STREAMLINING PREDICTION IN BAYESIAN DEEP LEARNING

Anonymous authors
Paper under double-blind review

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 transformers, such as ViT and GPT-2, and assess its performance on regression and classification tasks.

1 INTRODUCTION

Recent progress and adoption of deep learning models, has led to a sharp increase of interest in improving their reliability and robustness. In applications such as aided medical diagnosis (Begoli et al., 2019), autonomous driving (Michelmore et al., 2020), or supporting scientific discovery (Psaros et al., 2023), providing reliable and robust predictions as well as identifying failure modes is vital. A principled approach to address these challenges is the use of Bayesian deep learning (BDL, Wilson & Izmailov, 2020; Papamarkou et al., 2024) that promises a *plug & play* framework for uncertainty quantification. However, while *plugging* the Bayesian approach into deep learning is relatively straightforward (Blundell et al., 2015; Gal & Ghahramani, 2016; Wu et al., 2019), the *play* part is typically severely hampered by computational and practical challenges (Wenzel et al., 2020; Foong et al., 2020; Gelberg et al., 2024; Coker et al., 2022; Kristiadi et al., 2023).

The key challenges associated with BDL, can roughly be divided into three parts: *(i)* defining a meaningful prior, *(ii)* estimating the posterior distribution, and *(iii)* performing inferences of interest, *e.g.*, making predictions for unseen data, detecting out-of-distribution settings, or analysing model sensitivities. While constructing a meaningful prior is an important research direction (Nalisnick, 2018; Meronen et al., 2021; Fortuin et al., 2021; Tran et al., 2022), it has been argued that the differentiating aspect of Bayesian deep learning is marginalisation (Wilson & Izmailov, 2020; Wilson, 2020) rather than the prior itself. Estimating the posterior distribution has seen significant progress in recent years (Blundell et al., 2015; Maddox et al., 2019; Daxberger et al., 2021a) with a particular FIX (oXsH)

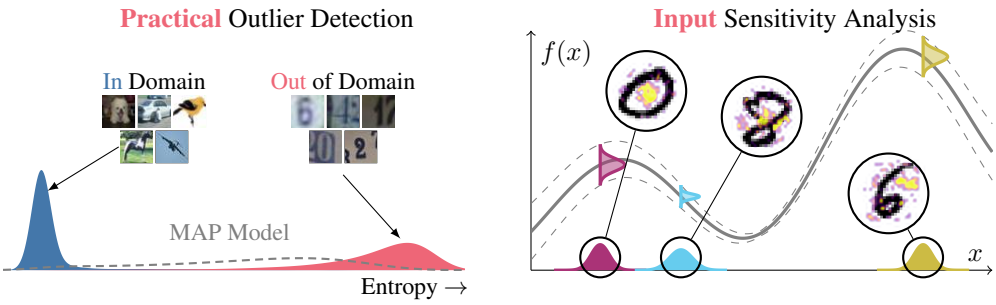


Figure 1: Our streamlined approach allows for *practical* outlier detection and sensitivity analysis. Locally linearizing the network function with local Gaussian approximations enables many relevant inference tasks to be solved analytically, helping render BDL a practical tool for downstream tasks.

focus on post-hoc approximations (Kristiadi et al., 2020; Daxberger et al., 2021b). However, while these approaches have shown promise in making BDL useful for real-world applications, they are tackling only part of the computational and practical challenges associated with using BDL.

In this work, we focus on streamlining BDL for downstream tasks by providing a straightforward and effective method to compute inferences of interest, *cf.*, Fig. 1. For this, we make the neural network locally linear with respect to the inputs. Thus, inferences, such as computing predictions, admit a closed-form solution and can be estimated efficiently. In particular, we propose to use local linearisation of non-linear activation functions at every layer of the network and use local Gaussian approximations at linear layers. Empirically, we find that local linearisation combined with Gaussian approximation of Bayesian neural networks provides accurate predictions, with useful predictive uncertainties, while being conceptually simple. Moreover, complex inference tasks w.r.t. the inputs, such as analysing model sensitivities to input perturbations, can be computed efficiently. Thus, allowing to truly account for all sources of uncertainties.

Contributions: ***(i)*** We propose layer-wise local linearisation and local Gaussian approximations of neural networks to streamline BDL for downstream tasks (Sec. 3). ***(ii)*** We discuss how to handle different covariance structures and architecture choices (Sec. 3.2 & Sec. 3.3). ***(iii)*** Finally, we present an empirical assessment of our approach on regression and classification tasks, and showcase its utility for uncertainty quantification, out-of-domain detection, and sensitivity analysis (Sec. 4).

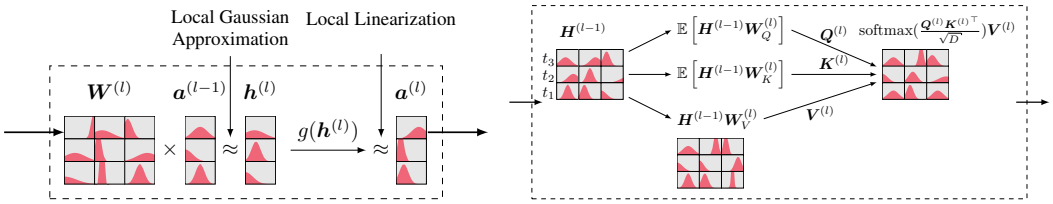
2 Related Work

To estimate the posterior in BDL, variational inference (VI, Blei et al., 2017; Zhang et al., 2018) utilizes a variational approximation to the true posterior distribution and minimizes a divergence measure between both distributions. A typically choice for the variational family is a factorized Gaussian distribution, chosen for computational reasons. Early works on mean-field VI (MFVI) and related approaches, require a modification on the model structure (Blundell et al., 2015) to perform a reparametrization of the variational distribution. Recent work by Shen et al. (2024) developed an optimiser to ease the use of MFVI, and have shown good performance on large-scale models such as ResNets (He et al., 2016) and GPT (Radford et al., 2019). However, VI-based methods typically require Monte Carlo estimation to perform inferences, which can be problematic in practice due to additional computational overhead.

A recent trend in BDL are post-hoc methods, such as the Laplace approximation (LA, MacKay, 1992a), which can be applied directly on the trained model without modification (Kristiadi et al., 2020; Daxberger et al., 2021a). Daxberger et al. (2021b) extended the applicability of LAs by showing that treating a subset of parameters Bayesian can still give good predictive uncertainties. Moreover, Immer et al. (2021b) proposed the linearised LA by performing a global linearisation, which is principled under the Generalised Gauss–Newton approximation to the Hessian, and has shown promise in providing useful predictive uncertainties. Recent works applied LA in various large-scale applications, such as large language models (Yang et al., 2024; Kampen et al., 2024) and dynamic neural networks (Meronen et al., 2024).

In addition, various tailored ensemble-based methods for BDL have been proposed, such as Monte Carlo dropout (Gal & Ghahramani, 2016), deep ensembles (Lakshminarayanan et al., 2017), and stochastic weight averaging-Gaussian (Maddox et al., 2019). While some works on deep ensembles enable estimating the predictive distribution in a single forward pass (Eschenhagen et al., 2021; Havasi et al., 2021), most methods typically require multiple forward passes to estimate the predictive distribution and do not explicate an approximation to the posterior distribution. **FIX** (oXsH)

More recently, there has been a trend on exploring deterministic computations in BDL to avoid the need for sampling (Goulet et al., 2021; Giordano et al., 2024; Burroni et al., 2024). In particular, Wu et al. (2019) derived an analytically training objective for VI by using moment-matching at each layer of the network. However, the solutions to the moment-matching have to be derived manually for each type of activation function, making it impractical in practice. More recently, Goulet et al. (2021) proposed local linearisation of the network to perform message-passing on the network under a mean-field assumption. Moreover, Petersen et al. (2024) used a local linearisation of the network to propagate aleatoric uncertainties over the input through a deterministic network. In addition, Dhawan et al. (2023) investigated local linearisations of activation functions to estimate the function space **FIX** (tEnH)



(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.

distance of two neural networks, for example, relevant in continual learning settings. In contrast, our work disentangles the approximation of the posterior distribution and the computation of inferences w.r.t. the posterior distribution. Hence, providing a streamlined framework to propagate all forms of uncertainties through Bayesian neural networks.

3 Method: Streamlining Bayesian Deep Learning

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 $p(y \mid \boldsymbol{x}, \mathcal{D})$.

Recently, much progress has been made in efficiently approximating the posterior distribution for BDL, including scaling mean-field variational inference (Shen et al., 2024) to large-scale models and performing post-hoc estimation using the Laplace approximations (Daxberger et al., 2021a). A common thread is the use of a tractable distribution q to approximate the posterior distribution $q(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta} \mid \mathcal{D})$, typically chosen to be a Gaussian distribution. Consequently, the posterior predictive distribution is typically approximated using Monte Carlo integration, *i.e.*, by sampling from q, to estimate the integral in Eq. (1), with the exception of the linearised Laplace approximation (Immer et al., 2021b). However, while using a Gaussian approximation facilitates efficient computation of the approximate posterior distribution, sampling from the high-dimensional Gaussian can be challenging (Vono et al., 2022) and result in a high computational overhead.

We will now shift our focus on estimating integrals of the form of Eq. (1) and assume that an approximation to the posterior distribution $q(\boldsymbol{\theta})$ is given. Further, we will assume that q is in the family of stable distributions, for which a linear combination of two independent random variables with this distribution has the same distribution. Gaussian distribution is a typical example of stable distribution. Note that marginalisation tasks such as in Eq. (1) appear in many scenarios, *e.g.*, active learning (MacKay, 1992a; Gal et al., 2017; Smith et al., 2023), model selection (Immer et al., 2021a; MacKay, 1996), or outlier detection (Wilson & Izmailov, 2020), and pose a reappearing challenge in downstream applications of BDL. **FIX** (oXsH)

3.1 Streamlining Computations with Local Approximations

Let the weights and biases of the mth 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 kth 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 (Petersen et al., 2024). However, for hidden layers the distribution over pre-activations is generally not of the same family as the posterior distribution (Wolinski & Arbel, 2022). 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 mth 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 kth and the jth 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

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

A detailed derivation alongside an empirical evaluation of the approximation quality can be found in Apps. A.1 and A.2. Depending on the structure of the covariance matrix, we can further simplify the computation of the covariance matrix, which we will discuss in Sec. 3.3.

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.*, Särkkä & Svensson, 2023), 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. (2024) 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 MacKay (1992b); Kristiadi et al. (2020).

Intuition One way to understand the resulting approximation is as a piecewise linear function (or multilinear function). Globally, the function will still be non-linear, but locally it will behave linear. In contrast to the original model, which composes piecewise linear functions in case of a ReLU network, our approximation composes linear functions locally. And we obtain a piecewise linear function due to the local composition, which allows us to the capture nonlinear nature of the model. **FIX** (tEnH)

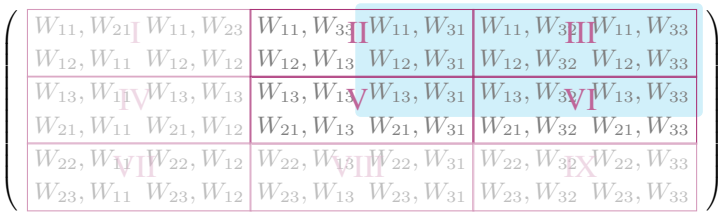


Figure 3: To retrieve the highlighted submatrix $\mathbb{C}\mathrm{ov}[\boldsymbol{W}[1,:],\boldsymbol{W}[2,:]]$ of the covariance for $\boldsymbol{W} \in \mathbb{R}^{2\times 3}$, we identify the Kronecker blocks that contain the covariance of interest (II, III, V, and VI), explicate those blocks in memory, and then retrieve the relevant submatrix.

3.2 Architecture Choices

By combining local Gaussian approximations for linear layers and local linearisation for non-linear activation functions, we can analytically compute the distribution over activations at each layer in a single forward pass. In case of a multi-layer perceptron (MLP) and common architecture choices, the described approach can be directly applied to each layer of the network. However, for more complex architectures such as attention, further considerations are needed to streamline the computation path. Fig. 2 illustrates the computation path for MLPs and attention layers.

Attention layers Each block in a transformer (Vaswani et al., 2017) constitutes: multi-head attention, an MLP, layer normalisation, and a residual connection. For the MLP part, the propagation is the same as previously described. Further, layer normalisation and residual connections are linear transformations and, hence, the resulting distribution can be obtained analytically. Treating the multi-head attention block is more involved as the softmax activation function 'squashes' the distribution on pre-activations. We describe our method below and further details are in App. A.6.

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. Further we denote the key, query and value in an attention blocks as $\boldsymbol{Q} = \boldsymbol{H}\boldsymbol{W}_Q$, $\boldsymbol{K} = \boldsymbol{H}\boldsymbol{W}_K$, and $\boldsymbol{V} = \boldsymbol{H}\boldsymbol{W}_V$. Then the output of attention layer is given as follows $\mathrm{Attention}(\boldsymbol{H}) = \mathrm{Softmax}\left(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{D}\right)\boldsymbol{V}$. For computational reasons we will assume the input distribution to the multi-head attention block to have a diagonal covariance structure. As pushing a random vectors over a softmax activation may require further approximations and will not result in an output with distribution close to a Gaussian distribution. Hence, we treat the query and key matrices as deterministically given. A possible remedy is to leverage an approximation to the softmax function such as Lu et al. (2021). **FIX** (8AJz)

Consequently, the attention scores are given as:

$$\mathrm{Attention}(\boldsymbol{H}) = \mathrm{Softmax}\left(\frac{\mathbb{E}\left[\boldsymbol{H}\right]\mathbb{E}\left[\boldsymbol{W}_Q\right]\left(\mathbb{E}\left[\boldsymbol{H}\right]\mathbb{E}\left[\boldsymbol{W}_K\right]\right)^\top}{\sqrt{D}}\right)\boldsymbol{V}, \tag{7}$$

where $\boldsymbol{V}$ follows a stable distributions. Due to linearity, the resulting distribution can again be obtained analytically.

3.3 Covariance Structure

Computing the full covariance of the posterior is usually infeasible due to high computational and memory cost. We describe our methods for two most common covariance approximations and will briefly discuss the computational cost in case of a full and diagonal covariance structure.

Full covariance When the posterior has full covariance, for the mth linear layer the computational complexity for computing $\mathbb{C}\mathrm{ov}[h_k, h_l]$ is $\mathcal{O}([\mathrm{D}_{\text{in}}^{(m)}]^2)$. Consequently, computing the covariance of the activations for the mth layer adds to $\mathcal{O}([\mathrm{D}_{\text{out}}^{(m)}]^2[\mathrm{D}_{\text{in}}^{(m)}]^2)$. Computing the local linearisation for element-wise activation functions results in a complexity of $\mathcal{O}([\mathrm{D}_{\text{out}}^{(l)}]^2)$. Hence, we obtain a total cost of $\mathcal{O}(\sum_{m=1}^{M}[\mathrm{D}_{\text{out}}^{(m)}]^2[\mathrm{D}_{\text{in}}^{(m)}]^2 + [\mathrm{D}_{\text{out}}^{(m)}]^2)$ for a network with M layers. As the computational cost is directly linked to the number of parameters and their correlation structure, a natural way to reduce

the computational cost is to either exploit structure in covariance matrix or consider only a subset of parameters, in the spirit of subnetwork Laplace (Daxberger et al., 2021b). We will focus on the covariance structure in the following as consider a subset of parameters trivially extends from our discussion.

Diagonal approximation In case the correlations between model parameters are ignored, as in mean-field variational inference, the computation of the pre-activation covariance reduces to:

$$\mathbb{C}\mathrm{ov}\left[h_k^{(m)}, h_l^{(m)}\right] = \sum_{1 \leq i,j \leq \mathrm{D}_{\text{in}}} \mathbb{E}\left[W_{ki}^{(m)}\right] \mathbb{E}\left[W_{lj}^{(m)}\right] \mathbb{C}\mathrm{ov}\left[a_i^{(m-1)}, a_j^{(m-1)}\right], \tag{8}$$

and variance of the kth pre-activation is given as: $\mathbb{V}\mathrm{ar}[h_k^{(m)}] =$

$$\sum_{1 \leq i \leq \mathrm{D}_{\text{in}}} \mathbb{E}\left[a_i^{(m)}\right]^2 \mathbb{V}\mathrm{ar}\left[W_{ki}^{(m)}\right] + \mathbb{V}\mathrm{ar}\left[b_k^{(m)}\right] + \mathbb{V}\mathrm{ar}\left[a_i^{(m-1)}\right] \left(\mathbb{E}\left[W_{ki}^{(m)}\right]^2 + \mathbb{V}\mathrm{ar}\left[W_{ki}^{(m)}\right]\right). \tag{9}$$

Hence, assuming a diagonal covariance structure can help in reducing the computational burden. If only the variance of the layer output is of interest, the computational cost can be further reduced and adds to a total of $\mathcal{O}(\sum_{m=1}^{M} \mathrm{D}_{\text{out}}^{(m)} \mathrm{D}_{\text{in}}^{(m)} + \mathrm{D}_{\text{out}}^{(m)})$. Further details are given in App. A.3.

Kronecker-factorisation (KFAC) Another common choice for approximating the posterior covariance is the use of a Kronecker-factorisation (KFAC) (Martens & Grosse, 2015), popularised in the context of Laplace approximations (Ritter et al., 2018). In this case, the posterior covariance $\boldsymbol{\Sigma}$ is given by a Kronecker product of two factors, *i.e.*, $\boldsymbol{\Sigma} = (\boldsymbol{A} \otimes \boldsymbol{B} + \lambda^2 \mathbf{I})^{-1}$ where $\otimes$ denotes the Kronecker product and $\lambda^2 \mathbf{I}$ is a prior precision. Note that in case of a non-zero prior precision, the covariance cannot be expressed in the form of a Kronecker matrix multiplication. Denote the eigenvectors and eigenvalues of $\boldsymbol{A}$ as $\boldsymbol{U}_A$ and $\boldsymbol{\Lambda}_A$, respectively, we approximate the $\boldsymbol{\Sigma}$ as follows:

$$\begin{aligned} \boldsymbol{\Sigma} = (\boldsymbol{A} \otimes \boldsymbol{B} + \lambda^2 \mathbf{I})^{-1} &= \left((\boldsymbol{U}_A \boldsymbol{\Lambda}_A \boldsymbol{U}_A^\top) \otimes (\boldsymbol{U}_B \boldsymbol{\Lambda}_B \boldsymbol{U}_B^\top) + \lambda^2 \mathbf{I}\right)^{-1} \qquad \text{(Eigen Decomposition)} \\ &\approx ((\boldsymbol{U}_A \otimes \boldsymbol{U}_B)(\boldsymbol{\Lambda}_A + \lambda \mathbf{I}_A))^{-1} \otimes \left((\boldsymbol{\Lambda}_B + \lambda \mathbf{I}_B)(\boldsymbol{U}_A \otimes \boldsymbol{U}_B)^\top\right)^{-1}. \end{aligned} \tag{10}$$

To compute the covariance of the pre-activations, we need to retrieve the covariance between the weights of the kth unit and the weights of the lth unit, which corresponds to the kth and l^{th} row in $\boldsymbol{W}$, *i.e.*, $\mathbb{C}\mathrm{ov}[\boldsymbol{W}[k,:], \boldsymbol{W}[l,:]]$. In case of KFAC Laplace approximations, accessing $\mathbb{C}\mathrm{ov}[\boldsymbol{W}[k,:], \boldsymbol{W}[l,:]]$ cannot be done directly. Therefore, we developed a block retrieval method to retrieve $\mathbb{C}\mathrm{ov}[\boldsymbol{W}[k,:], \boldsymbol{W}[l,:]]$ without explicating the full covariance matrix in memory.

The key idea is to first identify the Kronecker blocks that contain the covariance of interest and then retrieve the submatrix by reconstructing only the relevant blocks. Fig. 3 illustrates the idea in a toy example for a weight matrix $\boldsymbol{W} \in \mathbb{R}^{2\times 3}$ and a submatrix of interest $\mathbb{C}\mathrm{ov}[\boldsymbol{W}[1,:], \boldsymbol{W}[2,:]]$. Our method only reconstructs blocks contain the sub-covariance of interest (II, III, V, and VI) and then retrieves the relevant submatrix. Further details are given in App. A.4.

4 Experiments

We demonstrate *practical applicability* of our approach on classification/regression tasks (Sec. 4.1), large-scale classification results with ViT/GPT models (Sec. 4.2), and sensitivity estimation (Sec. 4.3).

Data sets We use a selection of data sets from UCI repository (Kelly et al., 2023) for the regression experiments. For classification, we experiment on MNIST (LeCun et al., 1998), FMNIST (Xiao et al., 2017), as well as the 11-class data sets OrganCMNIST and OrganSMNIST from MedMNIST (Yang et al., 2023). To assess our method on higher-dimensional settings, we experiment with CIFAR-10 and CIFAR-100 (Krizhevsky & Hinton, 2009), DTD (Cimpoi et al., 2014), RESISC (Cheng et al., 2017) and a subsampled version of ImageNet-R (Hendrycks et al., 2021) with 100 classes to reduce the memory overhead for LA. For the GPT model we used the BOOLQ, WIC, and MRPC tasks from GLUE (Wang et al., 2019b) and SuperGLUE (Wang et al., 2019a) benchmarks.

Posterior approximations We adopt the Laplace approximation (LA, MacKay (1996)) and mean-field variational inference (MFVI, Blei et al., 2017) for approximating the posterior distribution of the network parameters. For LA, we estimate the full covariance for the regression experiments, while we use diagonal or KFAC approximations for the covariance where applicable in the classification

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)	**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

experiments. We compare our method using local Gaussian approximation and local linearisation against Monte Carlo (MC) sampling and a global linearised model (GLM, Immer et al., 2021b). For MFVI, we adopt the IVON optimiser (Shen et al., 2024) to obtain the posterior approximation with a diagonal covariance structure by default, 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. (2024). For the MFVI and LA sampling baselines, we used $1,000$ MC samples in the regression and classification experiments in Sec. 4.1, and 50 MC samples for the ViT and GPT-2 in Sec. 4.2. For our method, we fit an additional scaling factor on the predictive variance by minimising the NLPD on the validation set, similar to the pseudo-count used in Ritter et al. (2018). **FIX** (KRRC) **FIX** (8AJz)

Network architectures We experiment with one or two layer multi-layer perceptron (MLP) on the UCI regression data sets with details given in App. B.1. For MNIST, FMNIST, OrganCMNIST and OrganSMNIST, we use an MLP with layers containing $784 - 128 - 64 - C$ neurons, where C is the number of classes. For CIFAR-10/100, DTD, RESISC and ImageNet-R, we fine-tune a Vision Transformer (ViT) (Dosovitskiy et al., 2021) base model pre-trained on ImageNet-1k (Deng et al., 2009). For the GPT model, we use the pre-trained GPT-2 base model from Hugging Face Transformers (Wolf et al., 2019) and fine-tune it on the respective tasks.

Evaluation metrics For the regression experiments, we measure the negative log predictive density (NLPD) and root-mean-square error (RMSE) for each method. In the classification experiments, we use accuracy (ACC), NLPD, and expected calibration error (ECE) for comparing the methods. We use a paired t-test with $p = 0.05$ to bold results with significant statistical difference when reporting the results. For assessing out-of-distribution (OOD) robustness, we use a Gaussian kernel density estimator with variance 0.25 on the histogram of the predictive entropy evaluated on the test set.

4.1 Does our Method Provide Useful Uncertainty Estimates?

Regression We experiment on a selection of data sets from the UCI repository and run a 5-fold cross validation to report results for each data set. We use either MFVI and LA to obtain the posterior approximation and separately compare our method against their corresponding prediction approaches. Table 1 shows that our method achieves better NLPD in general than the predictions with sampling for both MFI and LA. Moreover, our method performs on par with the GLM, even though our method results in a locally linearised network w.r.t. the inputs. Similar conclusions are made inspecting Table 8.

Classification Here, we assess our method on MNIST-like classification tasks. For the LA, we use KFAC approximation of the covariance to reduce the memory overhead. In Table 2, we report the ACC and NLPD with their standard errors and the ECE for each method. Our method achieves similar ACC with the baselines, while outperforming them on the NLPD and ECE metrics. In App. B.2, we assess our method on robustness to OOD data. We evaluate an MLP trained on MNIST on rotated versions of the test set. Our method consistently reduces overconfidence on OOD data, *cf.*, Fig. 8.

Ours vs. moment-matching To verify the viability of local linearisation, we compare our method against moment-matching (MM) used in Wu et al. (2019). We apply MM instead of local linearisation

Table 2: Performance metrics on the MNIST-like data sets for each method with the standard error for ACC and NLPD. Our method achieves better NLPD and ECE than the baselines.

Metrics	Methods	MNIST	FMNIST	OrganCMNIST	OrganSMNIST
ACC ↑	LA Sampling	0.972±0.002	0.868±0.004	0.734±0.008	0.591±0.010
	LA GLM	**0.975**±0.002	**0.882**±0.003	**0.824**±0.007	**0.634**±0.009
	LA Ours	**0.975**±0.002	**0.881**±0.003	**0.826**±0.008	**0.630**±0.009
	MFVI Sampling	**0.974**±0.002	**0.843**±0.004	0.620±0.010	**0.467**±0.010
	MFVI Ours	**0.974**±0.002	**0.842**±0.004	**0.630**±0.010	0.454±0.010
NLPD ↓	LA Sampling	0.210±0.003	0.556±0.008	1.135±0.017	1.614±0.021
	LA GLM	**0.089**±0.004	0.548±0.018	0.875±0.44	1.967±0.070
	LA Ours	**0.089**±0.005	**0.397**±0.010	**0.710**±0.021	**1.365**±0.025
	MFVI Sampling	0.179±0.014	2.010±0.051	4.775±0.140	6.095±0.141
	MFVI Ours	**0.086**±0.005	**0.529**±0.011	**1.492**±0.026	**1.895**±0.022
ECE ↓	LA Sampling	0.122	0.151	0.292	0.261
	LA GLM	0.009	0.078	**0.038**	0.170
	LA Ours	**0.004**	**0.012**	0.122	**0.151**
	MFVI Sampling	0.018	0.144	0.251	0.385
	MFVI Ours	**0.003**	**0.013**	**0.145**	**0.067**

Table 3: Performance comparison between our method and Moment-Matching (MM) on the MNIST-like classification tasks. We report the ACC and NLPD with standard errors and the ECE. Our method outperforms MM despite being simpler and more applicable to various distributions.

Metrics	Methods	MNIST	FMNIST	OrganCMNIST	OrganSMNIST
ACC ↑	MM	0.971±0.002	**0.882**±0.003	0.771±0.004	0.600±0.010
	Ours	**0.975**±0.002	**0.881**±0.003	**0.816**±0.004	**0.614**±0.010
NLPD ↓	MM	0.103±0.005	0.463±0.014	0.838±0.015	1.401±0.036
	Ours	**0.090**±0.005	**0.435**±0.010	**0.733**±0.010	**1.282**±0.024
ECE ↓	MM	0.014	0.051	**0.023**	0.110
	Ours	**0.006**	**0.022**	0.127	**0.081**

to our setting, assuming a diagonal posterior approximation from LA. In Table 3, we show the results for our method against MM on MNIST-like classification tasks. Our approach outperforms MM across the data sets and the metrics, except for the ECE on OrganCMNIST. Compared to MM, our method is applicable to any differentiable activation function and any type of stable distribution (Petersen et al., 2024), while MM requires tailored derivations for each case and, hence, is less *plug-and-play*.

4.2 Is our Method Scalable?

We demonstrate that our method is applicable to large-scale networks by experimenting with pre-trained ViT and GPT-2 models. In particular, we experiment with applying our method on either the attention layer or the MLP after the attention layer in the last two (ViT)/four (GPT) transformer blocks. For each target data set, we fine-tune the layers we obtain the posterior approximation for. We assume a diagonal posterior, to reduce the memory overhead. Table 5 shows the results when fine-tuning ViT models and obtaining the posterior approximation from the attention layers. We observe that our method achieves better or on par NLPD and ECE compared to the baselines for both LA and MFVI across all data sets while maintaining similar ACC as the baselines.

In Table 4 we show the results for a GPT-2 model with LA on the MLP layers. We observe that our method systematically outperforms sampling, while achieving similar performance to GLM in some cases. Thus indicating that our method is applicable to different application domains. We present additional results for ViT models in App. B.

Table 4: Performance on language understanding tasks.

Metrics	Methods	BoolQ	WIC	MRPC
ACC ↑	LA Sampling	0.383±0.009	0.500±0.020	0.335±0.011
	LA GLM	**0.698**±0.008	**0.611**±0.019	**0.710**±0.011
	LA Ours	0.641±0.008	0.500±0.020	0.665±0.011
NLPD ↓	LA Sampling	0.814±0.006	0.858±0.024	1.219±0.018
	LA GLM	**0.650**±0.014	**0.730**±0.027	**0.660**±0.023
	LA Ours	0.684±0.001	**0.719**±0.010	**0.629**±0.009
ECE ↓	LA Sampling	0.259	0.261	0.484
	LA GLM	**0.086**	0.126	0.145
	LA Ours	0.129	**0.118**	**0.054**

We also assess the robustness to out-of-distribution (OOD) data for our method and the baselines. In particular, we take the ViT network fine-tuned on CIFAR-10 and evaluate its predictive entropy on the SVHN data set (Netzer et al., 2011). Fig. 4 shows the kernel density of the predictive entropy computed on the test sets of CIFAR-10 and SVHN, where the model should have high entropy for data

Table 5: Performance metrics using ViT with posterior approximation on the attention layers with the standard error for ACC and NLPD. Our method achieves better NLPD and ECE in general and achives similar ACC compared to the baselines.

Metrics	Methods	CIFAR-10	CIFAR-100	DTD	RESISC	IMAGENET-R
ACC ↑	LA Sampling	0.971±0.002	**0.882**±0.003	**0.715**±0.010	**0.892**±0.004	0.731±0.012
	LA GLM	**0.976**±0.002	0.879±0.003	**0.718**±0.010	**0.891**±0.004	**0.739**±0.012
	LA Ours	**0.976**±0.002	**0.880**±0.003	**0.719**±0.010	**0.892**±0.004	**0.739**±0.012
	MFVI Sampling	**0.975**±0.002	**0.880**±0.003	**0.732**±0.010	**0.867**±0.004	**0.730**±0.012
	MFVI Ours	**0.975**±0.002	**0.880**±0.003	**0.734**±0.010	**0.867**±0.004	**0.728**±0.012
NLPD ↓	LA Sampling	0.170±0.004	**0.444**±0.012	1.238±0.028	0.461±0.009	1.208±0.048
	LA GLM	0.092±0.007	0.459±0.012	1.197±0.029	0.385±0.010	**1.180**±0.047
	LA Ours	**0.086**±0.006	**0.456**±0.012	**1.068**±0.035	**0.352**±0.012	1.267±0.043
	MFVI Sampling	0.133±0.011	0.641±0.022	**1.091**±0.048	1.010±0.041	1.577±0.083
	MFVI Ours	**0.088**±0.006	**0.468**±0.013	**1.007**±0.035	**0.617**±0.019	**1.234**±0.052
ECE ↓	LA Sampling	**0.006**	**0.022**	0.197	0.129	0.070
	LA GLM	0.011	0.024	0.155	0.053	**0.057**
	LA Ours	0.008	0.027	**0.040**	**0.016**	0.132
	MFVI Sampling	0.015	0.070	0.075	0.079	0.118
	MFVI Ours	**0.008**	**0.025**	**0.042**	**0.017**	**0.036**

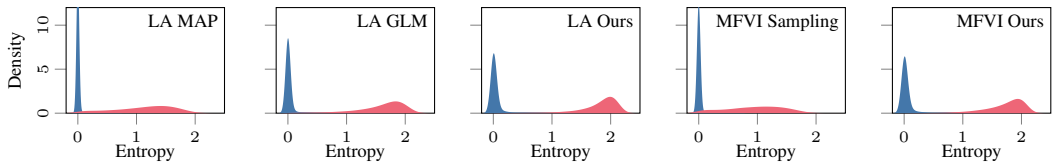


Figure 4: Kernel density plots over the predictive entropy from a ViT network finetuned on CIFAR-10 (blue, in-distribution) and data from SVHN (red, out-of-distribution). Our method results in a clear separation between the in- and out-of-distribution data.

different from the training data. Although our method is slightly underconfident on the in-distribution data, the entropy for in-distribution and OOD data is clearly separated, especially for MFVI.

4.3 Can our Method Estimate Input Sensitivies?

We demonstrate that our method can estimate sensitivities w.r.t. the inputs to the network. For this, we use an 3-class MLP trained on the digits $0/6/8$. Our goal is to estimate sensitivity maps by assuming that the input images $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}, \boldsymbol{\Sigma})$ are distributed according to a Gaussian centred at the pixel values with diagonal covariance $\boldsymbol{\Sigma}$. We optimise the input covariance of each image by minimising the loss

$$\ell = \sum_{n=1}^{N} \text{cross-entropy}(f(\boldsymbol{x}_n), y_n) - \mathcal{H}(\mathcal{N}(\boldsymbol{x}_n, \Sigma_n)). \tag{11}$$

In words, we jointly minimise the cross-entropy loss, after analytically propagating the input distribution through the network, while maximising the entropy $\mathcal{H}(\mathcal{N}(\boldsymbol{x}_n, \Sigma_n))$ of the input distribution. The optimisation is stopped once the difference in NLPD between the current iteration and initial condition is more than 0.1. Fig. 5 shows examples of the resulting sensitivity maps for a deterministic MLP (MAP) and the same MLP with last-layer LA (Bayes). We observe that the largest sensitivity for the digits 0 and 8 are generally in the middle, while for 6 in the upper right corner. The Bayes model shows less spurious sensitivities across the pixels compared to the MAP model. Thus, indicating that incorporating all sources of uncertainties can lead to a more interpretable sensitivity analysis.

5 Discussion & Conclusion

In this work, we proposed to streamline Bayesian deep learning through local linearisation and local Gaussian approximations of the network. For this, we discussed the propagation in different neural network architectures and covariance structures. In particular, we discussed how to handle Kronecker-factorised posterior covariances and transformer architectures. We showed through a series of experiments that our method obtains high predictive performance, provides useful predictive uncertainties, and can be used for sensitivity analysis. Our method helps in making BDL more useful in practice and expands the use-cases and sources of uncertainties that can be considered.

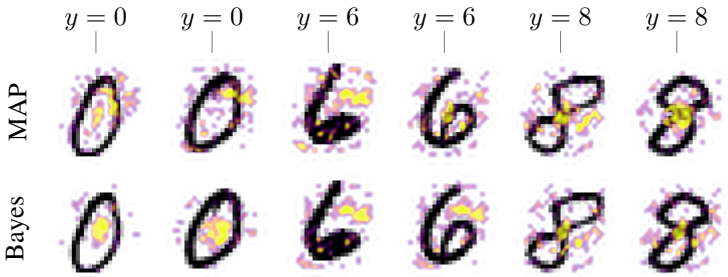


Figure 5: Pixel sensitivity maps of an MLP trained on a subset of MNIST digits (classes $0/6/8$). The two rows show sensitivities to pixel perturbations for the MLP (MAP) and MLP with last-layer Laplace approximation (Bayes). The sensitivities are visualised in range (0.5 1.0). The Bayes MLP shows less spurious sensitivities across the pixels compared to MAP.

In future work, we aim to apply our approach to tasks with larger number of output classes, explore additional use-case scenarios in which our streamlined approach can be beneficial, and scale to even larger networks. Moreover, we aim to further investigate computational benefits obtained by exploiting the posterior covariance structure and sparsity in the network.

FIX (8AJz) & (KRRC)

Limitations The local linearisation of activation functions induces an error that depends on both the activation function as well as the location and scale of the distribution over the input to the activation function. Moreover, we assume independence between the activations and model parameters for the local Gaussian approximation in linear layers and residual connections, which may incur a loss of information in the propagation. Especially, the independence assumption in residual block is potentially harmful, and relaxing it would be a valuable future direction. Further, it would be interesting to estimate the induced approximation error to identify potential failure modes. Finally, we assume access to a validation set for fitting the scaling factor of the predictive posterior distribution, which is currently done using a grid search. Interesting future step is the use of the marginal likelihood.

References

Edmon Begoli, Tanmoy Bhattacharya, and Dimitri Kusnezov. The need for uncertainty quantification in machine-assisted medical decision making. *Nature Machine Intelligence*, 1(1):20–23, 2019. 1

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017. 2, 6

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *Proceedings of the 32th International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1613–1622. PMLR, 2015. 1, 2

Javier Burroni, Justin Domke, and Daniel Sheldon. Sample average approximation for black-box variational inference. In *Proceedings of the 40th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2024. 2

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017. 6

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3606–3613. IEEE Computer Society, 2014. 6

Beau Coker, Wessel P Bruinsma, David R Burt, Weiwei Pan, and Finale Doshi-Velez. Wide mean-field bayesian neural networks ignore the data. In *Proceedings of the twenty fifth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 131 of *Proceedings of Machine Learning Research*, pp. 5276–5333. PMLR, 2022. 1

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux - effortless bayesian deep learning. In *Advances in Neural Information Processing Systems (NeurIPS) 34*, volume 34, pp. 20089–20103. Curran Associates, Inc., 2021a. 1, 2, 3, 30

Erik Daxberger, Eric Nalisnick, James U Allingham, Javier Antorán, and José Miguel Hernández-Lobato. Bayesian deep learning via subnetwork inference. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2510–2521. PMLR, 2021b. 2, 6

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255. IEEE Computer Society, 2009. 7

Nikita Dhawan, Sicong Huang, Juhan Bae, and Roger Baker Grosse. Efficient parametric approximations of neural network function space distance. In *Proceedings of the 40th International Conference on Machine Learning ICML*, Proceedings of Machine Learning Research, pp. 7795–7812. PMLR, 2023. 2

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. 7

Runa Eschenhagen, Erik Daxberger, Philipp Hennig, and Agustinus Kristiadi. Mixtures of laplace approximations for improved post-hoc uncertainty in deep learning. In *NeurIPS workshop on Bayesian deep learning*, 2021. 2

Andrew Foong, David Burt, Yingzhen Li, and Richard Turner. On the expressiveness of approximate inference in bayesian neural networks. In *Advances in Neural Information Processing Systems (NeurIPS) 33*, volume 33, pp. 15897–15908. Curran Associates, Inc., 2020. 1

Vincent Fortuin, Adrià Garriga-Alonso, Sebastian W Ober, Florian Wenzel, Gunnar Rätsch, Richard E Turner, Mark van der Wilk, and Laurence Aitchison. Bayesian neural network priors revisited. In *International Conference on Learning Representations (ICLR)*, 2021. 1

Yarin Gal and Zoubin 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*, pp. 1050–1059. PMLR, 2016. 1, 2

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, pp. 1183–1192. PMLR, 2017. 3

Yoav Gelberg, Tycho F. A. van der Ouderaa, Mark van der Wilk, and Yarin Gal. Variational inference failures under model symmetries: Permutation invariant posteriors for bayesian neural networks. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*, 2024. 1

Ryan Giordano, Martin Ingram, and Tamara Broderick. Black box variational inference with a deterministic objective: Faster, more accurate, and even more black box. *Journal of Machine Learning Research*, 25(18):1–39, 2024. 2

James-A Goulet, Luong Ha Nguyen, and Saeid Amiri. Tractable approximate gaussian inference for bayesian neural networks. *Journal of Machine Learning Research*, 22(251):1–23, 2021. 2

Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew Mingbo Dai, and Dustin Tran. Training independent subnetworks for robust prediction. In *International Conference on Learning Representations (ICLR)*, 2021. 2

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. IEEE Computer Society, 2016. 2

Dan Hendrycks, Kevin Zhao, Sebastian Basart, Jacob Steinhardt, and Dawn Song. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8340–8349. IEEE, 2021. 6

Alexander Immer, Matthias Bauer, Vincent Fortuin, Gunnar Rätsch, and Khan Mohammad Emtiyaz. Scalable marginal likelihood estimation for model selection in deep learning. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, pp. 4563–4573. PMLR, 2021a. 3

Alexander Immer, Maciej Korzepa, and Matthias 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*, pp. 703–711. PMLR, 2021b. 2, 3, 7

Peter JT Kampen, Gustav RS Als, and Michael Riis 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

Raymond Kan. From moments of sum to moments of product. *Journal of Multivariate Analysis*, 99 (3):542–554, 2008. 17

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The UCI machine learning repository, 2023. URL: https://archive.ics.uci.edu. 6

Agustinus Kristiadi, Matthias Hein, and Philipp 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*, pp. 5436–5446. PMLR, 2020. 2, 4

Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, and Vincent Fortuin. Promises and pitfalls of the linearized laplace in bayesian optimization. In *Fifth Symposium on Advances in Approximate Bayesian Inference*, 2023. 1

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Toronto, ON, Canada, 2009. 6

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS) 30*, volume 30, pp. 6402–6413. Curran Associates, Inc., 2017. 2

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 6

Jiachen Lu, Jinghan Yao, Junge Zhang, Xiatian Zhu, Hang Xu, Weiguo Gao, Chunjing Xu, Tao Xiang, and Li Zhang. Soft: Softmax-free transformer with linear complexity. In *Advances in Neural Information Processing Systems NeurIPS 34*, pp. 21297–21309. Curran Associates, Inc., 2021. 5

David JC MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992a. 2, 3

David JC MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992b. 4

David JC MacKay. Bayesian methods for backpropagation networks. In *Models of neural networks III: association, generalization, and representation*, pp. 211–254. Springer, 1996. 3, 6

Wesley J. Maddox, Pavel Izmailov, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems (NeurIPS) 32*, volume 32, pp. 13132–13143. Curran Associates, Inc., 2019. 1, 2

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, pp. 2408–2417. PMLR, 2015. 6

Lassi Meronen, Martin Trapp, and Arno Solin. Periodic activation functions induce stationarity. In *Advances in Neural Information Processing Systems (NeurIPS) 34*, volume 34, pp. 1673–1685. Curran Associates, Inc., 2021. 1

Lassi Meronen, Martin Trapp, Andrea Pilzer, Le Yang, and Arno Solin. Fixing overconfidence in dynamic neural networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2680–2690, 2024. 2

Rhiannon Michelmore, Matthew Wicker, Luca Laurenti, Luca Cardelli, Yarin Gal, and Marta Kwiatkowska. Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. In *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 7344–7350. IEEE, 2020. 1

Saralees Nadarajah and Tibor K Pogány. On the distribution of the product of correlated normal random variables. *Comptes Rendus. Mathématique*, 354(2):201–204, 2016. 17

Eric Thomas Nalisnick. *On priors for Bayesian neural networks*. University of California, Irvine, 2018. 1

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011. 8

Theodore Papamarkou, Maria Skoularidou, Konstantina Palla, Laurence Aitchison, Julyan Arbel, David B. Dunson, Maurizio Filippone, Vincent Fortuin, Philipp Hennig, José Miguel Hernández-Lobato, Aliaksandr Hubin, Alexander Immer, Theofanis Karaletsos, Mohammad Emtiyaz Khan, Agustinus Kristiadi, Yingzhen Li, Stephan Mandt, Christopher Nemeth, Michael A. Osborne, Tim G. J. Rudner, David Rügamer, Yee Whye Teh, Max Welling, Andrew Gordon Wilson, and Ruqi 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

Felix Petersen, Aashwin Ananda Mishra, Hilde Kuehne, Christian Borgelt, Oliver Deussen, and Mikhail Yurochkin. Uncertainty quantification via stable distribution propagation. In *International Conference on Learning Representations (ICLR)*, 2024. 2, 4, 8

Apostolos F Psaros, Xuhui Meng, Zongren Zou, Ling Guo, and George Em Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics*, 477:111902, 2023. 1

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. 2

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations (ICLR)*, 2018. 6, 7

Simo Särkkä and Lennart Svensson. *Bayesian filtering and smoothing*, volume 17. Cambridge university press, 2023. 4

Yuesong Shen, Nico Daheim, Bai Cong, Peter Nickl, Gian Maria Marconi, Bazan Clement Emile Marcel Raoul, Rio Yokota, Iryna Gurevych, Daniel Cremers, Mohammad Emtiyaz Khan, and Thomas 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, 3, 7, 30

Freddie Bickford Smith, Andreas Kirsch, Sebastian Farquhar, Yarin Gal, Adam Foster, and Tom Rainforth. Prediction-oriented bayesian active learning. In *Proceedings of the twenty sixth International Conference on Artificial Intelligence and Statistics (AISTATS)*, Proceedings of Machine Learning Research, pp. 7331–7348. PMLR, 2023. 3

Ba-Hien Tran, Simone Rossi, Dimitrios Milios, and Maurizio Filippone. All you need is a good functional prior for bayesian deep learning. *Journal of Machine Learning Research*, 23(74):1–56, 2022. 1

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS) 30*, volume 30. Curran Associates, Inc., 2017. 5, 15, 20

Maxime Vono, Nicolas Dobigeon, and Pierre Chainais. High-dimensional gaussian sampling: a review and a unifying approach based on a stochastic proximal point algorithm. *SIAM Review*, 64 (1):3–56, 2022. 3

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems (NeurIPS) 32*, volume 32, pp. 3266–3280. Curran Associates, Inc., 2019a. 6, 24

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations (ICLR)*, 2019b. 6, 24

Florian Wenzel, Kevin Roth, Bastiaan Veeling, Jakub Swiatkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really? In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10248–10259. PMLR, 2020. 1

Andrew Gordon Wilson. The case for bayesian deep learning. *arXiv preprint arXiv:2001.10995*, 2020. 1

Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *Advances in Neural Information Processing Systems (NeurIPS) 33*, volume 33, pp. 4697–4708. Curran Associates, Inc., 2020. 1, 3

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019. 7

Pierre Wolinski and Julyan Arbel. Gaussian pre-activations in neural networks: Myth or reality? *arXiv preprint arXiv:2205.12379*, 2022. 4

Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E. Turner, José Miguel Hernández-Lobato, and Alexander L. Gaunt. Deterministic variational inference for robust bayesian neural networks. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 2, 7

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 6

Adam X Yang, Maxime Robeyns, Xi Wang, and Laurence Aitchison. Bayesian low-rank adaptation for large language models. In *International Conference on Learning Representations (ICLR)*, 2024. 2

Jiancheng Yang, Rui Shi, Donglai Wei, Zeju Liu, Lin Zhao, Bilian Ke, Ziyang Shi, Yunzhu Li, Xiaoyang Hu, Yang Gao, Ye Xu, Daniel L. Rubin, and Holger R. Roth. Medmnist v2: A large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):1–14, 2023. doi: 10.1038/s41597-023-02552-1. 6

Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018. 2

APPENDICES

The appendices are structured as follows: App. A presents the derivations of our method in detail. App. B describes the experimental setup and additional experimental results.

A DERIVATIONS

We derive how to propagate the distribution in a deterministic way in this section. See Table 6 for the list of notation that will be used throughout this section.

We first derive the general result in App. A.1 where the posterior covariance has full structure in linear layer and evaluate the quality of local Gaussian approximation in App. A.2. Next, in App. A.3 and App. A.4 we give the derivation for diagonal and KFAC covariance respectively. Then, App. A.5 shows the derivation for activation functions. Finally, App. A.6 describes how we apply our method in a transformer network (Vaswani et al., 2017).

Table 6: 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

A.1 DERIVATION FOR FULL COVARIANCE STRUCTURE

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

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 mth 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

$$\mathbb{E}\left[h_k\right] = \mathbb{E}\left[\sum_{i=1}^{\mathrm{D_{in}}} W_{ki}a_i + b_k\right] \tag{12}$$

$$= \sum_{i=1}^{\mathrm{D_{in}}} \mathbb{E}\left[W_{ki}a_i + b_k\right] \tag{13}$$

$$= \sum_{i=1}^{\mathrm{D_{in}}} \mathbb{E}\left[W_{ki}a_i\right] + \mathbb{E}\left[b_k\right] \tag{14}$$

$$\approx \sum_{i=1}^{\mathrm{D_{in}}} \mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[a_i\right] + \mathbb{E}\left[b_k\right]. \qquad \text{(Assumption 2)}$$

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

$$\mathbb{C}\mathrm{ov}\left[h_k, h_l\right] = \mathbb{C}\mathrm{ov}\left[\sum_{i=1}^{\mathrm{D_{in}}} a_iW_{ki} + b_k, \sum_{i=1}^{\mathrm{D_{in}}} a_iW_{li} + b_l\right] \tag{15}$$

$$= \mathbb{C}\mathrm{ov}\left[\sum_{i=1}^{\mathrm{D_{in}}} a_iW_{ki}, \sum_{i=1}^{\mathrm{D_{in}}} a_iW_{li}\right] + \mathbb{C}\mathrm{ov}\left[\sum_{i=1}^{\mathrm{D_{in}}} a_iW_{ki}, b_l\right] + \mathbb{C}\mathrm{ov}\left[\sum_{i=1}^{\mathrm{D_{in}}} a_iW_{li}, b_k\right] + \mathbb{C}\mathrm{ov}\left[b_k, b_l\right] \tag{16}$$

$$= \sum_{1\le i,j\le D_{\text{in}}} \mathbb{C}\mathrm{ov}\left[a_iW_{ki}, a_jW_{lj}\right] + \sum_{1\le i\le D_{\text{in}}} \left(\mathbb{C}\mathrm{ov}\left[a_iW_{ki}, b_l\right] + \mathbb{C}\mathrm{ov}\left[a_iW_{li}, b_k\right]\right) + \mathbb{C}\mathrm{ov}\left[b_k, b_l\right] \tag{17}$$

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

$$\mathbb{C}\mathrm{ov}\left[a_iW_{ki}, a_jW_{lj}\right]$$

$$= \mathbb{E}\left[(a_iW_{ki} - \mathbb{E}\left[a_iW_{ki}\right])(a_jW_{lj} - \mathbb{E}\left[a_jW_{lj}\right])\right] \tag{18}$$

$$= \mathbb{E}\left[a_iW_{ki}a_jW_{lj} - a_iW_{ki}\mathbb{E}\left[a_jW_{lj}\right] - \mathbb{E}\left[a_iW_{ki}\right]a_jW_{lj} + \mathbb{E}\left[a_iW_{ki}\right]\mathbb{E}\left[a_jW_{lj}\right]\right] \tag{19}$$

$$= \mathbb{E}\left[a_ia_jW_{ki}W_{lj}\right] - \mathbb{E}\left[a_iW_{ki}\right]\mathbb{E}\left[a_jW_{lj}\right] - \mathbb{E}\left[a_iW_{ki}\right]\mathbb{E}\left[a_jW_{lj}\right] + \mathbb{E}\left[a_iW_{ki}\right]\mathbb{E}\left[a_jW_{lj}\right] \tag{20}$$

$$\approx \mathbb{E}\left[a_ia_j\right]\mathbb{E}\left[W_{ki}W_{lj}\right] - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[a_j\right]\mathbb{E}\left[W_{lj}\right] \qquad \text{(Assumption 2)}$$

$$= (\mathbb{E}\left[a_i\right]\mathbb{E}\left[a_j\right] + \mathbb{C}\mathrm{ov}\left[a_i, a_j\right])(\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[W_{lj}\right] + \mathbb{C}\mathrm{ov}\left[W_{ki}, W_{lj}\right]) - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[a_j\right]\mathbb{E}\left[W_{lj}\right] \tag{21}$$

$$= \mathbb{E}\left[a_i\right]\mathbb{E}\left[a_j\right]\mathbb{C}\mathrm{ov}\left[W_{ki}, W_{lj}\right] + \mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[W_{lj}\right]\mathbb{C}\mathrm{ov}\left[a_i, a_j\right] + \mathbb{C}\mathrm{ov}\left[a_i, a_j\right]\mathbb{C}\mathrm{ov}\left[W_{ki}, W_{lj}\right]. \tag{22}$$

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

$$\mathbb{C}\mathrm{ov}\left[a_iW_{ki}, b_l\right] = \mathbb{E}\left[(a_iW_{ki} - \mathbb{E}\left[a_iW_{ki}\right])(b_l - \mathbb{E}\left[b_l\right])\right] \tag{23}$$

$$\approx \mathbb{E}\left[(a_iW_{ki} - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right])(b_l - \mathbb{E}\left[b_l\right])\right] \qquad \text{(Assumption 2)}$$

$$= \mathbb{E}\left[a_iW_{ki}b_l - a_iW_{ki}\mathbb{E}\left[b_l\right] - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]b_l + \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right]\right] \tag{24}$$

$$= \mathbb{E}\left[a_iW_{ki}b_l\right] - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right] \tag{25}$$

$$\approx \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}b_l\right] - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right] \qquad \text{(Assumption 2)}$$

$$= \mathbb{E}\left[a_i\right]\left(\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right] + \mathbb{C}\mathrm{ov}\left[W_{ki}, b_l\right]\right) - \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[b_l\right] \tag{26}$$

$$= \mathbb{E}\left[a_i\right]\mathbb{C}\mathrm{ov}\left[W_{ki}, b_l\right]. \tag{27}$$

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

$$\sum_{1\leq i,j\leq \mathrm{D}_{\text{in}}} \mathbb{C}\text{ov}\left[a_i W_{ki}, a_j W_{lj}\right] + \sum_{i=1}^{\mathrm{D}_{\text{in}}} \left(\mathbb{E}\left[a_i\right] \mathbb{C}\text{ov}\left[W_{ki}, b_l\right] + \mathbb{E}\left[a_i\right] \mathbb{C}\text{ov}\left[W_{li}, b_k\right]\right) + \mathbb{C}\text{ov}\left[b_k, b_l\right], \tag{28}$$

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

$$\mathbb{E}\left[a_i\right] \mathbb{E}\left[a_j\right] \mathbb{C}\text{ov}\left[W_{ki}, W_{lj}\right] + \mathbb{E}\left[W_{ki}\right] \mathbb{E}\left[W_{lj}\right] \mathbb{C}\text{ov}\left[a_i, a_j\right] + \mathbb{C}\text{ov}\left[a_i, a_j\right] \mathbb{C}\text{ov}\left[W_{ki}, W_{lj}\right]. \tag{29}$$

Note that $\sum_{1\leq i,j\leq \mathrm{D}_{\text{in}}} \mathbb{C}\text{ov}[a_i W_{ki}, a_j W_{lj}]$ in Eq. (28) could be rewrite into the form of matrix multiplication for efficient implementation:

$$\sum_{1\leq i,j\leq \mathrm{D}_{\text{in}}} \mathbb{C}\text{ov}\left[a_i W_{ki}, a_j W_{lj}\right] \tag{30}$$

$$= \sum_{1\leq i,j\leq \mathrm{D}_{\text{in}}} \mathbb{E}\left[a_i\right] \mathbb{E}\left[a_j\right] \mathbb{C}\text{ov}\left[W_{ki}, W_{lj}\right] + \mathbb{E}\left[W_{ki}\right] \mathbb{E}\left[W_{lj}\right] \mathbb{C}\text{ov}\left[a_i, a_j\right] + \mathbb{C}\text{ov}\left[a_i, a_j\right] \mathbb{C}\text{ov}\left[W_{ki}, W_{lj}\right] \tag{31}$$

$$= \sum \begin{bmatrix} \mathbb{E}\left[a_1\right] \mathbb{E}\left[a_1\right] \mathbb{C}\text{ov}[W_{k1}, W_{l1}] & \dots & \mathbb{E}\left[a_1\right] \mathbb{E}\left[a_{\mathrm{D}_{\text{in}}}\right] \mathbb{C}\text{ov}[W_{k1}, W_{l\,\mathrm{D}_{\text{in}}}] \\ \vdots & \vdots & \vdots \\ \mathbb{E}\left[a_{\mathrm{D}_{\text{in}}}\right] \mathbb{E}\left[a_1\right] \mathbb{C}\text{ov}[W_{k\,\mathrm{D}_{\text{in}}}, W_{l1}] & \dots & \mathbb{E}\left[a_1\right] \mathbb{E}\left[a_{\mathrm{D}_{\text{in}}}\right] \mathbb{C}\text{ov}[W_{k\,\mathrm{D}_{\text{in}}}, W_{l\,\mathrm{D}_{\text{in}}}] \end{bmatrix} \tag{32}$$

$$\odot \begin{bmatrix} \mathbb{C}\text{ov}[W_{k1}, W_{l1}] & \dots & \mathbb{C}\text{ov}[W_{k1}, W_{l\,\mathrm{D}_{\text{in}}}] \\ \vdots & \vdots & \vdots \\ \mathbb{C}\text{ov}[W_{k\,\mathrm{D}_{\text{in}}}, W_{l1}] & \dots & \mathbb{C}\text{ov}[W_{k\,\mathrm{D}_{\text{in}}}, W_{l\,\mathrm{D}_{\text{in}}}] \end{bmatrix} \tag{33}$$

$$+ \sum \begin{bmatrix} \mathbb{E}\left[W_{k1}\right] \mathbb{E}\left[W_{l1}\right] & \dots & \mathbb{E}\left[W_{k1}\right] \mathbb{E}\left[W_{l\,\mathrm{D}_{\text{in}}}\right] \\ \vdots & \vdots & \vdots \\ \mathbb{E}\left[W_{k\,\mathrm{D}_{\text{in}}}\right] \mathbb{E}\left[W_{l1}\right] & \dots & \mathbb{E}\left[W_{k\,\mathrm{D}_{\text{in}}}\right] \mathbb{E}\left[W_{l\,\mathrm{D}_{\text{in}}}\right] \end{bmatrix} \odot \begin{bmatrix} \mathbb{C}\text{ov}[a_1, a_1] & \dots & \mathbb{C}\text{ov}[a_1, a_{\mathrm{D}_{\text{in}}}] \\ \vdots & \vdots & \vdots \\ \mathbb{C}\text{ov}[a_{\mathrm{D}_{\text{in}}}, a_1] & \dots & \mathbb{C}\text{ov}[a_{\mathrm{D}_{\text{in}}}, a_{\mathrm{D}_{\text{in}}}] \end{bmatrix} \tag{34}$$

$$+ \sum \begin{bmatrix} \mathbb{C}\text{ov}[a_1, a_1] & \dots & \mathbb{C}\text{ov}[a_1, a_{\mathrm{D}_{\text{in}}}] \\ \vdots & \vdots & \vdots \\ \mathbb{C}\text{ov}[a_{\mathrm{D}_{\text{in}}}, a_1] & \dots & \mathbb{C}\text{ov}[a_{\mathrm{D}_{\text{in}}}, a_{\mathrm{D}_{\text{in}}}] \end{bmatrix} \odot \begin{bmatrix} \mathbb{C}\text{ov}[W_{k1}, W_{l1}] & \dots & \mathbb{C}\text{ov}[W_{k1}, W_{l\,\mathrm{D}_{\text{in}}}] \\ \vdots & \vdots & \vdots \\ \mathbb{C}\text{ov}[W_{k\,\mathrm{D}_{\text{in}}}, W_{l1}] & \dots & \mathbb{C}\text{ov}[W_{k\,\mathrm{D}_{\text{in}}}, W_{l\,\mathrm{D}_{\text{in}}}] \end{bmatrix} \tag{35}$$

A.2 Error Induced Through Local Gaussian Approximation

In this section we provide analysis of the error induced through the local Gaussian approximation. Recall we made these two assumptions for the derivation:

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

We first examine the error induced by A2 on the moments for $a_i W_{ki}$. Given two correlated univariate Gaussian x_1 and x_2 with the joint being

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbb{E}\left[x_1\right] \\ \mathbb{E}\left[x_2\right] \end{bmatrix}, \begin{bmatrix} \sigma_{x_1}^2 & \mathbb{C}\text{ov}[x_1, x_2] \\ \mathbb{C}\text{ov}[x_1, x_2] & \sigma_{x_2}^2 \end{bmatrix}\right), \tag{36}$$

from Nadarajah & Pogány (2016); Kan (2008), although the distribution form of $x_1 x_2$ is no longer Gaussian and intractable, its mean and variance can be computed analytically as

$$\mathbb{E}\left[x_1 x_2\right] = \mathbb{E}\left[x_1\right] \mathbb{E}\left[x_2\right] + \mathbb{C}\text{ov}\left[x_1, x_2\right], \tag{37}$$

$$\mathbb{V}\text{ar}\left[x_1 x_2\right] = \sigma_1^2 \sigma_2^2 + \sigma_1^2 \mathbb{E}\left[x_2\right]^2 + \sigma_2^2 \mathbb{E}\left[x_1\right]^2 + \left(\sigma_1^2 \sigma_2^2 + 2\mathbb{E}\left[x_1\right] \mathbb{E}\left[x_2\right]\right) \mathbb{C}\text{ov}\left[x_1, x_2\right]. \tag{38}$$

Applying the above result in our case, we have

$$\mathbb{E}\left[a_i W_{ki}\right] = \mathbb{E}\left[a_i\right] \mathbb{E}\left[W_{ki}\right] + \mathbb{C}\text{ov}\left[a_i, W_{ki}\right], \tag{39}$$

$$\mathbb{V}\text{ar}\left[a_i W_{ki}\right] = \sigma_{a_i}^2 \sigma_{W_{ki}}^2 + \sigma_{a_i}^2 \mathbb{E}\left[W_{ki}\right]^2 + \sigma_{W_{ki}}^2 \mathbb{E}\left[a_i\right]^2 + \left(2\mathbb{E}\left[a_i\right] \mathbb{E}\left[W_{ki}\right] + \sigma_{a_i}^2 \sigma_{W_{ki}}^2\right) \mathbb{C}\text{ov}\left[a_i, W_{ki}\right]. \tag{40}$$

As $\mathbb{C}\text{ov}[a_i, W_{ki}]$ is intractable, in A2 we ignore the correlation between a_i and W_{ki}, which results in

$$\mathbb{E}\left[a_i W_{ki}\right] \approx \mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right] \cancel{+ \mathbb{C}\text{ov}\left[a_i, W_{ki}\right]}, \tag{41}$$

$$\mathbb{V}\text{ar}\left[a_i W_{ki}\right] \approx \sigma_{a_i}^2 \sigma_{W_{ki}}^2 + \sigma_{a_i}^2 \mathbb{E}\left[W_{ki}\right]^2 + \sigma_{W_{ki}}^2 \mathbb{E}\left[a_i\right]^2 \cancel{+ (2\mathbb{E}\left[a_i\right]\mathbb{E}\left[W_{ki}\right] + \sigma_{a_i}^2 \sigma_{W_{ki}}^2)\, \mathbb{C}\text{ov}\left[a_i, W_{ki}\right]}. \tag{42}$$

Note that in the case of diagonal posterior covariance, as each parameters are independent from each other, A2 holds automatically. In this case we recover the correct mean and variance for $a_i W_{ki}$.

Now, we examine the error induced by A1 and A2 through Monte-Carlo estimation. Fig. 6 provides a simulation result illustrating the error induced by the local Gaussian approximation on $a_i W_{ki}$. We plot the results for weights with the largest absolute magnitude of a MLP trained on MNIST. We find this approximation to work well in practice, but fail to capture potential skewness of the distributions.

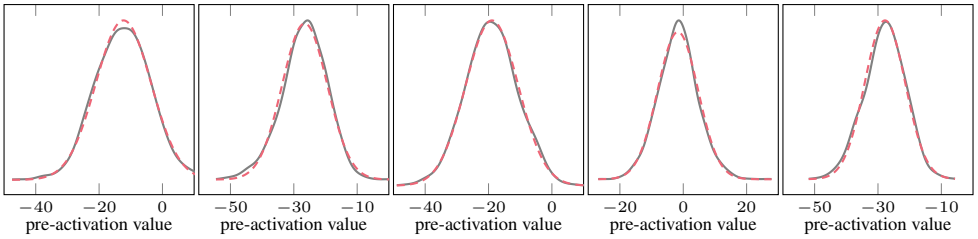


Figure 6: Comparison between Monte-Carlo estimates —— of the distribution over $a_i W_{ki}$ and our analytic Gaussian approximation - - -.

A.3 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, when $k \neq l$ we have $\mathbb{C}\text{ov}[h_k, h_l] =$

$$\sum_{1 \leq i,j \leq \mathrm{D}_{\text{in}}} \mathbb{C}\text{ov}\left[a_i W_{ki}, a_j W_{lj}\right] + \sum_{i=1}^{\mathrm{D}_{\text{in}}} \left(\mathbb{E}\left[a_i\right]\mathbb{C}\text{ov}\left[W_{ki}, b_l\right] + \mathbb{E}\left[a_i\right]\mathbb{C}\text{ov}\left[W_{li}, b_k\right]\right) + \mathbb{C}\text{ov}\left[b_k, b_l\right] \tag{43}$$

$$= \sum_{1 \leq i,j \leq \mathrm{D}_{\text{in}}} \mathbb{C}\text{ov}\left[a_i W_{ki}, a_j W_{lj}\right] \tag{44}$$

$$= \sum_{1 \leq i,j \leq \mathrm{D}_{\text{in}}} \mathbb{E}\left[a_i\right]\mathbb{E}\left[a_j\right]\mathbb{C}\text{ov}\left[W_{ki}, W_{lj}\right] + \mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[W_{lj}\right]\mathbb{C}\text{ov}\left[a_i, a_j\right] + \mathbb{C}\text{ov}\left[a_i, a_j\right]\mathbb{C}\text{ov}\left[W_{ki}, W_{lj}\right] \tag{45}$$

$$= \sum_{1 \leq i,j \leq \mathrm{D}_{\text{in}}} \mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[W_{lj}\right]\mathbb{C}\text{ov}\left[a_i, a_j\right]. \tag{46}$$

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

$$\sum_{1 \leq i,j \leq \mathrm{D}_{\text{in}}} \mathbb{C}\text{ov}\left[a_i W_{ki}, a_j W_{kj}\right] + \sum_{i=1}^{\mathrm{D}_{\text{in}}} \left(\mathbb{E}\left[a_i\right]\mathbb{C}\text{ov}\left[W_{ki}, b_k\right] + \mathbb{E}\left[a_i\right]\mathbb{C}\text{ov}\left[W_{ki}, b_k\right]\right) + \mathbb{V}\text{ar}\left[b_k\right] \tag{47}$$

$$= \sum_{1 \leq i \leq \mathrm{D}_{\text{in}}} \mathbb{C}\text{ov}\left[a_i W_{ki}, a_i W_{ki}\right] + \mathbb{V}\text{ar}\left[b_k\right] \tag{48}$$

$$= \sum_{1 \leq i \leq \mathrm{D}_{\text{in}}} \mathbb{E}\left[a_i\right]^2 \mathbb{V}\text{ar}\left[W_{ki}\right] + \mathbb{E}\left[W_{ki}\right]^2 \mathbb{V}\text{ar}\left[a_i\right] + \mathbb{V}\text{ar}\left[a_i\right]\mathbb{V}\text{ar}\left[W_{ki}\right] + \mathbb{V}\text{ar}\left[b_k\right]. \tag{49}$$

Note that as

$$\mathbb{V}\mathrm{ar}\left[h_k^{(m)}\right] = \sum_{1\le i\le \mathrm{D_{in}}} \mathbb{E}\left[a_i^{(m-1)}\right]^2 \mathbb{V}\mathrm{ar}\left[W_{ki}^{(m)}\right] + \mathbb{E}\left[W_{ki}^{(m)}\right]^2 \mathbb{V}\mathrm{ar}\left[a_i^{(m-1)}\right] \tag{50}$$

$$+ \sum_{1\le i\le \mathrm{D_{in}}} \mathbb{V}\mathrm{ar}\left[a_i^{(m-1)}\right] \mathbb{V}\mathrm{ar}\left[W_{ki}^{(m)}\right] + \mathbb{V}\mathrm{ar}\left[b_k^{(m)}\right], \tag{51}$$

the variance of $h_k^{(m)}$ will only rely on the variance of the activations of previous layers, *i.e.*, $\mathbb{V}\mathrm{ar}[a_i^{(m-1)}]$. In the case of element-wise activation functions, $\mathbb{V}\mathrm{ar}[a_i^{(m-1)}]$ will only rely on $\mathbb{V}\mathrm{ar}[h_i^{(m-1)}]$ as now the Jacobian of activation is diagonal. As a result, in the case where we only need the variance of the input, we could drop the computation of $\mathbb{C}\mathrm{ov}[h_k, h_l]$ and only compute the variance for each layer, which will largely reduce the computation cost.

A.4 Derivation for Kronecker Covariance Structure

In KFAC, the Hessian is represented in Kronecker product form $\mathbf{Hess} = \boldsymbol{A} \otimes \boldsymbol{B}$. Denote the prior precision as λ^2, then the posterior covariance is

$$\boldsymbol{\Sigma} = (\mathbf{Hess} + \lambda^2 \mathbf{I})^{-1} = (\boldsymbol{A} \otimes \boldsymbol{B} + \lambda^2 \mathbf{I})^{-1} \tag{52}$$

As there is no closed form for the inverse, to express the covariance in the form of Kronecker product as well, we approximate the covariance as

$$\begin{aligned}
\boldsymbol{\Sigma} &= (\boldsymbol{A} \otimes \boldsymbol{B} + \lambda^2 \mathbf{I})^{-1} && (53)\\
&= \left[(\boldsymbol{U}_A \boldsymbol{\Lambda}_A \boldsymbol{U}_A^\top) \otimes (\boldsymbol{U}_B \boldsymbol{\Lambda}_B \boldsymbol{U}_B^\top) + \lambda^2 \mathbf{I}\right]^{-1} && \text{(Eigen Decomposition)}\\
&\approx \left[(\boldsymbol{U}_A (\boldsymbol{\Lambda}_A + \lambda \mathbf{I}_A) \boldsymbol{U}_A^\top) \otimes (\boldsymbol{U}_B (\boldsymbol{\Lambda}_B + \lambda \mathbf{I}_B) \boldsymbol{U}_B^\top)\right]^{-1} && (54)\\
&= \underbrace{\left(\boldsymbol{U}_A (\boldsymbol{\Lambda}_A + \lambda \mathbf{I}_A) \boldsymbol{U}_A^\top\right)^{-1}}_{\boldsymbol{C}} \otimes \underbrace{\left(\boldsymbol{U}_B (\boldsymbol{\Lambda}_B + \lambda \mathbf{I}_B) \boldsymbol{U}_B^\top\right)^{-1}}_{\boldsymbol{D}}. && ((\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1})
\end{aligned}$$

Recall for an efficient implementation for computing $\sum_{1\le i,j\le \mathrm{D_{in}}} \mathbb{C}\mathrm{ov}[a_i W_{ki}, a_j W_{lj}]$ (Eq. (35)), we need to retrieve the covariance between the kth row of weight and l^{th} row of weight, which is a $\mathrm{D_{in}} \times \mathrm{D_{in}}$ matrix:

$$\mathbb{C}\mathrm{ov}\left[\boldsymbol{W}[k,:], \boldsymbol{W}[l,:]\right] = \begin{bmatrix} \mathbb{C}\mathrm{ov}[W_{k1}, W_{l1}] & \dots & \mathbb{C}\mathrm{ov}[W_{k1}, W_{l\,\mathrm{D_{in}}}] \\ \vdots & \ddots & \vdots \\ \mathbb{C}\mathrm{ov}[W_{k\,\mathrm{D_{in}}}, W_{l1}] & \dots & \mathbb{C}\mathrm{ov}[W_{k\,\mathrm{D_{in}}}, W_{l\,\mathrm{D_{in}}}] \end{bmatrix}. \tag{55}$$

As $\boldsymbol{\Sigma} \approx \boldsymbol{C} \otimes \boldsymbol{D}$ where $\boldsymbol{C} \in \mathrm{D_{in}} \times \mathrm{D_{in}}$ and $\boldsymbol{D} \in \mathrm{D_{out}} \times \mathrm{D_{out}}$, the posterior covariance is represented by a total number of $\mathrm{D_{in}} \times \mathrm{D_{in}}$ matrix with size $\mathrm{D_{out}} \times \mathrm{D_{out}}$. Retrieving a $\mathrm{D_{in}} \times \mathrm{D_{in}}$ matrix from it is not trivial. In the toy example as shown in Fig. 7, for a $\mathrm{D_{in}} = 3$ and $\mathrm{D_{out}} = 2$ matrix $\boldsymbol{W}$, its covariance is represented by a total number of 9 ($\mathrm{D_{in}} \times \mathrm{D_{in}}$) matrix $I, II, \dots, IX$ with shape 2×2 ($\mathrm{D_{out}} \times \mathrm{D_{out}}$). To retrieve $\mathbb{C}\mathrm{ov}[\boldsymbol{W}[1,:], \boldsymbol{W}[2,:]]$, we need to first decide which Kronecker blocks contains it (in this case block II, III, V and VI) and reconstruct these Kronecker blocks. Then, we retrieve $\mathbb{C}\mathrm{ov}[\boldsymbol{W}[1,:], \boldsymbol{W}[2,:]]$ from the reconstructed blocks.

In general, the retrieval process consists of two steps: (1) identifying the block indices within the Kronecker product matrix that correspond to the required covariance block, and (2) extracting the covariance of interests from the constructed block

Identifying Block Indices We first identify the Kronecker blocks that contains the covariance of interest. This is achieved by calculating the block indiced for $\boldsymbol{C}$ which is later used to construct Kronecker blcoks. Specifically, the start and end positions of the covariance block corresponding to rows k and l can be computed as:

$$\text{row_start} = \left\lfloor \frac{k \cdot D_{\text{in}}}{D_{\text{out}}} \right\rfloor, \tag{56}$$

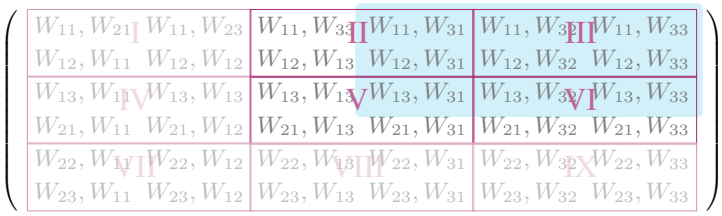


Figure 7: To retrieve the highlighted submatrix $\mathbb{C}\mathrm{ov}[\boldsymbol{W}[1,:], \boldsymbol{W}[2,:]]$ of the covariance for $\boldsymbol{W} \in \mathbb{R}^{2\times 3}$, we identify the Kronecker blocks that contain the covariance of interest (II, III, V, and VI), explicate those blocks in memory, and then retrieve the relevant submatrix.

$$\text{row_end} = \left\lceil \frac{(k+1)\cdot D_{\text{in}}}{D_{\text{out}}} \right\rceil, \tag{57}$$

$$\text{col_start} = \left\lfloor \frac{l\cdot D_{\text{in}}}{D_{\text{out}}} \right\rfloor, \tag{58}$$

$$\text{col_end} = \left\lceil \frac{(l+1)\cdot D_{\text{in}}}{D_{\text{out}}} \right\rceil. \tag{59}$$

Then, we can construct the Kronecker blocks that contain the covariance of interest by $\boldsymbol{C}[\text{row_start} : \text{row_end}, \text{col_start} : \text{col_end}]\boldsymbol{D}$.

Extract the Covariance Once we have $\boldsymbol{C}[\text{row_start} : \text{row_end}, \text{col_start} : \text{col_end}]\boldsymbol{D}$, as we know the covariance we need to retrieve has shape $\mathrm{D}_{\text{in}} \times \mathrm{D}_{\text{in}}$, we only need to compute the start row and column index, which can be computed as

$$\text{select_row_start} = (k\cdot D_{\text{in}}) \bmod D_{\text{out}}, \tag{60}$$

$$\text{select_col_start} = (l\cdot D_{\text{in}}) \bmod D_{\text{out}}. \tag{61}$$

A.5 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 $\mathbb{E}[\boldsymbol{h}]$:

$$\boldsymbol{a} = g(\boldsymbol{h}) \tag{62}$$

$$\approx g(\mathbb{E}[\boldsymbol{h}]) + \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}(\boldsymbol{h} - \mathbb{E}[\boldsymbol{h}]). \tag{63}$$

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

$$\boldsymbol{h} \sim \mathcal{N}(\mathbb{E}[\boldsymbol{h}], \boldsymbol{\Sigma}_h) \tag{64}$$

$$\boldsymbol{h} - \mathbb{E}[\boldsymbol{h}] \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_h) \tag{65}$$

$$\boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}(\boldsymbol{h} - \mathbb{E}[\boldsymbol{h}]) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}{}^{\top}\boldsymbol{\Sigma}_h\boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}) \tag{66}$$

$$g(\mathbb{E}[\boldsymbol{h}]) + \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}(\boldsymbol{h} - \mathbb{E}[\boldsymbol{h}]) \sim \mathcal{N}(g(\mathbb{E}[\boldsymbol{h}]), \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}{}^{\top}\boldsymbol{\Sigma}_h\boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}) \tag{67}$$

$$\boldsymbol{a} \underset{\text{approx}}{\sim} \mathcal{N}(\boldsymbol{a}; g(\mathbb{E}[\boldsymbol{h}]), \boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}{}^{\top}\boldsymbol{\Sigma}_h\boldsymbol{J}_g|_{\boldsymbol{h}=\mathbb{E}[\boldsymbol{h}]}). \tag{68}$$

A.6 Transformer Block

There are four components in each transformer block (Vaswani et al., 2017): (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 and residual connection, as Gaussian distribution is closed under linear transformation, push distribution over them is straightforward. We describe

how to push distribution through attention layers below. Note for computational reasons, we always assume the input has diagonal covariance.

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

$$\boldsymbol{Q} = \boldsymbol{H}\boldsymbol{W}_Q, \quad \boldsymbol{K} = \boldsymbol{H}\boldsymbol{W}_K, \quad \boldsymbol{V} = \boldsymbol{H}\boldsymbol{W}_V, \tag{69}$$

and the output of attention block is

$$\text{Attention}(\boldsymbol{H}) = \text{Softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}})\boldsymbol{V}. \tag{70}$$

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:

$$\boldsymbol{Q} = \mathbb{E}\left[\boldsymbol{H}\right]\mathbb{E}\left[\boldsymbol{W}_Q\right], \quad \boldsymbol{K} = \mathbb{E}\left[\boldsymbol{H}\right]\mathbb{E}\left[\boldsymbol{W}_K\right]. \tag{71}$$

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 kth and the lth value is

$$\mathbb{C}\text{ov}\left[v_k, v_l\right] = \mathbb{C}\text{ov}\left[\sum_{i=1}^{D} W_{V_{ki}} h_i, \sum_{j=1}^{D} W_{V_{lj}} h_j\right] \tag{72}$$

$$= \sum_{i=1}^{D}\sum_{j=1}^{D} \mathbb{C}\text{ov}\left[W_{V_{ki}} h_i, W_{V_{lj}} h_j\right]. \tag{73}$$

When treating $\boldsymbol{W}_V$ deterministically, we have

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

When $\boldsymbol{W}_V$ is a isotropic Gaussian, we have

$$\begin{aligned}
\mathbb{C}\text{ov}\left[v_k, v_l\right] &= \sum_{1 \le i,j \le D} \mathbb{C}\text{ov}\left[W_{V_{ki}} h_i, W_{V_{lj}} h_j\right] && (74) \\
&\approx \sum_{1 \le i,j \le D} \left(\mathbb{E}\left[h_i\right]\mathbb{E}\left[h_j\right] + \mathbb{C}\text{ov}\left[h_i, h_j\right]\right)\mathbb{C}\text{ov}\left[W_{ki}, W_{lj}\right] + \mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[W_{lj}\right]\mathbb{C}\text{ov}\left[h_i, h_j\right] \\
& && \text{(assumption A2)} \\
&= \sum_{1 \le i,j \le D} \mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[W_{lj}\right]\mathbb{C}\text{ov}\left[h_i, h_j\right] && (\boldsymbol{W}_V \text{ is isotropic Gaussian}) \\
&\approx \sum_{1 \le i \le D} \mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[W_{li}\right]\mathbb{V}\text{ar}\left[h_i\right]. \\
& && \text{(ignore correlation between } \boldsymbol{h} \text{ for computational reason)}
\end{aligned}$$

$$
\begin{aligned}
\mathbb{V}\mathrm{ar}\left[v_k\right] &= \sum_{1\le i,j\le D} \mathbb{C}\mathrm{ov}\left[W_{V_{ki}}h_i, W_{V_{kj}}h_j\right] && \text{(definition)}\\
&\approx \sum_{1\le i,j\le D} \left(\mathbb{E}\left[h_i\right]\mathbb{E}\left[h_j\right] + \mathbb{C}\mathrm{ov}\left[h_i,h_j\right]\right)\mathbb{C}\mathrm{ov}\left[W_{ki},W_{kj}\right] + \mathbb{E}\left[W_{ki}\right]\mathbb{E}\left[W_{kj}\right]\mathbb{C}\mathrm{ov}\left[h_i,h_j\right] \\
& && \text{(assumption A2)}\\
&= \sum_{1\le i\le D} \left(\mathbb{E}\left[h_i\right]^2 + \mathbb{V}\mathrm{ar}\left[h_i\right]\right)\mathbb{V}\mathrm{ar}\left[W_{ki}\right] + \mathbb{E}\left[W_{ki}\right]^2\mathbb{V}\mathrm{ar}\left[h_i\right]. \\
& && (\boldsymbol{W}_V \text{ is isotropic Gaussian})
\end{aligned} \tag{75}
$$

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.

A.7 Convolutional Neural Network

The derivation for convolutional layer is very similar to fully connected layer as convolution layer can be considered as a shared weight fully connected layer. We first give the derivation for convolutional layer, then discuss pooling layers in convolutional neural network.

Denote the pixel value at (i,j) of c_{in}th channel as $a_{c_{\text{in}}}[i,j]$, the c_{in}th channel of convolutional kernel corresponding to c_{out}th output channel as $W_{c_{\text{out}},c_{\text{in}}}[i,j]$ and the pixel value at (k,l) of the c_{out}th output channel as $h_{c_{\text{out}}}[k,l]$. Then suppose there are C_{in} channel in total and the kernel size is $K_h \times K_w$, we can write convolutional layer as

$$
h_{c_{\text{out}}}[k,l] = \sum_{c_{\text{in}}=1}^{C_{\text{in}}}\sum_{i=1}^{K_h}\sum_{j=1}^{K_w} a_{c_{\text{in}}}[k+i-1, l+j-1]W_{c_{\text{out}},c_{\text{in}}}[i,j]. \tag{76}
$$

Derivation of mean Following our assumption that $a_{c_{\text{in}}}[k+i-1,l+j-1]$ is uncorrelated with $W_{c_{\text{out}},c_{\text{in}}}[i,j]$, we have

$$
\mathbb{E}\left[h_{c_{\text{out}}}[k,l]\right] = \sum_{c_{\text{in}}=1}^{C_{\text{in}}}\sum_{i=1}^{K_h}\sum_{j=1}^{K_w} \mathbb{E}\left[a_{c_{\text{in}}}[k+i-1,l+j-1]\right]\mathbb{E}\left[W_{c_{\text{out}},c_{\text{in}}}[i,j]\right] \tag{77}
$$

Derivation of covariance The covariance between pixels of the c_{out}th output channel are given as

$$
\begin{aligned}
&\mathbb{C}\mathrm{ov}\left[h_{c_{\text{out}}}[k_1,l_1], h_{c_{\text{out}}}[k_2,l_2]\right] && (78)\\
&= \mathbb{C}\mathrm{ov}\left[\sum_{c_{\text{in},1}=1}^{C_{\text{in}}}\sum_{i_1=1}^{K_h}\sum_{j_1=1}^{K_w} a_{c_{\text{in},1}}[k_1+i_1-1,l_1+j_1-1]W_{c_{\text{out}},c_{\text{in}}}[i_1,j_1]\,, \right. && (79)\\
&\qquad \left. \sum_{c_{\text{in},2}=1}^{C_{\text{in}}}\sum_{i_2=1}^{K_h}\sum_{j_2=1}^{K_w} a_{c_{\text{in},2}}[k_2+i_2-1,l_2+j_2-1]W_{c_{\text{out}},c_{\text{in}}}[i_2,j_2]\right] && (80)\\
&= \sum_{c_{\text{in},1}=1}^{C_{\text{in}}}\sum_{i_1=1}^{K_h}\sum_{j_1=1}^{K_w}\sum_{c_{\text{in},2}=1}^{C_{\text{in}}}\sum_{i_2=1}^{K_h}\sum_{j_2=1}^{K_w} \mathbb{C}\mathrm{ov}\left[a_{c_{\text{in},1}}[k_1+i_1-1,l_1+j_1-1]W_{c_{\text{out}},c_{\text{in}}}[i_1,j_1]\right. && (81)\\
&\qquad \left. a_{c_{\text{in},2}}[k_2+i_2-1,l_2+j_2-1]W_{c_{\text{out}},c_{\text{in}}}[i_2,j_2]\right]. && (82)
\end{aligned}
$$

Using earlier result in Eq. (22), we have

$$\mathbb{C}\text{ov}\left[a_{\text{in},1}[k_1+i_1-1, l_1+j_1-1]W_{c_{\text{out}},c_{\text{in}}}[i_1,j_1], a_{\text{in},2}[k_2+i_2-1, l_2+j_2-1]W_{c_{\text{out}},c_{\text{in}}}[i_2,j_2]\right] \tag{83}$$

$$\approx \mathbb{E}\left[a_{\text{in},1}[k_1+i_1-1, l_1+j_1-1]a_{\text{in},2}[k_2+i_2-1, l_2+j_2-1]\right]\mathbb{E}\left[W_{c_{\text{out}},c_{\text{in}}}[i_1,j_1]W_{c_{\text{out}},c_{\text{in}}}[i_2,j_2]\right] \tag{84}$$

$$-\mathbb{E}\left[a_{\text{in},1}[k_1+i_1-1, l_1+j_1-1]\right]\mathbb{E}\left[a_{\text{in},2}[k_2+i_2-1, l_2+j_2-1]\right]\mathbb{E}\left[W_{c_{\text{out}},c_{\text{in}}}[i_1,j_1]\right]\mathbb{E}\left[W_{c_{\text{out}},c_{\text{in}}}[i_2,j_2]\right] \tag{85}$$

$$=\mathbb{E}\left[a_{\text{in},1}[k_1+i_1-1, l_1+j_1-1]\right]\mathbb{E}\left[a_{\text{in},2}[k_2+i_2-1, l_2+j_2-1]\right]\mathbb{C}\text{ov}\left[W_{c_{\text{out}},c_{\text{in}}}[i_1,j_1], W_{c_{\text{out}},c_{\text{in}}}[i_2,j_2]\right] \tag{86}$$

$$+\mathbb{E}\left[W_{c_{\text{out}},c_{\text{in}}}[i_1,j_1]\right]\mathbb{E}\left[W_{c_{\text{out}},c_{\text{in}}}[i_2,j_2]\right]\mathbb{C}\text{ov}\left[a_{\text{in},1}[k_1+i_1-1, l_1+j_1-1], a_{\text{in},2}[k_2+i_2-1, l_2+j_2-1]\right] \tag{87}$$

$$+\mathbb{C}\text{ov}\left[a_{\text{in},1}[k_1+i_1-1, l_1+j_1-1], a_{\text{in},2}[k_2+i_2-1, l_2+j_2-1]\right]\mathbb{C}\text{ov}\left[W_{c_{\text{out}},c_{\text{in}}}[i_1,j_1], W_{c_{\text{out}},c_{\text{in}}}[i_2,j_2]\right] \tag{88}$$

B Additional Experiments

B.1 Regression

Table 7 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 8 we report the Root Mean Square Error (RMSE), our method 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 7: UCI regression experiment setup.

Data Set 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

B.2 Classification

Table 9 gives the classification data sets information and the neural network structure we used for the MLP experiment. We use ReLU activation for MLP.

OOD Experiments with MLP To test our method on out-of-distribution (OOD) data, we first evaluate the MNIST-trained MLP on rotated versions of the test set as shown in Fig. 8. The rotation degree interval is $10°$ from $0-180°$. We observe that with increasing rotation degree, our method achieves a lower NLPD compared to LA MAP and MFVI Sampling, while being close compared with LA Sampling and GLM. Also, our method achieves similar NLPD for both LA and MFVI posterior approximations across the rotation degrees. All methods perform on par on the ACC. In Fig. 9, 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 8: 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

Table 9: Classification experiment setup.

Data Set Name	(n, d)	Network Structure
MNIST	(50000, 784)	d-128-64-10
FMNIST	(50000, 784)	d-128-64-10
ORGANCMNIST	(12975, 784)	d-128-64-11
ORGANSMNIST	(13932, 784)	d-128-64-11

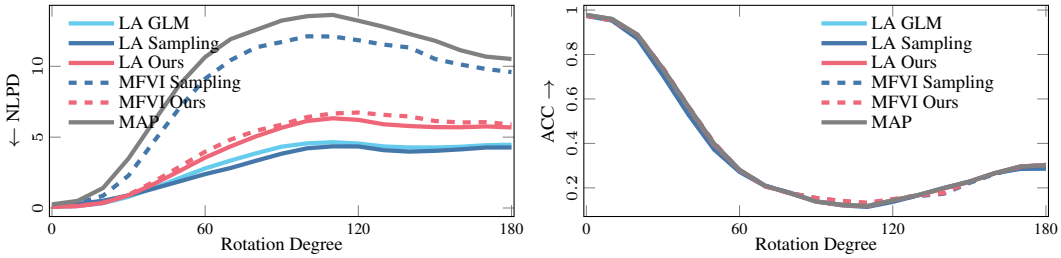


Figure 8: NLPD and ACC for MNIST-trained MLP on rotated versions of the MNIST test set. The rotation degree interval is $10°$ from $0 - 180°$. Our method achieves similar NLPD for both LA and MFVI posterior approximations.

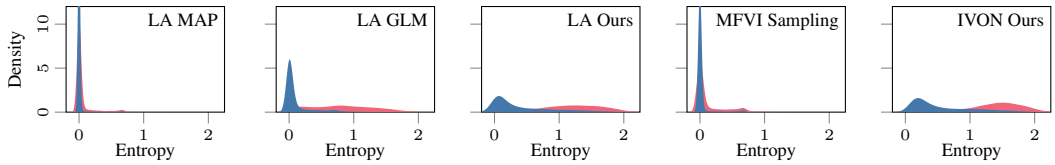


Figure 9: Kernel density plots over the predictive entropy from an MLP trained on FMNIST (blue, in-distribution) and data from MNIST (red, out-of-distribution). Our method results in a clear separation between the in- and out-of-distribution data.

Our method applied to MLP in ViT In Table 10 we report the results for fine tuning the MLPS after the attention layers in the last two transformer block in ViT and later treating them Bayesian. We observe that our method achieves better or on par NLPD and ECE compared to the baselines for both LA and MFVI across all data sets while maintaining similar ACC as the baselines.

We present results for GPT-2 on tasks from GLUE (Wang et al., 2019b) and SuperGLUE (Wang et al., 2019a) benchmark. These natural language understanding tasks could be turned into classification tasks with the prompt shown in Table 11. We add a classification layer on top of the encoder and use the embedding of the last token in each input to do classification.

Table 10: Performance metrics using ViT with posterior approximation on MLPs after the attention layers with the standard error for ACC and NLPD. Our method achieves better NLPD and ECE in general and achives similar ACC compared to the baselines.

Metrics	Methods	CIFAR-10	CIFAR-100	DTD	RESISC	IMAGENET-R
ACC ↑	LA Sampling	0.971±0.002	0.855±0.004	0.656±0.011	0.812±0.005	0.589±0.013
	LA GLM	**0.974**±0.002	0.873±0.003	**0.714**±0.010	0.886±0.004	**0.687**±0.012
	LA Ours	**0.976**±0.002	**0.884**±0.003	**0.716**±0.010	**0.909**±0.004	**0.713**±0.012
	MFVI Sampling	**0.978**±0.001	**0.896**±0.003	**0.727**±0.010	**0.870**±0.004	**0.733**±0.012
	MFVI Ours	**0.978**±0.001	**0.895**±0.003	**0.720**±0.010	**0.868**±0.004	**0.732**±0.012
NLPD ↓	LA Sampling	0.169±0.004	1.043±0.010	2.035±0.022	1.304±0.011	2.330±0.041
	LA GLM	**0.089**±0.005	0.602±0.011	1.260±0.029	0.568±0.011	1.584±0.045
	LA Ours	**0.088**±0.006	**0.457**±0.012	**1.078**±0.036	**0.318**±0.013	**1.339**±0.047
	MFVI Sampling	0.124±0.011	0.480±0.018	**1.277**±0.060	1.098±0.043	1.489±0.081
	MFVI Ours	**0.080**±0.005	**0.437**±0.013	**1.146**±0.040	**0.651**±0.020	**1.206**±0.053
ECE ↓	LA Sampling	0.078	0.349	0.442	0.431	0.331
	LA GLM	**0.005**	0.097	0.174	0.157	0.155
	LA Ours	0.006	**0.031**	**0.055**	**0.019**	**0.087**
	MFVI Sampling	0.014	0.040	0.083	0.075	0.115
	MFVI Ours	**0.005**	**0.033**	**0.053**	**0.024**	**0.041**

Table 11: Prompt templates for fine-tuning GPT-2 on natural language understanding tasks.

Task	Prompt
MRPC	Answer whether sentence 2 is equivalent to sentence 1. Sentence 1: {`sentence1`}. Sentence 2: {`sentence2`}. Answer:
WiC	Select whether word {`word`} has the same meaning in these two sentences. Sentence 1: {`sentence1`}. Sentence 2: {`sentence2`}. Answer:
BoolQ	Answer the question with only True or False. Passage: {`passage`}. Question: {`question`}. Answer:

B.3 IMAGE PIXEL SENSITIVITY

We trained a 4 layer MLP classifier on MNIST digits zero and eight using a batch size of 64, learning rate of $1e-3$, weight decay set to $1e-5$, and for 50 epochs. We used a subset of 0.1% of the training data as held-out validation set and assumed a full covariance Gaussian distribution for each input centred at the pixel values of the datum and with a fixed co-variance of 0.1. We then computed the pixel sensitivities for the trained model by learning the pixel-wise input covariance matrices by minimizing the negative log-likelihood of the held-out validation set and jointly maximizing the entropy of the input distributions. The optimization was performed using Adam with a learning rate of $5e-3$ until the validation loss dropped below a divergence to the intial loss of $1e-1$. Doing so typically took around 900 iterations. Fig. 10 shows some additional examples with the input-dependent sensitivties. Furthermore, we experiment with using an inter-class covariance shared between the images containing the same digit with the loss

$$\ell = \sum_{n=1}^{N} \text{cross-entropy}(f(\boldsymbol{x}_n), y_n) - \mathcal{H}(\mathcal{N}(\boldsymbol{x}_n, \Sigma_{c=y_n})), \tag{89}$$

where $\Sigma_{c=y_n}$ is the intra-class covariance for class c. Fig. 11 shows examples of the input sensitivities when learning the intra-class covariance, where we observe that the input sensitivities are similar between the determinisitic MLP (MAP) and the Bayes MLP.

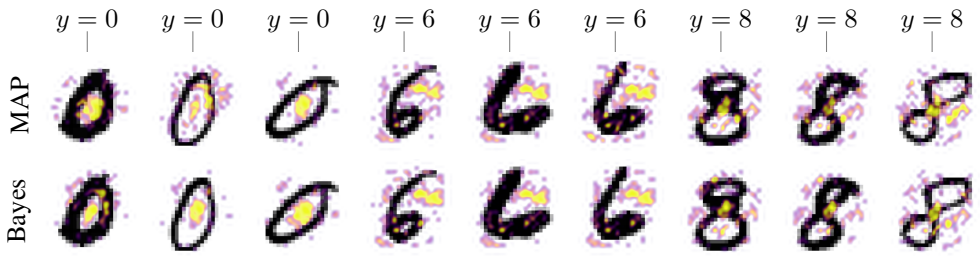


Figure 10: Pixel sensitivity of MLP classifiers trained on binary classification tasks $(0/6/8)$ for MNIST digits. The rows show the sensitivity of the MAP predictor to pixel perturbations, and the pixel sensitivity for a last-layer Laplace approximation. The predictive distribution is approximated analytically in both cases. We observe that the Bayesian model using a Laplace approximation has less spurious sensitivities to pixel perturbations indicating that it is more robust to input perturbations. The sensitivities are visualised in the range (0.5 1.0)

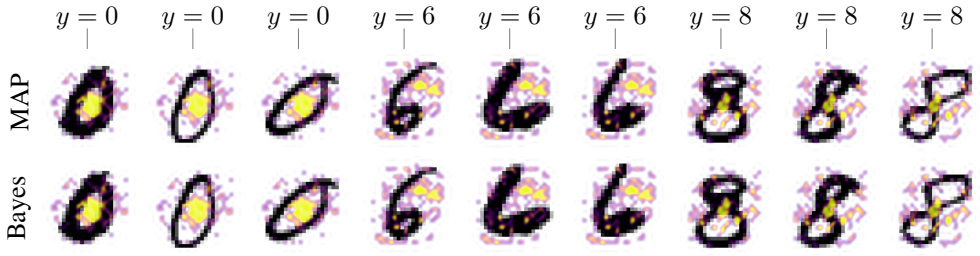


Figure 11: Per-class pixel sensitivities for MLP classifiers trained on classification tasks $(0/6/8)$ where we learn an intra-class covariance. The sensitivities are visualised in the range (0.5 1.0)

B.4 Effect of the Number of MC Samples on Performance

FIX (KRRC)

We investigate the influence of number of samples on performance.

On regression tasks with small scale neural network (two layer MLP), we run experiments with the range of $[100, 500, 1000, 5000, 10000, 50000]$. On classification tasks with medium scale neural network (four layer MLP), we run experiments with the range of $[100, 500, 1000, 5000, 10000, 25000]$. On classification tasks with large scale neural network (ViT-Base), we run experiments with the range of $[10, 20, \ldots, 100]$. The results are reported in Figs. 12 to 14. The number of samples we used to report results in the main paper is shown in the dashed line.

In Fig. 12 and Fig. 14, we observe that for regression on small scale network and classification on large scale network, the performance saturates when set the number of samples to 50 samples (classification with ViT-Base) or 1000 samples (regression with two layer MLP). In Fig. 13, the performance saturates with 1000 samples with LA. For MFVI the performance saturates with 5000 samples, as the improvement gained on NLPD is marginal (from 2.13 to 2.11) from 1000 samples to 5000 samples, in experiment we set the number of samples as 1000 for MFVI as well.

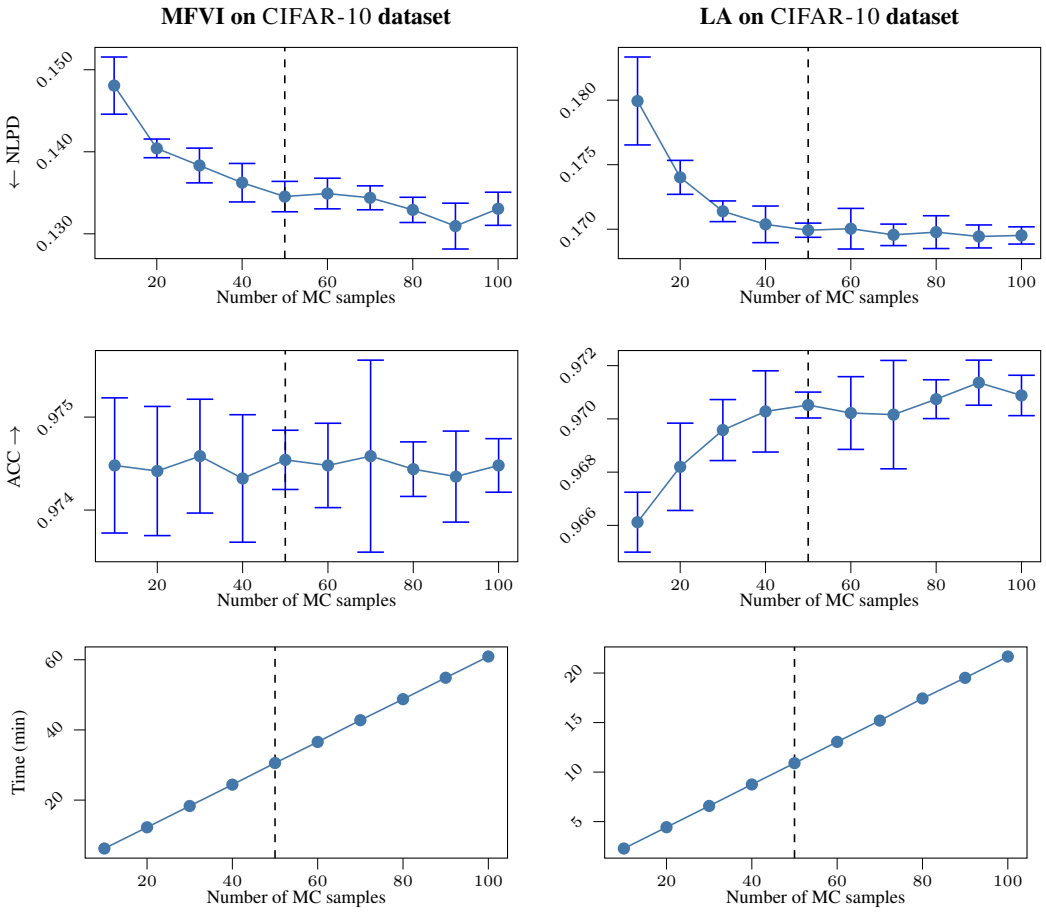


Figure 12: Effects of the number of MC samples on performance for LA and MFVI on CIFAR-10 with ViT-Base model. In the results reported in main paper, we set the number of MC samples to 50 (dashed line).

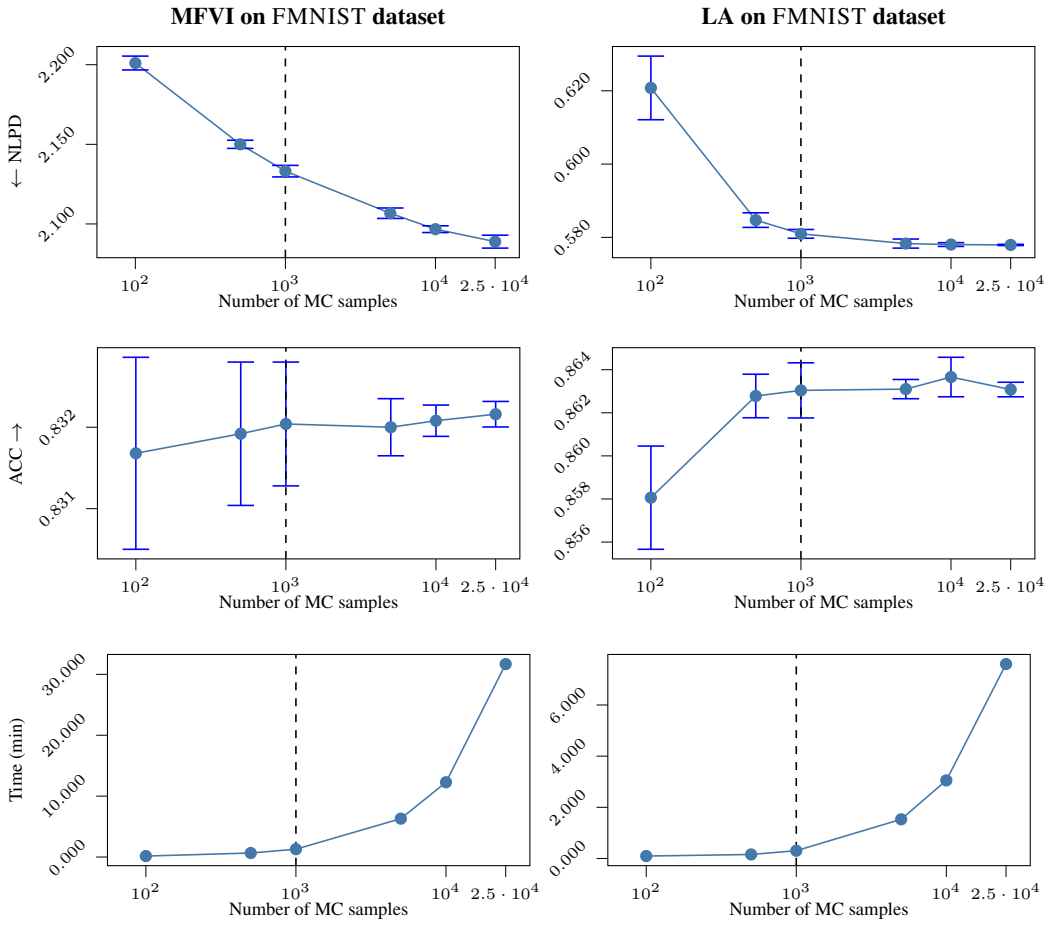


Figure 13: Effects of the number of MC samples on performance for LA and MFVI on FMNIST. In the results reported in main paper, we set the number of MC samples to 1000 (dashed line).

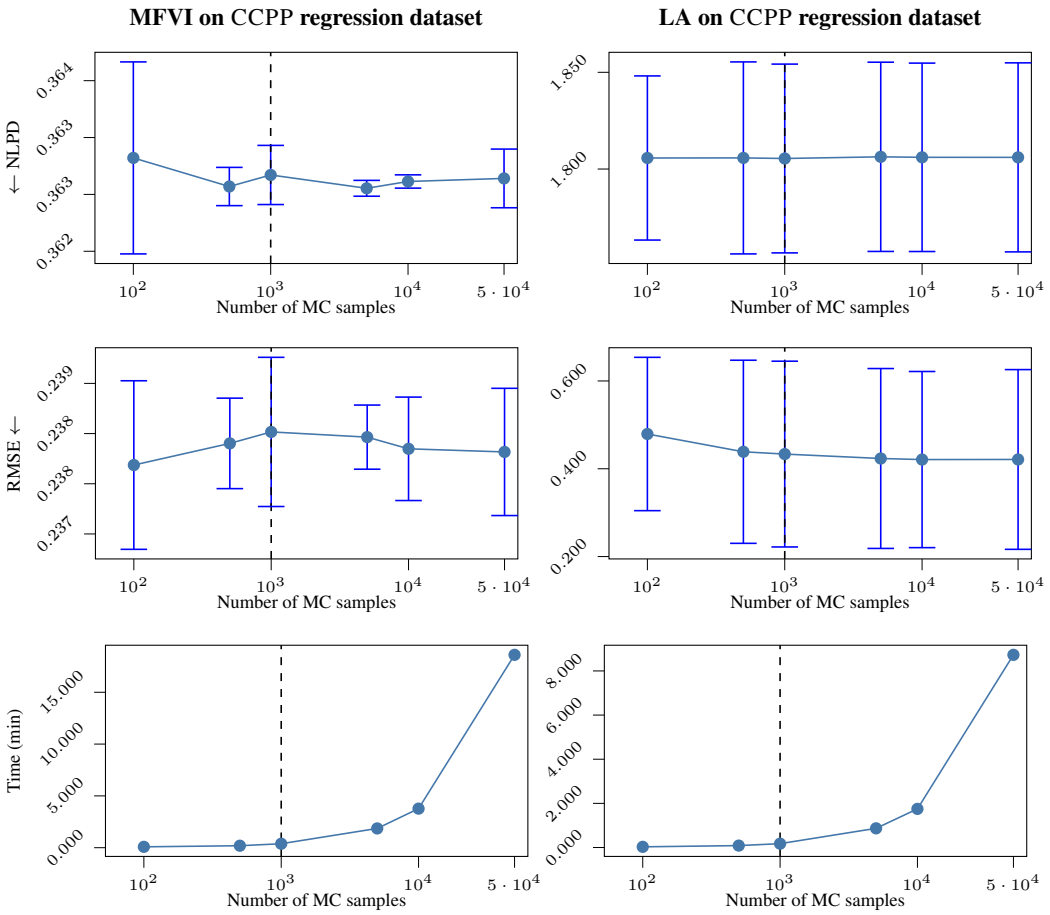


Figure 14: Effects of the number of MC samples on performance for LA and MFVI on regression tasks. In the results reported in main paper, we set the number of MC samples to 1000 (dashed line).

B.5 Estimating Degree of Local Linearity

FIX (tEnH)

We performed an additional experiment to assess the degree of local linearity of a trained MLP with ReLU activation functions. In particular, for trained MLP $f(\cdot)$, we are estimating the expected absolute error

$$\delta_{\text{Lin}} = \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \left[|f(\boldsymbol{z}(1 \pm \epsilon)) - f(\boldsymbol{z})(1 \pm \epsilon)| \right], \tag{90}$$

where $\epsilon \geq 0$ and δ_{Lin} is zero for any ϵ if $f(\cdot)$ is linear around each $\boldsymbol{z}$.

In our experiments we vary ϵ in the range of $\epsilon \in [1\text{e}{-6}, 1\text{e}{-5}, \ldots, 1]$ for a fully connected ReLU MLP with layers with sizes $[784, 128, 64, 10]$ trained on MNIST digits. After training, we removed the softmax operation on the last layer and measure the local linearisation error on the logits. We estimated the error on a random subset of 124 validation data points and estimated the range of the inputs and the function outputs on the same subset. The range of input values is 3.246 and the range of the function outputs varies between 153.072 and 291.168. Fig. 15 shows the results for each of the ten output dimensions scaled relative to their respective range. We observe that the trained ReLU MLP obtains low expected absolute error and behaves locally linear to a certain degree.

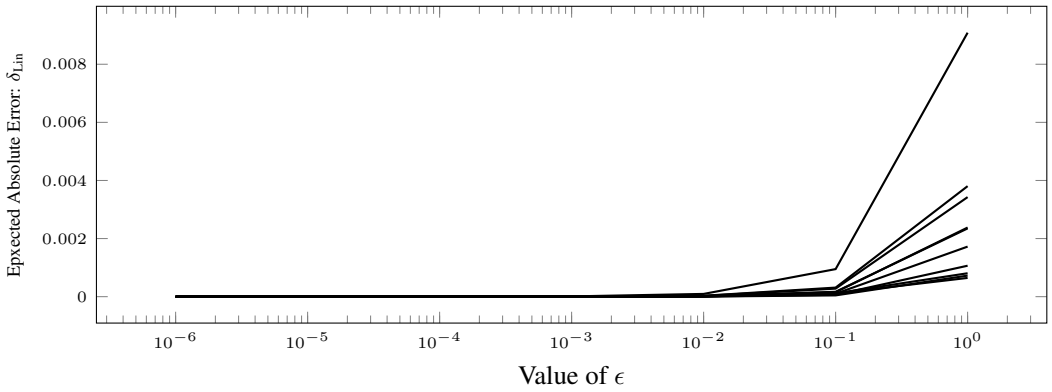


Figure 15: Estimated divergence from a locally linear function as a function of ϵ. Note that a value of zero means that the function behaves locally like a linear function.

B.6 Comparison of Value Covariance in Transformers

One way to improve efficiency in transformer is dropping the correlation between values, *i.e.*, drop the correlation $\mathbb{C}\text{ov}[v_k, v_l]$ given in Eq. (74). We compare the performance of both approximation and the results are given in Tables 12 to 14. Both approximation results in almost the same results for NLPD, ACC and ECE.

Table 12: Negative Log Predictive Density (NLPD) for ViT with posterior approximation n the attention layers. We compare only considering variance for value $\boldsymbol{V}$ and considering full covariance. For MFVI and LA, both approximation results in almost the same result.

	Mean Field Variational Inference		*Laplace Approximation*	
Dataset	Full Covariance	Only Variance	Full Covariance	Only Variance
CIFAR-10	0.088 ± 0.006	0.088 ± 0.006	0.086 ± 0.006	0.086 ± 0.006
CIFAR-100	0.468 ± 0.013	0.467 ± 0.012	0.456 ± 0.012	0.456 ± 0.012
DTD	1.007 ± 0.035	1.007 ± 0.035	1.068 ± 0.035	1.068 ± 0.035
RESISC	0.617 ± 0.019	0.616 ± 0.019	0.352 ± 0.012	0.352 ± 0.012
ImageNet-R	1.234 ± 0.052	1.233 ± 0.052	1.267 ± 0.043	1.267 ± 0.043

B.7 Runtime Experiment

We compared the runtime of our method against sampling (using the 'torch-laplace' library Daxberger et al. (2021a) for Laplace and the IVON Shen et al. (2024)) and the GLM implementation of the 'torch-laplace' library for diagonal posterior covariances. For our streamlined approach on ViT,

Table 13: Accuracy (ACC) for ViT with posterior approximation n the attention layers. We compare only considering variance for value $\boldsymbol{V}$ and considering full covariance. For MFVI and LA, both approximation results in almost the same result.

Dataset	*Mean Field Variational Inference*		*Laplace Approximation*	
	Full Covariance	Only Variance	Full Covariance	Only Variance
CIFAR-10	0.975 ± 0.002	0.975 ± 0.004	0.976 ± 0.002	0.976 ± 0.004
CIFAR-100	0.880 ± 0.003	0.880 ± 0.009	0.880 ± 0.003	0.880 ± 0.009
DTD	0.734 ± 0.010	0.734 ± 0.012	0.719 ± 0.010	0.719 ± 0.012
RESISC	0.867 ± 0.004	0.867 ± 0.009	0.892 ± 0.004	0.892 ± 0.008
IMAGENET-R	0.728 ± 0.012	0.728 ± 0.012	0.739 ± 0.012	0.739 ± 0.012

Table 14: Expected Calibration Error (ECE) for ViT with posterior approximation n the attention layers. We compare only considering variance for value $\boldsymbol{V}$ and considering full covariance. For MFVI and LA, both approximation results in almost the same result.

Dataset	*Mean Field Variational Inference*		*Laplace Approximation*	
	Full Covariance	Only Variance	Full Covariance	Only Variance
CIFAR-10	0.008	0.008	0.008	0.008
CIFAR-100	0.027	0.026	0.025	0.023
DTD	0.040	0.040	0.042	0.043
RESISC	0.016	0.016	0.017	0.020
IMAGENET-R	0.132	0.132	0.036	0.039

we assessed two cases: (i) propagating $\mathbb{C}\mathrm{ov}[v_k, v_l]$ covariance terms (*cf.*, Eq. (74)) through the transformer (+Cov), and (ii) ignoring $\mathbb{C}\mathrm{ov}[v_k, v_l]$ covariance terms. We used a pre-trained ViT base model on CIFAR-10 and a pre-trained MLP on MNIST. For comparision we also list the runtime for a single forward pass. For this, we ran experiments on an NVIDIA H100 80GB GPU for 400 data points, batchsize of one, and for each data point we repeated the measurement ten times. To account for code compilation overheads, we droped the first run on each data point. We report the mean and standard deviation of the runtime (in milliseconds) over the remaining nine runs and all 400 data points. The results are shown in Table 15. For ViT, we can see that our method without $\mathbb{C}\mathrm{ov}[v_k, v_l]$ covariance terms has a comparable runtime to a single forward pass in the deterministic model. When additionally accounting for covariance terms, we obtain slight speed improvements over GLM but overall comparable performance. Note that our implemention is not optimised for speed and larger speedups may be obtained by optimising the code. For MLP, we obtain slight speed improvements over LA GLM but overall comparable performance.

Model	Methods	AVG. RUNTIME ($\pm$ STD) $\downarrow$
ViT	MAP	$3.737_{\pm 0.093}$
	LA Sampling	$190.806_{\pm 0.137}$
	LA GLM	$17.191_{\pm 0.734}$
	MFVI Sampling	$207.854_{\pm 0.307}$
	Ours (+ Cov)	$14.728_{\pm 0.144}$
	Ours	$4.350_{\pm 0.079}$
MLP	MAP	$0.069_{\pm 0.001}$
	LA Sampling	$98.584_{\pm 3.737}$
	LA GLM	$1.656_{\pm 0.049}$
	MFVI Sampling	$190.302_{\pm 0.466}$
	Ours	$0.542_{\pm 0.073}$

Table 15: Wallclock times for ViT base on CIFAR-10 and MLP on MNIST in milliseconds.