565.

[2] P. Hennig. *Approximate Inference in Graphical Models*. PhD thesis, November 2010.

[3] Alex Graves. Practical variational inference for neural networks. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2348–2356. Curran Associates, Inc., 2011.

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

[5] David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Comput.*, 4(3):448–472, May 1992. ISSN 0899-7667.

[6] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

[7] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2171–2180, Lille, France, 07–09 Jul 2015. PMLR.

[8] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 370–378, Cadiz, Spain, 09–11 May 2016. PMLR.

[9] Nicolas Brosse, Carlos Riquelme, Alice Martin, Sylvain Gelly, and Éric Moulines. On last-layer algorithms for classification: Decoupling representation from uncertainty estimation. *arXiv preprint arXiv:2001.08049*, 2020.

[10] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks. *arXiv preprint arXiv:2002.10118*, 2020.

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

[12] Shengyang Sun, Changyou Chen, and Lawrence Carin. Learning structured weight uncertainty in Bayesian neural networks. In *Artificial Intelligence and Statistics*, pages 1283–1292, 2017.

[13] James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *ICML*, 2015.