Fast Predictive Uncertainty for Classification with Bayesian Deep Networks (Supplementary material)

Marius Hobbhahn[1] **Agustinus Kristiadi**[1] **Philipp Hennig**[1,2]

[1]University of Tübingen
[2]Max-Planck Institute for Intelligent Systems

1 APPENDIX A

FIGURES

The parameters of Figure **??** are from left to right $\alpha, \beta = (0.8, 0.9), (4, 2,), (2, 7)$.

CHANGE OF VARIABLE FOR PDF

Let $\mathbf{z}$ 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 $p_{\mathbf{y}}$:

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

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.

CORRECTION FOR SUM(Y)=0

We know that the product rule of Gaussians yields

$$p(x|Ax = y) = \frac{p(x, y)}{p(y)} \tag{2}$$

$$= \mathcal{N}(x; \mu + \Sigma A^\top (A\Sigma A^\top)^{-1}(y - A\mu), \Sigma - \Sigma A^\top (A\Sigma A^\top)^{-1} A\Sigma) \tag{3}$$

In our particular setup we have

$$p(x) = \mathcal{N}(x; \mu, \Sigma) \tag{4}$$

with constraint

$$p(I|x) = \delta(1x^\top - 0) = \lim_{\epsilon \to \infty} \mathcal{N}(0; 1^\top x, \frac{1}{\epsilon}) \tag{5}$$

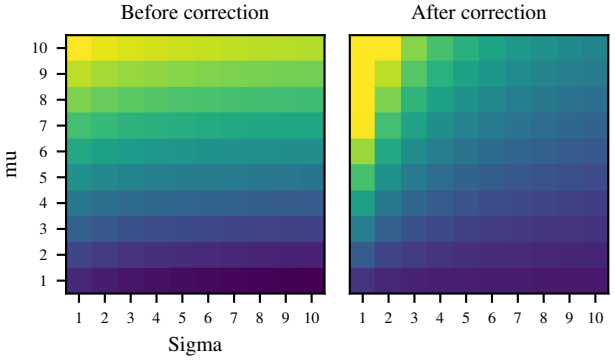

Figure 1: Contourplot showing the scaling behavior of μ and Σ. In the left figure, we see that Sigma has nearly no influence on the scaling. Our correction in the right figure fixes that. Contour levels show the first entry of α on a log-scale.

Therefore we get

$$p(x|I) = \mathcal{N}(x; \mu + \Sigma 1(1^\top \Sigma 1 - \frac{1}{\epsilon})^{-1}(0 - 1^\top \mu), \tag{6}$$

$$\Sigma - \Sigma 1(1^\top \Sigma 1 - \frac{1}{\epsilon})^{-1} 1^\top \Sigma)$$

$$= \mathcal{N}\left(x; \mu - \frac{\Sigma \mathbf{1}\mathbf{1}^\top \mu}{\mathbf{1}^\top \Sigma \mathbf{1}}, \Sigma - \frac{\Sigma \mathbf{1}\mathbf{1}^\top \Sigma}{\mathbf{1}^\top \Sigma \mathbf{1}}\right) \tag{7}$$

VARIANCE CORRECTION

As described in the main text, the original Laplace Bridge scales worse with Σ than sampling and applying the softmax. In Figure 1 you can see a contourplot that shows the scaling of mean and variance with and without correction. As suggested, the Variance has nearly no influence on the result before the correction but our correction fixes that.

Some reviewers wanted to understand how we derived the equations for our correction, so here is a short informal explanation. During the experimentation with the LB, we

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

found that it doesn't approximate the sample distribution well when Σ gets large. We then understood why (as detailed in the limitations section) and proposed a fix for these scenarios without damaging its behavior in all other scenarios. We experimented with multiple fixes and the result you see in the paper is the one that fulfilled most of our criteria. Therefore, the correction doesn't come from a principled theoretical derivation but is motivated by the theoretical findings.

2 APPENDIX (DERIVATION OF LB)

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

$$\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) := \frac{\Gamma\left(\sum_{k=1}^{K}\alpha_k\right)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\pi_k^{\alpha_k-1}, \tag{8}$$

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

$$\pi_k(\mathbf{z}) := \frac{\exp(z_k)}{\sum_{l=1}^{K}\exp(z_l)}, \tag{9}$$

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:

$$\mathrm{Dir}_{\mathbf{z}}(\boldsymbol{\pi}(\mathbf{z})|\boldsymbol{\alpha}) := \frac{\Gamma\left(\sum_{k=1}^{K}\alpha_k\right)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\pi_k(\mathbf{z})^{\alpha_k}, \tag{10}$$

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 10 transforms into the Dirichlet over $\mathbf{z}$. First, we consider the special case where $\boldsymbol{\pi}$ is confined to an $I-1$ dimensional subspace satisfying $\sum_i \boldsymbol{\pi}_i = c$. In this subspace we can represent φ by an $I-1$ dimensional vector φ such that

$$\pi_i = \varphi_i \quad i, ..., I-1 \tag{11}$$

$$\pi_I = c - \sum_{i}^{I-1}\varphi_i \tag{12}$$

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

$$z_i = \mathbf{a}_i \quad i, ..., I-1 \tag{13}$$

$$z_I = 1 - \sum_{i}^{I-1}\mathbf{a}_i \tag{14}$$

then we can find the density over $\mathbf{a}$ (which is proportional to the required density over $\mathbf{z}$) from the density over φ (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

$$\begin{aligned} J_{ik} = \frac{\partial\varphi_i}{\partial\mathbf{a}_l} &= \sum_{j}^{I}\frac{\partial\boldsymbol{\pi}_i}{\partial\mathbf{z}_j}\frac{\partial\mathbf{z}_j}{\partial\mathbf{a}_k} \\ &= \delta_{ik}\boldsymbol{\pi}_i - \boldsymbol{\pi}_i\boldsymbol{\pi}_k + \boldsymbol{\pi}_i\boldsymbol{\pi}_I = \boldsymbol{\pi}_i(\delta_{ik} - (\boldsymbol{\pi}_k - \boldsymbol{\pi}_I)) \end{aligned} \tag{15}$$

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

$$\begin{aligned} \det J &= \prod_{i=1}^{I-1}\boldsymbol{\pi}_i \times \det[I - n\boldsymbol{\pi}^{+^T}] \\ &= \prod_{i=1}^{I-1}\boldsymbol{\pi}_i \times (1 - n\cdot\boldsymbol{\pi}^+) \\ &= \prod_{i=1}^{I-1}\boldsymbol{\pi}_i \times \left(1 - \sum_k \boldsymbol{\pi}_k^+\right) = I\prod_{i=1}^{I}\boldsymbol{\pi}_i \end{aligned} \tag{16}$$

Therefore, using Equation 10 we find that

$$P(\boldsymbol{\pi}) = \frac{P(\mathbf{z})}{|\det\mathbf{J}|} \propto \prod_{i=1}^{I}\boldsymbol{\pi}_i^{\alpha_i-1} \tag{17}$$

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 (DERIVATION OF INVERSION)

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 $\boldsymbol{\mu}$.

$$\mu_k = \log\alpha_k - \frac{1}{K}\sum_{l=1}^{K}\log\alpha_l \tag{18}$$

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

$$\log p_{\mathbf{z}}(\mathbf{z}|\alpha) = \sum_k \alpha_k \pi_k \tag{19}$$

Using π_k as the softmax of $\mathbf{y}$ as shown in Equation 9 we can find the elements of the Hessian $\mathbf{L}$

$$L_{kl} = \hat{\alpha}(\delta_{kl}\hat{\pi_k} - \hat{\pi_k}\hat{\pi_l}) \tag{20}$$

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 $\mathbf{L} = \mathbf{A} + \mathbf{X}\mathbf{B}\mathbf{X}^\top$ and inverting it with the Schur-complement. You can find the derivation in [Hennig, 2010]. This process results in the inverse of the Hessian

$$L^{-1}_{kl} = \delta_{kl}\frac{1}{\alpha_k} - \frac{1}{K}\left[\frac{1}{\alpha_k} + \frac{1}{\alpha_l} - \frac{1}{K}\left(\sum_u^K \frac{1}{\alpha_u}\right)\right] \tag{21}$$

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 $\boldsymbol{\Sigma} = \mathbf{L}^{-1}$ can be calculated as

$$\Sigma_{kk} = \frac{1}{\alpha_k}\left(1 - \frac{2}{K}\right) + \frac{1}{K^2}\sum_l^k \frac{1}{\alpha_l}. \tag{22}$$

To invert this mapping we transform Equation 18 to

$$\alpha_k = e^{\mu_k}\prod_l^K \alpha_l^{1/K} \tag{23}$$

by applying the logarithm and re-ordering some parts. Inserting this into Equation 22 and re-arranging yields

$$\prod_l^K \alpha_l^{1/K} = \frac{1}{\boldsymbol{\Sigma}_{kk}}\left[e^{-\mu}\left(1 - \frac{2}{K}\right) + \frac{1}{K^2}\sum_u^K e^{-\mu_u}\right] \tag{24}$$

which can be re-inserted into Equation 23 to give

$$\alpha_k = \frac{1}{\Sigma_{kk}}\left(1 - \frac{2}{K} + \frac{e^{\mu_k}}{K^2}\sum_l^K e^{-\mu_k}\right) \tag{25}$$

which is the final mapping. With Equations 18 and 22 we are able to map from Dirichlet to Gaussian and with Equation 25 we are able to map the inverse direction.

4 APPENDIX (EXPERIMENTAL DETAILS)

The exact experimental setups, i.e. network architectures, learning rates, random seeds, etc. can be found in the accompanying GitHub repository [1]. This section is used to justify some of the decisions we made during the process in more detail, highlight some miscellaneous interesting things and showcase the additional experiments promised in the main paper.

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 [Graves, 2011, Blundell et al., 2015] and Laplace approximations for NNs [MacKay, 1992, Ritter et al., 2018]. 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

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta}) = p(\boldsymbol{\theta}) \prod_{(\mathbf{x},t)\in\mathcal{D}} p(y = t|\boldsymbol{\theta}, \mathbf{x})\,, \tag{26}$$

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

[1] https://github.com/mariushobbhahn/LB_for_BNNs_official

Table 1: Comparing the extended probit approximation with the normalized version of the LB norm in the KFAC setting. The probit approximation seems to break down in the MNIST scenarios.

		KFAC Probit				KFAC LB norm			
Train	**Test**	**MMC** ↓	**AUROC** ↑	**ECE** ↓	**NLL** ↓	**MMC** ↓	**AUROC** ↑	**ECE** ↓	**NLL** ↓
MNIST	MNIST	0.105	0.000	2.258	0.883	0.975	0.000	0.043	0.018
MNIST	FMNIST	0.102	0.955	2.302	0.032	0.444	0.990	2.871	0.364
MNIST	notMNIST	0.103	0.922	2.300	0.043	0.409	0.986	2.854	0.294
MNIST	KMNIST	0.102	0.962	2.304	0.012	0.414	0.991	3.162	0.328
CIFAR10	CIFAR10	0.548	0.000	0.661	0.404	0.941	0.000	0.195	0.017
CIFAR10	CIFAR100	0.358	0.896	2.652	0.253	0.662	0.866	3.871	0.558
CIFAR10	SVHN	0.307	0.956	2.567	0.195	0.441	0.965	2.837	0.327

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. [2015], Wilson et al. [2016], Brosse et al. [2020], etc. and has been shown theoretically that it can mitigate overconfidence problems in ReLU networks [Kristiadi et al., 2020]. In this case, given the approximate last-layer posterior

$$p(\mathbf{W}^L|\mathcal{D}) \approx \mathcal{N}(\text{vec}(\mathbf{W}^L)|\text{vec}(\mathbf{W}^L_{\text{MAP}}), \mathbf{H}^{-1}_{\mathbf{W}^L}) \,, \quad (27)$$

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

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mathbf{W}^L_{\text{MAP}}\boldsymbol{\phi}(\mathbf{x}), (\boldsymbol{\phi}(\mathbf{x})^\top \otimes \mathbf{I})\mathbf{H}^{-1}_{\mathbf{W}^L}(\boldsymbol{\phi}(\mathbf{x}) \otimes \mathbf{I})) \,, \quad (28)$$

where $\otimes$ denotes the Kronecker product.

An even more efficient last-layer approximation can be obtained using a Kronecker-factored matrix normal distribution [Louizos and Welling, 2016, Sun et al., 2017, Ritter et al., 2018]. That is, we assume the posterior distribution to be

$$p(\mathbf{W}^L|\mathcal{D}) \approx \mathcal{MN}(\mathbf{W}^L|\mathbf{W}^L_{\text{MAP}}, \mathbf{U}, \mathbf{V}) \,, \quad (29)$$

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 $\mathbf{H}^{-1}_{\mathbf{W}^L}$ [Martens and Grosse, 2015] and $\mathcal{MN}$ denotes the Matrix Normal distribution. In this case, for any $\mathbf{x} \in \mathbb{R}^N$, one can easily show that the distribution over logits is given by

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mathbf{W}^L_{\text{MAP}}\boldsymbol{\phi}(\mathbf{x}), (\boldsymbol{\phi}(\mathbf{x})^\top \mathbf{V}\boldsymbol{\phi}(\mathbf{x}))\mathbf{U}) \,, \quad (30)$$

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

$$p(\mathbf{W}^L|\mathcal{D}) \approx \mathcal{N}(\text{vec}(\mathbf{W}^L)|\text{vec}(\mathbf{W}^L_{\text{MAP}}), \text{diag}(\boldsymbol{\sigma}^2)) \,, \quad (31)$$

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 $\text{vec}(\mathbf{W}^L_{\text{MAP}})$.

OOD DETECTION

The test scenarios are: A two-layer convolutional network trained on the MNIST dataset [LeCun, 1998]. The OOD datasets for this case are FMNIST [Xiao et al., 2017], notMNIST [Bulatov, 2011], and KMNIST [Clanuwat et al., 2018]. For larger datasets, i.e. CIFAR-10 [Krizhevsky et al., 2014], SVHN [Netzer et al., 2011], and CIFAR-100 [Krizhevsky et al., 2014], we use a ResNet-18 network [He et al., 2016]. In all scenarios, the networks are well-trained with 99% test accuracy on MNIST, 95.4% on CIFAR-10, 76.6% on CIFAR-100, and 100% on SVHN. For the sampling baseline, we use 100 posterior samples.

All network have been trained with conventional setups, i.e. we use ADAM with learning rate $1e-3$ and weight decay $5e-4$ for the MNIST experiments and SGD with a cosine annealing scheduler starting at learning rate 0.1 and momentum 0.9 for the CIFAR and SVHN experiments.

PROBIT VS LB

The KFAC setting of the probit comparison can be found in Table 1. Especially in the MNIST scenario the probit approximation seems to break down since even in-dist detection is at chance level. The LB, on the other hand, yields reasonable results.

References

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *ICML*, pages 1613–1622. PMLR, 2015.

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.

Yaroslav Bulatov. notMNIST dataset, 2011. URL http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical Japanese literature. *arXiv*, abs/1812.01718, 2018.

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

P. Hennig. *Approximate Inference in Graphical Models*. PhD thesis, University of Cambridge, November 2010.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being Bayesian, even just a bit, fixes overconfidence in relu networks. In *ICML*, pages 5436–5446. PMLR, 2020.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The CIFAR-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

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

Christos Louizos and Max Welling. Structured and efficient Variational deep learning with matrix Gaussian posteriors. In *ICML*, 2016.

David J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Comput.*, 4(3): 448âĂŞ472, May 1992. ISSN 0899-7667.

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

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

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

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 ICML*, volume 37 of *Proceedings of Machine Learning Research*, pages 2171–2180, Lille, France, 07–09 Jul 2015. PMLR.

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

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.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv*, abs/1708.07747, 2017.