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

Rui Li **Marcus Klasson** **Arno Solin** **Martin Trapp**
Department of Computer Science, Aalto University, Finland
{firstname.lastname}@aalto.fi

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.

See also extended paper at https://arxiv.org/abs/2411.18425.

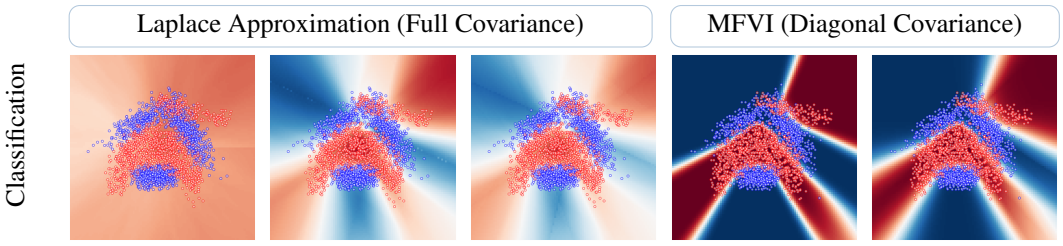
1 Introduction

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 [18], or supporting scientific discovery [22], 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, [27, 20]) 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 [3, 16, 4]. For the second step, in case of a Laplace approximation (LA, [11]), globally linearising the model function around the maximum *a posteriori* (MAP) estimate to perform inferences [13, 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 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

Workshop on Bayesian Decision-making and Uncertainty, 38th Conference on Neural Information Processing Systems (NeurIPS 2024).



(a) Sampling (b) GLM (c) Ours (d) Sampling (e) Ours

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.

approximation of neural networks to provide accurate predictive uncertainties and predictions, while 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 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 [11] where simple post-hoc Laplace approximation (LA) has shown promising results, LA has gained increasing attention ever since. Recent works applied LA methods in various applications, such as large language models [29, 10] and dynamic neural networks [17]. *(ii)*: Variational inference (VI) based methods: [3] showed mean-field VI (MFVI) could improve generalisation in small-scale neural network and [24] 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 [12] combines the outputs of multiple independently trained models to capture predictive uncertainty. Stochastic Weight Averaging-Gaussian [16], 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 and the usual go-to solution is Monte Carlo Estimation. For Laplace approximation, [8] proposed the linearised LA by performing a global linearisation and has shown promise in providing useful predictive uncertainties.

3 Methods

In Bayesian deep learning (BDL), predicting the output y (*e.g.*, class label, regression value) for an input $\boldsymbol{x} \in \mathcal{X}$ is performed by *marginalizing* out the model parameters $\boldsymbol{\theta}$ of the neural network $f_{\boldsymbol{\theta}}(\cdot)$ instead of trusting a single point estimate, *i.e.*,

$$p(y \mid \boldsymbol{x}, \mathcal{D}) = \int_{\boldsymbol{\theta}} p(y \mid f_{\boldsymbol{\theta}}(\boldsymbol{x}))\, p(\boldsymbol{\theta} \mid \mathcal{D})\, \mathrm{d}\boldsymbol{\theta}, \tag{1}$$

where $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ denotes the training data and the posterior distribution $p(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{p(\boldsymbol{\theta}, \mathcal{D})}{p(\mathcal{D})}$ is given by Bayes' rule. However, for most neural networks integrating over the high-dimensional parameter space is intractable, necessitating the use of approximations to compute the posterior distribution $p(\boldsymbol{\theta} \mid \mathcal{D})$ and the posterior predictive distribution $p(y \mid \boldsymbol{x}, \mathcal{D})$.

Let the weights and biases of the m^{th} linear layer of the network f be denoted as $\boldsymbol{W}^{(m)} \in \mathbb{R}^{\mathrm{D_{out}} \times \mathrm{D_{in}}}$ and $\boldsymbol{b}^{(m)} \in \mathbb{R}^{\mathrm{D_{out}}}$, respectively. Then the pre-activation $\boldsymbol{h}^{(m)}$ is given as $\boldsymbol{h}^{(m)} = \boldsymbol{W}^{(m)}\boldsymbol{a}^{(m-1)} + \boldsymbol{b}^{(m)}$, where $\boldsymbol{a}^{(m-1)} \in \mathbb{R}^{\mathrm{D_{in}}}$ is the activation of the previous layer. In case $m = 1$, then $\boldsymbol{a}^{(0)}$ corresponds to the input $\boldsymbol{x}$. We further denote the k^{th} element of $\boldsymbol{h}^{(m)}$ as $h_k^{(m)} = \sum_{i=1}^{\mathrm{D_{in}}} W_{ki}^{(m)} a_i^{(m-1)} + b_k^{(m)}$ and drop the superscript if it is clear from the context.

Given an approximate posterior distribution $q(\boldsymbol{\theta})$ with $\boldsymbol{\theta} = \{\boldsymbol{W}^{(m)}, \boldsymbol{b}^{(m)}\}_{m=1}^{M}$, we aim to compute the probability distribution of the activation $\boldsymbol{a}^{(m)}$ of each layer m. For this, we need to estimate the distribution of the pre-activation $\boldsymbol{h}^{(m)}$ and then compute an approximation to the activation $\boldsymbol{a}^{(m)}$ after application of a non-linear activation function $g(\cdot)$.

Approximating the pre-activation distribution In case the activation $\boldsymbol{a}^{(m-1)}$ is deterministically give, *i.e.*, for the input layer, we can compute the distribution over pre-activations analytically as a consequence of the stability of stable distributions under linear transformations [21]. However, for hidden layers the distribution over pre-activations is generally not of the same family as the posterior distribution [28]. Nevertheless, we will apply a local Gaussian approximation to the pre-activation at every hidden layer. Specifically, we make the assumption:

Assumption 3.1. *Assume that the activations of the previous layer $a_i^{(m-1)}$ and parameters of the m^{th} layer are independent.*

Then followed by a Gaussian approximation of $a_i^{(m-1)} W_{ki}^{(m)}$ for each i and each k, the mean of the pre-activation $\boldsymbol{h}^{(m)}$ is given as:

$$\mathbb{E}\left[\boldsymbol{h}^{(m)}\right] = \mathbb{E}\left[\boldsymbol{W}^{(m)}\right]\mathbb{E}\left[\boldsymbol{a}^{(m-1)}\right] + \mathbb{E}\left[\boldsymbol{b}^{(m)}\right], \tag{2}$$

and the covariance between the k^{th} and the j^{th} hidden unit is computed as:

$$\begin{aligned}\mathbb{C}\mathrm{ov}\left[h_k^{(m)}, h_l^{(m)}\right] &= \sum_{1\leq i,j\leq \mathrm{D_{in}}} \mathbb{C}\mathrm{ov}\left[a_i^{(m-1)}W_{ki}^{(m)}, a_j^{(m-1)}W_{lj}^{(m)}\right] + \mathbb{C}\mathrm{ov}\left[b_k^{(m)}, b_l^{(m)}\right] \\ &\quad + \sum_{1\leq i\leq \mathrm{D_{in}}} \mathbb{E}\left[a_i^{(m-1)}\right]\left(\mathbb{C}\mathrm{ov}\left[W_{ki}^{(m)}, b_l^{(m)}\right] + \mathbb{C}\mathrm{ov}\left[W_{li}^{(m)}, b_k^{(m)}\right]\right),\end{aligned} \tag{3}$$

where

$$\begin{aligned}\mathbb{C}\mathrm{ov}\left[a_i^{(m-1)}W_{ki}^{(m)}, a_j^{(m-1)}W_{lj}^{(m)}\right] &= \mathbb{E}\left[a_i^{(m-1)}\right]\mathbb{E}\left[a_j^{(m-1)}\right]\mathbb{C}\mathrm{ov}\left[W_{ki}^{(m)}, W_{lj}^{(m)}\right] \\ &\quad + \mathbb{E}\left[W_{ki}^{(m)}\right]\mathbb{E}\left[W_{lj}^{(m)}\right]\mathbb{C}\mathrm{ov}\left[a_i^{(m-1)}, a_j^{(m-1)}\right] \\ &\quad + \mathbb{C}\mathrm{ov}\left[a_i^{(m-1)}, a_j^{(m-1)}\right]\mathbb{C}\mathrm{ov}\left[W_{ki}^{(m)}, W_{lj}^{(m)}\right].\end{aligned} \tag{4}$$

Depending on the structure of the covariance matrix, we can further simplify the computation of the covariance matrix.

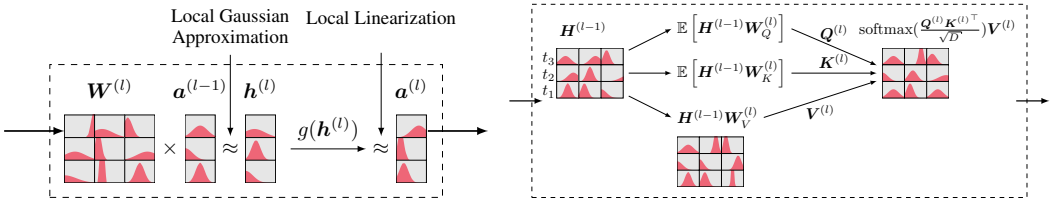
Approximating the activation distribution Let $g(\cdot)$ denote a non-linear activation function computing $\boldsymbol{a} = g(\boldsymbol{h})$ for a pre-activation $\boldsymbol{h}$. Inspired by the application of local linearisation in Bayesian filtering [*e.g.*, 23], we use a first order Taylor expansion of $g(\cdot)$ at the mean of the pre-activation $\mathbb{E}[\boldsymbol{h}]$. Specifically, we approximate $g(\boldsymbol{h})$ using

$$g(\boldsymbol{h}) \approx g(\mathbb{E}[\boldsymbol{h}]) + \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}(\boldsymbol{h} - \mathbb{E}[\boldsymbol{h}]), \tag{5}$$

where $\boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}$ is the Jacobian of $g(\cdot)$ at $\boldsymbol{h} = \mathbb{E}[\boldsymbol{h}]$. Then, as stable distributions are closed under linear transformations, the distribution of $\boldsymbol{a}$ can be computed analytically and is given as follows in case of a Gaussian distributed, *i.e.*,

$$\boldsymbol{a} \sim \mathcal{N}(g(\mathbb{E}[\boldsymbol{h}]), \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}^{\top}\boldsymbol{\Sigma}_{\boldsymbol{h}}\boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}). \tag{6}$$

Note that the quality of the local linearisation will depend on the scale of the distribution over the input $\boldsymbol{h}$. For ReLU activation functions, Petersen et al. [21] have shown that local linearisation provides the optimal Gaussian approximation of a univariate Gaussian distribution in total variation. For classification tasks, we employ a probit approximation [14, 11].



(a) Multi-layer Perceptron (MLP)

(b) Attention Layer

Figure 2: Illustration our approach for different network architectures. In MLPs, we can directly apply local Gaussian approximations and local linearisation of each layer. The distribution over activations is then propagated to the next layer. In attention layers, we treat the query $\boldsymbol{Q}$ and key $\boldsymbol{K}$ deterministically and only treat the value $\boldsymbol{V}$ as a random quantity, resulting in a straightforward propagation path. The resulting distribution is then propagated to the subsequent MLP layer.

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, which can be computed in a single forward pass. 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. B.6.

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. B.2 for details. In case of Kronecker factors, we developed a tailored block retrieval method for efficient propagation of uncertainties, see App. B.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. (3) and (4) that the computational cost to obtain $(\mathbb{C}\mathrm{ov}[h_k, h_l])$ is $\mathcal{O}(\mathrm{D}_{\text{in}}^{(l)^2})$. Therefore, computing the output covariance at the l^{th} linear layer will be in the order of $\mathcal{O}(\mathrm{D}_{\text{out}}^{(l)^2}\mathrm{D}_{\text{in}}^{(l)^2})$. For element-wise activation functions, the computational cost will be $\mathcal{O}(\mathrm{D}_{\text{out}}^{(l)^2})$. Hence, we obtain a total cost of $\mathcal{O}(\sum_{l=1}^{L}\mathrm{D}_{\text{out}}^{(l)^2}\mathrm{D}_{\text{in}}^{(l)^2} + \mathrm{D}_{\text{out}}^{(l)^2})$ for a network with L layers. By explointing the covariance structure, the total computational cost can be substantially reduced.

4 Experiments

We adopt the Laplace approximation (LA, [15]) and mean-field variational inference [MFVI, 2] for approximating the posterior distribution of the network parameters. We compare our method using local Gaussian approximation and local linearisation against Monte Carlo (MC) sampling and a global linearised model [GLM, 8]. For MFVI, we adopt the IVON optimiser [24] to obtain the posterior approximation, which has been shown to be effective and scalable to large-scale classification tasks. Here, we compare our method against MC sampling from the posterior to make predictions as done in Shen et al. [24]. For the MFVI and LA sampling baselines, we used $1,000$ MC samples in the regression and MLP classification experiments, and 50 MC samples for the ViT classification experiments. For our method, we addionally fit a scale factor, multiplied to the predictive variance, by minimizing the negative log predictive density (NLPD) on the validation set. This is necessary, as the predictive variance in case of deep and wide network with diagonal covariance structure can be large. We use a paired t-test with $p = 0.05$ and bold results with significant statistical difference.

Regression We experiment with multi-layer perceptron (MLP) for regression. See App. C.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 For MLP, we train it from scratch and treat all layers Bayesian. For ViT, we fine-tune the MLP layers in the last two blocks in a pre-trained Vision transformer (ViT) base model [5] and

Table 1: Negative log predictive density ↓ on UCI regression. 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)	**1.287**±0.069	**1.136**±0.182	3.795±0.110	**1.047**±0.172	1.443±0.077
LD	(345, 5)	**1.346**±0.280	**1.369**±0.440	2.221±0.110	**1.495**±0.580	**1.474**±0.648
AM	(398, 7)	1.004±0.052	**0.807**±0.087	1.812±0.065	**0.492**±0.279	**0.478**±0.309
REV	(414, 6)	1.076±0.059	**0.925**±0.091	1.932±0.045	0.859±0.129	**0.833**±0.156
FF	(517, 12)	**2.160**±3.003	**2.333**±3.671	2.086±0.292	**1.584**±0.950	**1.596**±1.217
ITT	(1020, 33)	0.937±0.047	**0.841**±0.065	1.681±0.069	0.825±0.095	**0.756**±0.164
CCS	(1030, 8)	0.939±0.068	**0.828**±0.108	1.612±0.048	0.319±0.109	**0.234**±0.161
ASN	(1503, 5)	0.962±0.054	**0.899**±0.065	1.788±0.045	0.422±0.109	**0.396**±0.133
CAC	(1994, 127)	0.973±0.092	**0.920**±0.118	1.848±0.055	**1.281**±0.069	2.662±1.096
PT	(5875, 19)	0.976±0.069	**0.940**±0.074	0.984±0.101	**0.576**±0.181	0.651±0.306
CCPP	(9568, 4)	0.365±0.040	**0.352**±0.042	1.345±0.085	**−0.062**±0.182	**−0.062**±0.200
Bold Count		3/11	11/11	0/11	7/11	8/11

later treat them Bayesian. See App. C.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, for both MLP and ViT, we obtain better performance when compared with sampling and GLM.

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	0.179±0.014	**0.086**±0.005	0.210±0.003	**0.089**±0.004	**0.089**±0.005
FMNIST	MLP	2.010±0.051	**0.529**±0.011	0.556±0.008	0.548±0.018	**0.397**±0.010
CIFAR-10	ViT	0.124±0.011	**0.080**±0.005	0.169±0.004	**0.089**±0.005	**0.088**±0.006
CIFAR-100	ViT	0.480±0.018	**0.437**±0.013	1.043±0.010	0.602±0.011	**0.457**±0.012

Robustness to Out-of-distribution We now assess the robustness to out-of-distribution (OOD) data for our method and the baselines. In Fig. 3, we take the ViT network fine-tuned on CIFAR-10 and evaluate its predictive entropy on the SVHN data set [19]. Our method can distinguish between in-distribution and OOD data better than the LA MAP and MFVI Sampling. Although our method underfits on the in-distribution data, the separation between is clear for the OOD data similar. For results on MLP, see App. C.2.

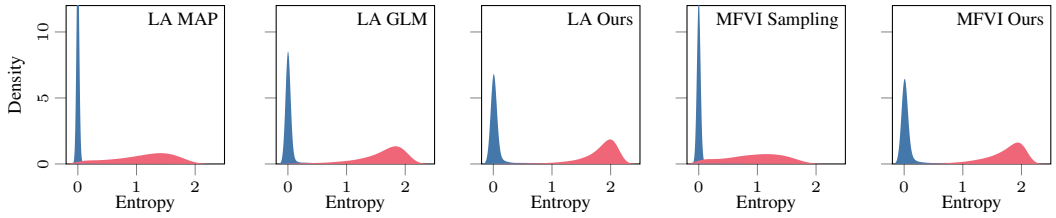


Figure 3: Kernel density plots over the predictive entropy from a ViT network finetuned on CIFAR-10 (blue, in-distribution) and data from SVHN (red, out-of-distribution). Our method results in a clear separation between the in- and out-of-distribution data.

5 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.

Acknowledgements

AS and RL acknowledge funding from the Research Council of Finland (grant number 339730). MT acknowledges funding from the Research Council of Finland (grant number 347279). MK acknowledges funding from the Finnish Center for Artificial Intelligence (FCAI). We acknowledge CSC – IT Center for Science, Finland, for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CSC (Finland) and the LUMI consortium through CSC. We acknowledge the computational resources provided by the Aalto Science-IT project.

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] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017. 4

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

[4] 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. Curran Associates, Inc., 2021. 1, 12

[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 (ICLR)*, 2021. 4

[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)*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059. PMLR, 2016. 2

[7] M. N. Gibbs. *Bayesian Gaussian processes for regression and classification*. Phd thesis, University of Cambridge, 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, 2, 4

[9] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*, pages 876–885. AUAI Press, 2018. 2

[10] P. J. Kampen, G. R. Als, and M. R. Andersen. Towards scalable bayesian transformers: Investigating stochastic subset selection for nlp. In *Proceedings of the 40th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2024. 2

[11] 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)*, volume 119 of *Proceedings of Machine Learning Research*, pages 5436–5446. PMLR, 2020. 1, 2, 3

[12] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS) 30*, volume 30, pages 6402–6413. Curran Associates, Inc., 2017. 2

[13] N. D. Lawrence. *Variational inference in probabilistic models*. PhD thesis, University of Cambridge, 2001. 1

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

[15] D. J. MacKay. Bayesian methods for backpropagation networks. In *Models of neural networks III: association, generalization, and representation*, pages 211–254. Springer, 1996. 4

[16] 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. Curran Associates, Inc., 2019. 1, 2

[17] L. Meronen, M. Trapp, A. Pilzer, L. Yang, and A. Solin. Fixing overconfidence in dynamic neural networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2680–2690, 2024. 2

[18] 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

[19] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011. 5

[20] T. Papamarkou, M. Skoularidou, K. Palla, L. Aitchison, J. Arbel, D. B. Dunson, M. Filippone, V. Fortuin, P. Hennig, J. M. Hernández-Lobato, A. Hubin, A. Immer, T. Karaletsos, M. E. Khan, A. Kristiadi, Y. Li, S. Mandt, C. Nemeth, M. A. Osborne, T. G. J. Rudner, D. Rügamer, Y. W. Teh, M. Welling, A. G. Wilson, and R. Zhang. Position: Bayesian deep learning is needed in the age of large-scale ai. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*. PMLR, 2024. 1

[21] F. Petersen, A. A. Mishra, H. Kuehne, C. Borgelt, O. Deussen, and M. Yurochkin. Uncertainty quantification via stable distribution propagation. In *International Conference on Learning Representations (ICLR)*, 2024. 3

[22] 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

[23] S. Särkkä and L. Svensson. *Bayesian filtering and smoothing*, volume 17. Cambridge university press, 2023. 3

[24] 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)*, volume 235 of *Proceedings of Machine Learning Research*. PMLR, 2024. 2, 4

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

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS) 30*, volume 30. Curran Associates, Inc., 2017. 12

[27] A. G. Wilson and P. Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *Advances in Neural Information Processing Systems (NeurIPS) 33*, volume 33, pages 4697–4708. Curran Associates, Inc., 2020. 1

[28] P. Wolinski and J. Arbel. Gaussian pre-activations in neural networks: Myth or reality? *arXiv preprint arXiv:2205.12379*, 2022. 3

[29] A. X. Yang, M. Robeyns, X. Wang, and L. Aitchison. Bayesian low-rank adaptation for large language models. In *International Conference on Learning Representations (ICLR)*, 2024. 2

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

Supplementary Material

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

A Notation

We list notation that will be used throughout the appendix in Table 3.

Table 3: Notation.

$\boldsymbol{x}$	lowercase bolder letter, vector
$\boldsymbol{W}$	uppercase bold letter, matrix
$\mathcal{D}$	set
x_i	i^{th} element of $\boldsymbol{x}$
W_{ki}	k^{th} row, i^{th} column of $\boldsymbol{W}$
$\boldsymbol{W}[k,:]$	k^{th} row of a matrix
k, l	dimension of the output
i, j	dimension of the input
d	data feature dimension
n, N	number of data points
C	total number of classes
m	layer index

B Derivations

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

B.1 Derivation for General Covariance Structure

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

We make the following assumptions to obtain a tractable distribution on the pre-activation:

- Assumption 1: We assume each $a_i^{(m-1)} W_{ki}^{(m)}$ is a Gaussian distribution.
- Assumption 2: We assume that the activations of the previous layer $a_i^{(m-1)}$ and parameters of the m^{th} layer are independent.

From assumption 1, because now $a_i^{(m-1)} W_{ki}^{(m)}$ and $b_k^{(m)}$ are all Gaussian distributions, $h_k^{(m)}$ will follow Gaussian distribution as well. We call this local Gaussian approximation as we approximate each local component $a_i^{(m-1)} W_{ki}^{(m)}$ with a Gaussian. As now each $h_k^{(m)}$ is a Gaussian, $\boldsymbol{h}^{(m)}$ will be jointly Gaussian. We derive its mean and covariance and drop the layer index if it is clear from the context.

Derivation of mean As a_i is assumed to be uncorrected with W_{ki}, we have

$$\begin{aligned}
\mathbb{E}\left[h_k\right] &= \mathbb{E}\left[\sum_{i=1}^{\mathrm{D_{in}}} W_{ki} a_i + b_k\right] && (7)\\
&= \sum_{i=1}^{\mathrm{D_{in}}} \mathbb{E}\left[W_{ki} a_i + b_k\right] && (8)\\
&= \sum_{i=1}^{\mathrm{D_{in}}} \mathbb{E}\left[W_{ki} a_i\right] + \mathbb{E}\left[b_k\right] && (9)\\
&\approx \sum_{i=1}^{\mathrm{D_{in}}} \mathbb{E}\left[W_{ki}\right] \mathbb{E}\left[a_i\right] + \mathbb{E}\left[b_k\right]. && \text{(Assumption 2)}
\end{aligned}$$

Derivation of covariance The covariance between the k^{th} and l^{th} pre-activation can be written as

$$\begin{aligned}
\mathbb{C}\mathrm{ov}\left[h_k, h_l\right] &= \mathbb{C}\mathrm{ov}\left[\sum_{i=1}^{\mathrm{D_{in}}} a_i W_{ki} + b_k, \sum_{i=1}^{\mathrm{D_{in}}} a_i W_{li} + b_l\right] && (10)\\
&= \mathbb{C}\mathrm{ov}\left[\sum_{i=1}^{\mathrm{D_{in}}} a_i W_{ki}, \sum_{i=1}^{\mathrm{D_{in}}} a_i W_{li}\right] + \mathbb{C}\mathrm{ov}\left[\sum_{i=1}^{\mathrm{D_{in}}} a_i W_{ki}, b_l\right] + \mathbb{C}\mathrm{ov}\left[\sum_{i=1}^{\mathrm{D_{in}}} a_i W_{li}, b_k\right]\\
&\qquad + \mathbb{C}\mathrm{ov}\left[b_k, b_l\right] && (11)\\
&= \sum_{1 \le i,j \le D_{\text{in}}} \mathbb{C}\mathrm{ov}\left[a_i W_{ki}, a_j W_{lj}\right] + \sum_{1 \le i \le D_{\text{in}}} \left(\mathbb{C}\mathrm{ov}\left[a_i W_{ki}, b_l\right] + \mathbb{C}\mathrm{ov}\left[a_i W_{li}, b_k\right]\right)\\
&\qquad + \mathbb{C}\mathrm{ov}\left[b_k, b_l\right] && (12)
\end{aligned}$$

We first derive the form of $\mathbb{C}\mathrm{ov}[a_i W_{ki}, a_i W_{li}]$:

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

Then we drive the form of $\mathbb{C}\mathrm{ov}[a_i W_{ki}, b_l]$:

$$\begin{aligned}
\mathbb{C}\mathrm{ov}\left[a_i W_{ki}, b_l\right] &= \mathbb{E}\left[(a_i W_{ki} - \mathbb{E}\left[a_i W_{ki}\right])(b_l - \mathbb{E}\left[b_l\right])\right] && (18)\\
&\approx \mathbb{E}\left[(a_i W_{ki} - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right])(b_l - \mathbb{E}\left[b_l\right])\right] && \text{(Assumption 2)}\\
&= \mathbb{E}\left[a_i W_{ki} b_l - a_i W_{ki}\mathbb{E}\left[b_l\right] - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right] b_l + \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right]\right] && (19)\\
&= \mathbb{E}\left[a_i W_{ki} b_l\right] - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right] && (20)\\
&\approx \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki} b_l\right] - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right] && \text{(Assumption 2)}\\
&= \mathbb{E}\left[a_i\right]\left(\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right] + \mathbb{C}\mathrm{ov}\left[W_{ki}, b_l\right]\right) - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right] && (21)\\
&= \mathbb{E}\left[a_i\right]\mathbb{C}\mathrm{ov}\left[W_{ki}, b_l\right]. && (22)
\end{aligned}$$

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

$$\sum_{1\le i,j\le \mathrm{D_{in}}} \mathbb{C}\mathrm{ov}\left[a_i W_{ki}, a_j W_{lj}\right] + \sum_{i=1}^{\mathrm{D_{in}}} \left(\mathbb{E}\left[a_i\right]\mathbb{C}\mathrm{ov}\left[W_{ki}, b_l\right] + \mathbb{E}\left[a_i\right]\mathbb{C}\mathrm{ov}\left[W_{li}, b_k\right]\right) + \mathbb{C}\mathrm{ov}\left[b_k, b_l\right], \quad (23)$$

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

$$\mathbb{E}\left[a_i\right]\mathbb{E}\left[a_j\right]\mathbb{C}\mathrm{ov}\left[W_{ki}, W_{lj}\right] + \mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[W_{lj}\right]\mathbb{C}\mathrm{ov}\left[a_i, a_j\right] + \mathbb{C}\mathrm{ov}\left[a_i, a_j\right]\mathbb{C}\mathrm{ov}\left[W_{ki}, W_{lj}\right]. \quad (24)$$

Note that $\sum_{1\le i,j\le \mathrm{D_{in}}} \mathbb{C}\mathrm{ov}[a_i W_{ki}, a_j W_{lj}]$ in Eq. (23) could be rewrite into the form of matrix multiplication for efficient implementation:

$$\begin{aligned}
&\sum_{1\le i,j\le \mathrm{D_{in}}} \mathbb{C}\mathrm{ov}\left[a_i W_{ki}, a_j W_{lj}\right] && (25)\\
&= \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] && (26)\\
&= \sum \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} && (27)\\
&\qquad \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} && (28)\\
&+ \sum \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} && (29)\\
&+ \sum \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} && (30)
\end{aligned}$$

B.2 Derivation for Diagonal Covariance Structure

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

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

$$\sum_{1\leq i,j\leq \mathrm{D_{in}}} \mathbb{C}\text{ov}\left[a_i W_{ki}, a_j W_{lj}\right] + \sum_{i=1}^{\mathrm{D_{in}}} \left(\mathbb{E}\left[a_i\right] \mathbb{C}\text{ov}\left[W_{ki}, b_l\right] + \mathbb{E}\left[a_i\right] \mathbb{C}\text{ov}\left[W_{li}, b_k\right]\right) + \mathbb{C}\text{ov}\left[b_k, b_l\right] \tag{31}$$

$$= \sum_{1\leq i,j\leq \mathrm{D_{in}}} \mathbb{C}\text{ov}\left[a_i W_{ki}, a_j W_{lj}\right] \tag{32}$$

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

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

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

$$\sum_{1\leq i,j\leq \mathrm{D_{in}}} \mathbb{C}\text{ov}\left[a_i W_{ki}, a_j W_{kj}\right] + \sum_{i=1}^{\mathrm{D_{in}}} \left(\mathbb{E}\left[a_i\right] \mathbb{C}\text{ov}\left[W_{ki}, b_k\right] + \mathbb{E}\left[a_i\right] \mathbb{C}\text{ov}\left[W_{ki}, b_k\right]\right) + \mathbb{V}\text{ar}\left[b_k\right] \tag{35}$$

$$= \sum_{1\leq i\leq \mathrm{D_{in}}} \mathbb{C}\text{ov}\left[a_i W_{ki}, a_i W_{ki}\right] + \mathbb{V}\text{ar}\left[b_k\right] \tag{36}$$

$$= \sum_{1\leq i\leq \mathrm{D_{in}}} \mathbb{E}\left[a_i\right]^2 \mathbb{V}\text{ar}\left[W_{ki}\right] + \mathbb{E}\left[W_{ki}\right]^2 \mathbb{V}\text{ar}\left[a_i\right] + \mathbb{V}\text{ar}\left[a_i\right] \mathbb{V}\text{ar}\left[W_{ki}\right] + \mathbb{V}\text{ar}\left[b_k\right] \tag{37}$$

B.3 Derivation for Kronecker Covariance Structure

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

$$\boldsymbol{h} = \boldsymbol{A} \otimes \boldsymbol{B} \tag{38}$$

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

$$\mathbf{P} = \boldsymbol{h} + \lambda^2 \mathbf{I} = \boldsymbol{A} \otimes \boldsymbol{B} + \lambda^2 \mathbf{I} \tag{39}$$

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:

$$\mathbf{P} \approx (\boldsymbol{U}_A \otimes \boldsymbol{U}_B)((\boldsymbol{\Lambda}_A + \lambda \mathbf{I}_A) \otimes (\boldsymbol{\Lambda}_B + \lambda \mathbf{I}_B))(\boldsymbol{U}_A \otimes \boldsymbol{U}_B)\top \tag{40}$$

$$= \left([(\boldsymbol{U}_A(\boldsymbol{\Lambda}_A + \lambda \mathbf{I}_A)) \otimes (\boldsymbol{U}_B(\boldsymbol{\Lambda}_B + \lambda \mathbf{I}_B))]\,(\boldsymbol{U}_A \otimes \boldsymbol{U}_B)^\top\right)^{-1} \tag{41}$$

$$= (\boldsymbol{U}_A(\boldsymbol{\Lambda}_A + \lambda \mathbf{I}_A)\boldsymbol{U}_A^\top)^{-1} \otimes (\boldsymbol{U}_B(\boldsymbol{\Lambda}_B + \lambda \mathbf{I}_B)\boldsymbol{U}_B^\top)^{-1} \tag{42}$$

$$= (\boldsymbol{U}_A \otimes \boldsymbol{U}_B)(\boldsymbol{\Lambda}_A \otimes \boldsymbol{\Lambda}_B + \lambda^2 \mathbf{I})(\boldsymbol{U}_A \otimes \boldsymbol{U}_B)^\top + \lambda \mathbf{I}_A \otimes \boldsymbol{\Lambda}_B + \boldsymbol{\Lambda}_A \otimes \lambda \mathbf{I}_B, \tag{43}$$

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

Recall for an efficient implementation for computing $\sum_{1\leq i,j\leq \mathrm{D_{in}}} \mathbb{C}\text{ov}[a_i W_{ki}, a_j W_{lj}]$ in Eq. (30), 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:

$$\mathbb{C}\text{ov}\left[\boldsymbol{W}[k,:], \boldsymbol{W}[l,:]\right] = \begin{bmatrix} \mathbb{C}\text{ov}[W_{k1}, W_{l1}] & \dots & \mathbb{C}\text{ov}[W_{k1}, W_{l\,\mathrm{D_{in}}}] \\ \vdots & \vdots & \vdots \\ \mathbb{C}\text{ov}[W_{k\,\mathrm{D_{in}}}, W_{l1}] & \dots & \mathbb{C}\text{ov}[W_{k\,\mathrm{D_{in}}}, W_{l\,\mathrm{D_{in}}}] \end{bmatrix} \tag{44}$$

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 $\mathbb{C}\mathrm{ov}[\boldsymbol{W}[k,:], \boldsymbol{W}[l,:]]$.

B.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 $\mathbb{E}[\boldsymbol{h}]$:

$$\begin{aligned} \boldsymbol{a} &= g(\boldsymbol{h}) && (45)\\ &\approx g(\mathbb{E}[\boldsymbol{h}]) + \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}(\boldsymbol{h} - \mathbb{E}[\boldsymbol{h}]). && (46) \end{aligned}$$

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

$$\begin{aligned} \boldsymbol{h} &\sim \mathcal{N}(\mathbb{E}[\boldsymbol{h}], \boldsymbol{\Sigma}_h) && (47)\\ \boldsymbol{h} - \mathbb{E}[\boldsymbol{h}] &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_h) && (48)\\ \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}(\boldsymbol{h} - \mathbb{E}[\boldsymbol{h}]) &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}{}^\top \boldsymbol{\Sigma}_h \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}) && (49)\\ g(\mathbb{E}[\boldsymbol{h}]) + \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}(\boldsymbol{h} - \mathbb{E}[\boldsymbol{h}]) &\sim \mathcal{N}(g(\mathbb{E}[\boldsymbol{h}]), \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}{}^\top \boldsymbol{\Sigma}_h \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}) && (50)\\ \boldsymbol{a} &\underset{\text{approx}}{\sim} \mathcal{N}(\boldsymbol{a}; g(\mathbb{E}[\boldsymbol{h}]), \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}{}^\top \boldsymbol{\Sigma}_h \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}). && (51) \end{aligned}$$

B.5 Probit Approximation for Classification

Following [4], 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 [14, 25]

$$\begin{aligned} p(y^* \mid x^*) &= \int_{\mathbb{R}} \mathrm{sigmoid}(f^*)\, \mathcal{N}\left(f^* \mid \mu^*, {\sigma^*}^2\right) \mathrm{d}f^* && (52)\\ &\approx \int \Phi(f^*)\, \mathcal{N}\left(f^* \mid \mu^*, {\sigma^*}^2\right) \mathrm{d}f^* && (53)\\ &= \sigma\left(\frac{\mu^*}{\sqrt{1 + \frac{\pi}{8}{\sigma^*}^2}}\right). && (54) \end{aligned}$$

Multi-class [7]

$$\begin{aligned} p(\mathbf{y}^* \mid \mathbf{x}^*) &= \int_{\mathbb{R}^C} \mathrm{softmax}(\mathbf{f}^*)\, \mathcal{N}(\mathbf{f}^* \mid \mu^*, \boldsymbol{\Sigma}^*)\, \mathrm{d}\mathbf{f}^*\\ &\overset{\text{j-th element}}{\approx} \frac{\exp(\tau_i)}{\sum_{j=1}^C \exp(\tau_j)}, \text{ where } \tau_j = \frac{\mu_j^*}{\sqrt{1 + \frac{\pi}{8}\Sigma_{jj}^*}} \end{aligned} \tag{55}$$

B.6 Transformer Block

There are four components in each transformer block [26]: (1) multi-head attention; (2) MLP; (3) layer normalisation; and (4) residual connection. For MLP blocks, the propagation is the same as described above. For layer normalisation, as Gaussian distribution is closed under linear transformation, push distribution over it is straightforward. For residual connection, we assume the input and output are independent. We describe how to push distribution through attention layers below. Note for computational reasons, we always assume the input has diagonal covariance.

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

$$\boldsymbol{Q} = \boldsymbol{H}\boldsymbol{W}_Q, \quad \boldsymbol{K} = \boldsymbol{H}\boldsymbol{W}_K, \quad \boldsymbol{V} = \boldsymbol{H}\boldsymbol{W}_V, \tag{56}$$

and the output of attention block is

$$\text{Attention}(\boldsymbol{H}) = \text{Softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}})\boldsymbol{V}. \tag{57}$$

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

$$\boldsymbol{Q} = \mathbb{E}\left[\boldsymbol{H}\right]\mathbb{E}\left[\boldsymbol{W}_Q\right], \quad \boldsymbol{K} = \mathbb{E}\left[\boldsymbol{H}\right]\mathbb{E}\left[\boldsymbol{W}_K\right]. \tag{58}$$

For $\boldsymbol{V}$, for simplicity we describe our approximation for a single token $\boldsymbol{h}$ whose value is $\boldsymbol{v} = \boldsymbol{W}_V\boldsymbol{h}$ with k^{th} element being $v_k = \sum_{i=1}^{D} W_{V_{ki}} h_i$. Assuming $\boldsymbol{h}$ is a Gaussian, the covariance between the k^{th} and the l^{th} value is

$$\mathbb{C}\text{ov}\left[v_k, v_l\right] = \mathbb{C}\text{ov}\left[\sum_{i=1}^{D} W_{V_{ki}} h_i, \sum_{j=1}^{D} W_{V_{lj}} h_j\right] \tag{59}$$

$$= \sum_{i=1}^{D}\sum_{j=1}^{D} \mathbb{C}\text{ov}\left[W_{V_{ki}} h_i, W_{V_{lj}} h_j\right]. \tag{60}$$

We have

$$\begin{aligned}
\mathbb{C}\text{ov}\left[v_k, v_l\right] &= \sum_{i=1}^{D}\sum_{j=1}^{D} \mathbb{C}\text{ov}\left[W_{V_{ki}} h_i, W_{V_{lj}} h_j\right] && \text{(definition)} \\
&= \sum_{i=1}^{D}\sum_{j=1}^{D} W_{V_{ki}} W_{V_{lj}}\, \mathbb{C}\text{ov}\left[h_i, h_j\right] && (\boldsymbol{W}_V \text{ deterministic}) \\
&\approx \sum_{i=1}^{D} W_{V_{ki}} W_{V_{li}}\, \mathbb{V}\text{ar}\left[h_i\right]. && \text{(ignore correlation between } \boldsymbol{h} \text{ for computational reason)}
\end{aligned}$$

$$\begin{aligned}
\mathbb{V}\text{ar}\left[v_k\right] &= \sum_{1\le i,j\le D} \mathbb{C}\text{ov}\left[W_{V_{ki}} h_i, W_{V_{kj}} h_j\right] && \text{(definition)} \\
&\approx \sum_{1\le i,j\le D} \left(\mathbb{E}\left[h_i\right]\mathbb{E}\left[h_j\right] + \mathbb{C}\text{ov}\left[h_i, h_j\right]\right)\mathbb{C}\text{ov}\left[W_{ki}, W_{kj}\right] + \mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[W_{kj}\right]\mathbb{C}\text{ov}\left[h_i, h_j\right] \\
& && \text{(assumption A2)} \\
&= \sum_{1\le i\le D} \left(\mathbb{E}\left[h_i\right]^2 + \mathbb{V}\text{ar}\left[h_i\right]\right)\mathbb{V}\text{ar}\left[W_{ki}\right] + \mathbb{E}\left[W_{ki}\right]^2 \mathbb{V}\text{ar}\left[h_i\right]. \\
& && (\boldsymbol{W}_V \text{ is isotropic Gaussian})
\end{aligned} \tag{61}$$

Once we have the distribution over $\boldsymbol{V}$, the distribution over Attention$(\boldsymbol{H})$ becomes a 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. As we assume all input is isotropic, here we only need to compute the variance for each dimension.

C Experiment

C.1 Regression

Table 4 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 5 we report the Root Mean Square

Table 4: 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

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

C.2 Classification

Table 6 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 7 we report the test accuracy, our method results in matching or better performance compared with sampling and GLM, indicating the effectiveness of our method.

Table 6: 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
CIFAR-100	(50000, 3, 32, 32)	ViT-base

In Fig. 4, we show kernel density plots over the predictive entropy of an FMNIST-trained MLP evaluated on MNIST. Our method can distinguish between in-distribution and OOD data better than the LA MAP and MFVI Sampling. Although our method underfits on the in-distribution data, the separation between is clear for the OOD data similar.

Table 7: 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	MLP	**0.974**±0.002	**0.974**±0.002	0.972±0.002	**0.975**±0.002	**0.975**±0.002
FMNIST	MLP	**0.843**±0.004	**0.842**±0.004	0.868±0.004	**0.882**±0.003	**0.881**±0.003
CIFAR-10	ViT	**0.978**±0.001	**0.978**±0.001	0.971±0.002	**0.974**±0.002	**0.976**±0.002
CIFAR-100	ViT	**0.896**±0.003	**0.895**±0.003	0.855±0.004	0.873±0.003	**0.884**±0.003

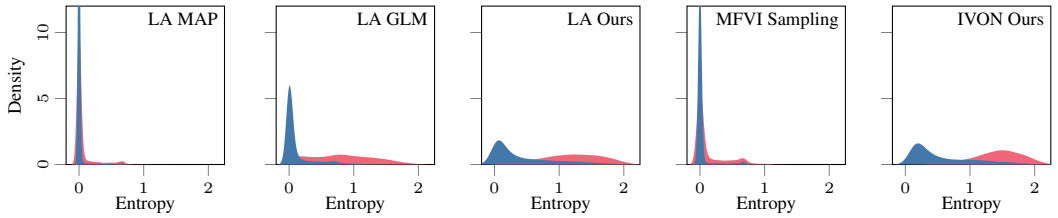


Figure 4: Kernel density plots over the predictive entropy from an MLP trained on FMNIST (blue, in-distribution) and data from MNIST (red, out-of-distribution). Our method results in a clear separation between the in- and out-of-distribution data.