Formatting Instructions for TMLR Journal Submissions

Anonymous authors
Paper under double-blind review

Abstract

Learning the posterior distribution of a deep neural network is a difficult task. Even when using a Gaussian approximation, one faces the daunting challenge of estimating the $D \times D$ posterior covariance matrix, where D, the number of model parameters, can reach billions by today's standards. Some simplification has to be made, and one option is to use a low-rank plus diagonal approximation. One particular Gaussian distribution with a covariance matrix of this form is the factor analysis (FA) model. This paper introduces a novel approach to learning a FA posterior of a neural network via variational inference (VI). The algorithm - aptly named VIFA due to its use of VI and FA - is model-agnostic and can be readily applied to any type of neural network architecture with no extra effort. Crucially, the implementation scales to high-dimensional deep neural networks. Experiments demonstrate its effectiveness in learning the posterior and making predictions with uncertainty estimates, while keeping the computational overhead small compared to standard neural network training.

1 Introduction

In recent years, deep learning (Goodfellow et al., 2016b) has come to dominate many areas of machine learning. Deep learning models continue to achieve state of the art results in several domains, such as CoAtNet-7 (Dai et al., 2021) for image classification and Megatron-LM (Shoeybi et al., 2019) for language modelling. One reason for their success is the availability of huge training sets, such as the 300 millions images in JFT-300M (Sun et al., 2017) or the 100 million tokens in WikiText-103 (Merity et al., 2016). The number of training examples, however, pales in comparison to the number of learnable parameters in some deep models. CoAtNet-7, for example, has 2.4 billion parameters, whereas Megatron-LM has a whopping 8.3 billion. This is a common feature of deep learning: the number of model parameters is often greater than the number of training examples. In this scenario, the model is severely *underspecified* by the data and many different settings of the parameters are able to explain the training set equally well. In some domains, choosing a single setting of the parameters and predicting a point estimate for each test example is satisfactory. However, for other critical applications, some measure of the uncertainty in the prediction is also required.

This is where Bayesian deep learning comes in. Instead of choosing a single parameter vector $\boldsymbol{\theta} \in \mathbb{R}^D$ for the neural network, predictions are made using *all* possible parameter vectors weighted by their posterior probabilities, given the training data $\mathcal{D}$. Formally, the posterior predictive distribution of the output y given the input $\boldsymbol{x}$ is

$$p(y|\boldsymbol{x}, \mathcal{D}) = \int p(y|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}. \tag{1}$$

In practice, computing this integral is intractable due to the dimensionality and non-linearities of a deep neural network. However, the integral can be approximated by a *Bayesian model average*,

$$p(y|\boldsymbol{x}, \mathcal{D}) \approx \frac{1}{L} \sum_{l=1}^{L} p(y|\boldsymbol{x}, \boldsymbol{\theta}_l), \quad \boldsymbol{\theta}_l \sim p(\boldsymbol{\theta}|\mathcal{D}). \tag{2}$$

Note that this requires samples from the *posterior*, $p(\boldsymbol{\theta}|\mathcal{D})$, which is defined as

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}, \tag{3}$$

where $p(\mathcal{D}|\boldsymbol{\theta})$ is the *likelihood* that $\boldsymbol{\theta}$ generated $\mathcal{D}$, $p(\boldsymbol{\theta})$ is the *prior* and $p(\mathcal{D})$ is the *marginal likelihood*. Unfortunately, this calculation is also intractable due to the high-dimensional integral

$$p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}. \tag{4}$$

A common alternative is to approximate $p(\boldsymbol{\theta}|\mathcal{D})$ with a Gaussian distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^D$ and covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{D\times D}$, denoted by $\mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$. The key practical challenge is then finding a way to approximate $\boldsymbol{\Sigma}$, since the full $D \times D$ covariance matrix will not fit into memory for large D.

One of the simplest approaches is to use a diagonal - or *mean-field* - approximation (Blundell et al., 2015; Graves, 2011; Hernández-Lobato & Adams, 2015; Khan et al., 2018; Ranganath et al., 2014). This is convenient in the sense that the covariance matrix is completely specified by D parameters, but it does not allow for any posterior correlations between different elements of $\boldsymbol{\theta}$. A more flexible approach is a low-rank plus diagonal approximation, $\boldsymbol{F}\boldsymbol{F}^T + \Psi$, where $\boldsymbol{F} \in \mathbb{R}^{D\times K}$ and Ψ is a $D \times D$ diagonal matrix. This preserves some of the off-diagonal structure of the covariance matrix and is practically feasible for $K \ll D$. Two recent methods which adopt this approach are SWAG (Maddox et al., 2019) and SLANG (Mishkin et al., 2018). While both algorithms have achieved promising results, they also have fundamental limitations. In the case of SWAG, it does not make use of all the available data to fit the low-rank plus diagonal covariance matrix, whereas the current implementation of SLANG only supports fully-connected neural networks and would require non-trivial modifications for other types of architectures.

The work in this paper is based on the low-rank plus diagonal factor analysis (FA) model (Barber, 2012), which can help to overcome these limitations. The main contribution is a novel variational inference (VI) algorithm called VIFA which can be applied to any neural network architecture to approximate its posterior with a FA distribution.

TODO: add overview if different sections.

2 Background

This section presents the background necessary to understanding the rest of the paper. Bayesian linear regression, whose posterior can be computed in closed-form, provides a nice testbed for our algorithm. Moreover, it can be viewed as a simple neural network with a single affine layer from input to output. Factor analysis is then introduced, as this is the basis of the main contributions in this paper. Finally, variational inference is descrived as a pre-requisite for algorithms which use this approach for learn the posterior distribution of a neural network.

2.1 Bayesian Linear Regression

A linear regression model is a mapping from vector inputs $\boldsymbol{x} \in \mathbb{R}^D$ to scalar outputs $y \in \mathbb{R}$ via an affine transformation. Given a set of observed input-output pairs, $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, it is assumed that each output is generated according to

$$y_n = \boldsymbol{\theta}^\top \boldsymbol{x}_n + \epsilon_n \tag{5}$$

for some unknown $\boldsymbol{\theta} \in \mathbb{R}^D$. The underlying signal, $f(\boldsymbol{x}_n) = \boldsymbol{\theta}^\top \boldsymbol{x}_n$, is corrupted by additive noise, $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$, for some $\sigma > 0$ (Barber, 2012). The model is often written with an explicit bias term, but this can be absorbed into $\boldsymbol{\theta}$ by adding a constant of one to the input, leading to the expression in Equation (5). Due to the additive noise, each y_n is a random variable, conditioned on $\boldsymbol{x}_n$ and $\boldsymbol{\theta}$. Since ϵ_n is Gaussian distributed, the conditional pdf of y_n is

$$p(y_n|\boldsymbol{\theta}, \boldsymbol{x}_n) = \mathcal{N}\big(\boldsymbol{\theta}^\top \boldsymbol{x}_n, \sigma^2\big) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{1}{2\sigma^2}\big(y_n - \boldsymbol{\theta}^\top \boldsymbol{x}_n\big)^2\Big). \tag{6}$$

Assuming that the observations in $\mathcal{D}$ are independent and identically distributed (iid), the log-likelihood of $\boldsymbol{\theta}$ having generated the data is

$$\begin{aligned}
\log p(\mathcal{D}|\boldsymbol{\theta}) &= \sum_{n=1}^{N} \big[\log p(y_n|\boldsymbol{\theta}, \boldsymbol{x}_n) + \log p(\boldsymbol{x}_n) \big] \\
&= \sum_{n=1}^{N} \Big[-\frac{1}{2} \log 2\pi\sigma^2 - \frac{1}{2\sigma^2} \big(y_n - \boldsymbol{\theta}^\top \boldsymbol{x}_n\big)^2 \Big] + \sum_{n=1}^{N} \log p(\boldsymbol{x}_n) \\
&= -\frac{1}{2\sigma^2} \sum_{n=1}^{N} \big(y_n - \boldsymbol{\theta}^\top \boldsymbol{x}_n\big)^2 - \frac{N}{2} \log \sigma^2 - \frac{N}{2} \log 2\pi + \sum_{n=1}^{N} \log p(\boldsymbol{x}_n) \\
&= -\frac{\beta}{2} \sum_{n=1}^{N} \big(y_n - \boldsymbol{\theta}^\top \boldsymbol{x}_n\big)^2 + \frac{N}{2} \log \beta + \text{constant},
\end{aligned} \tag{7}$$

where $\beta = \frac{1}{\sigma^2}$ is the *noise precision* (Barber, 2012). In Bayesian linear regression, a prior distribution for $\boldsymbol{\theta}$ is also specified. A common choice is

$$p(\boldsymbol{\theta}) = \mathcal{N}\big(\mathbf{0}, \alpha^{-1}\boldsymbol{I}\big) = \Big(\frac{\alpha}{2\pi}\Big)^{\frac{D}{2}} \exp\Big(-\frac{\alpha}{2} \boldsymbol{\theta}^\top \boldsymbol{\theta} \Big) \tag{8}$$

for some $\alpha > 0$, which is a hyperparameter known as the *prior precision* (Barber, 2012). Given the log-likelihood and the prior, it turns out that the posterior distribution of $\boldsymbol{\theta}$ is Gaussian and can be computed in closed-form. Formally,

$$p(\boldsymbol{\theta}|\mathcal{D}) = \mathcal{N}(\boldsymbol{A}^{-1}\boldsymbol{b}, \boldsymbol{A}^{-1}), \tag{9}$$

where

$$\boldsymbol{A} = \alpha\boldsymbol{I} + \beta \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top \quad \text{and} \quad \boldsymbol{b} = \beta \sum_{n=1}^{N} y_n \boldsymbol{x}_n. \tag{10}$$

See Appendix A.1 for a full derivation.

2.2 Factor Analysis

Factor analysis (FA) is a latent variable model which generates observations $\boldsymbol{\theta} \in \mathbb{R}^D$ as follows. First, a latent vector $\boldsymbol{h} \in \mathbb{R}^K$, for some $K < D$, is sampled from $p(\boldsymbol{h}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Next, $\boldsymbol{h}$ is transformed onto a K-dimensional linear subspace of $\mathbb{R}^D$ by left-multiplying it by a *factor loading* matrix $\boldsymbol{F} \in \mathbb{R}^{D \times K}$. The origin of this subspace is then shifted by adding a bias term $\boldsymbol{c} \in \mathbb{R}^D$. Finally, the data is perturbed by adding some zero mean Gaussian noise $\epsilon \in \mathbb{R}^D$ sampled from $\mathcal{N}(\mathbf{0}, \Psi)$, where Ψ is a $D \times D$ diagonal matrix (Barber, 2012). Putting all this together, an observation $\boldsymbol{\theta} \in \mathbb{R}^D$ is generated according to

$$\boldsymbol{\theta} = \boldsymbol{F}\boldsymbol{h} + \boldsymbol{c} + \epsilon. \tag{11}$$

In the context of this paper, an observation $\boldsymbol{\theta}$ is the parameter vector of a neural network.

It follows from Equation (11) that, given $\boldsymbol{h}$, the observations $\boldsymbol{\theta}$ are Gaussian distributed with mean $\boldsymbol{F}\boldsymbol{h} + \boldsymbol{c}$ and covariance Ψ (Barber, 2012). Formally,

$$p(\boldsymbol{\theta}|\boldsymbol{h}) = \mathcal{N}\big(\boldsymbol{F}\boldsymbol{h} + \boldsymbol{c}, \Psi\big) = \frac{1}{\sqrt{(2\pi)^D |\Psi|}} \exp\Big(-\frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{F}\boldsymbol{h} - \boldsymbol{c})^\top \Psi^{-1} (\boldsymbol{\theta} - \boldsymbol{F}\boldsymbol{h} - \boldsymbol{c}) \Big), \tag{12}$$

where $|\Psi|$ is the *determinant* of Ψ. From (Barber, 2012), integrating $p(\boldsymbol{\theta}|\boldsymbol{h})$ over $\boldsymbol{h}$ gives the marginal distribution

$$p(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{c}, \boldsymbol{F}\boldsymbol{F}^\top + \Psi\big). \tag{13}$$

2.3 Variational Inference

For linear regression, computing the posterior in Equation (9) is possible because the log-likelihood, $p(\mathcal{D}|\boldsymbol{\theta})$, is a quadratic expression of $\boldsymbol{\theta}$. However, when $\boldsymbol{\theta}$ is the parameter vector of a neural network with even a single non-linear hidden layer, the log-likelihood is not so simple and the posterior cannot be written in closed-form. Moreover, numerical methods which evaluate the marginal likelihood, $p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$, are intractable for high-dimensional $\boldsymbol{\theta}$. An alternative strategy is to approximate the posterior with a Gaussian distribution, $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where the parameters $\boldsymbol{\mu}$ and Σ are chosen to minimise some measure of dissimilarity between $q(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta}|\mathcal{D})$. Because $q(\boldsymbol{\theta})$ is essentially a variable in an optimisation procedure, this approach is known as *variational inference* (VI) (Barber, 2012). A common measure of dissimilarity between two distributions is the Kullback-Leibler (KL) divergence Barber (2012), which in this case is

$$\begin{aligned} D_{KL}[q(\boldsymbol{\theta})\|p(\boldsymbol{\theta}|\mathcal{D})] &= \mathbb{E}_{q(\boldsymbol{\theta})}\Big[\log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathcal{D})}\Big] \\ &= \mathbb{E}_{q(\boldsymbol{\theta})}[\log q(\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) - \log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\mathcal{D})]. \end{aligned} \tag{14}$$

Since $p(\mathcal{D})$ is constant, it can be ignored and the solution can be obtained by solving

$$\min_{\boldsymbol{\mu},\Sigma} \big[\mathbb{E}_{q(\boldsymbol{\theta})}[\log q(\boldsymbol{\theta})] - \mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\boldsymbol{\theta})] - \mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})]\big]. \tag{15}$$

One way to tackle this is via an iterative gradient algorithm. Depending on the structure of Σ and the form of the prior $p(\boldsymbol{\theta})$, it may be possible to compute the partial derivatives $\nabla_{\boldsymbol{\mu},\Sigma}\mathbb{E}_{q(\boldsymbol{\theta})}[\log q(\boldsymbol{\theta})]$ and $\nabla_{\boldsymbol{\mu},\Sigma}\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\boldsymbol{\theta})]$ exactly (Kingma & Welling, 2013). However, when the likelihood is parameterised by a neural network, is not possible to obtain $\nabla_{\boldsymbol{\mu},\Sigma}\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})]$ analytically. Alternatively, the gradient can be approximated by Monte Carlo estimates of the form

$$\nabla_{\boldsymbol{\mu},\Sigma}\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] \approx \nabla_{\boldsymbol{\mu},\Sigma}\left(\frac{1}{L}\sum_{l=1}^{L}\left(\frac{N}{M}\sum_{(\boldsymbol{x},y)\in\mathcal{B}_l}\log p(y|\boldsymbol{x},\boldsymbol{\theta}_l)\right)\right), \tag{16}$$

where L is the Monte Carlo average size, each $\boldsymbol{\theta}_l \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ and each $\mathcal{B}_l$ is a mini-batch of M training examples sampled from the full dataset $\mathcal{D}$ of size N. Normally it would not be possible to compute these partial derivatives with respect to $\boldsymbol{\mu}$ and Σ, given that they only take part via the sampling operation. However, this can in fact be done by using the *re-parameterisation trick* (Goodfellow et al., 2016a). Instead of sampling $\boldsymbol{\theta}_l$ from $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ directly, a random variable $\boldsymbol{z}_l$ is sampled from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and then transformed to $\boldsymbol{\theta}_l$ via a deterministic operation involving $\boldsymbol{\mu}$ and Σ. This simple trick means that Equation (16) can be evaluated and used in conjunction with SGD or any of its variants to optimise the objective in Equation (15).

3 Variational Inference with a Factor Analysis Posterior (VIFA)

In Section 2.3, a general VI algorithm was outlined for learning an approximate Gaussian posterior of the parameter vector of a neural network. Suppose that the approximate posterior is a FA model. That is, $q(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{c}, \boldsymbol{F}\boldsymbol{F}^\top + \Psi\big)$ for some $\boldsymbol{c} \in \mathbb{R}^D$, $\boldsymbol{F} \in \mathbb{R}^{D\times K}$ and $D \times D$ diagonal matrix Ψ. Then the gradient of the VI objective in Equation (15) becomes

$$\nabla_{\boldsymbol{c},\boldsymbol{F},\Psi}\mathbb{E}_{q(\boldsymbol{\theta})}[\log q(\boldsymbol{\theta})] - \nabla_{\boldsymbol{c},\boldsymbol{F},\Psi}\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\boldsymbol{\theta})] - \nabla_{\boldsymbol{c},\boldsymbol{F},\Psi}\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})], \tag{17}$$

where the expectation is taken over samples $\boldsymbol{\theta} \sim \mathcal{N}\big(\boldsymbol{c}, \boldsymbol{F}\boldsymbol{F}^\top + \Psi\big)$. These derivatives are derived in the following sections.

3.1 Partial Derivatives of the Variational Distribution

The logarithm of the pdf of the variational distribution is

$$\log q(\boldsymbol{\theta}) = -\frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{c})^\top(\boldsymbol{F}\boldsymbol{F}^\top + \Psi)^{-1}(\boldsymbol{\theta}-\boldsymbol{c}) - \frac{1}{2}\log|\boldsymbol{F}\boldsymbol{F}^\top + \Psi| - \frac{D}{2}\log 2\pi. \tag{18}$$

It is given in (Petersen & Pedersen, 2012) that, for any symmetric matrix $\boldsymbol{A}$ and stochastic vector $\boldsymbol{x}$ with mean $\boldsymbol{m}$ and covariance $\boldsymbol{M}$,

$$\mathbb{E}[\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}] = \operatorname{Tr}(\boldsymbol{A}\boldsymbol{M}) + \boldsymbol{m}^\top \boldsymbol{A}\boldsymbol{m}. \tag{19}$$

Setting $\boldsymbol{S} = \boldsymbol{F}\boldsymbol{F}^\top + \Psi$ and noting that $\boldsymbol{\theta} - \boldsymbol{c}$ has mean $\mathbf{0}$ and covariance $\boldsymbol{S}$ over $q(\boldsymbol{\theta})$, it follows that

$$\begin{aligned}
\mathbb{E}_{q(\boldsymbol{\theta})}[\log q(\boldsymbol{\theta})] &= -\frac{1}{2}\operatorname{Tr}\left(\boldsymbol{S}^{-1}\boldsymbol{S}\right) - \frac{1}{2}\mathbf{0}\boldsymbol{S}^{-1}\mathbf{0} - \frac{1}{2}\log|\boldsymbol{S}| - \frac{D}{2}\log 2\pi \\
&= -\frac{1}{2}\operatorname{Tr}(\boldsymbol{I}) - \frac{1}{2}\log|\boldsymbol{S}| - \frac{D}{2}\log 2\pi \\
&= -\frac{D}{2} - \frac{1}{2}\log|\boldsymbol{S}| - \frac{D}{2}\log 2\pi.
\end{aligned} \tag{20}$$

Hence,

$$\nabla_{\boldsymbol{c}}\mathbb{E}_{q(\boldsymbol{\theta})}[\log q(\boldsymbol{\theta})] = \mathbf{0} \tag{21}$$

and, using the identity $\nabla_{\boldsymbol{X}}\log|\boldsymbol{X}| = \boldsymbol{X}^{-\top}$ from (Petersen & Pedersen, 2012) and the fact that $\boldsymbol{S}$ is symmetric,

$$\nabla_{\boldsymbol{S}}\mathbb{E}_{q(\boldsymbol{\theta})}[\log q(\boldsymbol{\theta})] = -\frac{1}{2}\boldsymbol{S}^{-1}. \tag{22}$$

It then follows from the chain rule of calculus that

$$\begin{aligned}
\nabla_{\boldsymbol{F}}\mathbb{E}_{q(\boldsymbol{\theta})}[\log q(\boldsymbol{\theta})] &= -\frac{1}{2}\boldsymbol{S}^{-1}(2\boldsymbol{F}) \\
&= -\boldsymbol{S}^{-1}\boldsymbol{F}
\end{aligned} \tag{23}$$

and

$$\begin{aligned}
\nabla_{\Psi}\mathbb{E}_{q(\boldsymbol{\theta})}[\log q(\boldsymbol{\theta})] &= -\frac{1}{2}\operatorname{diag}\left(\operatorname{diag}\left(\boldsymbol{S}^{-1}\boldsymbol{I}\right)\right) \\
&= -\frac{1}{2}\operatorname{diag}\left(\operatorname{diag}\left(\boldsymbol{S}^{-1}\right)\right).
\end{aligned} \tag{24}$$

3.2 Partial Derivatives of the Prior

Let the prior be $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \alpha^{-1}\boldsymbol{I})$ for some precision $\alpha > 0$. Then

$$\begin{aligned}
\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\boldsymbol{\theta})] &= \mathbb{E}_{q(\boldsymbol{\theta})}\Big[-\frac{1}{2}\boldsymbol{\theta}^\top \alpha\boldsymbol{I}\boldsymbol{\theta} - \frac{1}{2}\log|\alpha^{-1}\boldsymbol{I}| - \frac{D}{2}\log 2\pi\Big] \\
&= -\frac{\alpha}{2}\mathbb{E}_{q(\boldsymbol{\theta})}\Big[\boldsymbol{\theta}^\top \boldsymbol{I}\boldsymbol{\theta}\Big] + \text{constant} \\
&= -\frac{\alpha}{2}\big(\operatorname{Tr}(\boldsymbol{F}\boldsymbol{F}^\top + \Psi) + \boldsymbol{c}^\top\boldsymbol{c}\big) + \text{constant},
\end{aligned} \tag{25}$$

where the last line follows from Equation (19). Hence,

$$\nabla_{\boldsymbol{c}}\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\boldsymbol{\theta})] = -\alpha\boldsymbol{c}, \tag{26}$$

$$\nabla_{\boldsymbol{F}}\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\boldsymbol{\theta})] = -\alpha\boldsymbol{F}, \tag{27}$$

$$\nabla_{\Psi}\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\boldsymbol{\theta})] = -\frac{\alpha}{2}\boldsymbol{I}, \tag{28}$$

which follow from the matrix calculus identities in (Petersen & Pedersen, 2012).

3.3 Partial Derivatives of the Likelihood

Let the log-likelihood, $\log p(\mathcal{D}|\boldsymbol{\theta})$, be parameterised by a neural network with weights $\boldsymbol{\theta}$. Due to the functional form of a non-linear neural network, $\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})]$ cannot be computed in closed-form. Recall from Section 2.3 that a Monte Carlo estimate can instead be constructed from mini-batches of training data. Formally,

$$\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] \approx \frac{1}{L}\sum_{l=1}^{L}\left(\frac{N}{M}\sum_{(\boldsymbol{x},y)\in\mathcal{B}_l}\log p(y|\boldsymbol{x},\boldsymbol{\theta}_l)\right), \tag{29}$$

where L is the Monte Carlo average size, each $\mathcal{B}_l$ is a mini-batch of M training examples sampled from the full dataset $\mathcal{D}$ of size N and each $\boldsymbol{\theta}_l \sim q(\boldsymbol{\theta})$. In order to obtain the partial derivatives of this expression with respect to $\boldsymbol{c}$, $\boldsymbol{F}$ and Ψ, the re-parameterisation trick (Goodfellow et al., 2016a) can be used. First note that, by re-writing Equation (11), sampling $\boldsymbol{\theta}_l \sim q(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{c}, \boldsymbol{F}\boldsymbol{F}^\top + \Psi\big)$ is equivalent to setting

$$\boldsymbol{\theta}_l = \boldsymbol{F}\boldsymbol{h}_l + \boldsymbol{c} + \Psi^{1/2}\boldsymbol{z}_l \tag{30}$$

from some $\boldsymbol{h}_l \sim p(\boldsymbol{h}) = \mathcal{N}(\boldsymbol{0}^K, \boldsymbol{I}^{K\times K})$ and $\boldsymbol{z}_l \sim p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}^D, \boldsymbol{I}^{D\times D})$, where the exponent is applied to Ψ element-wise. It then follows that

$$\begin{aligned}\nabla_{\boldsymbol{c},\boldsymbol{F},\Psi}\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] &= \nabla_{\boldsymbol{c},\boldsymbol{F},\Psi}\mathbb{E}_{p(\boldsymbol{h})p(\boldsymbol{z})}[\log p(\mathcal{D}|\boldsymbol{\theta})] \\ &\approx -\frac{1}{L}\sum_{l=1}^{L}\left(N\cdot\nabla_{\boldsymbol{c},\boldsymbol{F},\Psi}\left(-\frac{1}{M}\sum_{(\boldsymbol{x},y)\in\mathcal{B}_l}\log p(y|\boldsymbol{x},\boldsymbol{\theta}_l)\right)\right).\end{aligned} \tag{31}$$

Since $\boldsymbol{\theta}_l = \boldsymbol{F}\boldsymbol{h}_l + \boldsymbol{c} + \Psi^{1/2}\boldsymbol{z}_l$, the chain rule of calculus can be used to obtain these partial derivatives. Formally,

$$\nabla_{\boldsymbol{c}}\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] \approx -\frac{1}{L}\sum_{l=1}^{L}N\boldsymbol{g}_{\boldsymbol{\theta}_l}, \tag{32}$$

$$\nabla_{\boldsymbol{F}}\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] \approx -\frac{1}{L}\sum_{l=1}^{L}N\boldsymbol{g}_{\boldsymbol{\theta}_l}\boldsymbol{h}_l^\top, \tag{33}$$

$$\nabla_{\Psi}\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] \approx \operatorname{diag}\left(\operatorname{diag}\left(-\frac{1}{L}\sum_{l=1}^{L}\frac{N}{2}\boldsymbol{g}_{\boldsymbol{\theta}_l}\boldsymbol{z}_l^\top\Psi^{-1/2}\right)\right), \tag{34}$$

where

$$\boldsymbol{g}_{\boldsymbol{\theta}_l} = \nabla_{\boldsymbol{\theta}_l}\left(-\frac{1}{M}\sum_{(\boldsymbol{x},y)\in\mathcal{B}_l}\log p(y|\boldsymbol{x},\boldsymbol{\theta}_l)\right). \tag{35}$$

Note that $\boldsymbol{g}_{\boldsymbol{\theta}_l}$ is the gradient of the average negative log-likelihood of $\boldsymbol{\theta}_l$, where the average is computed over the training examples in the mini-batch $\mathcal{B}_l$. This is convenient, since this quantity can be computed efficiently using the back-propagation algorithm (Rumelhart et al., 1986). Indeed, deep learning frameworks such as PyTorch (Paszke et al., 2019) compute $\boldsymbol{g}_{\boldsymbol{\theta}_l}$ as standard.

3.4 Practical Implementation

In order to compute the partial derivatives in Equations (23) and (24), the inverse of $\boldsymbol{S} = \boldsymbol{F}\boldsymbol{F}^\top + \Psi \in \mathbb{R}^{D\times D}$ is required. Using the Woodbury identity Petersen & Pedersen (2012),

$$\begin{aligned}(\boldsymbol{F}\boldsymbol{F}^\top + \Psi)^{-1} &= \Psi^{-1} - \Psi^{-1}\boldsymbol{F}(\boldsymbol{I} + \boldsymbol{F}^\top\Psi^{-1}\boldsymbol{F})^{-1}\boldsymbol{F}^\top\Psi^{-1} \\ &= \Psi^{-1} - (\psi^{-1}\odot\boldsymbol{F})\big(\boldsymbol{I} + \boldsymbol{F}^\top(\psi^{-1}\odot\boldsymbol{F})\big)^{-1}(\psi^{-1}\odot\boldsymbol{F})^\top \\ &= \Psi^{-1} - \boldsymbol{A}(\boldsymbol{I} + \boldsymbol{F}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top,\end{aligned} \tag{36}$$

where $\psi = \operatorname{diag}(\Psi)$ and $\boldsymbol{A} = \psi^{-1} \odot \boldsymbol{F}$ (with broadcasting). It follows that

$$\begin{aligned}
\nabla_{\boldsymbol{F}} \mathbb{E}_{q(\boldsymbol{\theta})}[\log q(\boldsymbol{\theta})] &= -(\boldsymbol{F}\boldsymbol{F}^\top + \Psi)^{-1}\boldsymbol{F} \\
&= -\Psi^{-1}\boldsymbol{F} + \boldsymbol{A}(\boldsymbol{I} + \boldsymbol{F}^\top\boldsymbol{A})^{-1}(\boldsymbol{A}^\top\boldsymbol{F}) \\
&= -\boldsymbol{A} + \boldsymbol{A}(\boldsymbol{I} + \boldsymbol{F}^\top\boldsymbol{A})^{-1}(\boldsymbol{A}^\top\boldsymbol{F}) \\
&= -\boldsymbol{A} + \boldsymbol{A}(\boldsymbol{I} + \boldsymbol{B})^{-1}\boldsymbol{B}^\top \\
&= -\boldsymbol{A} + \boldsymbol{C}\boldsymbol{B}^\top,
\end{aligned} \tag{37}$$

where $\boldsymbol{B} = \boldsymbol{F}^\top\boldsymbol{A}$ and $\boldsymbol{C} = \boldsymbol{A}(\boldsymbol{I} + \boldsymbol{B})^{-1}$. Also,

$$\begin{aligned}
\nabla_{\psi} \mathbb{E}_{q(\boldsymbol{\theta})}[\log q(\boldsymbol{\theta})] &= -\frac{1}{2}\operatorname{diag}\left((\boldsymbol{F}\boldsymbol{F}^\top + \Psi)^{-1}\right) \\
&= -\frac{1}{2}\operatorname{diag}\left(\Psi^{-1} - \boldsymbol{A}(\boldsymbol{I} + \boldsymbol{F}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\right) \\
&= -\frac{1}{2}\psi^{-1} + \frac{1}{2}\operatorname{diag}\left(\boldsymbol{C}\boldsymbol{A}^\top\right) \\
&= -\frac{1}{2}\psi^{-1} + \frac{1}{2}\operatorname{sum}\big(\boldsymbol{C} \odot \boldsymbol{A},\ \dim = 1\big).
\end{aligned} \tag{38}$$

Note that these partial derivatives involve no explicit use of $D \times D$ matrices. All other references to $D \times D$ matrices can be removed by writing the partial derivatives in Equations (28) and (34) with respect to ψ rather than Ψ. Formally,

$$\nabla_{\psi} \mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\boldsymbol{\theta})] = -\frac{\alpha}{2}\mathbf{1}^D, \tag{39}$$

$$\nabla_{\psi} \mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] \approx -\frac{1}{L}\sum_{l=1}^{L}\frac{N}{2}\boldsymbol{g}_{\boldsymbol{\theta}_l} \odot \left(\psi^{-1/2} \odot \boldsymbol{z}_l\right). \tag{40}$$

Finally, the re-parameterisation $\psi = \exp\gamma$ can be used to ensure that the variances remain positive. Since $\nabla_\gamma \psi = \psi$, it follows from the chain rule that the corresponding partial derivatives with respect to γ are

$$\nabla_{\gamma} \mathbb{E}_{q(\boldsymbol{\theta})}[\log q(\boldsymbol{\theta})] = -\frac{1}{2} + \frac{1}{2}\operatorname{sum}\big(\boldsymbol{C} \odot \boldsymbol{A},\ \dim = 1\big) \odot \psi, \tag{41}$$

$$\nabla_{\gamma} \mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\boldsymbol{\theta})] = -\frac{\alpha}{2}\psi, \tag{42}$$

$$\nabla_{\gamma} \mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] \approx -\frac{1}{L}\sum_{l=1}^{L}\frac{N}{2}\boldsymbol{g}_{\boldsymbol{\theta}_l} \odot \left(\psi^{1/2} \odot \boldsymbol{z}_l\right). \tag{43}$$

Collecting the partial derivatives in Equation (17), the final update rules are

$$\boldsymbol{C} \leftarrow \boldsymbol{C} - \eta_{\boldsymbol{C}}\left(\alpha\boldsymbol{C} + \frac{1}{L}\sum_{l=1}^{L} N\boldsymbol{g}_{\boldsymbol{\theta}_l}\right), \tag{44}$$

$$\boldsymbol{F} \leftarrow \boldsymbol{F} - \eta_{\boldsymbol{F}}\left(-\boldsymbol{A} + \boldsymbol{C}\boldsymbol{B}^\top + \alpha\boldsymbol{F} + \frac{1}{L}\sum_{l=1}^{L} N\boldsymbol{g}_{\boldsymbol{\theta}_l}\boldsymbol{h}_l^\top\right), \tag{45}$$

$$\gamma \leftarrow \gamma - \eta_{\gamma}\left(-\frac{1}{2} + \frac{1}{2}\operatorname{sum}\big(\boldsymbol{C} \odot \boldsymbol{A},\ \dim = 1\big) \odot \psi + \frac{\alpha}{2}\psi + \frac{1}{L}\sum_{l=1}^{L}\frac{N}{2}\boldsymbol{g}_{\boldsymbol{\theta}_l} \odot \left(\psi^{1/2} \odot \boldsymbol{z}_l\right)\right) \tag{46}$$

for learning rates $\eta_{\boldsymbol{C}}, \eta_{\boldsymbol{F}}, \eta_\gamma > 0$. Pseudo code for the practical implementation is given in Algorithm 1. Being a VI method for approximating a posterior with a FA model, the algorithm is named VIFA.

Algorithm 1 VIFA

Input: Dataset $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, observation dimension D, latent dimension K, prior precision $\alpha > 0$, iterations T, mini-batch size M, Monte Carlo average size L, learning rates $\eta_{\boldsymbol{C}}, \eta_{\boldsymbol{F}}, \eta_\gamma > 0$

1: Initialise $\boldsymbol{C} = \boldsymbol{0}^D$, $\boldsymbol{F} \in \mathbb{R}^{D\times K}$, $\psi = \boldsymbol{1}^D$
2: $\gamma \leftarrow \log \psi$
3: Initialise $\hat{\boldsymbol{g}}_{\boldsymbol{C}} = \boldsymbol{0}^D$, $\hat{\boldsymbol{g}}_{\boldsymbol{F}} = \boldsymbol{0}^{D\times K}$, $\hat{\boldsymbol{g}}_\gamma = \boldsymbol{0}^D$
4: $\boldsymbol{A} \leftarrow \psi^{-1} \odot \boldsymbol{F}$ (with broadcasting)
5: $\boldsymbol{B} \leftarrow \boldsymbol{F}^\top \boldsymbol{A}$
6: $\boldsymbol{C} \leftarrow \boldsymbol{A}(\boldsymbol{I} + \boldsymbol{B})^{-1}$
7: **for** $t = 1, \ldots, T$ **do**
8: Sample a mini-batch $\mathcal{B}$ of M training examples from $\mathcal{D}$
9: Sample $\boldsymbol{h} \sim \mathcal{N}(\boldsymbol{0}^K, \boldsymbol{I}^{K\times K})$
10: Sample $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}^D, \boldsymbol{I}^{D\times D})$
11: $\boldsymbol{\theta} \leftarrow \boldsymbol{F}\boldsymbol{h} + \boldsymbol{C} + \psi^{1/2} \odot \boldsymbol{z}$
12: $\boldsymbol{g}_{\boldsymbol{\theta}} \leftarrow \nabla_{\boldsymbol{\theta}} \big(-\frac{1}{M} \sum_{(\boldsymbol{x},y)\in\mathcal{B}} \log p(y|\boldsymbol{x}, \boldsymbol{\theta}) \big)$ (via back-propagation)
13: $\hat{\boldsymbol{g}}_{\boldsymbol{C}} \leftarrow \hat{\boldsymbol{g}}_{\boldsymbol{C}} + N\boldsymbol{g}_{\boldsymbol{\theta}}$
14: $\hat{\boldsymbol{g}}_{\boldsymbol{F}} \leftarrow \hat{\boldsymbol{g}}_{\boldsymbol{F}} + N\boldsymbol{g}_{\boldsymbol{\theta}}\boldsymbol{h}^\top$
15: $\hat{\boldsymbol{g}}_\gamma \leftarrow \hat{\boldsymbol{g}}_\gamma + \frac{N}{2}\boldsymbol{g}_{\boldsymbol{\theta}} \odot \big(\psi^{1/2} \odot \boldsymbol{z}\big)$
16: **if** $t \bmod L = 0$ **then**
17: $\boldsymbol{C} \leftarrow \boldsymbol{C} - \eta_{\boldsymbol{C}} \big(\alpha\boldsymbol{C} + \frac{1}{L}\hat{\boldsymbol{g}}_{\boldsymbol{C}}\big)$
18: $\boldsymbol{F} \leftarrow \boldsymbol{F} - \eta_{\boldsymbol{F}} \big(-\boldsymbol{A} + \boldsymbol{C}\boldsymbol{B}^\top + \alpha\boldsymbol{F} + \frac{1}{L}\hat{\boldsymbol{g}}_{\boldsymbol{F}}\big)$
19: $\gamma \leftarrow \gamma - \eta_\gamma \big(-\frac{1}{2} + \frac{1}{2}\mathrm{sum}(\boldsymbol{C} \odot \boldsymbol{A},\ \dim = 1) \odot \psi + \frac{\alpha}{2}\psi + \frac{1}{L}\hat{\boldsymbol{g}}_\gamma\big)$
20: $\psi \leftarrow \exp\gamma$
21: Reset $\hat{\boldsymbol{g}}_{\boldsymbol{C}}, \hat{\boldsymbol{g}}_{\boldsymbol{F}}$ and $\hat{\boldsymbol{g}}_\gamma$ according to line 3
22: Recalculate $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{C}$ according to lines 4-6
23: **end if**
24: **end for**
25: **return** $\boldsymbol{C}, \boldsymbol{F}, \psi$

4 Experiments

In this section, we test the performance of VIFA algorithm on both synthetic and real datasets. We first present the results of using VIFA to estimate the posterior distribution of linear regression models. Also, for simple neural networks, the ability of VIFA to make predictions with uncertainty estimates is tested in comparison to SLANG. Finally, we apply VIFA algorithm on modern Convolutional Neural Networks, which illustrates the model agnostic nature of the algorithm.

4.1 Experiments on Synthetic Dataset

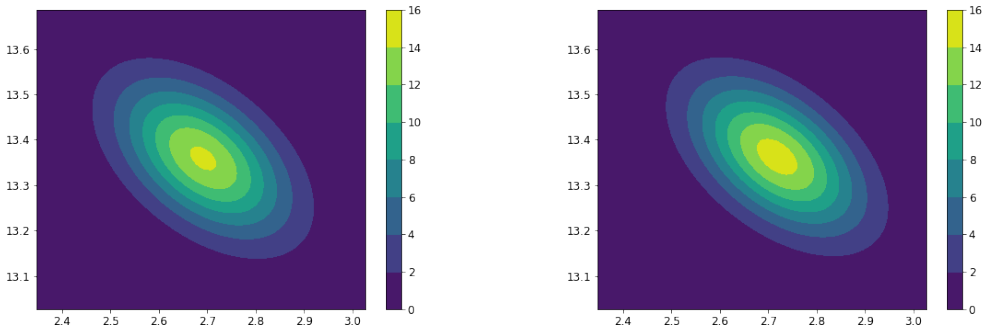
4.1.1 Linear regression posterior estimation with synthetic data

The purpose of these simulations is to test that VIFA is able to learn the posterior distribution of a very simple linear regression model with two learnable parameters.

Methodology Synthetic data was generated as follows. First, 1000 inputs $\boldsymbol{x} \in \mathbb{R}^2$ were sampled from a multivariate zero mean Gaussian distribution with unit variances and covariances of 0.5. Next, the linear regression parameter vector $\boldsymbol{\theta} \in \mathbb{R}^2$ was sampled from $\mathcal{N}(\boldsymbol{\theta}; 0, \alpha^{-1}\boldsymbol{I})$ with $\alpha = 0.01$. Then the outputs $y \in \mathbb{R}$ were generated according to the equation $y = \boldsymbol{\theta}^T\boldsymbol{x} + \epsilon$, where $\epsilon \sim \mathcal{N}(\epsilon; 0, \beta^{-1})$ with $\beta = 0.1$.

Using this data, the true posterior distribution was evaluated in closed form (see 9) and an approximate posterior with latent dimension $K = 1$ was estimated via VIFA. VIFA ran for 5000 epochs with a batch size of $M = 100$ and a Monte Carlo average size of $L = 10$. The learning rates $\eta_{\boldsymbol{c}}$, $\eta_{\boldsymbol{F}}$ and $\eta_{\boldsymbol{\gamma}}$ were set to 0.01, 0.0001 and 0.01, respectively. The reasoning for using a smaller learning rate for $\boldsymbol{F}$ was that its contribution to the full covariance matrix is $\boldsymbol{F}\boldsymbol{F}^T$. Since this is regression, the likelihood function used in VIFA was set to $\mathcal{N}(y; \boldsymbol{\theta}^T\boldsymbol{x}, \beta^{-1})$. Finally, to improve numerical stability, any gradients with Frobenius norm greater than 10 were rescaled to have norm of exactly 10.

Results and discussion Figure 1 shows a qualitative comparison between the ground truth and approximate linear regression posteriors. It can be seen that the contours of the approximate posterior are very similar to the ground truth in term of position, direction and density. To test the stability of VIFA, multiple random seeds are ran and quantitative results are shown in A.2.



(a) Ground truth (b) VIFA

Figure 1: The ground truth posterior pdf of a linear regression model with two learnable parameters, plus the pdf of a FA model with a single latent dimension which was fit to the same data using VIFA.

4.1.2 Logistic regression posterior estimation with synthetic data

The purpose of these simulations is to test that VIFA is able to learn the posterior distribution of a very simple logistic regression model with two learnable parameters.

Methodology Synthetic inputs $\boldsymbol{x} \in \mathbb{R}^2$ and the logistic regression parameter vector $\boldsymbol{\theta} \in \mathbb{R}^2$ were sampled in the same way as in Section 4.1.1. However, this time 3000 inputs were used. To generate each output $y \in \{0, 1\}$, the positive class probability was first evaluated as $p = \sigma(\boldsymbol{\theta}^T \boldsymbol{x})$, where $\sigma : \mathbb{R} \to [0, 1]$ denotes the logistic sigmoid function. Then a random number was sampled uniformly from the interval $[0, 1]$ and y was set to 1 if this number was less than p, else 0.

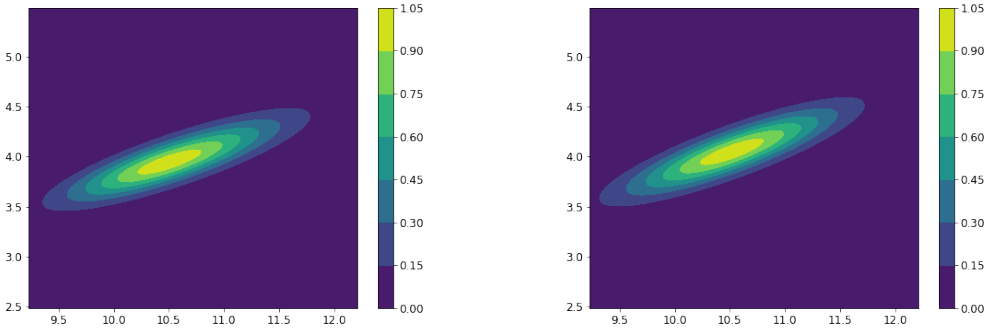
Unlike linear regression, there is no closed form solution for the true posterior of a logistic regression model. In this case, the ground truth posterior was evaluated by first looping over a 2D grid around the true parameter vector $\boldsymbol{\theta}$ and evaluating the unnormalised log posterior probability at each point in the grid. Formally, this is

$$\mathcal{N}(\boldsymbol{\theta}; 0, \alpha^{-1}\boldsymbol{I}) \prod_{n=1}^{3000} \sigma(\boldsymbol{\theta}^T \boldsymbol{x}_n)^{y_n} \sigma(\boldsymbol{\theta}^T \boldsymbol{x}_n)^{1-y_n}. \tag{47}$$

Then the values in the grid were scaled such that the maximum value was equal to 1. This posterior is only correct up to a constant, but suffices for a qualitative comparison.

The posterior was then approximated via VIFA using the exact same hyperparameters as in Section 4.1.1. This time, the likelihood function used in VIFA was set to binary cross-entropy. That is, $\sigma(\boldsymbol{\theta}^T \boldsymbol{x}_n)^{y_n} \sigma(\boldsymbol{\theta}^T \boldsymbol{x}_n)^{1-y_n}$.

Results and discussion Figure 2 shows a qualitative comparison between the ground truth and approximate logistic regression posteriors. It can be seen that the contours of the approximate posterior are very similar to the ground truth in term of position, direction and density.



(a) Ground truth

(b) VIFA

Figure 2: The ground truth posterior pdf of a logistic regression model with two learnable parameters, plus the pdf of a FA model with a single latent dimension which was fit to the same data using VIFA. Both posteriors are scaled such that the maximum value is equal to 1.

4.2 Experiments on UCI Datasets

Given the positive results achieved by VIFA in the experiments using synthetic data, the purpose of these experiments is to test how well VIFA is able to approximate the posterior of linear regression models fit to real datasets with more dimensions.

4.2.1 Linear regression posterior estimation with UCI datasets

Methodology Four regression datasets from the UCI Machine Learning Repository (Dua & Graff, 2017) were used. Namely, the Energy Efficiency[1], Boston Housing[2], Concrete Compressive Strength[3] and Yacht Hydrodynamics[4] datasets. More details about these datasets are given in Appendix A.3. Each dataset was randomly split into validation and test sets of equal size. The learning rate and number of epochs were tuned on the validation set to minimise the 2-Wasserstein distance between the true and approximate posteriors, and then the algorithm was evaluated once on the test set. To reduce the number of hyperparameters, a single learning rate η was used to optimise all parameters of the approximate posterior. That is, $\eta_{\boldsymbol{c}} = \eta_{\boldsymbol{F}} = \eta_{\boldsymbol{\gamma}} = \eta$. The latent dimension was set to $K = 3$ for all datasets. Also, the mini-batch size and Monte Carlo average size were set to $M = 100$ and $L = 10$, respectively, since these values worked well for the experiments with synthetic data in Section 4.1.1. The `BayesianRidge` estimator from the scikit-learn library (Pedregosa et al., 2011) was used to compute the ground truth posterior. This algorithm automatically selects α and β via the Bayesian method described in (MacKay, 1992). These ground truth values were also used when running VIFA. In particular, β was used in the calculation of the average negative log-likelihood term. The selected VIFA hyperparameter values for each dataset are given in Table 6 in Appendix A.3. Finally, before running the algorithms, each input variable was re-scaled to have zero mean and unit standard deviation, and while training, gradient norms were clipped at a maximum value of 10.

Results and discussion Figures 3 show a qualitative comparison of the true posterior of a linear regression model and the approximate posterior learned by VIFA, in the case of the Yacht Hydrodynamics and Boston Housing datasets. In the interest of space, qualitative comparisons for the Energy Efficiency and Concrete Compressive Strength datasets can be found in Figures 4 and 5 in Appendix A.3, along with quantitative results for all four datasets in Table 7.

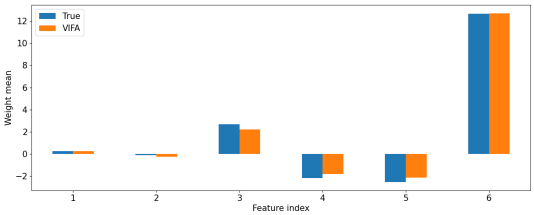
The results for Yacht Hydrodynamics and Boston Housing make for an interesting comparison, since these datasets have dimension 6 and 13, respectively, but in both cases the latent dimension was set to 3. In the case of Yacht Hydrodynamics, Figure 3 (e) shows that the off-diagonal entries of the approximate covariance matrix are a good match for the ground truth. In comparison, while the approximate covariance matrix for Boston Housing captures the general structure of the ground truth, the two matrices are noticeably different, as shown in Figure 3 (f). On closer inspection, most of the discrepancies appear to be caused by small covariances in the ground truth which are set closer to zero in the approximation. The diagonal entries of the covariance matrix are also better approximated in the case of Yacht Hydrodynamics, as can be seen by comparing Figure 3 (c) with Figure 3 (d). Although the Boston Housing variances appear to be ordered correctly with respect to each other, most of them are underestimated. Quantitatively, the results in Table 7 show that the relative distance of the approximate covariance matrix from the ground truth is an order of magnitude less for Yacht Hydrodynamics: 0.0391 compared to 0.3185 for Boston Housing. However, the relative distance of the approximate posterior mean from the ground truth is actually worse for Yacht Hydrodynamics: 0.0435 compared to 0.0262 for Boston Housing. This is not necessarily surprising, since while the quality of the posterior covariance approximation is dependant on the data dimension relative to the latent dimension, the posterior mean can in theory be approximated exactly, irrespective of the data dimension.

In summary, across the four datasets, VIFA demonstrates good capacity to approximate the true posterior distribution of linear regressions models. However, the quality of the covariance approximation naturally deteriorates as the data dimension becomes larger in relation to the latent dimension.

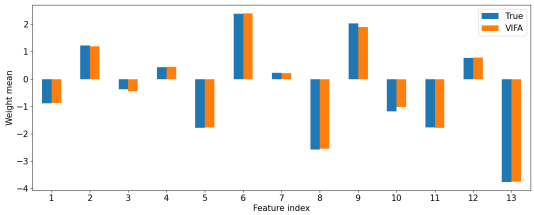
4.2.2 Neural network regression with UCI datasets

Given the encouraging results achieved by VIFA when applied to linear regression, the purpose of the following experiments is to test VIFA on simple neural networks with a single hidden layer. Unlike linear regression, it is not possible to compute the true posterior distribution of the parameter vector of a non-linear

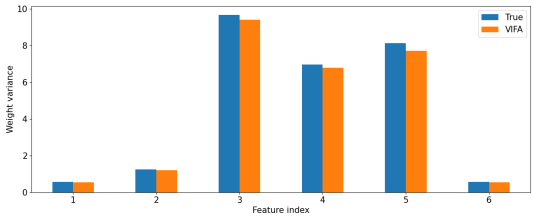
[1] https://archive.ics.uci.edu/ml/datasets/energy+efficiency
[2] https://archive.ics.uci.edu/ml/machine-learning-databases/housing
[3] https://archive.ics.uci.edu/ml/datasets/concrete+compressive+strength
[4] http://archive.ics.uci.edu/ml/datasets/yacht+hydrodynamics



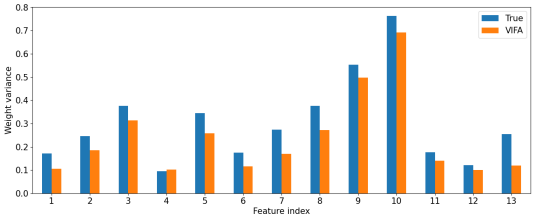
(a) Yacht Hydrodynamics posterior mean



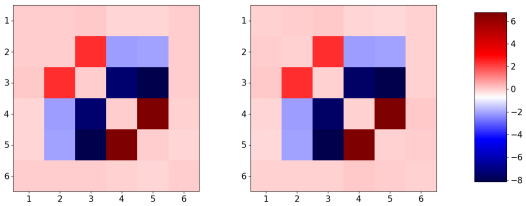
(b) Boston Housing posterior mean



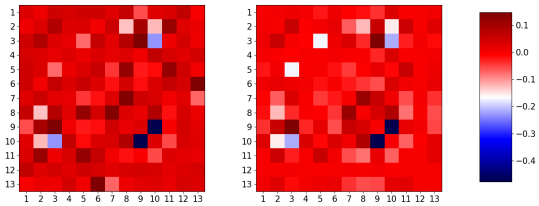
(c) Yacht Hydrodynamics posterior variance



(d) Boston Housing posterior variance



(e) Yacht Hydrodynamics posterior covariance



(f) Boston Housing posterior covariance

Figure 3: Comparison of the ground truth posterior of linear regression models fit to the Yacht Hydrodynamics and the Boston Housing datasets, and the approximate posteriors learned by VIFA. Variances and covariance are plotted separately due to the difference in their magnitude. In plots (e) and (f), the diagonal entries of the covariance matrices have been set to zero.

neural network in closed-form. Therefore, the focus of these experiments is on the quality of predictions made by neural networks trained via VIFA in comparison to SLANG.

Methodology The setup for the experiments largely followed the methodology adopted by the authors of SLANG (Mishkin et al., 2018). Each neural network had a single hidden layer with 50 rectified linear units (ReLU) and a single, unactivated output. The UCI regression datasets from Table 5 were used once again. Each dataset was split 20 times into different train and tests sets[5]. For each train-test split, VIFA hyperparameters were tuned using the training data and then the best configuration was evaluated once on the test set. The hyperparameters which were tuned were the learning rate η, the precision α of the prior and the precision β of the additive target noise distribution. As with SLANG, these were optimised via 30 iterations of hyperparameter search, with 5-fold cross-validation performed in each iteration to estimate the marginal log-likelihood. However, while SLANG used Bayesian optimisation, VIFA was tuned via random search to keep things simple. During the search, values for η, α and β were sampled log-uniformly from $[0.01, 0.02]$, $[0.01, 10]$ and $[0.01, 1]$, respectively. To improve numerical stability, any gradients with Frobenius norm greater than 10 were rescaled to have norm of exactly 10. The other hyperparameters were fixed to the same values as SLANG. That is, latent dimension $K = 1$, number of epochs $T = 120$, mini-batch size $M = 10$ and Monte Carlo average size $L = 4$. Although vanilla SGD is employed in Algorithm 1, a whole host of gradient optimisers can be used in conjunction with VIFA. In this case, the Adam optimiser (Kingma & Ba, 2014a) was used to perform the gradient updates. Before each training run, the features and target were normalised by subtracting their mean and dividing by their standard deviation. These statistics were computed on training data only, including within each separate fold of cross-validation. When training, the average negative Gaussian log-likelihood was used on line 12 of Algorithm 1. When testing, both the average marginal Gaussian log-likelihood and root mean squared error (RMSE) were computed, using a Monte Carlo average size of 100. Full details of how the training loss and test metrics were computed are given in Appendix A.4. The same experiments were re-run for SLANG[6] with some minor adjustments to the authors' original setup to allow for a fairer comparison with VIFA. In particular, the Monte Carlo average size when testing and the hyperparameter ranges for α and β were adjusted to match those of VIFA. Finally, the VIFA experiments were also run for linear regression models. The only difference being that the hyperparameter range for the learning rate η was set to $[0.001, 0.01]$, as this appeared to work better that $[0.01, 0.02]$ during some initial cross-validation runs.

Results and Discussion Test metrics for VIFA and SLANG, which were averaged over the 20 train-test splits, are given in Table 1. In terms of negative marginal log-likelihood (NMLL), neither algorithm is better than the other on all datasets. For Boston Housing and Concrete Compressive Strength, the mean results for neural networks are essentially the same when standard errors are taken into account. SLANG achieves lower mean NMLL on Yacht Hydrodynamics, but performs extremely poorly on Energy Efficiency with respect to the same metric. It is worth noting that this was caused by a single trial which returned NMLL of 4.04×10^8.

As for mean RMSE, VIFA is the best algorithm on all datasets by some distance. Moreover, the standard errors on the RMSE show that VIFA is very consistent on different train-test splits, whereas the performance of SLANG varies substantially. This is especially true in the case of Energy Efficiency, which returned RMSE of 2.84×10^4 in one trial.

If the outliers for NMLL and RMSE are removed, the SLANG results are much better. Table 8 in Appendix A.4 provides the median results over the 20 train-test splits. From this perspective, SLANG is far more competitive with VIFA. However, the fact that SLANG occasionally fails to generalise to the test data is worrying. In a real prediction scenario where the true targets of the data are not known, such poor performance could have serious consequences. In contrast, VIFA was able to generalise to all 20 test sets with similar performance across the board.

Another clear advantage that VIFA has over SLANG is speed. In all experiments involving neural networks, the average runtime of VIFA is roughly half that of SLANG. VIFA generally runs slightly faster when learning a linear regression posterior (although, surprisingly, not in the case of Yacht Hydrodynamics). However, the

[5]Data splits are available at `https://github.com/yaringal/DropoutUncertaintyExps`
[6]Using the code available at `https://github.com/aaronpmishkin/SLANG`

Table 1: Mean test results for prediction experiments with UCI regression datasets. Metrics are the average negative marginal log-likelihood (NMLL) and root mean squared error (RMSE). LR and NN refer to linear regression and neural network, respectively. The mean results over 20 different train-test splits are shown. The results for NMLL and RMSE include standard errors. The runtime refers to the total runtime for a single train-test split. That is, 30 rounds of 5-fold cross-validation on the training set followed by fitting the best model to the full training set and evaluating on the test set. All experiments were executed on an Apple M1 chip (2020) with an 8-core CPU and 16GB of RAM. The best results on each row are highlighted in bold.

Metric	**Dataset**	**VIFA-LR**	**VIFA-NN**	**SLANG-NN**
NMLL	Energy	$\mathbf{2.57 \pm 0.02}$	$\mathbf{2.53 \pm 0.02}$	$2.13 \times 10^7 \pm 2.01 \times 10^7$
	Boston	3.07 ± 0.12	$\mathbf{2.66 \pm 0.06}$	$\mathbf{2.73 \pm 0.06}$
	Concrete	3.77 ± 0.02	$\mathbf{3.34 \pm 0.01}$	$\mathbf{3.32 \pm 0.03}$
	Yacht	3.65 ± 0.04	2.36 ± 0.06	$\mathbf{1.63 \pm 0.04}$
RMSE	Energy	3.07 ± 0.06	$\mathbf{2.90 \pm 0.06}$	$1.50 \times 10^3 \pm 1.42 \times 10^3$
	Boston	4.64 ± 0.23	$\mathbf{3.64 \pm 0.23}$	15.87 ± 4.36
	Concrete	10.34 ± 0.15	$\mathbf{6.77 \pm 0.09}$	22.27 ± 6.64
	Yacht	8.96 ± 0.29	$\mathbf{2.51 \pm 0.09}$	8.03 ± 4.58
Runtime	Energy	**22.10**	22.59	40.38
(minutes)	Boston	**14.78**	16.18	32.97
	Concrete	**25.23**	28.41	52.31
	Yacht	11.78	**11.58**	20.61

NMLL and RMSE results indicate that there are non-linear relationships in the datasets which are better captured by the neural network.

In summary, the choice between VIFA and SLANG depends on the dataset and evaluation metric. However, VIFA certainly has the advantage of being more efficient and appears far less sensitive to variations in the training and test data.

4.3 Experiments on Medical Imaging Dataset

To illustrate the model-agnostic nature of VIFA algorithm, we adopt our algorithm to Convolutional Neural Networks (CNN) and evaluate its effectiveness for medical image diagnosis, which is often regarded as a safety-critical task where prediction reliability matters significantly. In particular, we focus on a diabetic retinopathy detection problem in India [7]. The dataset contains 3662 samples, and there are 5 severity levels of diabetic retinopathy, which mean label $y \in \{0, 1, 2, 3, 4\}$. To simplify the problem, we follow the strategy used in Leibig et al. (2017) to convert all instance labels into binary forms. This is done by categorizing the classes into two groups: sight-threatening diabetic retinopathy and non-sight-threatening diabetic retinopathy. The former includes cases of moderate diabetic retinopathy or more severe (classes 2, 3, and 4), while the latter includes cases with no or mild diabetic retinopathy (classes 0 and 1).

4.3.1 Methodology

The Convolutional Neural Network model we select is the resnet-18 model, which contains roughly 11.2M trainable parameters. Due to the non-linear nature of the neural network model, there is no closed-form solution for the true posterior of resnet-18 model on the dataset. In this case, our focus is on the prediction performance made by Bayesian model averaging of resnet-18 model trained via VIFA algorithm. To evaluate its effectiveness, we also train resnet-18 model via standard gradient descent and use the performance as a baseline.

In the preliminary experiment of apply original VIFA algorithm to resnet-18 model, we observe no performance increase on train and validation sets during training. This phenomenon might due to the fact that

[7] APTOS 2019 Blindness Detection Dataset, https://www.kaggle.com/competitions/aptos2019-blindness-detection/overview

the mean and the covariance of a factor analysis distribution for high dimensional models are challenging to be learned simultaneously from random start [8]. Therefore, we 'decouple' the learning of mean parameters $\boldsymbol{c}$ and covariance parameter $\boldsymbol{F}$, ψ by utilizing a 2-stages training strategy. In the first stage of training, covariance-related parameters $\boldsymbol{F}$ and ψ stay unchanged with small absolute values (less than 1e-5), and the parameter $\boldsymbol{c}$ is trained starting from pre-trained resnet-18 weights as initialization. During the second stage, the learning rates for $\boldsymbol{F}$ and ψ gradually increases from 0 to target values, while keeping the learning rate for $\boldsymbol{c}$ constant. Learning rate decay scheduler is in use for $\boldsymbol{c}$ during both stages. Compared to the original VIFA, this modification can further reduce computational cost since no gradient computations for $\boldsymbol{F}$ and ψ is needed in first stage. To keep things simple, learning rates for $\boldsymbol{c}$, $\boldsymbol{F}$, and ψ are set to the same values η, which are optimized as hyper-parameter on the validation dataset. We run a total of 10 epochs, with 2 epochs in stage 1. The FA distribution has a latent dimension of 1, and the gradients for parameter update are computed based on mini-batches of size 16. All gradients are clipped to have Frobenius norm less than or equal to 10, and parameter update is performed after 12 times of gradients calculations. We use Adam optimizer (Kingma & Ba, 2014b) to perform gradient updates.

In order to report robust results, test metrics are averaged over 5 train-valid-test splits with split ratios 50%, 20%, 30% respectively. For each data split, hyper-parameters prior precision α and learning rate η are tuned via 6 times random searches, where α and η are log-uniformly sampled from [0.01, 10] and [1e-6, 5e-4] respectively. After the best hyper-parameter configuration are found for the current split, evaluation metrics are computed on test data with Monte Carlo average size S being 100.

4.3.2 Prediction with Uncertainty

One advantage of Bayesian deep learning methods compared to traditional Frequentist training is that we are able to obtain predictive uncertainty at the same time as getting predictions. Based on this uncertainty, Band et al. (2022) propose an automated diagnostic workflow for medical imaging: When given input, a model generates a prediction along with an associated uncertainty estimation. If the uncertainty estimation falls below a specified reference threshold, indicating low uncertainty, the diagnosis proceeds without additional examination. However, if the threshold is not met, a medical professional is consulted for further review. In general, we desire a model's predictive uncertainty to have a strong correlation with the accuracy of its predictions. High-quality predictive uncertainty estimates can be a fail-safe against false predictions (Band et al., 2022).

There are multiple ways to define test sample uncertainty [9] One definition adopted in Band et al. (2022) is that for any given test sample $\boldsymbol{x}_*$ the predictive uncertainty is equal to the entropy of the predictive distribution:

$$H(\mathbb{E}_{\boldsymbol{\theta}\sim Q(\boldsymbol{\theta})}[p(\mathrm{y}_* \mid f(\boldsymbol{x}_*;\boldsymbol{\theta}))]) = -\sum_{c\in\{0,1\}} \mathbb{E}_{\boldsymbol{\theta}\sim Q(\boldsymbol{\theta})}[p(\mathrm{y}_* = c \mid f(\boldsymbol{x}_*;\boldsymbol{\theta}))] \log \mathbb{E}_{\boldsymbol{\theta}\sim Q(\boldsymbol{\theta})}[p(\mathrm{y}_* = c \mid f(\boldsymbol{x}_*;\boldsymbol{\theta}))] \tag{48}$$

where H represents the Shannon entropy, $f(\boldsymbol{x}_*;\boldsymbol{\theta})$ is logit value, $p(\mathrm{y}_* = c \mid f(\boldsymbol{x}_*;\boldsymbol{\theta}))$ is a binary cross-entropy likelihood function and $Q(\boldsymbol{\theta})$ is the variational distribution. In practice, the expectation $\mathbb{E}_{\boldsymbol{\theta}\sim Q(\boldsymbol{\theta})}[p(\mathrm{y}_* \mid f(\boldsymbol{x}_*;\boldsymbol{\theta}))]$ is approximated by S Monte Carlo samples, $\mathbb{E}_{\boldsymbol{\theta}\sim Q(\boldsymbol{\theta})}[p(\mathrm{y}_* \mid f(\boldsymbol{x}_*;\boldsymbol{\theta}))] \approx \frac{1}{S}\sum_i^S p(\mathrm{y}_* \mid f(\boldsymbol{x}_*;\boldsymbol{\theta}^{(i)}))$, here $\{\boldsymbol{\theta}^{(i)}\}_{i=1}^S$ are sampled from (trained) variational distribution $Q(\boldsymbol{\theta})$, and $p(\mathrm{y}_* \mid f(\boldsymbol{x}_*;\boldsymbol{\theta}^{(i)}))$ denotes the predictive distribution given parameter realization $\boldsymbol{\theta}^{(i)}$.

[8] One explanation is that in the process of learning a FA posterior, mean updating and covariance updating are closely related. An unlearned covariance will lead to terrible sampling of model parameters $\boldsymbol{\theta}$, thus directing mean parameter to a wrong way.

[9] In literature, the predictive uncertainty are usually argued to have different sources, such as aleatoric uncertainty and epistemic uncertainty, and different definitions may lead to varied implications (Ulmer et al., 2023; D'Angelo & Fortuin, 2021). However in this paper, we do not distinguish them, and just regard predictive uncertainty as a measure which reflects to what extent we trust the model's prediction result.

Another way to define predictive uncertainty is by measuring the model disagreement (D'Angelo & Fortuin, 2021), which is computed as:

$$\mathcal{MD}^2(\boldsymbol{x}_*) = \sum_{c\in\{0,1\}} \mathbb{E}_{\boldsymbol{\theta}\sim Q(\boldsymbol{\theta})}[(p(\mathrm{y}_* = c \mid f(\boldsymbol{x}_*;\boldsymbol{\theta})) - \mathbb{E}_{\boldsymbol{\theta}\sim Q(\boldsymbol{\theta})}[p(\mathrm{y}_* = c \mid f(\boldsymbol{x}_*;\boldsymbol{\theta}))])^2] \tag{49}$$

This quality represents how much 'disagreement' exist among the distribution of models. In practice, this quantity is approximated by $\mathcal{MD}^2(\boldsymbol{x}_*) \approx \sum_{c\in\{0,1\}} \frac{1}{S}\sum_{\boldsymbol{\theta}^{(i)}\in\{\boldsymbol{\theta}^{(1)},...,\boldsymbol{\theta}^{(S)}\}}(p(\mathrm{y}_* = c \mid f(\boldsymbol{x}_*;\boldsymbol{\theta}^{(i)})) - \frac{1}{S}\sum_{\boldsymbol{\theta}^{(i)}\in\{\boldsymbol{\theta}^{(1)},...,\boldsymbol{\theta}^{(S)}\}}[p(\mathrm{y}_* = c \mid f(\boldsymbol{x}_*;\boldsymbol{\theta}^{(i)}))])^2$, where $\{\boldsymbol{\theta}^{(1)},...,\boldsymbol{\theta}^{(S)}\}$ are Monte Carlo samples from the variational distribution $Q(\boldsymbol{\theta})$. It is easy to see if all models agree on the prediction $\boldsymbol{x}_*$, the disagreement measure $\mathcal{MD}^2(\boldsymbol{x}_*)$ becomes zero. On the other hand, the larger the score $\mathcal{MD}^2(\boldsymbol{x}_*)$, the server disagreement exists between predictive distributions.

By computing uncertainty scores for each test sample using either of the two aforementioned methods, we can implement selective prediction in the automated diagnostic workflow. This involves sorting the test samples based on their uncertainty scores and evaluating the model's performance on the 90%, 80%, 70%, 60%, and 50% most certain samples. With high-quality uncertainties, we expect the model's performance to increase as we extract fewer and fewer test samples with ascending predictive uncertainty. From another perspective, the assessment score of selective prediction also reflects the utility of the uncertainty obtained from Bayesian deep learning methods.

4.3.3 Results and discussion

Prediction Performance Table 2 shows a comparison between the performance of VIFA and traditional gradient descent training of resnet-18 model. For most metrics, such as Accuracy, F1-score and AU-ROC, the performances of VIFA are very competitive to that of Gradient Descent and their value are very close to each other. VIFA leads to a better Precision value while Gradient Descent has more satisfying Recall score, but their differences are not significant. The only exceptional metric is the training time, for which the time used for Gradient Descent training is around 6% less than VIFA training. However, it is noticeable that VIFA gets reliable predictive uncertainties simultaneously and a little bit more computational time compared to this achievable uncertainty quantification is reasonable.

Table 2: Mean test results for resnet-18 model on diabetic retinopathy detection task. The performances of applying VIFA and Gradient Descent algorithms are compared. Metrics include accuracy, precision, recall, F1-score, and Area Under Receiver Operating Characteristic curve (AU-ROC). The mean results over 5 different train-valid-test splits are shown. The results for test metrics include standard errors. The runtime refers to the total runtime for a single train-valid-test split. All experiments were executed on machines with Intel(R) Xeon(R) Silver 4114 CPU and NVIDIA GeForce RTX 2080 Ti GPU. The best results on each row are highlighted in bold (no score is bolded if both scores are competitive).

	VIFA	**Gradient Descent**
Accuracy	0.928±0.003	0.927±0.004
Precision	**0.921±0.009**	0.914±0.009
Recall	0.904±0.014	**0.907±0.009**
F1-Score	0.912±0.004	0.911±0.005
AU-ROC	0.980±0.002	0.979±0.002
Time(minutes)	463.2±2.8	**435.7±1.8**

Uncertainty Effectiveness Table 3 shows the test accuracy of selective prediction with different uncertainty levels (Results for F1 score and AU-ROC are in Table 9, 10 in the Appendix). Predictive uncertainties calculated from two approaches: predictive entropy and model disagreement score are in use. We observe that prediction performance continuously increase with the decline of predictive uncertainty. The best performance reached when extracting 50% of most certain samples for assessment, which is the lowest uncertainty we accept. This result illustrates that predictive uncertainty has negative correlation with prediction accuracy, which shows the effectiveness of the uncertainty from the VIFA-resnet model.

Table 3: Mean test accuracies of selective prediction under different uncertainty levels. Uncertainty scores calculated from two distinct approaches are employed, which are predictive entropy and model disagreement. Test samples are arranged in ascending order based on their uncertainty scores. 'Proportion of Samples' indicates the percentage of ordered samples used for evaluation. The results for test accuracy include standard errors. The best results in each column are highlighted in bold.

Proportion of Samples	**Predictive Entropy**	**Model Disagreement**
90%	0.957±0.005	0.956±0.005
80%	0.973±0.004	0.975±0.003
70%	0.983±0.002	0.984±0.002
60%	0.991±0.002	0.99±0.002
50%	**0.994±0.001**	**0.994±0.001**

5 Related Work

6 Conclusion

This work presents a novel Variational Inference algorithm named VIFA for posterior distribution estimation of neural network models. We adopt Factor Analysis (FA) as our variational distribution whose low rank plus diagonal structure for covariance matrix effectively alleviate computational burden when neural networks are large. Moreover, VIFA is model-agnostic, making it can be readily applied to any network architecture.

To assess VIFA, we conduct posterior estimation experiments for linear regression models on both synthetic and real world datasets. In all cases, VIFA was able to learn a good approximation of the true posterior covariance. Moreover, VIFA also achieved encouraging results in the neural network predictions experiments in comparison to SLANG (Mishkin et al., 2018), in terms of RMSE, marginal likelihood and runtime. Finally, to validate the claim that VIFA is model agnostic, experiments of applying VIFA on Convolutional Neural Networks for real world medical imaging tasks are conducted. Though the practical implementation of VIFA training has been modified a little bit to makes the algorithm works, the new algorithm becomes even more efficient and achieve satisfying performance for prediction under uncertainty. The model agnostic nature of VIFA also opens new research directions for being applied on many other senarios, such as generative models (Harshvardhan et al., 2020) and transformers (Vaswani et al., 2017).

References

Neil Band, Tim GJ Rudner, Qixuan Feng, Angelos Filos, Zachary Nado, Michael W Dusenberry, Ghassen Jerfel, Dustin Tran, and Yarin Gal. Benchmarking bayesian deep learning on diabetic retinopathy detection tasks. *arXiv preprint arXiv:2211.12717*, 2022.

David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. *International conference on machine learning*, pp. 1613–1622, 2015.

Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. CoAtNet: Marrying convolution and attention for all data sizes. *Proceedings of Advances in Neural Information Processing Systems*, 34, 2021.

Francesco D'Angelo and Vincent Fortuin. Repulsive deep ensembles are bayesian. *Advances in Neural Information Processing Systems*, 34:3451–3465, 2021.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016a.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016b.

Alex Graves. Practical variational inference for neural networks. *Proceeding of Advances in Neural Information Processing Systems*, 24, 2011.

GM Harshvardhan, Mahendra Kumar Gourisaria, Manjusha Pandey, and Siddharth Swarup Rautaray. A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, 38:100285, 2020.

José M Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. *International Conference on Machine Learning*, pp. 1861–1869, 2015.

Mohammad E Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable Bayesian deep learning by weight-perturbation in Adam. *Proceedings of the International Conference on Machine Learning*, 35:2611—-2620, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014a.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014b.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports*, 7(1):1–14, 2017.

David J C MacKay. Bayesian interpolation. *Computation and Neural Systems*, 4:415–447, 1992.

Wesley Maddox, Timur Garipov, Pavel Izmailov, Dmitry Vetrov, and Andrew G Wilson. A simple baseline for Bayesian uncertainty in deep learning. *Proceedings of Advances in Neural Information Processing Systems*, 32, 2019.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Aaron Mishkin, Frederik Kunstner, Didrik Nielsen, Mark Schmidt, and Mohammad E Khan. SLANG: Fast structured covariance approximations for Bayesian deep learning with natural gradient. *Proceedings of Advances in Neural Information Processing Systems*, 31:6248–6258, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *Proceedings of Advances in Neural Information Processing Systems*, 32:8024–8035, 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Kaare B Petersen and Michael S Pedersen. *The Matrix Cookbook*. Technical University of Denmark, 2012.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. *Artificial Intelligence and Statistics*, pp. 814–822, 2014.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017.

Dennis Ulmer, Christian Hardmeier, and Jes Frellsen. Prior and posterior networks: A survey on evidential deep learning methods for uncertainty estimation. *arXiv preprint arXiv:2110.03051*, 1(3), 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

A Appendix

You may include other additional sections here.

A.1 Linear Regression Posterior

This section completes the derivation of the linear regression posterior from Section 2.1. Recall from Equation (7) that

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = -\frac{\beta}{2}\sum_{n=1}^{N}\left(y_n - \boldsymbol{\theta}^\top \boldsymbol{x}_n\right)^2 + \frac{N}{2}\log\beta + \text{constant}. \tag{50}$$

Also, applying the logarithm to the prior in Equation (8),

$$\begin{aligned}\log p(\boldsymbol{\theta}) &= \frac{D}{2}\log\frac{\alpha}{2\pi} - \frac{\alpha}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta} \\ &= -\frac{\alpha}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta} + \frac{D}{2}\log\alpha + \text{constant}.\end{aligned} \tag{51}$$

Suppose that the prior precision α and the noise precision β are fixed. Then it follows from Bayes' rule that

$$\begin{aligned}\log p(\boldsymbol{\theta}|\mathcal{D}) &= \log\frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \\ &= \log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log p(\mathcal{D}) \\ &= -\frac{\beta}{2}\sum_{n=1}^{N}\left(y_n - \boldsymbol{\theta}^\top\boldsymbol{x}_n\right)^2 - \frac{\alpha}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta} + \text{constant} \\ &= -\frac{\beta}{2}\sum_{n=1}^{N}\left(y_n^2 - 2y_n\boldsymbol{\theta}^\top\boldsymbol{x}_n + (\boldsymbol{\theta}^\top\boldsymbol{x}_n)^2\right) - \frac{\alpha}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta} + \text{constant} \\ &= -\frac{\beta}{2}\sum_{n=1}^{N}y_n^2 + \beta\sum_{n=1}^{N}y_n\boldsymbol{\theta}^\top\boldsymbol{x}_n - \frac{\beta}{2}\sum_{n=1}^{N}\boldsymbol{\theta}^\top\boldsymbol{x}_n\boldsymbol{x}_n^\top\boldsymbol{\theta} - \frac{\alpha}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta} + \text{constant} \\ &= \Big(\beta\sum_{n=1}^{N}y_n\boldsymbol{x}_n\Big)^\top\boldsymbol{\theta} - \frac{1}{2}\boldsymbol{\theta}^\top\Big(\alpha\boldsymbol{I} + \beta\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^\top\Big)\boldsymbol{\theta} + \text{constant} \\ &= \boldsymbol{b}^\top\boldsymbol{\theta} - \frac{1}{2}\boldsymbol{\theta}^\top\boldsymbol{A}\boldsymbol{\theta} + \text{constant},\end{aligned} \tag{52}$$

where

$$\boldsymbol{A} = \alpha \boldsymbol{I} + \beta \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top \quad \text{and} \quad \boldsymbol{b} = \beta \sum_{n=1}^{N} y_n \boldsymbol{x}_n. \tag{53}$$

Now, using a result from (Barber, 2012) to complete the square of Equation (52),

$$\begin{aligned} \log p(\boldsymbol{\theta}|\mathcal{D}) &= \boldsymbol{b}^\top \boldsymbol{\theta} - \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{A}\boldsymbol{\theta} + \text{constant} \\ &= -\frac{1}{2}\big(\boldsymbol{\theta} - \boldsymbol{A}^{-1}\boldsymbol{b}\big)^\top \boldsymbol{A}\big(\boldsymbol{\theta} - \boldsymbol{A}^{-1}\boldsymbol{b}\big) + \frac{1}{2}\boldsymbol{b}^\top \boldsymbol{A}^{-1}\boldsymbol{b} + \text{constant} \\ &= -\frac{1}{2}\big(\boldsymbol{\theta} - \boldsymbol{A}^{-1}\boldsymbol{b}\big)^\top \boldsymbol{A}\big(\boldsymbol{\theta} - \boldsymbol{A}^{-1}\boldsymbol{b}\big) + \text{constant} \\ &= -\frac{1}{2}\big(\boldsymbol{\theta} - \boldsymbol{m}\big)^\top \boldsymbol{S}^{-1}\big(\boldsymbol{\theta} - \boldsymbol{m}\big) + \text{constant}, \end{aligned} \tag{54}$$

where

$$\boldsymbol{m} = \boldsymbol{A}^{-1}\boldsymbol{b} \quad \text{and} \quad \boldsymbol{S} = \boldsymbol{A}^{-1}. \tag{55}$$

Hence, the posterior distribution of $\boldsymbol{\theta}$ is

$$p(\boldsymbol{\theta}|\mathcal{D}) = \mathcal{N}(\boldsymbol{m}, \boldsymbol{S}). \tag{56}$$

A.2 Quantitative Results of Linear Regression Posterior for Synthetic Data

Table 4: Distances between the true posterior distribution of the parameter vector of a linear regression model and the approximate FA posterior estimated using VIFA. Relative distances between the true and approximate means and covariances are shown. Each relative distance is the Frobenius norm of the difference between the true parameter and the approximate parameter divided by the Frobenius norm of the true parameter. Also shown is the 2-Wasserstein distance between the true posterior Gaussian distribution and the approximate Gaussian distribution, divided by the dimension of the distribution. Each distance is the mean value over ten trials with different random seeds, and standard errors are also given.

Rel. Dist. from Mean	**Rel. Dist. from Covariance**	**Scaled 2-Wasserstein Dist.**
0.0031 ± 0.0005	0.0983 ± 0.0129	0.0194 ± 0.0023

A.3 UCI Linear Regression Posterior Estimation

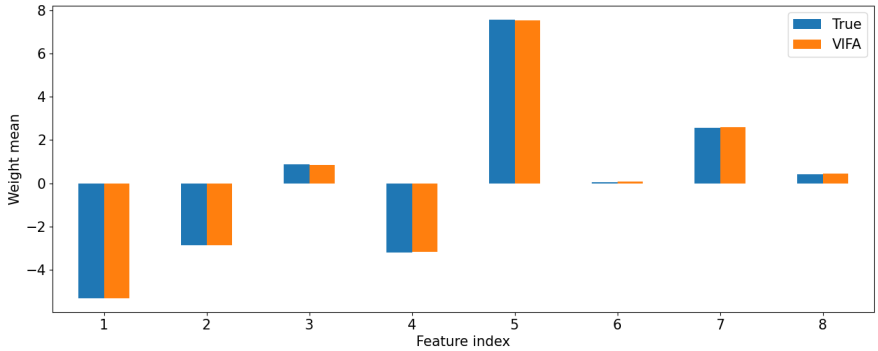
Table 5 shows the number of instances and input variables in each UCI regression dataset. Note that the original Energy Efficiency dataset has two target variables, heating load and cooling load, but in these experiments only the heating load was used.

Table 5: UCI regression datasets used in experiments.

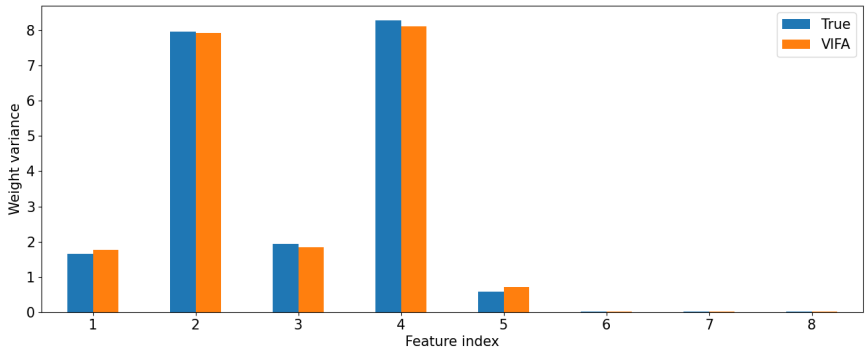
Dataset	**No. of Instances**	**No. of Input Variables**
Energy Efficiency	768	8
Boston Housing	506	13
Concrete Compressive Strength	1030	8
Yacht Hydrodynamics	308	6

Table 6 shows the hyperparameter values used in the experiments.

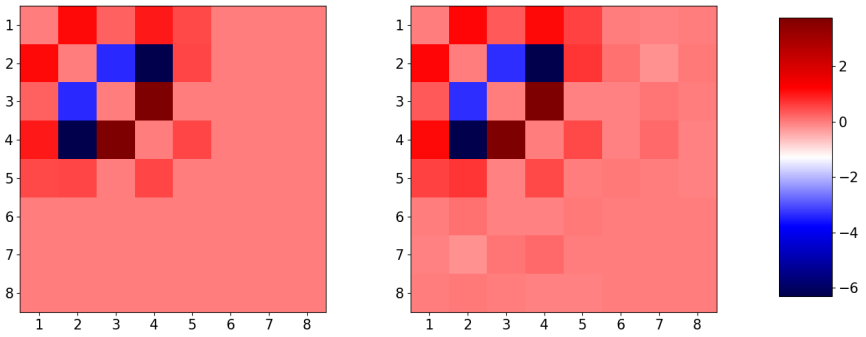
Recall that Section 4.2.1 presented qualitative comparisons of the true posterior of a linear regression model and the approximate posterior learned by VIFA for the Boston Housing and Yacht Hydrodynamics datasets. Figure 4 and Figure 5 show the same comparisons for the Energy Efficiency and Concrete Compressive Strength datasets, respectively. Finally, Table 7 contains quantitative measures of the difference between the true and estimate posteriors for all four UCI datasets.



(a) Comparison of true and estimated posterior means

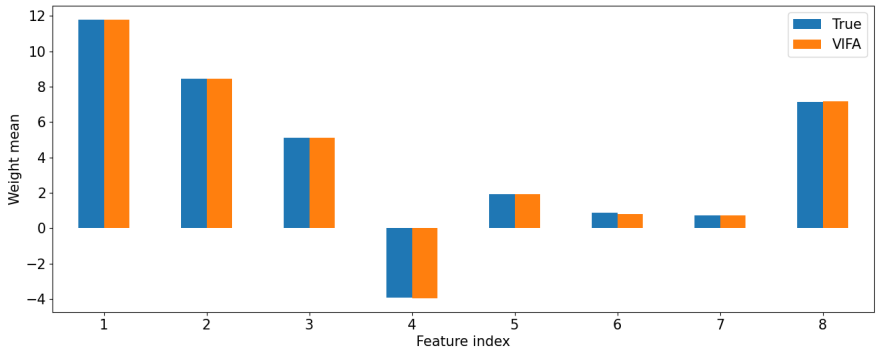


(b) Comparison of true and estimated posterior variances

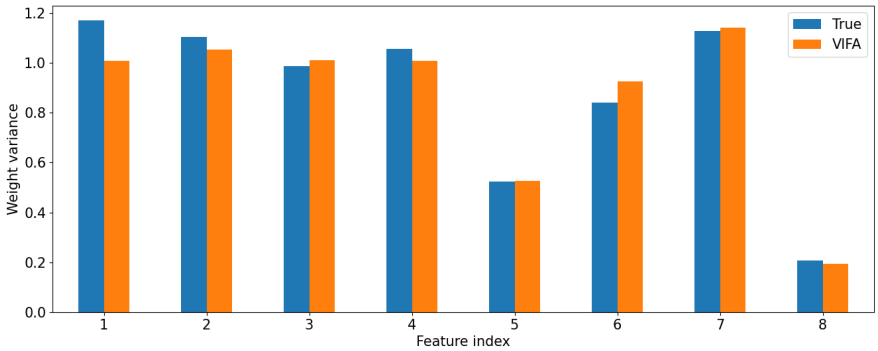


(c) Comparison of true and estimated posterior covariances

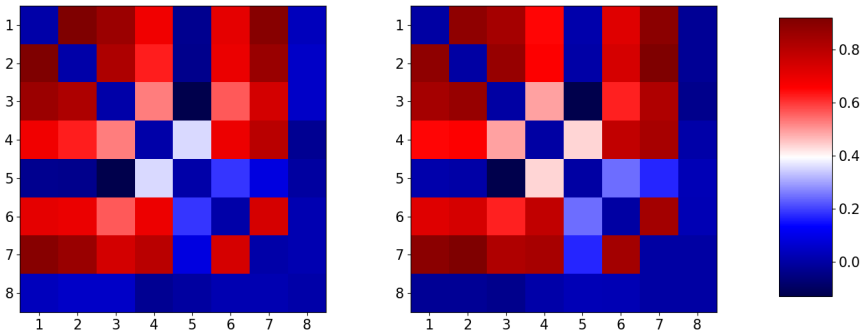
Figure 4: Comparison of the ground truth posterior of a linear regression model fit to the Energy Efficiency dataset, and the approximate posterior learned by VIFA. Variances and covariance are plotted separately due to the difference in their magnitude. In plot (c), the diagonal entries of the covariance matrices have been set to zero.



(a) Comparison of true and estimated posterior means



(b) Comparison of true and estimated posterior variances



(c) Comparison of true and estimated posterior covariances

Figure 5: Comparison of the ground truth posterior of a linear regression model fit to the Concrete Compressive Strength dataset, and the approximate posterior learned by VIFA. Variances and covariance are plotted separately due to the difference in their magnitude. In plot (c), the diagonal entries of the covariance matrices have been set to zero.

Table 6: VIFA hyperparameters for UCI linear regression experiments.

Dataset	K	**Epochs**	M	L	η	α	β
Energy Efficiency	3	25,000	100	10	0.01	0.0608	0.1246
Boston Housing	3	25,000	100	10	0.001	0.2859	0.0429
Concrete Compressive Strength	3	20,000	100	10	0.01	0.0254	0.0101
Yacht Hydrodynamics	3	45,000	100	10	0.01	0.0291	0.0114

Table 7: For each UCI dataset, the distance between the true posterior distribution of the parameter vector of a linear regression model and the approximate FA posterior estimated by VIFA. Relative distances between the true and approximate means and covariances are shown. Each relative distance is the Frobenius norm of the difference between the true parameter and the approximate parameter divided by the Frobenius norm of the true parameter. Also shown is the 2-Wasserstein distance between the true posterior Gaussian distribution and the approximate Gaussian distribution, divided by the number of dimensions in the dataset.

Dataset	**Rel. Dist. from Mean**	**Rel. Dist. from Covar.**	**Scaled Wasserstein Dist.**
Energy	0.0051	0.0421	0.0564
Boston	0.0262	0.3185	0.0468
Concrete	0.0047	0.0840	0.0278
Yacht	0.0435	0.0391	0.1210

A.4 UCI Neural Network Predictions

This section provides more details and results for the neural network prediction experiments in Section 4.2.2. Firstly, important details which are worth making explicit are the loss functions used in the experiments. When training, the average negative log-likelihood term used on line 12 of Algorithm 1 was set to

$$-\frac{1}{M}\sum_{(\boldsymbol{x},y)\in\mathcal{B}}\log\mathcal{N}\big(y;f(\boldsymbol{x};\boldsymbol{\theta}),\beta^{-1}\big), \tag{57}$$

where $\mathcal{B}$ is a mini-batch of M training examples, $f(\boldsymbol{x};\boldsymbol{\theta})$ is the prediction of the neural network for input $\boldsymbol{x}$ and parameters $\boldsymbol{\theta}$ and β is a hyperparameter.

When testing, the marginal log-likelihood of a test example $(\boldsymbol{x}, y)$ is

$$\begin{aligned}\log p(y|\boldsymbol{x},\alpha,\beta,\mathcal{D}) &= \log\int p(y|\boldsymbol{x},\boldsymbol{\theta},\beta)p(\boldsymbol{\theta}|\alpha,\beta,\mathcal{D})d\boldsymbol{\theta}\\ &= \log\int\mathcal{N}\big(y;f(\boldsymbol{x};\boldsymbol{\theta}),\beta^{-1}\big)p(\boldsymbol{\theta}|\alpha,\beta,\mathcal{D})d\boldsymbol{\theta},\end{aligned} \tag{58}$$

where $\mathcal{D}$ is the training data and $p(\boldsymbol{\theta}|\alpha,\beta,\mathcal{D})$ is the approximate posterior learned by VIFA or SLANG for hyperparameters α and β. This integral cannot be computed exactly. Instead, it was approximated by a sample average. Formally,

$$\begin{aligned}\log p(y|\boldsymbol{x},\alpha,\beta,\mathcal{D}) &\approx \log\left(\frac{1}{L}\sum_{l=1}^{L}\mathcal{N}\big(y;f(\boldsymbol{x};\boldsymbol{\theta}_l),\beta^{-1}\big)\right)\\ &= \log\left(\sum_{l=1}^{L}\mathcal{N}\big(y;f(\boldsymbol{x};\boldsymbol{\theta}_l),\beta^{-1}\big)\right)-\log L,\end{aligned} \tag{59}$$

where each $\boldsymbol{\theta}_l \sim p(\boldsymbol{\theta}|\alpha,\beta,\mathcal{D})$. This quantity was averaged over the full test set to approximate the average marginal log-likelihood. In addition, the test root mean squared error (RMSE) was also computed, where the squared error of a single test example is

$$\left(y-\frac{1}{L}\sum_{l=1}^{L}f(\boldsymbol{x};\boldsymbol{\theta}_l)\right)^2. \tag{60}$$

When computing these metrics, the Monte Carlo average size was set to $L = 100$.

Recall that Table 1 provides the tests metrics for the experiments averaged over 20 different train-test splits. The corresponding median results are given in Table 8.

Table 8: Median test results for prediction experiments with UCI regression datasets. Metrics are the average negative marginal log-likelihood (NMLL) and root mean squared error (RMSE). LR and NN refer to linear regression and neural network, respectively. The median results over 20 different train-test splits are shown. The best results on each row are highlighted in bold.

Metric	Dataset	VIFA-LR	VIFA-NN	SLANG-NN
NMLL	Energy	2.57	2.54	**1.19**
	Boston	2.88	**2.65**	2.68
	Concrete	3.76	3.33	**3.32**
	Yacht	3.62	2.38	**1.59**
RMSE	Energy	3.04	2.93	**1.19**
	Boston	4.31	**3.51**	6.47
	Concrete	10.27	**6.77**	9.03
	Yacht	9.07	2.59	**1.21**

A.5 Uncertainty on Medical Imaging

Table 9: Mean F1-scores of selective prediction under different uncertainty levels. Uncertainty scores calculated from two distinct approaches are employed, which are predictive entropy and model disagreement. Test samples are arranged in ascending order based on their uncertainty scores. 'Proportion of Samples' indicates the percentage of ordered samples used for testing. The results for F1 scores include standard errors. The best results in each column are highlighted in bold.

Proportion of Samples	Predictive Entropy	Model Disagreement
90%	0.946±0.007	0.945±0.007
80%	0.965±0.005	0.966±0.004
70%	0.975±0.004	0.976±0.004
60%	0.986±0.003	0.985±0.003
50%	**0.989±0.002**	**0.989±0.002**

Table 10: Mean AU-ROC scores of selective prediction under different uncertainty levels. Uncertainty scores calculated from two distinct approaches are employed, which are predictive entropy and model disagreement. Test samples are arranged in ascending order based on their uncertainty scores. 'Proportion of Samples' indicates the percentage of ordered samples used for testing. The results for AU-ROC include standard errors. The best results in each column are highlighted in bold.

Proportion of Samples	Predictive Entropy	Model Disagreement
90%	0.984+0.002	0.984+0.003
80%	0.989+0.002	0.989+0.002
70%	0.992+0.002	0.992+0.002
60%	0.994+0.002	0.994+0.002
50%	**0.995+0.001**	**0.995+0.002**