Posterior Inferred, Now What?
Streamlining Prediction in Bayesian Deep Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

The rising interest in Bayesian deep learning (BDL) has led to a plethora of methods for estimating the posterior distribution. However, efficient computation of inferences, such as predictions, has been largely overlooked with Monte Carlo integration remaining the standard. In this work we examine streamlining prediction in BDL through a single forward pass without sampling. For this we use local linearisation on activation functions and local Gaussian approximations at linear layers. Thus allowing us to analytically compute an approximation to the posterior predictive distribution. We showcase our approach for both MLP and transformer architectures and assess its performance on regression and classification tasks.

1 Introduction

Through the success of machine learning models in real-world applications, ensuring their reliability and robustness has become a key concern. In particular, in applications such as aided medical diagnosis [1], autonomous driving [15], or supporting scientific discovery [17], providing reliable predictions, identifying failure modes, and identify how to reduce uncertainties of the system is vital. Uncertainty quantification is at the core of these topics with Bayesian deep learning (BDL, [20, 16]) providing a promising paradigm for assessing uncertainties effectively and efficiently.

The central goal in BDL is to make inferences w.r.t. the posterior distribution over the probabilistic model (the parameters or the function itself). For example, to compute the expected prediction, estimate model uncertainties, or use it within acquisition functions in active learning. For this, we need to first estimate the posterior distribution and secondly make inferences of interest based on the estimated posterior. While both of these steps typically involve intractable integration, only the first step has seen significant progress in recent years [2, 14, 3]. For the second step, in case of a Laplace approximation (LA, [10]), globally linearising the model function around the maximum *a posteriori* (MAP) estimate to perform inferences [12, 8] has shown promise in providing good predictive uncertainty. However, for all other posterior approximation methods, sampling based approximations remain to be the default. Given the high dimensionality of neural networks, sophisticated sampling methods are usually computationally prohibited and vanilla Monte-Carlo sampling is typically employed.

In this work, we tackle this problem by streamlining the prediction in BDL through local linearisation of activation functions and by utilising local Gaussian approximations at linear layers. Instead of a sample based approximation, which requires multiple re-evaluations of the network, we analytically approximate the posterior predictive distribution in a single forward pass through the network, making our methods well-suited for large-scale applications. Moreover, in contrast to global linearisation, our method is suitable for more complex inference tasks as the neural network function becomes locally linear with respect to the inputs. Empirically, we find that local linearisation and local Gaussian approximation of neural networks to provide accurate predictive uncertainties and predictions, while

Submitted to Workshop on Bayesian Decision-making and Uncertainty, 38th Conference on Neural Information Processing Systems (BDU at NeurIPS 2024). Do not distribute.

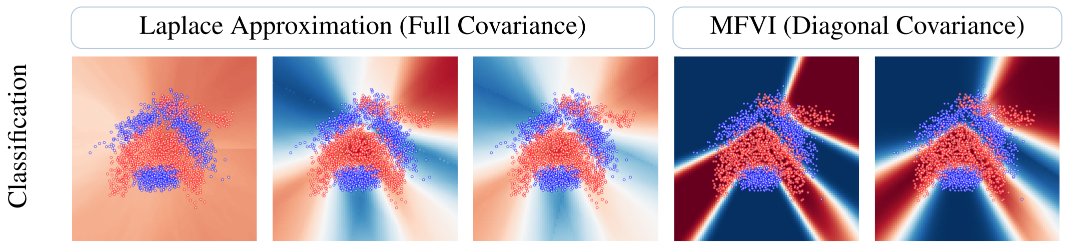


Figure 1: Ours gives better predictive uncertainties and decision boundaries compared with sampling in both Laplace approximation (LA) and mean-field variational inference (MFVI), while having matching performance with global linearised model (GLM) in LA.

being conceptually simple. Fig. 1 shows the posterior predictive densities for our proposal, compared to sampling based approximations and global linearisation in case of a Laplace approximation.

The contributions of our work can be summarised as follows: *(i)* We propose a sampling-free and deterministic method for approximating the posterior predictive distribution through local linearisation of activation functions and local Gaussian approximations in neural networks. *(ii)* We show how to exploit different covariance structures of the approximate posterior and present a streamlined prediction path for both MLP and transformer architectures. *(iii)* We evaluate our method on regression and classification tasks and find that our method result in good predictive performance.

2 Methods

We denote the model parameters as $\boldsymbol{\theta}$ and the training set as $\mathcal{D}$. Given the inferred approximate posterior $q(\boldsymbol{\theta} \mid \mathcal{D}) = \mathcal{N}(\mathbb{E}[\boldsymbol{\theta}], \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$, we aim to approximate the posterior predictive distribution for a new data point $\boldsymbol{x}^*$ in a tractable form with a single forward pass, *i.e.*, approximate $p(\boldsymbol{y}^* \mid \boldsymbol{x}^*, \mathcal{D}) = \int p(\boldsymbol{y}^* \mid \boldsymbol{x}^*, \boldsymbol{\theta}) q(\boldsymbol{\theta} \mid \mathcal{D}) \mathrm{d}\boldsymbol{\theta}$. This problem can be divided into two sub-problems: *(i)* estimate the output distribution at linear layers, and *(ii)* propagating this resulting distribution through a non-linear activation function. We will tackle these two sub-problems separately one after another.

Local Gaussian Approximations for Linear Layers Denote the weight and bias of the l^{th} linear layer as $\boldsymbol{W}^{(l)} \in \mathbb{R}^{\mathrm{D_{out}} \times \mathrm{D_{in}}}$ and $\boldsymbol{b}^{(l)} \in \mathbb{R}^{\mathrm{D_{out}}}$ respectively, and its input as $\boldsymbol{a}^{(l-1)} \in \mathbb{R}^{\mathrm{D_{in}}}$. Then the output $\boldsymbol{h}^{(l)}$ is given as $\boldsymbol{h}^{(l)} = \boldsymbol{W}^{(l)} \boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}$, where we use $h_k^{(l)}$ to denote the k^{th} element. We drop the superscript if it is clear from the context. Assumung that $\boldsymbol{W}$, $\boldsymbol{b}$ and $\boldsymbol{a}$ are Gaussian distributed, we make the following two assumptions to obtain a tractable approximation on $\boldsymbol{h}$: *(i)* we assume $a_i W_{ki}$ is Gaussian, and *(ii)* we assume a_i and W_{ki} are uncorrelated.

Under assumption *(i)*, as the sum of Gaussian random variables ($a_i W_{ki}$) is still Gaussian, h_k will be Gaussian as well. Conseuqently, $\boldsymbol{h}$ will be jointly Gaussian distributed. The mean of $\boldsymbol{h}$ is given as $\mathbb{E}[\boldsymbol{h}] = \mathbb{E}[\boldsymbol{W}] \mathbb{E}[\boldsymbol{a}] + \mathbb{E}[\boldsymbol{b}]$ and the covariance between the k^{th} and the l^{th} hidden unit is computed as follows: $\mathbb{C}\mathrm{ov}[h_k, h_l] =$

$$\sum_{1 \le i,j \le \mathrm{D_{in}}} \mathbb{C}\mathrm{ov}[a_i W_{ki}, a_j W_{lj}] + \sum_{1 \le i \le \mathrm{D_{in}}} (\mathbb{E}[a_i] \mathbb{C}\mathrm{ov}[W_{ki}, b_l] + \mathbb{E}[a_i] \mathbb{C}\mathrm{ov}[W_{li}, b_k]) + \mathbb{C}\mathrm{ov}[b_k, b_l], \quad (1)$$

where $\mathbb{C}\mathrm{ov}[a_i W_{ki}, a_j W_{lj}] =$

$$\mathbb{E}[a_i] \mathbb{E}[a_j] \mathbb{C}\mathrm{ov}[W_{ki}, W_{lj}] + \mathbb{E}[W_{ki}] \mathbb{E}[W_{lj}] \mathbb{C}\mathrm{ov}[a_i, a_j] + \mathbb{C}\mathrm{ov}[a_i, a_j] \mathbb{C}\mathrm{ov}[W_{ki}, W_{lj}]. \quad (2)$$

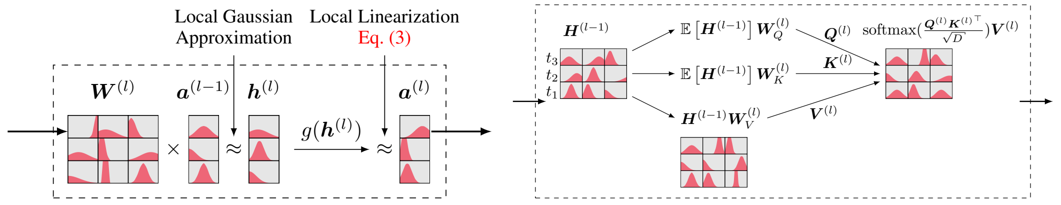
Note that structure of the posterior covariance influences the computational cost of the approximation.

Local Linearizations of Activation Functions Let $g(\cdot)$ denote a non-linear activation function computing $\boldsymbol{a} = g(\boldsymbol{h})$ for an input $\boldsymbol{h}$. Given $\boldsymbol{h} \sim \mathcal{N}(\mathbb{E}[\mathbf{h}], \boldsymbol{\Sigma}_{\mathbf{h}})$, we use a first order Taylor expansion of $g(\cdot)$ at the input mean $\mathbb{E}[\boldsymbol{h}]$ to obtain a tractable approximation of the distribution over $\boldsymbol{a}$, *i.e.*,

$$g(\boldsymbol{h}) \approx g(\mathbb{E}[\boldsymbol{h}]) + \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]} (\boldsymbol{h} - \mathbb{E}[\boldsymbol{h}]), \quad (3)$$

where $\boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}$ is the Jacobian of $g(\cdot)$ at $\boldsymbol{h} = \mathbb{E}[\boldsymbol{h}]$. As Gaussian distributions are closed under linear transformations, now $\boldsymbol{a}$ will also be Gaussian distributed, *i.e.*,

$$\boldsymbol{a} \sim \mathcal{N}(g(\mathbb{E}[\boldsymbol{h}]), \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}^{\top} \boldsymbol{\Sigma}_{\boldsymbol{h}} \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}). \quad (4)$$


Local Gaussian Approximation
Local Linearization Eq. (3)
$\boldsymbol{W}^{(l)}$
$\boldsymbol{a}^{(l-1)}$
$\boldsymbol{h}^{(l)}$
$g(\boldsymbol{h}^{(l)})$
$\boldsymbol{a}^{(l)}$
$\boldsymbol{H}^{(l-1)}$
t_3
t_2
t_1
$\mathbb{E}\left[\boldsymbol{H}^{(l-1)}\right]\boldsymbol{W}_Q^{(l)}$
$\mathbb{E}\left[\boldsymbol{H}^{(l-1)}\right]\boldsymbol{W}_K^{(l)}$
$\boldsymbol{H}^{(l-1)}\boldsymbol{W}_V^{(l)}$
$\boldsymbol{Q}^{(l)}$
$\boldsymbol{K}^{(l)}$
$\boldsymbol{V}^{(l)}$
$\mathrm{softmax}(\frac{\boldsymbol{Q}^{(l)}\boldsymbol{K}^{(l)\top}}{\sqrt{D}})\boldsymbol{V}^{(l)}$

(a) Multi-layer Perceptron (MLP) (b) Attention Block

Figure 2: Illustration of streamlined prediction through different network architecures. In MLPs, we perform a local Gaussian approximation for linear layers and locally linearise the activation function at each layer. The distribution over activations is then proporgated to the next layer. In transformer architecures, we treat the query $\boldsymbol{Q}$ and key $\boldsymbol{K}$ deterministically and use a local Gaussian approximation to obtain a tractable distribution on the value $\boldsymbol{V}$. See App. C.6 for details.

Note that the quality of the local linearisation will depend on the scale of the distribution over the input $\boldsymbol{h}$. Combining local Gaussian approximations for linear layers and local linearisation for non-linear activation functions results in a tractable approximation to the posterior predictive distribution. Fig. 2 illustrates our streamlined prediction for multi-layer perceptrons (MLP) and attention blocks in tranformers, for a detailed description on the approach for transformers see App. C.6. Note that the mean and covariance of the posterior predictive distribution can be computed in a single forward pass.

Covariance Structure Computing the full covariance of the posterior is usually infeasible due to high computational and memory cost. Diagonal approximation and Kronecker-factorization of the covariance/precision are two of the most common aprpoaches. For diagonal covariance, calculating the posterior predictive distribution is straightforward, see App. C.2 for details. In case of Kronecker factors, we developed a tailored block retrieval method for efficient propagation of uncertainties, see App. C.3 for details. Note that other covariance structures can exploited in a similar fashion.

Computational Complexity We will briefly discuss the computational complexity of our method for the case of full covariance. Observe from Eqs. (1) and (2) that the computational cost to obtain $(\mathbb{C}\mathrm{ov}[h_k, h_l])$ is $\mathcal{O}({\mathrm{D}_{\mathrm{in}}^{(l)}}^2)$. Therefore, computing the output covariance at the l^{th} linear layer will be in the order of $\mathcal{O}({\mathrm{D}_{\mathrm{out}}^{(l)}}^2 {\mathrm{D}_{\mathrm{in}}^{(l)}}^2)$. For element-wise activation functions, the computational cost will be $\mathcal{O}({\mathrm{D}_{\mathrm{out}}^{(l)}}^2)$. Hence, we obtain a total cost of $\mathcal{O}(\sum_{l=1}^{L} {\mathrm{D}_{\mathrm{out}}^{(l)}}^2 {\mathrm{D}_{\mathrm{in}}^{(l)}}^2) + {\mathrm{D}_{\mathrm{out}}^{(l)}}^2)$ for a network with L layers. By exploiting the covariance structure, the total computational cost can be substantially reduced.

3 Experiments

We evaluate our method on regression and classification tasks. We choose the Laplace approximation (LA, [3]) and mean-field variational inference (MFVI, [18]) to esimate the posterior. We compare predicitions based on sampling, global linearisation (GLM, [8]), and our method. We use a paired t-test with $p = 0.05$ and bold results with significant statistical difference. For our method, we addionally fit a scale factor, multiplied to the predictive variance, by minimizing the negative log predictive density (NLPD) on the training set. This is necessary, as the predictive variance in case of deep and wide network with diagonal covariance structure can be large.

Regression We experiment with multi-layer perceptron (MLP) for regression. See App. D.1 for experiment setup details and additional results. We use full covariance for LA. As shown in Table Table 1, for MFVI our proposal (Ours) result in better performance than sampling on 8 data sets and matches the performance on the remaining 3 data sets. For LA, our approach obtains better performance than sampling on all data sets.

Classification We train an MLP from scratch and fine-tune a pre-trained Vision transformer (ViT) base model [5]. See App. D.2 for experiment setup details and additional results. With LA, we use a Kronecker-factorized covariance for MLPs and a diagonal covariance for ViT models. As shown in Table 2, we obtain better performance when compared with sampling and GLM. For ViT, fine-tuning on SVHN with MFVI failed, resulting in unreliable results.

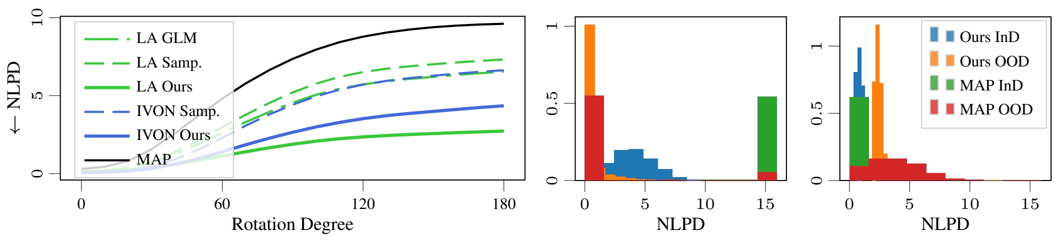
Table 1: Negative log predictive density ↓ on UCI regression data sets. Ours results in better or matching performance compared with sampling and GLM, indicating the effectiveness of our method.

		MFVI (Diagonal Covariance)		*Laplace Approximation (Full Covariance)*		
	(n, d)	Sampling	Ours	Sampling	GLM	Ours
SERVO	(167, 4)	$\mathbf{1.287{\pm}0.069}$	$\mathbf{1.136{\pm}0.182}$	$3.795{\pm}0.110$	$\mathbf{1.047{\pm}0.172}$	$1.443{\pm}0.077$
LD	(345, 5)	$\mathbf{1.346{\pm}0.280}$	$\mathbf{1.369{\pm}0.440}$	$2.221{\pm}0.110$	$\mathbf{1.495{\pm}0.580}$	$\mathbf{1.474{\pm}0.648}$
AM	(398, 7)	$1.004{\pm}0.052$	$\mathbf{0.807{\pm}0.087}$	$1.812{\pm}0.065$	$\mathbf{0.492{\pm}0.279}$	$\mathbf{0.478{\pm}0.309}$
REV	(414, 6)	$1.076{\pm}0.059$	$\mathbf{0.925{\pm}0.091}$	$1.932{\pm}0.045$	$0.859{\pm}0.129$	$\mathbf{0.833{\pm}0.156}$
FF	(517, 12)	$\mathbf{2.160{\pm}3.003}$	$\mathbf{2.333{\pm}3.671}$	$2.086{\pm}0.292$	$\mathbf{1.584{\pm}0.950}$	$\mathbf{1.596{\pm}1.217}$
ITT	(1020, 33)	$0.937{\pm}0.047$	$\mathbf{0.841{\pm}0.065}$	$1.681{\pm}0.069$	$0.825{\pm}0.095$	$\mathbf{0.756{\pm}0.164}$
CCS	(1030, 8)	$0.939{\pm}0.068$	$\mathbf{0.828{\pm}0.108}$	$1.612{\pm}0.048$	$0.319{\pm}0.109$	$\mathbf{0.234{\pm}0.161}$
ASN	(1503, 5)	$0.962{\pm}0.054$	$\mathbf{0.899{\pm}0.065}$	$1.788{\pm}0.045$	$0.422{\pm}0.109$	$\mathbf{0.396{\pm}0.133}$
CAC	(1994, 127)	$0.973{\pm}0.092$	$\mathbf{0.920{\pm}0.118}$	$1.848{\pm}0.055$	$\mathbf{1.281{\pm}0.069}$	$2.662{\pm}1.096$
PT	(5875, 19)	$0.976{\pm}0.069$	$\mathbf{0.940{\pm}0.074}$	$0.984{\pm}0.101$	$\mathbf{0.576{\pm}0.181}$	$0.651{\pm}0.306$
CCPP	(9568, 4)	$0.365{\pm}0.040$	$\mathbf{0.352{\pm}0.042}$	$1.345{\pm}0.085$	$\mathbf{-0.062{\pm}0.182}$	$\mathbf{-0.062{\pm}0.200}$
Bold Count		3	11	0	7	8

Table 2: Negative log predictive density ↓ on classification data sets. Ours results in better or matching performance when compared with sampling, indicating the effectiveness of our approximation.

		MFVI (Diagonal Covariance)		*LA (Kron. Cov. for MLP, Diag. Cov. for ViT)*		
		Sampling	Ours	Sampling	GLM	Ours
MNIST	MLP	$0.081{\pm}0.087$	$\mathbf{0.066{\pm}0.050}$	$0.141{\pm}0.138$	$0.137{\pm}0.122$	$\mathbf{0.116{\pm}0.038}$
FMNIST	MLP	$0.746{\pm}0.323$	$\mathbf{0.458{\pm}0.131}$	$1.283{\pm}0.498$	$1.249{\pm}0.482$	$\mathbf{0.430{\pm}0.113}$
CIFAR-10	ViT	$\mathbf{0.580{\pm}1.305}$	$0.598{\pm}1.328$	$2.389{\pm}0.214$	$0.992{\pm}0.155$	$\mathbf{0.845{\pm}0.179}$
SVHN	ViT	$\mathbf{12.820{\pm}2.820}$	$\mathbf{12.820{\pm}2.820}$	$2.522{\pm}0.578$	$1.225{\pm}0.287$	$\mathbf{0.767{\pm}0.597}$

To test our method on out-of-distribution (OOD) data, we first evaluate the MNIST trained model on rotated MNIST as shown in Fig. 3a. We observe that with increasing roation degree, the increase in NLPD is less compared with other methods. In addition, we show OOD results on a FMNIST trained MLP and CIFAR-10 trained ViT model. For this, we evaluate the MLP on MNIST and the ViT on SVHN As shown in Figs. 3b and 3c, our method can distinguish between in-distribution (InD) and OOD better than the MAP estimate.



(a) Evaluate MNIST trained MLP on rotated MNIST. (b) FMNIST → MNIST. (c) CIFAR-10 → SVHN.

Figure 3: Fig. 3a shows the performance of MNIST-trained model on rotated MNIST and Ours results in lower NLPD. Figs. 3b and 3c shows the NLPD for InD and OOD data using the posterior inferred by LA. Compared with MAP, Ours results in a more clear distribution shift. These Out-of-distribution detection results indicate Ours has good OOD predictive uncertainty.

4 Discussion & Conclusion

In this work, we proposed to streamline prediction in Bayesian deep learning by local linearisation and local Gaussian approximations. For this, we discussed the propgation in different neural network architecures and covariance structures. We showed through a series of experiments that our method obtains high predictive performance, obtain good predictive uncertainties, and can distinguish between in-distribution and OOD data. In future work, we aim to extend our approach to other network architectures, such as convolutional layers, and utilize our approach in more complex inference tasks.

References

[1] E. Begoli, T. Bhattacharya, and D. Kusnezov. The need for uncertainty quantification in machine-assisted medical decision making. *Nature Machine Intelligence*, 1(1):20–23, 2019. 1

[2] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *Proceedings of the 32th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, pages 1613–1622. PMLR, 2015. 1, 7

[3] E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig. Laplace redux - effortless bayesian deep learning. In *Advances in Neural Information Processing Systems (NeurIPS) 34*, volume 34, pages 20089–20103. MIT Press, 2021. 1, 3, 11

[4] E. Daxberger, E. Nalisnick, J. U. Allingham, J. Antorán, and J. M. Hernández-Lobato. Bayesian deep learning via subnetwork inference. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, pages 2510–2521. PMLR, 2021. 7

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 3

[6] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, pages 1050–1059. PMLR, 2016. 7

[7] M. N. Gibbs. *Bayesian Gaussian processes for regression and classification*. PhD thesis, Citeseer, 1998. 12

[8] A. Immer, M. Korzepa, and M. Bauer. Improving predictions of bayesian neural nets via local linearization. In *Proceedings of the twenty forth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130 of *Proceedings of Machine Learning Research*, pages 703–711. PMLR, 2021. 1, 3, 7

[9] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 876–885. Association For Uncertainty in Artificial Intelligence (AUAI), 2018. 7

[10] A. Kristiadi, M. Hein, and P. Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, pages 5436–5446. PMLR, 2020. 1, 7

[11] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems (NeurIPS) 30*, 30:6402–6413, 2017. 7

[12] N. D. Lawrence. *Variational inference in probabilistic models*. PhD thesis, Citeseer, 2001. 1

[13] D. J. MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992. 12

[14] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems (NeurIPS) 32*, volume 32, pages 13132–13143. MIT Press, 2019. 1, 7

[15] R. Michelmore, M. Wicker, L. Laurenti, L. Cardelli, Y. Gal, and M. Kwiatkowska. Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 7344–7350. IEEE, 2020. 1

[16] T. Papamarkou, M. Skoularidou, K. Palla, L. Aitchison, J. Arbel, D. Dunson, M. Filippone, V. Fortuin, P. Hennig, A. Hubin, et al. Position paper: Bayesian deep learning in the age of large-scale ai. *arXiv preprint arXiv:2402.00809*, 2024. 1

[17] A. F. Psaros, X. Meng, Z. Zou, L. Guo, and G. E. Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics*, 477:111902, 2023. 1

[18] Y. Shen, N. Daheim, B. Cong, P. Nickl, G. M. Marconi, B. C. E. M. Raoul, R. Yokota, I. Gurevych, D. Cremers, M. E. Khan, and T. Möllenhoff. Variational learning is effective for large deep networks. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research. PMLR, 2024. 3, 7

[19] D. J. Spiegelhalter and S. L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20(5):579–605, 1990. 12

[20] A. G. Wilson and P. Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in Neural Information Processing Systems (NeurIPS) 33*, 33:4697–4708, 2020. 1

Posterior Inferred, Now What? Streamlining Prediction in Bayesian Deep Learning

Supplementary Material

We first introduce notation in App. A and related work in App. B. Then, we introduce the derivation of our method in App. C. At last, we describe the experiment setup and additional experiment results in App. D.

A Notation

We use lowercase bold letter for vector, *e.g.*, $\boldsymbol{x}$, and uppercase bold letter for matrix, *e.g.*, $\boldsymbol{W}$. We use subscript to denote element of vector and matrix, *e.g.*, x_i (i^{th} element) and W_{ki} (k^{th} row, i^{th} column). We use $W[k,:]$ to indicate the k^{th} row of a matrix.

B Related Work

Inferring Posterior in Bayesian Deep Learning There has been many methods developed which can be roughly grouped into three categories: *(i)* Laplace approximation based methods: Starting from [10] where simple post-hoc Laplace approximation (LA) has shown promising results on neural network, LA has gained increasing attention ever since. [4] has shown that treating a subnetwork Bayesian will also result in good predictive uncertainties. *(ii)*: Variational inference (VI) based methods: [2] showed mean-field VI (MFVI) could improve generalisation in small-scale neural network and [18] showned MFVI is effective for large-scale neural networks as well. *(iii)*: Others: Monte Carlo Dropout [6] aims to estimate predictive uncertainty by interpreting dropout in neural networks as a form of Bayesian approximation. Deep ensemble [11] combines the outputs of multiple independently trained models to capture predictive uncertainty. Stochastic Weight Averaging-Gaussian [14], which extends Stochastic Weight Averaging [9] by capturing the posterior distribution of model weights using a Gaussian approximation.

Making Prediction in Bayesian Deep Learning Little work has been done for this and the usual go-to solution is simple Monte Carlo Estimation. For Laplace approximation, [8] proposed a global liberalised model for better posterior prediction.

C Derivations

We derive the approximate posterior predictive distribution form in this section. App. C.1 is for the case where the covariance has full structure in linear layer. App. C.2 is for the case where the covariance has diagonal structure in linear layer. App. C.3 is for the case where the covariance has Kronecker-factorised structure in linear layer. App. C.4 is the derivation for activation layers. App. C.5 describes the probit approximation for approximate the posterior prediction for classification. App. C.6 describes how to apply our method for the transformer.

C.1 Derivation for General Covariance Structure

Denote the weight and bias of a linear layer as $\boldsymbol{W} \in \mathbb{R}^{\mathrm{D_{out}} \times \mathrm{D_{in}}}$ and $\boldsymbol{b} \in \mathbb{R}^{\mathrm{D_{out}}}$ respectively, and its input as $\boldsymbol{a} \in \mathbb{R}^{\mathrm{D_{in}}}$. The output is $\boldsymbol{h} = \boldsymbol{W}\boldsymbol{a} + \boldsymbol{b}$ with its k^{th} element being $h_k = \sum_{i=1}^{\mathrm{D_{in}}} W_{ki} a_i + b_k$.

We make the following two assumptions to obtain tractable distribution on the output:

- Assumption 1: We assume $a_i W_{ki}$ is a Gaussian distribution.
- Assumption 2: We assume a_i and W_{ki} are uncorrelated.

From assumption 1, given $\boldsymbol{W}$, $\boldsymbol{a}$, and $\boldsymbol{b}$ are all Gaussian, each h_k will be a Gaussian distribution. As a result, $\boldsymbol{h}$ will be a Gaussian distribution as well.

We now derive its mean and covariance. We first derive the mean for each h_k. As a_i and W_{ki} are uncorrelated, we have

$$\begin{aligned}
\mathbb{E}\left[h_k\right] &= \mathbb{E}\left[\sum_{i=1}^{\mathrm{D_{in}}} W_{ki} a_i + b_k\right] && (5)\\
&= \sum_{i=1}^{\mathrm{D_{in}}} \mathbb{E}\left[W_{ki} a_i + b_k\right] && (6)\\
&= \sum_{i=1}^{\mathrm{D_{in}}} \mathbb{E}\left[W_{ki} a_i\right] + \mathbb{E}\left[b_k\right] && (7)\\
&\approx \sum_{i=1}^{\mathrm{D_{in}}} \mathbb{E}\left[W_{ki}\right] \mathbb{E}\left[a_i\right] + \mathbb{E}\left[b_k\right] && (8)
\end{aligned}$$

We now derive the covariance. Define $h'_k = \sum_{i=1}^{\mathrm{D_{in}}} a_i W_{ki}$, we have

$$\begin{aligned}
\mathbb{C}\mathrm{ov}\left[h_k, h_l\right] &= \mathbb{C}\mathrm{ov}\left[h'_k + b_k, h'_l + b_l\right] && (9)\\
&= \mathbb{C}\mathrm{ov}\left[h'_k, h'_l\right] + \mathbb{C}\mathrm{ov}\left[h'_k, b_l\right] + \mathbb{C}\mathrm{ov}\left[h'_l, b_k\right] + \mathbb{C}\mathrm{ov}\left[b_k, b_l\right]. && (10)
\end{aligned}$$

where $\mathbb{C}\mathrm{ov}[h'_k, h'_l]$ is

$$\begin{aligned}
\mathbb{C}\mathrm{ov}\left[h'_k, h'_l\right] &= \mathbb{C}\mathrm{ov}\left[\sum_{1 \le i \le D_{\mathrm{in}}} a_i W_{ki}, \sum_{1 \le j \le D_{\mathrm{in}}} a_j W_{lj}\right] && (11)\\
&= \sum_{1 \le i \le D_{\mathrm{in}}} \sum_{1 \le j \le D_{\mathrm{in}}} \mathbb{C}\mathrm{ov}\left[a_i W_{ki}, a_j W_{lj}\right]. && (12)
\end{aligned}$$

To derive the form of $\mathbb{C}\mathrm{ov}[a_i W_{ki}, a_j W_{lj}]$, we use assumption 2:

$$\begin{aligned}
&\mathbb{C}\mathrm{ov}\left[a_i W_{ki}, a_j W_{lj}\right]\\
&= \mathbb{E}\left[(a_i W_{ki} - \mathbb{E}\left[a_i W_{ki}\right])(a_j W_{lj} - \mathbb{E}\left[a_j W_{lj}\right])\right] && (13)\\
&= \mathbb{E}\left[a_i W_{ki} a_j W_{lj} - a_i W_{ki} \mathbb{E}\left[a_j W_{lj}\right] - \mathbb{E}\left[a_i W_{ki}\right] a_j W_{lj} + \mathbb{E}\left[a_i W_{ki}\right] \mathbb{E}\left[a_j W_{lj}\right]\right] && (14)\\
&= \mathbb{E}\left[a_i a_j W_{ki} W_{lj}\right] - \mathbb{E}\left[a_i W_{ki}\right] \mathbb{E}\left[a_j W_{lj}\right] - \mathbb{E}\left[a_i W_{ki}\right] \mathbb{E}\left[a_j W_{lj}\right] + \mathbb{E}\left[a_i W_{ki}\right] \mathbb{E}\left[a_j W_{lj}\right] && (15)\\
&\approx \mathbb{E}\left[a_i a_j\right] \mathbb{E}\left[W_{ki} W_{lj}\right] - \mathbb{E}\left[a_i\right] \mathbb{E}\left[W_{ki}\right] \mathbb{E}\left[a_j\right] \mathbb{E}\left[W_{lj}\right] && \text{(Assumption 2)}\\
&= (\mathbb{E}\left[a_i\right] \mathbb{E}\left[a_j\right] + \mathbb{C}\mathrm{ov}\left[a_i, a_j\right])(\mathbb{E}\left[W_{ki}\right] \mathbb{E}\left[W_{lj}\right] + \mathbb{C}\mathrm{ov}\left[W_{ki}, W_{lj}\right]) - \mathbb{E}\left[a_i\right] \mathbb{E}\left[W_{ki}\right] \mathbb{E}\left[a_j\right] \mathbb{E}\left[W_{lj}\right] && (16)\\
&= \mathbb{E}\left[a_i\right] \mathbb{E}\left[a_j\right] \mathbb{C}\mathrm{ov}\left[W_{ki}, W_{lj}\right] + \mathbb{E}\left[W_{ki}\right] \mathbb{E}\left[W_{lj}\right] \mathbb{C}\mathrm{ov}\left[a_i, a_j\right] + \mathbb{C}\mathrm{ov}\left[a_i, a_j\right] \mathbb{C}\mathrm{ov}\left[W_{ki}, W_{lj}\right] && (17)
\end{aligned}$$

Now the only term left is $\mathbb{C}\text{ov}[h'_k, b_l]$, which can be written as

$$\begin{aligned}
\mathbb{C}\text{ov}\left[h'_k, b_l\right] &= \mathbb{C}\text{ov}\left[\sum_{i=1}^{\text{D}_\text{in}} a_i W_{ki}, b_l\right] \\
&= \sum_{i=1}^{\text{D}_\text{in}} \mathbb{C}\text{ov}\left[a_i W_{ki}, b_l\right] && (18) \\
&= \sum_{i=1}^{\text{D}_\text{in}} \mathbb{E}\left[(a_i W_{ki} - \mathbb{E}\left[a_i W_{ki}\right])(b_l - \mathbb{E}\left[b_l\right])\right] && (19) \\
&\approx \sum_{i=1}^{\text{D}_\text{in}} \mathbb{E}\left[(a_i W_{ki} - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right])(b_l - \mathbb{E}\left[b_l\right])\right] && \text{(Assumption 2)} \\
&= \sum_{i=1}^{\text{D}_\text{in}} \mathbb{E}\left[a_i W_{ki} b_l - a_i W_{ki}\mathbb{E}\left[b_l\right] - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right] b_l + \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right]\right] && (20) \\
&= \sum_{i=1}^{\text{D}_\text{in}} \mathbb{E}\left[a_i W_{ki} b_l\right] - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right] && (21) \\
&\approx \sum_{i=1}^{\text{D}_\text{in}} \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki} b_l\right] - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right] && \text{(Assumption 2)} \\
&= \sum_{i=1}^{\text{D}_\text{in}} \mathbb{E}\left[a_i\right]\left(\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right] + \mathbb{C}\text{ov}\left[W_{ki}, b_l\right]\right) - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right] && (22) \\
&= \sum_{i=1}^{\text{D}_\text{in}} \mathbb{E}\left[a_i\right]\mathbb{C}\text{ov}\left[W_{ki}, b_l\right] && (23)
\end{aligned}$$

Putting it together, we have $\mathbb{C}\text{ov}[h_k, h_l] =$

$$\sum_{1\le i,j\le \text{D}_\text{in}} \mathbb{C}\text{ov}\left[a_i W_{ki}, a_j W_{lj}\right] + \sum_{i=1}^{\text{D}_\text{in}} \left(\mathbb{E}\left[a_i\right]\mathbb{C}\text{ov}\left[W_{ki}, b_l\right] + \mathbb{E}\left[a_i\right]\mathbb{C}\text{ov}\left[W_{li}, b_k\right]\right) + \mathbb{C}\text{ov}\left[b_k, b_l\right], \quad (24)$$

where $\mathbb{C}\text{ov}[a_i W_{ki}, a_j W_{lj}] =$

$$\mathbb{E}\left[a_i\right]\mathbb{E}\left[a_j\right]\mathbb{C}\text{ov}\left[W_{ki}, W_{lj}\right] + \mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[W_{lj}\right]\mathbb{C}\text{ov}\left[a_i, a_j\right] + \mathbb{C}\text{ov}\left[a_i, a_j\right]\mathbb{C}\text{ov}\left[W_{ki}, W_{lj}\right]. \quad (25)$$

Note that the first term in Eq. (24) could be rewrite into the form of matrix multiplication which results in an efficient implementation:

$$\sum_{1\le i,j\le \mathrm{D_{in}}} \mathbb{C}\mathrm{ov}\left[a_i W_{ki}, a_j W_{lj}\right] \tag{26}$$

$$= \sum_{1\le i,j\le \mathrm{D_{in}}} \mathbb{E}\left[a_i\right]\mathbb{E}\left[a_j\right]\mathbb{C}\mathrm{ov}\left[W_{ki}, W_{lj}\right] + \mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[W_{lj}\right]\mathbb{C}\mathrm{ov}\left[a_i, a_j\right] + \mathbb{C}\mathrm{ov}\left[a_i, a_j\right]\mathbb{C}\mathrm{ov}\left[W_{ki}, W_{lj}\right] \tag{27}$$

$$= \begin{bmatrix} \mathbb{E}\left[a_1\right]\mathbb{E}\left[a_1\right]\mathbb{C}\mathrm{ov}[W_{k1}, W_{l1}] & \dots & \mathbb{E}\left[a_1\right]\mathbb{E}\left[a_{\mathrm{D_{in}}}\right]\mathbb{C}\mathrm{ov}[W_{k1}, W_{l\,\mathrm{D_{in}}}] \\ \vdots & \vdots & \vdots \\ \mathbb{E}\left[a_{\mathrm{D_{in}}}\right]\mathbb{E}\left[a_1\right]\mathbb{C}\mathrm{ov}[W_{k\,\mathrm{D_{in}}}, W_{l1}] & \dots & \mathbb{E}\left[a_1\right]\mathbb{E}\left[a_{\mathrm{D_{in}}}\right]\mathbb{C}\mathrm{ov}[W_{k\,\mathrm{D_{in}}}, W_{l\,\mathrm{D_{in}}}] \end{bmatrix} \tag{28}$$

$$\odot \begin{bmatrix} \mathbb{C}\mathrm{ov}[W_{k1}, W_{l1}] & \dots & \mathbb{C}\mathrm{ov}[W_{k1}, W_{l\,\mathrm{D_{in}}}] \\ \vdots & \vdots & \vdots \\ \mathbb{C}\mathrm{ov}[W_{k\,\mathrm{D_{in}}}, W_{l1}] & \dots & \mathbb{C}\mathrm{ov}[W_{k\,\mathrm{D_{in}}}, W_{l\,\mathrm{D_{in}}}] \end{bmatrix} \tag{29}$$

$$+ \begin{bmatrix} \mathbb{E}\left[W_{k1}\right]\mathbb{E}\left[W_{l1}\right] & \dots & \mathbb{E}\left[W_{k1}\right]\mathbb{E}\left[W_{l\,\mathrm{D_{in}}}\right] \\ \vdots & \vdots & \vdots \\ \mathbb{E}\left[W_{k\,\mathrm{D_{in}}}\right]\mathbb{E}\left[W_{l1}\right] & \dots & \mathbb{E}\left[W_{k\,\mathrm{D_{in}}}\right]\mathbb{E}\left[W_{l\,\mathrm{D_{in}}}\right] \end{bmatrix} \odot \begin{bmatrix} \mathbb{C}\mathrm{ov}[a_1, a_1] & \dots & \mathbb{C}\mathrm{ov}[a_1, a_{\mathrm{D_{in}}}] \\ \vdots & \vdots & \vdots \\ \mathbb{C}\mathrm{ov}[a_{\mathrm{D_{in}}}, a_1] & \dots & \mathbb{C}\mathrm{ov}[a_{\mathrm{D_{in}}}, a_{\mathrm{D_{in}}}] \end{bmatrix} \tag{30}$$

$$+ \begin{bmatrix} \mathbb{C}\mathrm{ov}[a_1, a_1] & \dots & \mathbb{C}\mathrm{ov}[a_1, a_{\mathrm{D_{in}}}] \\ \vdots & \vdots & \vdots \\ \mathbb{C}\mathrm{ov}[a_{\mathrm{D_{in}}}, a_1] & \dots & \mathbb{C}\mathrm{ov}[a_{\mathrm{D_{in}}}, a_{\mathrm{D_{in}}}] \end{bmatrix} \odot \begin{bmatrix} \mathbb{C}\mathrm{ov}[W_{k1}, W_{l1}] & \dots & \mathbb{C}\mathrm{ov}[W_{k1}, W_{l\,\mathrm{D_{in}}}] \\ \vdots & \vdots & \vdots \\ \mathbb{C}\mathrm{ov}[W_{k\,\mathrm{D_{in}}}, W_{l1}] & \dots & \mathbb{C}\mathrm{ov}[W_{k\,\mathrm{D_{in}}}, W_{l\,\mathrm{D_{in}}}] \end{bmatrix} \tag{31}$$

C.2 Derivation for Diagonal Covariance Structure

When the posterior has diagonal covariance, the mean $\mathbb{E}\left[h_k\right]$ will still be the same.

For covariance, note that as now the posterior is diagonal, when $k \neq l$, we have $\mathbb{C}\mathrm{ov}[h_k, h_l] =$

$$\sum_{1\le i,j\le \mathrm{D_{in}}} \mathbb{C}\mathrm{ov}\left[a_i W_{ki}, a_j W_{lj}\right] + \sum_{i=1}^{\mathrm{D_{in}}} \left(\mathbb{E}\left[a_i\right]\mathbb{C}\mathrm{ov}\left[W_{ki}, b_l\right] + \mathbb{E}\left[a_i\right]\mathbb{C}\mathrm{ov}\left[W_{li}, b_k\right]\right) + \mathbb{C}\mathrm{ov}\left[b_k, b_l\right] \tag{32}$$

$$= \sum_{1\le i,j\le \mathrm{D_{in}}} \mathbb{C}\mathrm{ov}\left[a_i W_{ki}, a_j W_{lj}\right] \tag{33}$$

$$= \sum_{1\le i,j\le \mathrm{D_{in}}} \mathbb{E}\left[a_i\right]\mathbb{E}\left[a_j\right]\mathbb{C}\mathrm{ov}\left[W_{ki}, W_{lj}\right] + \mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[W_{lj}\right]\mathbb{C}\mathrm{ov}\left[a_i, a_j\right] + \mathbb{C}\mathrm{ov}\left[a_i, a_j\right]\mathbb{C}\mathrm{ov}\left[W_{ki}, W_{lj}\right] \tag{34}$$

$$= \sum_{1\le i,j\le \mathrm{D_{in}}} \mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[W_{lj}\right]\mathbb{C}\mathrm{ov}\left[a_i, a_j\right] \tag{35}$$

For $k = l$, we have $\mathbb{V}\mathrm{ar}[h_k] =$

$$\sum_{1\le i,j\le \mathrm{D_{in}}} \mathbb{C}\mathrm{ov}\left[a_i W_{ki}, a_j W_{kj}\right] + \sum_{i=1}^{\mathrm{D_{in}}} \left(\mathbb{E}\left[a_i\right]\mathbb{C}\mathrm{ov}\left[W_{ki}, b_k\right] + \mathbb{E}\left[a_i\right]\mathbb{C}\mathrm{ov}\left[W_{ki}, b_k\right]\right) + \mathbb{V}\mathrm{ar}\left[b_k\right] \tag{36}$$

$$= \sum_{1\le i\le \mathrm{D_{in}}} \mathbb{C}\mathrm{ov}\left[a_i W_{ki}, a_i W_{ki}\right] + \mathbb{V}\mathrm{ar}\left[b_k\right] \tag{37}$$

$$= \sum_{1\le i\le \mathrm{D_{in}}} \mathbb{E}\left[a_i\right]^2 \mathbb{V}\mathrm{ar}\left[W_{ki}\right] + \mathbb{E}\left[W_{ki}\right]^2 \mathbb{V}\mathrm{ar}\left[a_i\right] + \mathbb{V}\mathrm{ar}\left[a_i\right]\mathbb{V}\mathrm{ar}\left[W_{ki}\right] + \mathbb{V}\mathrm{ar}\left[b_k\right] \tag{38}$$

C.3 Derivation for Kronecker Covariance Structure

In Kronecker approximation, the Hessian is represented in Kronecker product form:

$$\boldsymbol{h} = \boldsymbol{A} \otimes \boldsymbol{B} \tag{39}$$

Denote the prior precision as λ^2, then the posterior precision is

$$\mathbf{P} = \boldsymbol{h} + \lambda^2 \mathbf{I} = \boldsymbol{A} \otimes \boldsymbol{B} + \lambda^2 \mathbf{I} \tag{40}$$

To improve numerical stability, an eigen-decomposition is often performed on $\boldsymbol{A}$ and $\boldsymbol{B}$ in `Laplace Redux` library:

$$\begin{aligned}
\mathbf{P} &= (\boldsymbol{U}_A \boldsymbol{\Lambda}_A \boldsymbol{U}_A^\top) \otimes (\boldsymbol{U}_B \boldsymbol{\Lambda}_B \boldsymbol{U}_B^\top) + \lambda^2 \mathbf{I} && \text{(Definition)} \\
&= (\boldsymbol{U}_A \otimes \boldsymbol{U}_B)(\boldsymbol{\Lambda}_A \otimes \boldsymbol{\Lambda}_B)(\boldsymbol{U}_A \otimes \boldsymbol{U}_B)^\top + \lambda^2 \mathbf{I} && ((\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}))
\end{aligned}$$

For computational efficiency, for our forward pass we will represent the covariance as $\boldsymbol{C} \otimes \boldsymbol{D}$ form, which results in an approximation:

$$\begin{aligned}
\mathbf{P} &\approx (\boldsymbol{U}_A \otimes \boldsymbol{U}_B)((\boldsymbol{\Lambda}_A + \lambda \mathbf{I}_A) \otimes (\boldsymbol{\Lambda}_B + \lambda \mathbf{I}_B))(\boldsymbol{U}_A \otimes \boldsymbol{U}_B)\top & (41) \\
&= \left(\left[(\boldsymbol{U}_A(\boldsymbol{\Lambda}_A + \lambda \mathbf{I}_A)) \otimes (\boldsymbol{U}_B(\boldsymbol{\Lambda}_B + \lambda \mathbf{I}_B)) \right] (\boldsymbol{U}_A \otimes \boldsymbol{U}_B)^\top \right)^{-1} & (42) \\
&= (\boldsymbol{U}_A(\boldsymbol{\Lambda}_A + \lambda \mathbf{I}_A)\boldsymbol{U}_A^\top)^{-1} \otimes (\boldsymbol{U}_B(\boldsymbol{\Lambda}_B + \lambda \mathbf{I}_B)\boldsymbol{U}_B^\top)^{-1} & (43) \\
&= (\boldsymbol{U}_A \otimes \boldsymbol{U}_B)(\boldsymbol{\Lambda}_A \otimes \boldsymbol{\Lambda}_B + \lambda^2 \mathbf{I})(\boldsymbol{U}_A \otimes \boldsymbol{U}_B)^\top + \lambda \mathbf{I}_A \otimes \boldsymbol{\Lambda}_B + \boldsymbol{\Lambda}_A \otimes \lambda \mathbf{I}_B, & (44)
\end{aligned}$$

where the extra term introduced by the approximation is written in blue colour.

Recall for an efficient implementation for computing $\sum_{1 \le i,j \le \mathrm{D_{in}}} \mathbb{C}\mathrm{ov}[a_i W_{ki}, a_j W_{lj}]$ in Eq. (31), we need to retrieve the covariance between the k^{th} row of weight and l^{th} row of weight, which is a $\mathrm{D_{in}} \times \mathrm{D_{in}}$ matrix:

$$\mathbb{C}\mathrm{ov}\left[\boldsymbol{W}[k,:], \boldsymbol{W}[l,:]\right] = \begin{bmatrix} \mathbb{C}\mathrm{ov}[W_{k1}, W_{l1}] & \dots & \mathbb{C}\mathrm{ov}[W_{k1}, W_{l\,\mathrm{D_{in}}}] \\ \vdots & \vdots & \vdots \\ \mathbb{C}\mathrm{ov}[W_{k\,\mathrm{D_{in}}}, W_{l1}] & \dots & \mathbb{C}\mathrm{ov}[W_{k\,\mathrm{D_{in}}}, W_{l\,\mathrm{D_{in}}}] \end{bmatrix} \tag{45}$$

However, for posterior stored in Kronecker product form, we will have $\mathrm{D_{in}} \times \mathrm{D_{in}}$ numbers of $\mathrm{D_{out}} \times \mathrm{D_{out}}$ matrix, which complicates the retrieval of $\mathbb{C}\mathrm{ov}[\boldsymbol{W}[k,:], \boldsymbol{W}[l,:]]$.

C.4 Derivation for Activation Layers

For $\boldsymbol{a} = g(\boldsymbol{h})$ where $\boldsymbol{h} \sim \mathcal{N}(\boldsymbol{h}; \mathbb{E}[\boldsymbol{h}], \boldsymbol{\Sigma}_h)$ and $g(\cdot)$ is the activation function, we use local linearisation to approximate the distribution of $\boldsymbol{a}$. Specifically, we do a first-order Taylor expansion on $g(\cdot)$ at $\mathbb{E}[\boldsymbol{h}]$:

$$\begin{aligned}
\boldsymbol{a} &= g(\boldsymbol{h}) & (46) \\
&\approx g(\mathbb{E}[\boldsymbol{h}]) + \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}(\boldsymbol{h} - \mathbb{E}[\boldsymbol{h}]) & (47)
\end{aligned}$$

Given that Gaussian distribution is closed under linear transformation, we have

$$\begin{aligned}
\boldsymbol{h} &\sim \mathcal{N}(\mathbb{E}[\boldsymbol{h}], \boldsymbol{\Sigma}_h) & (48) \\
\boldsymbol{h} - \mathbb{E}[\boldsymbol{h}] &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_h) & (49) \\
\boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}(\boldsymbol{h} - \mathbb{E}[\boldsymbol{h}]) &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}^\top \boldsymbol{\Sigma}_h \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}) & (50) \\
g(\mathbb{E}[\boldsymbol{h}]) + \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}(\boldsymbol{h} - \mathbb{E}[\boldsymbol{h}]) &\sim \mathcal{N}(g(\mathbb{E}[\boldsymbol{h}]), \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}^\top \boldsymbol{\Sigma}_h \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}) & (51) \\
\boldsymbol{a} &\underset{\text{approx}}{\sim} \mathcal{N}(\boldsymbol{a}; g(\mathbb{E}[\boldsymbol{h}]), \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}^\top \boldsymbol{\Sigma}_h \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}) & (52)
\end{aligned}$$

C.5 Probit Approximation for Classification

Following [3], in classification we treat the logits before last layer activation (softmax) as model output f. Then we can use probit approximation to get posterior predictive:

Binary [13, 19]

$$
\begin{aligned}
p\left(y^* \mid x^*\right) &= \int_{\mathbb{R}} \operatorname{sigmoid}\left(f^*\right) \mathcal{N}\left(f^* \mid \mu^*, {\sigma^*}^2\right) \mathrm{d} f^* && (53)\\
&\approx \int \Phi\left(f^*\right) \mathcal{N}\left(f^* \mid \mu^*, {\sigma^*}^2\right) \mathrm{d} f^* && (54)\\
&= \sigma\left(\frac{\mu^*}{\sqrt{1+\frac{\pi}{8} {\sigma^*}^2}}\right). && (55)
\end{aligned}
$$

Multi-class [7]

$$
\begin{aligned}
p\left(\mathbf{y}^* \mid \mathbf{x}^*\right) &= \int_{\mathbb{R}^C} \operatorname{softmax}\left(\mathbf{f}^*\right) \mathcal{N}\left(\mathbf{f}^* \mid \mu^*, \boldsymbol{\Sigma}^*\right) \mathrm{d} \mathbf{f}^* \\
&\overset{\text{j-th element}}{\approx} \frac{\exp\left(\tau_i\right)}{\sum_{j=1}^C \exp\left(\tau_j\right)}, \text{ where } \tau_j = \frac{\mu_j^*}{\sqrt{1+\frac{\pi}{8} \Sigma_{jj}^*}}
\end{aligned} \qquad (56)
$$

C.6 Transformer Block

There are four components in each transformer block: (1) multi-head attention; (2) MLP; (3) layer normalisation; and (4) residual connection. We treat MLP bayesian and multi-head attention deterministic. For layer normalisation and residual connection, as Gaussian distribution is closed under linear transformation, push distribution over them is straightforward. For MLP, the computation is the same as described above. We describe how to push distribution through attention block below.
Attention Block Given an input $\boldsymbol{H} \in \mathbb{R}^{T \times D}$ where T is the number of tokens in the input sequence and D is the dimension of each token, denote the query, key and value matrices as $\boldsymbol{W}_Q \in \mathbb{R}^{D \times D}$, $\boldsymbol{W}_K \in \mathbb{R}^{D \times D}$, $\boldsymbol{W}_V \in \mathbb{R}^{D \times D}$ respectively, the key, query and value in an attention blocks are

$$
\boldsymbol{Q} = \boldsymbol{H}\boldsymbol{W}_Q, \quad \boldsymbol{K} = \boldsymbol{H}\boldsymbol{W}_K, \quad \boldsymbol{V} = \boldsymbol{H}\boldsymbol{V}_Q \qquad (57)
$$

and the output of attention block is

$$
\text{Attention}(\boldsymbol{H}) = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}}\right)\boldsymbol{V} \qquad (58)
$$

When the input $\boldsymbol{H}$ is a distribution, $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ will all be distribution as well. As pushing a distribution over softmax requires further approximation, for computational reason we ignore the distribution over $\boldsymbol{Q}$ and $\boldsymbol{K}$ and compute their value by using the mean of input:

$$
\boldsymbol{Q} = \mathbb{E}\left[\boldsymbol{H}\right]\boldsymbol{W}_Q, \quad \boldsymbol{K} = \mathbb{E}\left[\boldsymbol{H}\right]\boldsymbol{W}_K \qquad (59)
$$

We keep the distribution over $\boldsymbol{V}$ and compute its distribution according to the structure of input's covariance accordingly. Once we have the distribution over $\boldsymbol{V}$, getting the distribution over Attention$(\boldsymbol{H})$ will becomes obtain the distribution of linear combination of Gaussian, which is tractable. Then for multi-head attention, we assume each attention head's output is independent, which allows us to compute the distribution over the final output in tractable form.

D Experiment

D.1 Regression

Table 3 gives the UCI regression data set information and the neural network structure we used. For all neural networks, we use ReLU activation function. In Table 4 we report the Root Mean Square Error (RMSE), Ours results in matching or better performance compared with sampling and GLM, indicating the effectiveness of our method. Note that as the mean of the posterior prediction of our method is the same as the prediction made by setting the weights of the neural network to be the mean of the posterior, we result in the same prediction as GLM of LA, and hence the same performance.

Table 3: UCI regression experiment setup.

Dataset Name	Shorthand	(n, d)	Network Structure
SERVO	SERVO	(167, 4)	d-50-1
LIVER DISORDERS	LD	(345, 5)	d-50-1
AUTO MPG	AM	(398, 7)	d-50-1
REAL ESTATE VALUATION	REV	(414,6)	d-50-1
FOREST FIRES	FF	(517, 12)	d-50-1
INFRARED THERMOGRAPHY TEMPERATURE	ITT	(1020, 33)	d-100-1
CONCRETE COMPRESSIVE STRENGTH	CCS	(1030, 8)	d-100-1
AIRFOIL SELF-NOISE	ASN	(1503, 5)	d-100-1
COMMUNITIES AND CRIME	CAC	(1994, 127)	d-100-1
PARKINSONS TELEMONITORING	PT	(5875, 19)	d-50-50-1
COMBINED CYCLE POWER PLANT	CCPP	(9568, 4)	d-50-50-1

Table 4: Root Mean Square Error ↓ on UCI regression data sets. Ours results in better or matching performance compared with sampling and GLM, indicating the effectiveness of our method.

		MFVI (Diag. Cov.)		*Laplace Approximation (Full Cov.)*		
	(n, d)	Sampling	Ours	Sampling	GLM	Ours
SERVO	(167, 4)	0.749 ± 0.147	$\mathbf{0.740 \pm 0.143}$	1.632 ± 0.233	$\mathbf{0.658 \pm 0.141}$	$\mathbf{0.658 \pm 0.141}$
LD	(345, 5)	$\mathbf{0.884 \pm 0.273}$	$\mathbf{0.881 \pm 0.272}$	$\mathbf{0.989 \pm 0.441}$	$\mathbf{0.977 \pm 0.418}$	$\mathbf{0.977 \pm 0.418}$
AM	(398, 7)	$\mathbf{0.415 \pm 0.115}$	$\mathbf{0.417 \pm 0.113}$	0.505 ± 0.105	$\mathbf{0.371 \pm 0.103}$	$\mathbf{0.371 \pm 0.103}$
REV	(414, 6)	$\mathbf{0.563 \pm 0.096}$	$\mathbf{0.562 \pm 0.095}$	0.789 ± 0.130	$\mathbf{0.532 \pm 0.104}$	$\mathbf{0.532 \pm 0.104}$
FF	(517, 12)	$\mathbf{0.874 \pm 1.123}$	$\mathbf{0.874 \pm 1.124}$	$\mathbf{0.910 \pm 0.824}$	$\mathbf{0.852 \pm 0.792}$	$\mathbf{0.852 \pm 0.792}$
ITT	(1020, 33)	$\mathbf{0.481 \pm 0.057}$	$\mathbf{0.497 \pm 0.066}$	0.560 ± 0.075	$\mathbf{0.507 \pm 0.072}$	$\mathbf{0.507 \pm 0.072}$
CCS	(1030, 8)	$\mathbf{0.472 \pm 0.102}$	$\mathbf{0.476 \pm 0.106}$	0.494 ± 0.102	$\mathbf{0.301 \pm 0.057}$	$\mathbf{0.301 \pm 0.057}$
ASN	(1503, 5)	0.568 ± 0.062	$\mathbf{0.560 \pm 0.062}$	0.550 ± 0.069	$\mathbf{0.352 \pm 0.055}$	$\mathbf{0.352 \pm 0.055}$
CAC	(1994, 127)	$\mathbf{0.571 \pm 0.105}$	0.585 ± 0.092	1.481 ± 0.167	$\mathbf{0.703 \pm 0.101}$	$\mathbf{0.703 \pm 0.101}$
PT	(5875, 19)	0.601 ± 0.067	$\mathbf{0.590 \pm 0.068}$	0.479 ± 0.081	$\mathbf{0.410 \pm 0.076}$	$\mathbf{0.410 \pm 0.076}$
CCPP	(9568, 4)	$\mathbf{0.241 \pm 0.038}$	$\mathbf{0.241 \pm 0.038}$	0.358 ± 0.041	$\mathbf{0.224 \pm 0.037}$	$\mathbf{0.224 \pm 0.037}$
Bold Count		8	10	2	11	11

D.2 Classification

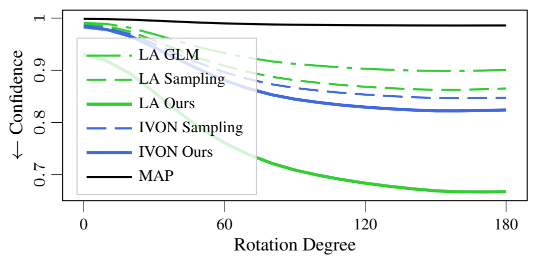
Table 5 gives the classification data sets information and the neural network structure we used. We use ReLU activation for MLP. For ViT, we make the MLP block in the last two transformer block and the classification head Bayesian, and treat the rest of the weight deterministically. In Table 6 we report the test accuracy, on SVHN the fine-tuning of MFVI failed and hence the bad performance. For the rest, Ours results in matching or better performance compared with sampling and GLM, indicating the effectiveness of our method.

Table 5: Classification experiment setup.

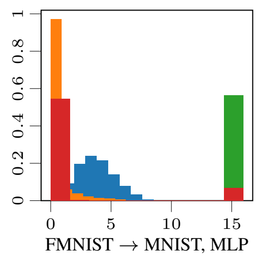
Dataset Name	(n, d)	Network Structure
MNIST	(50000, 784)	d-128-64-10
FMNIST	(50000, 784)	d-128-64-10
CIFAR-10	(50000, 3, 32, 32)	ViT-base
SVHN	(73257, 3, 32, 32)	ViT-base

Table 6: Accuracy ↑ on classification data sets. Ours results in better or matching performance compared with sampling and GLM, indicating the effectiveness of our method.

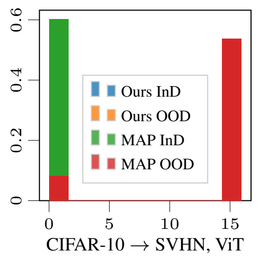
		MFVI (Diag. Cov.)		*LA (Kron. Cov. for MLP, Diag. Cov. for ViT)*		
		Sampling	Ours	Sampling	GLM	Ours
MNIST	ViT	$\mathbf{0.981 \pm 0.015}$	$\mathbf{0.981 \pm 0.015}$	$\mathbf{0.976 \pm 0.017}$	0.974 ± 0.018	0.974 ± 0.017
FMNIST	ViT	$\mathbf{0.864 \pm 0.043}$	$\mathbf{0.863 \pm 0.044}$	$\mathbf{0.873 \pm 0.041}$	$\mathbf{0.873 \pm 0.040}$	$\mathbf{0.871 \pm 0.042}$
CIFAR-10	ViT	$\mathbf{0.959 \pm 0.089}$	$\mathbf{0.959 \pm 0.089}$	0.109 ± 0.143	$\mathbf{0.972 \pm 0.072}$	$\mathbf{0.971 \pm 0.074}$
SVHN	ViT	$\mathbf{0.196 \pm 0.177}$	$\mathbf{0.196 \pm 0.177}$	0.197 ± 0.180	0.724 ± 0.200	$\mathbf{0.758 \pm 0.191}$



(a) Evaluate MNIST trained MLP on Rotated MNIST.



(b) Evaluate FMNIST trained MLP on FMNIST (InD) and MNIST (OOD).



(c) Evaluate CIFAR-10 trained ViT on CIFAR-10 (InD) and SVHN (OOD).

Figure 4: Fig. 4a shows the performance on rotated MNIST and Ours results in lower NLPD. Figs. 4b and 4c shows the NLPD for Ind and OOD data using the posterior inferred by MFVI. On CIFAR-10 as the posterior inferred by MFVI is extremely peaked (the highest variance being 0.0004), Ours has the almost same result as MAP. For FMNIST to MNIST, compared with MAP, Ours results in a more clear distribution shift. These Out-of-distribution detection results indicate Ours has good OOD predictive uncertainty.