Deka, B., and Nguyen, L. H., and Goulet, J-A. (2023).*Analytically tractable heteroscedastic uncertainty quantification in Bayesian neural networks for regression tasks. Neurocomputing. 127183, doi*

Analytically Tractable Heteroscedastic Uncertainty Quantification in Bayesian Neural Networks for Regression Tasks

Bhargob Deka* and Luong Ha Nguyen
Department of Civil, Geologic and Mining Engineering
Polytechnique Montreal, CANADA
and James-A. Goulet
Department of Civil, Geologic and Mining Engineering
Polytechnique Montreal, CANADA

January 12, 2024

Abstract

The *tractable approximate Gaussian inference* (TAGI) method allows for analytical parameter inference in Bayesian neural networks. In its current form, TAGI can only model homoscedastic aleatory uncertainty that is quantified by a constant error variance across the input covariate-domain. In this paper, we present the *approximate Gaussian variance inference* (AGVI) method that enables analytical inference of the error variance term as a Gaussian random variable. The combined framework regrouping TAGI and AGVI, referred to as TAGI-V, enables modeling heteroscedastic aleatory uncertainty in Bayesian neural networks. TAGI-V outperforms the homoscedastic version of TAGI in terms of predictive performance for the benchmark regression datasets. In comparison with other approximate inference methods, TAGI-V is an order of magnitude faster and exhibits a comparable or superior predictive performance.

Keywords: tractable approximate Gaussian inference, approximate Gaussian variance inference, aleatory uncertainty, closed-form inference, Gaussian multiplicative approximation

1 Introduction

Making informed decisions require quantifying the *predictive uncertainty* of machine learning models. This task involves modeling the epistemic uncertainty, which represents the lack of knowledge about a model's structure and parameters, and the aleatory uncertainty, which represents the inherent variability present in the data. By collecting more data, it is possible to reduce epistemic uncertainty, but not the aleatory one which can be further classified into *homoscedastic*, i.e., that remains constant over input values and *heteroscedastic*, i.e., that varies with input values.
Bayesian neural networks (BNN) [22, 25] provide a probabilistic framework for estimating the model's confidence in its parameters and predictions. However, exact Bayesian inference is computationally intractable for neural networks (NN) and as a result, many approximate inference methods have been proposed in the literature [1, 5, 16]. The *Hamiltonian Monte Carlo* (HMC) is a sampling method that is considered as the reference method for BNN [22, 12]. For most practical applications, the method lacks scalability and requires problem-specific parameter tuning [12].

*Corresponding author: bhargobdeka11@gmail.com

The *probabilistic backpropagation* (PBP) [12] uses expectation propagation to estimate the parameters in BNN. Unlike backpropagation, PBP computes the posterior distribution over the parameters by propagating backward the gradients of the marginal likelihood with respect to the parameters. The *Monte Carlo dropout* (MC-dropout) method uses *dropout* [30] as a Bayesian approximation for deriving an approximate predictive distribution. MC-dropout computes the predictive uncertainty by averaging over an ensemble of neural networks where each network is trained using dropout [16, 5]. MC-dropout is orders of magnitudes faster than PBP but requires hyperparameter tuning unlike PBP [5]. MC-dropout motivated researchers to look into ensemble methods in neural networks [16]. *Deep ensembles* [16] provided a simple and scalable method that combines ensembling techniques and adversarial training for obtaining improved predictive performance and out-of-distribution robustness [16].

Variational inference remains an active research area for BNN. Other notable works using variational inference include *Variational matrix Gaussian* (VMG) [17] and *probabilistic backpropagation with the matrix-variate Gaussian* (MVG) distribution (PBP-MV) [31]. VMG uses matrix-variate Gaussian (MVG) [10] priors over the weights compared to PBP which uses independent standard Gaussian priors. [34] identified that variational Bayes is limited in practical applications because of computational constraints and sensitivity to prior variances for weights. *Deterministic variational inference* (DVI) [34] was proposed to counter these limitations. DVI provides analytically tractable moment computation and an empirical Bayes [29] approach to automatically assign prior variances for the weights.

Another modern approach is the stochastic weight averaging Gaussian (SWAG) [18] that uses the information present within the SGD trajectory to approximate a Gaussian posterior distribution over the parameters. SWAG uses the solution provided by stochastic weight averaging (SWA) which are the mean and a low rank plus diagonal covariance matrix obtained from the SGD iterations to form the Gaussian distribution. Afterwards, sampling is carried out from this Gaussian distribution to perform *Bayesian model averaging* [6]. A recent approach that builds on this concept is the *subspace inference* (SI) [14] that also provides an alternative solution for performing inference in high-dimensional parameter space, especially for deep neural networks. The SI method is carried out by first constructing low-dimensional spaces for the parameters called the *subspace*, then performing posterior inference within these subspaces, and finally carrying out Bayesian model averaging leading to increase in accuracy and well-calibrated prediction uncertainty. Many different choices are available to select the type of subspace such as the principal component analysis (PCA) subspace, random subspace, or the mode-connected subspace [14]. While the posterior inference within the subspaces can be carried out either through variational inference or using MCMC methods such as the HMC or the *elliptical slice sampling* (ESS) [21]. Recently, [9] proposed the analytically *tractable approximate Gaussian inference* (TAGI) method which allows for analytical parameter inference in BNN, and which was shown to provide a competitive performance in comparison to networks trained with backpropagation [9, 23]. In its current form, TAGI can only model homoscedastic aleatory uncertainty quantified by a single error variance parameter that is constant across the input covariate-domain.

In this paper, we propose the analytically tractable *approximate Gaussian variance inference* (AGVI) method for modeling heteroscedastic aleatory uncertainty in the context of the TAGI framework. The method can be summarized in two key steps. First, we use the *Gaussian multiplicative approximation* (GMA) [9, 3] for establishing the relationship between the Gaussian random variables describing the error V, the square of the error V^2, and the expected value of V^2 to obtain the prior knowledge for the error variance σ_V^2. Second, we leverage these relationships and use the posterior knowledge of the error V to update our prior knowledge for σ_V^2. Using these two steps, the error variance can be inferred analytically as a Gaussian random variable. The proposed method, referred to as TAGI-V,

combines the TAGI framework with AGVI to model heteroscedastic aleatory uncertainty, thereby addressing one key limitation of the original TAGI method.
The paper is organized as follows: Section 1 provides an introduction and a survey of related works. Section 2 reviews the TAGI method for performing analytically tractable posterior inference of parameters in Bayesian neural networks. Section 3 presents the AGVI method for handling heteroscedastic uncertainty within the TAGI framework and Section 4 evaluates the performance of TAGI-V for regression tasks and provides a comparison with existing approximate inference methods.

2 Tractable Approximate Gaussian Inference

In this section, we review the TAGI method by [9] for obtaining the parameters' posterior probability density function (PDF) in Bayesian neural networks. Here, we summarize the key principles behind TAGI through a feedforward neural network (FNN) architecture for which the nomenclature and notations are as follows. We use lower case slanted letters for deterministic variables, upper case slanted letters for random variables, slanted lower case with bold font to denote vectors, and upright upper case with bold font for matrices. The typewriter style is used either for specific names or to represent the number of variables in a set, vector, or matrix. We consider a FNN with $\mathtt{L}$ hidden layers for learning the relationship between the input covariates $\boldsymbol{x} = [x_1\ x_2\ \cdots\ x_{\mathtt{X}}]^\intercal \in \mathbb{R}^{\mathtt{X}}$ and the observed system response $y \in \mathbb{R}^{\mathtt{Y}}$. Each j^{th} layer, $\forall j \in \{1, 2, \cdots, \mathtt{L}\}$ of this FNN consists of $\mathtt{A}$ hidden units $z_i^{(j)}, \forall i \in \{1, 2, \cdots, \mathtt{A}\}$ for which the corresponding activation units $a_i^{(j)} = \phi(z_i^{(j)})$ are obtained using an activation function $\phi(\cdot)$. The observation model describing the relationship between the observed system response y and the model outputs is given by

$$y = z^{(\mathtt{0})} + v, \quad v : V \sim \mathcal{N}(v; 0, \sigma_V^2), \tag{1}$$

where $z^{(\mathtt{0})} \in \mathbb{R}^{\mathtt{Y}}$ represents the vector of hidden units on the output layer ($\mathtt{0}$), and v represents the error with mean zero and variance σ_V^2. Figure 1 shows a graphical model representing the FNN for obtaining a single model output $z^{(\mathtt{0})}$ as a function of the input covariates $\boldsymbol{x}$. The green nodes represent the vector of hidden units and the directed arrows show the causal relationships between nodes. The parameters between any two layers j and $j+1$ are represented by $\boldsymbol{\theta}^{(j)},\ j \in \{1, 2, \cdots, \mathtt{L}\}$. The observation y, denoted by the blue node, is connected to the output unit $z^{(\mathtt{0})}$, and the error v to represent the model in Equation 1.

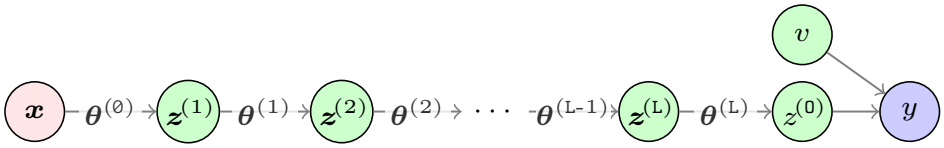


Figure 1: Representation of a FNN for obtaining a single model output $z^{(\mathtt{0})}$ as a function of the input covariates $\boldsymbol{x}$. The network comprises of $\mathtt{L}$ hidden layers having $\mathtt{A}$ hidden units in any layer $j \in \{1, 2, \cdots, \mathtt{L}\}$. The parameters between any two layers j and $j+1$ are represented by $\boldsymbol{\theta}^{(j)}$. The observation y, denoted by the blue node, is connected to the output unit $z^{(\mathtt{0})}$, and the error v to represent the model in Equation 1.

In a generic form, TAGI requires propagating uncertainties from the activation hidden units $\boldsymbol{A}^{(j)} \sim$

$\mathcal{N}(\boldsymbol{a}^{(j)}; \boldsymbol{\mu}_{\boldsymbol{A}}^{(j)}, \boldsymbol{\Sigma}_{\boldsymbol{A}}^{(j)})$ in hidden layer j to the i^{th} hidden unit in layer $j+1$ following

$$Z_i^{(j+1)} = \sum_{k=1}^{\mathtt{A}} W_{i,k}^{(j)} A_k^{(j)} + B_i^{(j)}, \tag{2}$$

where the parameters $W_{i,k}^{(j)}$ and $B_i^{(j)}$ are assumed to be Gaussian random variables. Equation 2 involves the product of pairs of weights W and activation units A for which the exact moments can be computed using the Gaussian multiplicative approximation (GMA) [9, 3], see Appendix A for further details. TAGI uses a 1^{st} order Taylor series approximation at the expected value of the hidden unit $\mu_{Z_i}^{(j+1)}$ to maintain the analytical tractability when propagating uncertainty through the activation function. Moreover, in order to maintain a linear computational complexity with respect to the number of parameters, first, the method employs a diagonal covariance matrix for both the parameters $\boldsymbol{\theta}$ and the hidden units $\boldsymbol{Z}^{(j)}$. Second, it uses a recursive layer-wise Gaussian inference approach that relies on the conditional independence between the hidden units $\boldsymbol{Z}^{(j-1)}$ and $\boldsymbol{Z}^{(j+1)}$ given that the hidden units $\boldsymbol{z}^{(j)}$ are known and the parameters $\boldsymbol{\theta}$ are independent for each layer.
A two-fold inference step is used for obtaining the posterior moments for the parameters $\boldsymbol{\theta}$ and hidden units $\boldsymbol{Z}^{(j)}$. First, the posterior expected value and diagonal covariance matrix for the output units are obtained such that

$$f(\boldsymbol{z}^{(\mathtt{0})}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{z}^{(\mathtt{0})}; \boldsymbol{\mu}_{\boldsymbol{Z}^{(\mathtt{0})}|\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{Z}^{(\mathtt{0})}|\boldsymbol{y}}), \tag{3}$$

$$\boldsymbol{\mu}_{\boldsymbol{Z}^{(\mathtt{0})}|\boldsymbol{y}} = \boldsymbol{\mu}_{\boldsymbol{Z}^{(\mathtt{0})}} + \boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(\mathtt{0})}}^{\intercal} \boldsymbol{\Sigma}_{\boldsymbol{Y}}^{-1} \left(\boldsymbol{y} - \boldsymbol{\mu}_{\boldsymbol{Y}}\right), \tag{4}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{Z}^{(\mathtt{0})}|\boldsymbol{y}} = \boldsymbol{\Sigma}_{\boldsymbol{Z}^{(\mathtt{0})}} - \boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(\mathtt{0})}}^{\intercal} \boldsymbol{\Sigma}_{\boldsymbol{Y}}^{-1} \boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(\mathtt{0})}}, \tag{5}$$

where $\boldsymbol{\mu}_{\boldsymbol{Z}^{(\mathtt{0})}}$ and $\boldsymbol{\mu}_{\boldsymbol{Y}}$ are the mean vectors for $\boldsymbol{Z}^{(\mathtt{0})}$ and $\boldsymbol{Y}$; $\boldsymbol{\Sigma}_{\boldsymbol{Z}^{(\mathtt{0})}}$ is the prior covariance matrix for $\boldsymbol{Z}^{(\mathtt{0})}$, $\boldsymbol{\Sigma}_{\boldsymbol{Y}}$ is the prior covariance matrix for $\boldsymbol{Y}$, and $\boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(\mathtt{0})}}$ is the prior cross-covariance between $\boldsymbol{Z}^{(\mathtt{0})}$ and $\boldsymbol{Y}$. Second, the Rauch-Tung-Striebel (RTS)[27] update equations are used to perform the layer-wise backward inference pass using the posterior knowledge for the output units obtained by Equations 3 – 5. The posterior moments for the parameters $\boldsymbol{\theta}$ and the hidden units $\boldsymbol{Z}$ are provided in Appendix A. The posterior inference for the parameters $\boldsymbol{\theta}$ is done recursively using either a single observation or a batch of them. Moreover, the recursive inference process is done over multiple epochs $\mathtt{E}$ to overcome the limitation of having weakly informative priors for the initial parameters $\boldsymbol{\theta}^{(\mathtt{0})}$.
TAGI provides an analytically tractable method for inferring the posterior expected values and diagonal covariance matrix of a neural network's parameters. However, a key limitation is that the original version of TAGI is only applicable to homoscedastic cases for which the error variance σ_V^2 is considered as a hyperparameter that needs to be identified separately from the analytical parameter inference. This limitation incurs a substantial computational burden and undermines the capacity to accurately characterize aleatory uncertainties.

3 Approximate Gaussian Variance Inference

In this section, we introduce the analytically tractable *approximate Gaussian variance inference* (AGVI) method for inferring the error variance σ_V^2 as a Gaussian random variable. Overall, this new method can be summarized in two steps; first, we establish the relationship between the Gaussian random variables describing the error V, the square of the error V^2, and the expected value of V^2, to obtain the prior knowledge for the error variance σ_V^2; second, we leverage these relationships, and use the posterior PDF of the error V to obtain the posterior knowledge for σ_V^2. Here, we describe AGVI

through the univariate case having a single σ_V^2 associated with the observation unit Y, whereas without modification, it can be extended for the multivariate case with diagonal covariance matrix $\boldsymbol{\Sigma_V} = \text{diag}(\boldsymbol{\sigma}_V^2)$.
For the first step, we employ GMA, introduced in Section 2 to model V^2 using a Gaussian random variable such that

$$f(v^2) = \mathcal{N}(v^2; \mu_{V^2}, \sigma_{V^2}^2). \tag{6}$$

Using Equation 6, and given that V is zero-mean, the variance for V is by definition equal to the expected value for V^2, so that the PDF of V is described by

$$f(v) = \mathcal{N}(v; 0, \mu_{V^2}). \tag{7}$$

Following Proof B.1 presented in Appendix B, we show that the PDF of V^2 can be described using only the expected value μ_{V^2} such that

$$f(v^2|\mu_{V^2}) = \mathcal{N}(v^2; \mu_{V^2}, 2\mu_{V^2}^2), \tag{8}$$

where using GMA, the variance for V^2 is $\sigma_{V^2}^2 = 2\mu_{V^2}^2$. In order to maintain the analytical tractability, we assume the hyperparameter μ_{V^2} in Equation 8 to be a Gaussian random variable described by $\overline{V^2} \sim \mathcal{N}(\overline{v^2}; \mu_{\overline{V^2}}, \sigma_{\overline{V^2}}^2)$, using which, the Equation 8 can be re-written as

$$f(v^2|\overline{v^2}) \quad = \quad \mathcal{N}(v^2; \overline{v^2}, 2(\overline{v^2})^2), \tag{9}$$

where the PDF for V^2 is defined using the knowledge of $\overline{v^2}$. Figure 2 shows the graphical model representing the relationships between the random variables V, V^2, and $\overline{V^2}$, denoted by the green nodes. The causal relationship between the nodes $\overline{V^2}$ and V^2 is shown by the directed arrow as defined in Equation 9. The undirected solid line between the nodes V^2 and V represents the one-to-one relationship between their moments as defined by Equations 6 & 7. Hence, we obtain the prior predictive PDF for V, through the prior predictive PDF for V^2. In Proof B.2 presented in Appendix B, we detail the procedure to obtain the prior predictive PDF for V^2, for which the moments are given by

$$\mu_{V^2} = \mu_{\overline{V^2}}, \tag{10}$$

$$\sigma_{V^2}^2 = 3\sigma_{\overline{V^2}}^2 + 2\mu_{\overline{V^2}}^2. \tag{11}$$

Using Equations 7 & 10, the prior predictive PDF of V is described by

$$f(v) = \mathcal{N}(v; 0, \mu_{\overline{V^2}}),$$

where the variance of V is $\sigma_V^2 = \mu_{\overline{V^2}}$. Therefore, we obtain the prior knowledge for σ_V^2 by using the prior PDF for $\overline{V^2}$ described by its moments $\mu_{\overline{V^2}}$ and $\sigma_{\overline{V^2}}^2$.
In order to obtain the moments for $\overline{V^2}$ as a function of the input covariates $\boldsymbol{x}$, we use a neural network having a two-headed output layer where the first output unit $Z^{(\mathtt{0})}$ models the expected value of the system response and the second output unit is $\overline{V^2}$. This network setup allows for handling *heteroscedastic aleatory uncertainty* in regression tasks. Figure 3 shows the graphical model for a feedforward network where the two output units are the random variables $Z^{(\mathtt{0})}$ and $\overline{V^2}$. The output unit for $\overline{V^2}$ has its own set of parameters $\boldsymbol{\theta}_{\overline{V^2}}^{(\mathtt{L})}$ connected to the last hidden layer $\mathtt{L}$ as shown in red. This graphical model also shows the causal relationship between the random variables Y, $Z^{(\mathtt{0})}$, and

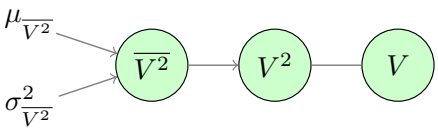


Figure 2: Graphical model representing the relationship between the random variables V, V^2, and $\overline{V^2}$, denoted by the green nodes. The causal relationship between the nodes $\overline{V^2}$ and V^2 is shown by the directed arrow as demonstrated by Equation 9. The undirected solid line between the nodes V^2 and V represents the one-to-one relationship between their moments as defined by Equations 6 & 7.

V, as per the observation model, along with the graphical model shown in Figure 2. This structure presents the flow of information from $\overline{V^2}$ to V, and then to the observation unit Y. Note that in order to restrict the possible values for $\overline{v^2}$ to the positive domain, we need to transform its value through an exponential activation function $\exp(\cdot)$. This lead to a log-normal PDF for which the moments are available in [7, §4.2.1]. The moments for the transformed random variable $\widetilde{\overline{V^2}}$ are

$$\begin{aligned} \mu_{\widetilde{\overline{V^2}}} &= \exp(\mu_{\overline{V^2}} + 0.5\sigma^2_{\overline{V^2}}), \\ \sigma^2_{\widetilde{\overline{V^2}}} &= \exp(2\mu_{\overline{V^2}} + \sigma^2_{\overline{V^2}}) \cdot (\exp(\sigma^2_{\overline{V^2}}) - 1), \\ \operatorname{cov}(\overline{V^2}, \widetilde{\overline{V^2}}) &= \sigma^2_{\overline{V^2}} \cdot \mu_{\widetilde{\overline{V^2}}}, \end{aligned} \tag{12}$$

where $\operatorname{cov}(\overline{V^2}, \widetilde{\overline{V^2}})$ is the covariance between the transformed random variable $\widetilde{\overline{V^2}}$ and the original $\overline{V^2}$.

In order to infer σ_V^2 following the structure provided in Figure 3, we need to divide the process in two steps. Considering that the vector of output units is $\boldsymbol{h} = [z^{(0)}\ v]^\intercal$, we first define the posterior PDF $f(\boldsymbol{h}|y)$ as

$$f(\boldsymbol{h}|y) = \frac{f(\boldsymbol{h}, y)}{f(y)} = \mathcal{N}(\boldsymbol{h}; \boldsymbol{\mu}_{\boldsymbol{H}|y}, \boldsymbol{\Sigma}_{\boldsymbol{H}|y}). \tag{13}$$

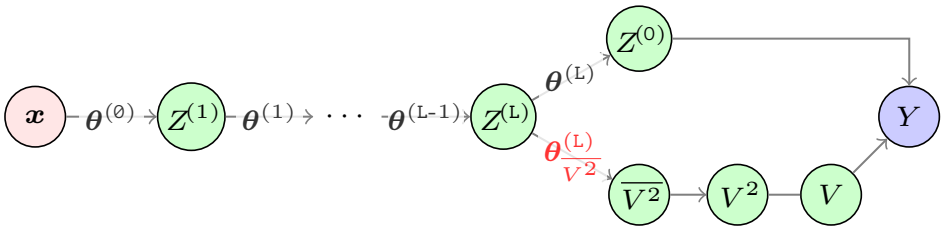


Figure 3: Network architecture for TAGI having a two-headed output layer for obtaining the random variables $Z^{(0)}$ and $\overline{V^2}$ as a function of the input covariates $\boldsymbol{x}$. The output unit for $\overline{V^2}$ has an additional set of parameters $\boldsymbol{\theta}^{(L)}_{\overline{V^2}}$ connected to the last hidden layer L as shown in red. Also, it shows the extended graphical model representing the causal relationship between the random variables Y, $Z^{(0)}$, and V, as per the observation model, along with the graphical model shown in Figure 2.

Using the same Gaussian conditional equations as presented in Section 2, the posterior mean vector $\boldsymbol{\mu}_{\boldsymbol{H}|y}$ and covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{H}|y}$ are obtained by

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{H}|y} &= \boldsymbol{\mu}_{\boldsymbol{H}} + \frac{\boldsymbol{\Sigma}_{\boldsymbol{H}Y}}{\sigma_Y^2}(y - \mu_Y), \\ \boldsymbol{\Sigma}_{\boldsymbol{H}|y} &= \boldsymbol{\Sigma}_{\boldsymbol{H}} - \frac{\boldsymbol{\Sigma}_{\boldsymbol{H}Y} \cdot \boldsymbol{\Sigma}_{\boldsymbol{H}Y}^{\intercal}}{\sigma_Y^2}. \end{aligned} \tag{14}$$

Second, the current knowledge of $\overline{V^2}$ is updated using the posterior PDF $f(v|y)$ derived from Equation 14. Following Proof B.3 presented in Appendix B, the posterior moments for V^2 and $\overline{V^2}$ are given by

$$\begin{aligned} \mu_{V^2|y} &= \mu_{V|y}^2 + \sigma_{V|y}^2, \\ \sigma_{V^2|y}^2 &= 2(\sigma_{V|y})^4 + 4\sigma_{V|y}^2\mu_{V|y}^2, \\ \mu_{\overline{V^2}|y} &= \mu_{\overline{V^2}} + k(\mu_{V^2|y} - \mu_{V^2}), \\ \sigma_{\overline{V^2}|y}^2 &= \sigma_{\overline{V^2}}^2 + k^2(\sigma_{V^2|y}^2 - \sigma_{V^2}^2), \\ k &= \frac{\sigma_{\overline{V^2}}^2}{\sigma_{V^2}^2}. \end{aligned}$$

The updated knowledge for $Z^{(\mathtt{0})}$ and $\overline{V^2}$ are used to obtain the posterior for the parameters and the hidden units using the layer-wise recursive inference detailed in Section 2. By combining the frameworks of TAGI and AGVI, we can perform the analytically tractable inference of a neural network's parameters as well as the error variance, and enable heteroscedastic aleatory uncertainty quantification for regression tasks. We refer to this method as TAGI-V.

Note that the overall complexity of TAGI is $\mathcal{O}(\mathtt{A}^2)$, where $\mathtt{A}$ signifies the number of activation units present within each layer [9]. This complexity scales linearly with the number of hidden layers $\mathtt{L}$. In TAGI-V, we simply modify the output layer that includes $\mathtt{A}$ additional weight terms associated with the output unit for the singular error variance term. This additional step has a complexity of $\mathcal{O}(\mathtt{A})$ and hence, do not change the overall complexity of the network, i.e., $\mathcal{O}(\mathtt{A}^2)$.

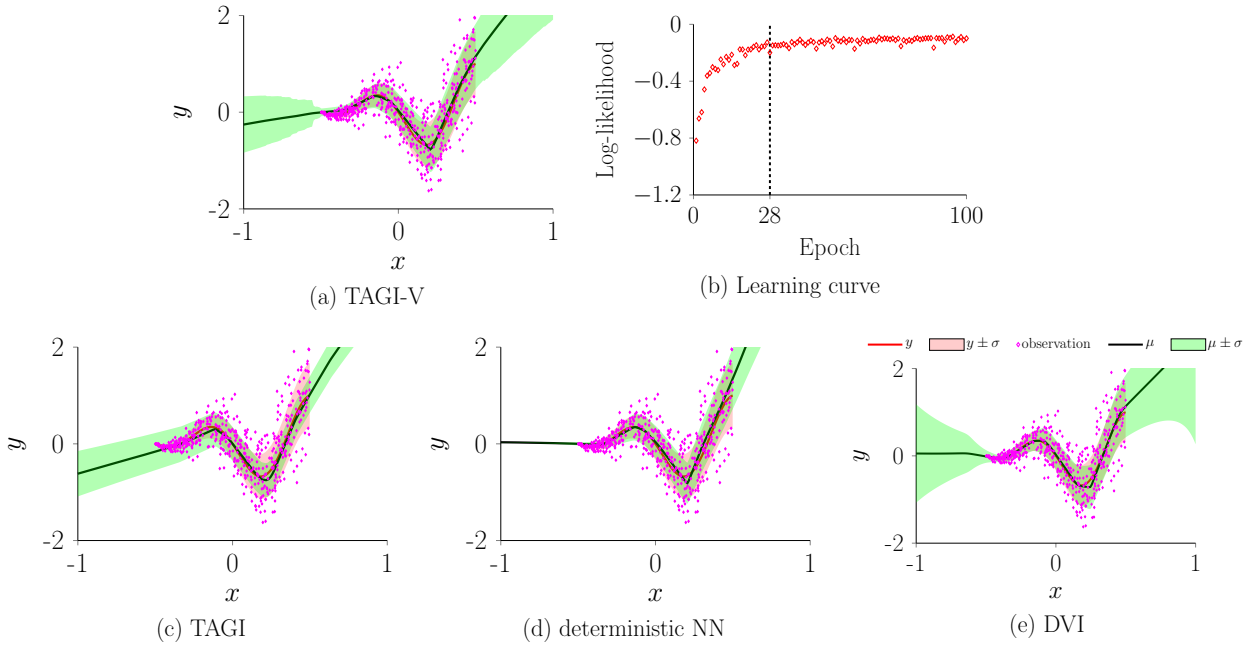
4 Experiments

In this section, we perform experiments using the TAGI-V method for a 1D toy problem and for the UCI regression benchmark datasets [12, 32]. We provide a comparative analysis with the approximate inference methods used for regression tasks in existing literature.

4.1 Toy Problem

We apply TAGI-V to the 1D heteroscedastic regression problem for $y = -(x+0.5)\cdot\sin(3\pi x)+v$, such that $V \sim \mathcal{N}(0, \sigma_V^2)$, where the heteroscedastic error variance is modeled by $\sigma_V^2 = 0.45 \cdot (x+0.5)^2$. We generate 500 observations sampled uniformly in the range $[-0.5, 0.5]$ and use a two-layer network of 128 hidden units with ReLU activation function. The prior weights and bias are initialized using He's approach [11] and the inference is carried out using one observation at a time. We compare our method with the homoscedastic original version of TAGI using the implementation from [9], a deterministic heteroscedastic neural network trained with backpropagation [26], and deterministic variational inference (DVI) as implemented by [34]. Figure 4 compares the true function used to

generate the data with the predictions described by the expected values and $\pm\sigma$ confidence regions for each of these methods. The results in Figure 4(a) show that TAGI-V is capable of handling



(a) TAGI-V (b) Learning curve (c) TAGI (d) deterministic NN (e) DVI

Figure 4: Application of TAGI-V to a toy problem having a heteroscedastic error variance modeled using $\sigma_V^2 = 0.45 \cdot (x + 0.5)^2$. The training data points are plotted in magenta, the true function $y = -(x + 0.5) \cdot \sin(3\pi x) + v$, and the $\pm 1\sigma$ confidence region are shown by the red solid line and red shaded region, and the model predictions and their $\pm 1\sigma$ confidence regions are shown by the black solid line and green shaded area, respectively. In (a) we show the predictions using TAGI-V and in (b) we show the learning curve showing the evolution of the test log-likelihood as a function of the number of epochs. Figures (c)-(e) show the predictions using the original version of TAGI [9], a deterministic neural network [26], and DVI [34], respectively.

heteroscedastic error variance in the region where training data is available and is able to extrapolate the confidence region outside the training region in order to represent a lack of knowledge. Figure 4(b) shows the learning curve representing the evolution of the test log-likelihood as a function of the number of epochs. We identify the optimal epoch for the toy problem to be $\texttt{E} = 28$ using an early-stopping procedure and patience of 5 epochs. In Figure 4(c), we can see that the original TAGI method is not capable of handling heteroscedastic error variance and can only model a constant one, both within as well as beyond the training region. Figure 4(d) shows the predictions using a deterministic NN trained with backpropagation that can, to a certain extent, model heteroscedastic uncertainty where training data is available but fails to extrapolate the uncertainty beyond the training region. As shown by Figure 4(e), DVI is capable of handling heteroscedastic error variance and is better than TAGI-V at extrapolating the confidence interval to represent the lack of knowledge outside the training region. However, DVI requires orders of magnitudes more epochs (20000) for achieving convergence compared to TAGI-V (28) as shown in Figure 4(b).

4.2 Regression Benchmarks

In this section, we evaluate the performance of TAGI-V for both the small and large UCI regression datasets. For both datasets, we compare our method with some of the approximate inference methods used for regression tasks in the literature.

4.2.1 Small UCI Datasets

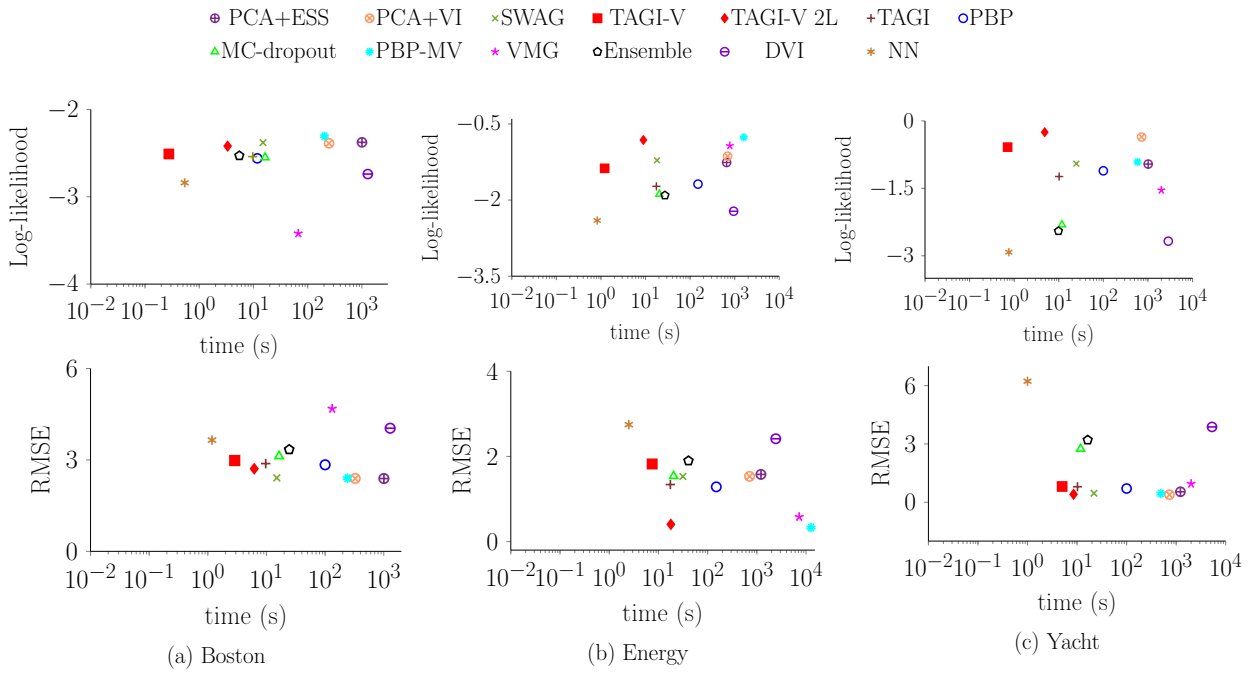
In this section, we compare our method for the small UCI regression datasets with some of the existing baselines such as PBP [12], MC-dropout [5, 20], deterministic NN [26], ensemble of neural networks [16], DVI [34], PBP-MV [31], VMG [17], the stochastic weight averaging Gaussian (SWAG) [18], two types of subspace inference (SI) methods [14] namely, principal component analysis with elliptical slice sampling (PCA+ESS) and the other with variational inference (PCA + VI), along with the original version of TAGI [9]. This study is conducted using the experimental set-up provided by [12] which has been extensively used in the literature to evaluate the predictive capacity of the approximate inference methods. The implementation details for each method is provided in Appendix C.

Each dataset is randomly split into a training and a test set having 90% and 10% of the data, respectively, and we maintain the same indices in both sets for each method. We consider 20 data splits to compute the average test performance. For comparative purposes, we consider a network having a single hidden layer of 50 units for each dataset, except for Protein, which has 100 units. For TAGI-V, the data is normalized, a ReLU activation function is used, and the batch size considered is $\mathtt{B} = 32$. The prior covariances for weights and bias are initialized using He's approach [11]. Note that the scaling factor associated with the prior variance of the weights for the mean as well as the error variance are tuned for each dataset for proper initialization. The details regarding the grid-search procedure employed are provided in Appendix D. Moreover, an early-stopping procedure is used to identify the optimal number of epochs for each dataset by dividing the training set into an $80 - 20\%$ train-validation sets, see details in Appendix D.

For comparison among methods, the *epoch setting* is common in the literature that provides the predictive accuracy of a particular method as it learns with each new epoch. However, for practical applications, it is equally important to understand the amount of training time required by each method to achieve a particular accuracy. Therefore, we introduce the *time setting*, where unlike the epoch setting, we obtain the learning curves as a function of the average training time for each method. The results for the epoch setting are provided in Appendix E that are obtained by training each method for a fixed 100 epochs.

Figure 5 provides the maximum test log-likelihood and the minimum RMSE values for the datasets Boston, Energy, and Yacht obtained under the time setting. In order to accommodate for the large disparities with respect to training time between the methods, we present the horizontal axis in log-scale (base 10). In addition, we have also included the maximum log-likelihood values, the minimum RMSE values as well as the learning curves for PBP-MV and VMG as obtained directly from [31]. The figures showing the maximum log-likelihood and the minimum RMSE values for the other datasets under the time setting are presented in Appendix E. The learning curves for all the datasets under both settings are provided in Appendix F.

While comparing the average training time per epoch between the methods, we find that TAGI-V is ≈ 100 times faster than PCA+ESS, PCA+VI, PBP-MV and VMG, ≈ 10 times faster than PBP, and ≈ 3 times faster than Ensemble. Even though methods such as MC-dropout and SWAG have an average training time per epoch equivalent to TAGI-V, they require more time than TAGI-V to converge to their best accuracy as presented in Figure 5. For MC-dropout, this observation



(a) Boston (b) Energy (c) Yacht

Figure 5: Comparison for the maximum test log-likelihood and the minimum test RMSE values obtained with TAGI-V and the existing baselines for the datasets a) Boston, b) Energy, and c) Yacht under the time setting. The horizontal axis represents the training time (s) in log scale (base 10) and the vertical axis represents either the test log-likelihood (top figure) or the test RMSE (bottom figure) in linear scale. The marked points are : TAGI-V (■), PBP (○) [12], MC-dropout (△) [5], DVI (⊖) [34], deterministic NN (✱) [26], Ensemble (⬠) [16], TAGI (✚) [9], TAGI-V 2L (♦) that represents a TAGI-V network of two layers and 100 hidden nodes, PBP-MV (●) [31], VMG (★) [17], SWAG (×) [18], the two subspace inference methods, i.e., PCA+ESS (⊕) , and PCA+VI (⊗) [14].

is validated by the fact that it takes orders of magnitude more epochs (≈ 4000) to achieve the predictive performance as mentioned in the article by [20]. Moreover, MC-dropout requires tuning several hyperparameters for e.g., learning rate, dropout rate, and precision parameter that also has a high computational requirement. Similarly, SWAG also has several hyperparameters such as learning rates for both SGD and SWAG, the maximum number of SWAG models, the number of Monte Carlo (MC) samples in forward pass, and the number of epochs for running SGD initially, that either requires manual tuning or finding the optimal values through cross-validation [18, 14]. The deterministic NN is the fastest method by a factor of ≈ 1.5 compared with TAGI-V, but it provides a poor predictive accuracy as shown in Figure 5. The average training time per epoch for all the methods is provided in Appendix G.

In terms of absolute predictive performance, PBP-MV reports the best test log-likelihood and test RMSE among all methods in 5 out of the 9 datasets (Boston, Concrete, Wine, Kin8nm, and Protein), VMG outperforms PBP-MV and all other methods in Power, DVI provides the best results in Naval, while TAGI-V performs the best in Yacht. Even though PBP-MV and VMG produce state-of-the-art results in terms of predictive accuracy, these methods take orders of magnitudes more computational time than all other methods, except DVI that also has a similar order of computational demand. The subspace inference methods are also capable of providing good accuracy especially for the

datasets Concrete, Kin8nm, Naval, Protein but takes more computational time than even PBP-MV to achieve those results. Even though there is not a clear distinction between the two subspace methods, we found that PCA+VI is faster and performs marginally better than PCA+ESS in most of the small UCI datasets. On the other hand, SWAG is a relatively low-cost method in terms of its training time that can also provide accuracy similar to the subspace methods. However, the large number of hyperparameters limits its practical applicability compared to TAGI-V that has at most two hyperparameters, i.e., the gain parameters for the prior variances of weights and biases, and converges to its best accuracy much faster than SWAG.
In order to further demonstrate the superiority of TAGI-V in comparison with approaches such as PBP-MV that can reach a high accuracy at the expense of computational efficiency, we have tested the performance of our method by using 2 layers and 100 hidden nodes (TAGI-V 2L) as represented by the red diamond in Figures 5. The two-layer network not only outperform PBP-MV and VMG for test log-likelihood in all datasets except in Concrete and Wine, but while doing so remains at two orders of magnitude faster than these methods. For the test RMSE, the two-layer network exceeds the performance of PBP-MV and VMG in 5 out of 9 datasets (Concrete, Energy, Kin8nm, Yacht and Power). These results demonstrate that TAGI-V can achieve the state-of-the-art predictive accuracy by using a larger neural network architecture while still being computationally faster than most approaches. Hence, in regard to practical applicability, TAGI-V has the potential to achieve superior predictive accuracy on regression tasks considering that it is one of the fastest methods not only in terms of its training time per epoch but also the minimum training time it requires to converge to those results.
It is to be mentioned that the results here for TAGI-V were obtained using the TAGI MATLAB library [4] on a 12 cores 3GHz CPU. Recently, [24] have released the *cuTAGI* (C++ & CUDA) along with the pyTAGI (Python) libraries with the TAGI-V formulation. The benchmark tests performed show that these new libraries are more than two times faster than the MATLAB library for the small UCI regression datasets.

4.2.2 Large UCI Datasets

In this section, TAGI-V is tested for the large UCI regression datasets: *elevators*, *keggdirected*, *keggundirected*, *pol*, and *skillcraft*. The experimental framework used here is provided by [32]. Each dataset is randomly split into a training and test set having 90% and 10% of the data. The experiment is carried out for 10 splits in order to compute the average test RMSE and normalized test log-likelihood values [14].
On all datasets, except skillcraft, a network of five hidden layers is used where the number of hidden units in each layer are: $[1000, 1000, 500, 50, 2]$. For skillcraft, a smaller network is used such that the structure is: $[1000, 500, 50, 2]$. A ReLU activation unit is used and the batch size considered is $\mathtt{B} = 10$. The prior variances for weights and bias are initialized using He's approach [11]. Similarly to the procedure for the small UCI datasets, the gain parameters associated with the prior variances of $\boldsymbol{\theta}$ for all the hidden layers and the output layer connected to the mean and error variance are tuned using a grid-search procedure. Also, an early-stopping procedure is used to stop the training process with a fixed patience of 3 epochs. The hyperparameters and the computational time for each dataset are provided in Appendix H.
Table 1 provides the normalized test log-likelihood and RMSE values for the large UCI regression datasets. A direct comparison is made with Bayesian inference methods such as the best performing *subspace inference method* [14] i.e., principal component analysis combined with variational inference (PCA+VI), along with the *stochastic weight averaging-Gaussian* (SWAG) [18], the *Bayesian final layers* (NL) [28], and the *stochastic gradient descent* (SGD) obtained from [14]. The results for

TAGI-V are averaged over 3 random seeds. The test log-likelihood values show that TAGI-V performs better than all other methods in 4 out of the 5 datasets. The TAGI-V method is also competitive for RMSE values where it provides the best results in 2 out of the 5 datasets, i.e., Elevators and KeggD, while it is second best for KeggU and Pol. Both PCA+ VI and NL outperform the others in two datasets.

Table 1: Normalized log-likelihood and RMSE comparison between TAGI-V and the existing Bayesian inference methods [14, 18, 28] on large UCI regression datasets (Rank legend: **first**). The $\pm\sigma$ represents one standard deviation computed over 10 splits.

Metrics	Datasets	TAGI-V	PCA + VI (SI)	SWAG	NL	SGD
log-likelihood	Elevators	$\mathbf{-0.319 \pm 0.027}$	-0.325 ± 0.019	-0.374 ± 0.021	-0.698 ± 0.039	-0.538 ± 0.108
	KeggD	$\mathbf{1.287 \pm 0.112}$	1.085 ± 0.031	1.080 ± 0.035	0.935 ± 0.265	1.012 ± 0.154
	KeggU	$\mathbf{0.793 \pm 1.034}$	0.757 ± 0.028	0.749 ± 0.029	0.670 ± 0.038	0.602 ± 0.224
	Pol	0.718 ± 0.108	$\mathbf{1.764 \pm 0.271}$	1.533 ± 1.084	-2.84 ± 0.226	1.073 ± 0.858
	Skillcraft	$\mathbf{-0.981 \pm 0.031}$	-1.179 ± 0.033	-1.180 ± 0.033	-1.002 ± 0.050	-1.162 ± 0.032
RMSE	Elevators	$\mathbf{0.087 \pm 0.002}$	0.088 ± 0.001	0.088 ± 0.001	0.101 ± 0.002	0.103 ± 0.035
	KeggD	$\mathbf{0.128 \pm 0.004}$	$\mathbf{0.128 \pm 0.029}$	0.129 ± 0.029	0.134 ± 0.036	0.132 ± 0.017
	KeggU	0.126 ± 0.004	0.160 ± 0.043	0.160 ± 0.043	$\mathbf{0.120 \pm 0.003}$	0.186 ± 0.034
	Pol	2.737 ± 0.135	$\mathbf{2.50 \pm 0.068}$	3.11 ± 0.070	4.380 ± 0.853	3.900 ± 6.003
	Skillcraft	0.445 ± 0.135	0.293 ± 0.015	0.293 ± 0.015	$\mathbf{0.253 \pm 0.011}$	0.288 ± 0.014

5 Conclusion

The TAGI-V method proposed in this article provides an analytical method for handling heteroscedastic aleatory uncertainty and overcomes the limitation of the original version of TAGI which can only handle homoscedastic error variance. The method proposed combines the TAGI framework that allows for the analytical inference of the parameters' posterior PDF in Bayesian neural networks, and AGVI that enables analytical inference of the error variance term as a Gaussian random variable. TAGI-V outperforms the original version of TAGI in terms of test log-likelihood while providing similar test RMSE values for all small UCI regression datasets. In comparison with other approximate inference methods, TAGI-V is an order of magnitude faster and exhibits comparable or superior predictive performance. The TAGI-V framework was also tested for large UCI regression datasets for which it provided better log-likelihood values in four out of the five datasets compared to the benchmark methods while providing competitive performance in terms of RMSE.

6 Acknowledgement

This project was financially supported by research grants from Natural Sciences and Engineering Research Council of Canada (NSERC), Hydro-Québec (HQ), Hydro-Québec's Research Institute (IREQ), and Institute For Data Valorization (IVADO).

A Appendix A

A.1 GMA Equations

In this section, we present the Gaussian multiplicative approximation (GMA), an approximate method for computing the first two moments for the product of two Gaussian random variables [9, 3]. Consider $\boldsymbol{X} = [X_1\ X_2\ X_3\ X_4]^\intercal$, a vector of Gaussian random variables such that $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$, with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Using the Gaussian moment generating function or 2^{nd} order Taylor series expansion, the following equations can be derived for the product of any two Gaussian random variables such that

$$
\begin{aligned}
\mathbb{E}[X_1X_2] &= \mu_1\mu_2 + \text{cov}(X_1, X_2),\\
\text{var}(X_1X_2) &= \sigma_1^2\sigma_2^2 + \text{cov}(X_1, X_2)^2\\
&\quad + 2\text{cov}(X_1, X_2)\mu_1\mu_2\\
&\quad + \sigma_1^2\mu_2^2 + \sigma_2^2\mu_1^2,\\
\text{cov}(X_3, X_1X_2) &= \text{cov}(X_1, X_3)\mu_2 + \text{cov}(X_2, X_3)\mu_1,\\
\text{cov}(X_1X_2, X_3X_4) &= \text{cov}(X_1, X_3)\text{cov}(X_2, X_4)\\
&\quad + \text{cov}(X_1, X_4)\text{cov}(X_2, X_3)\\
&\quad + \text{cov}(X_1, X_3)\mu_2\mu_4\\
&\quad + \text{cov}(X_1, X_4)\mu_2\mu_3\\
&\quad + \text{cov}(X_2, X_3)\mu_1\mu_4\\
&\quad + \text{cov}(X_2, X_4)\mu_1\mu_3.
\end{aligned}
$$

The detailed procedure to develop these equations are provided in the articles by [9, 3]. Using the GMA equations, an approximate Gaussian feedforward network (AG-FNN) can be constructed, where we can propagate moments analytically from the input layer to the output layer shown by

$$
\{\boldsymbol{\mu}_{\boldsymbol{Z}}^{(0)}, \boldsymbol{\Sigma}_{\boldsymbol{Z}}^{(0)}, \boldsymbol{\Sigma}_{\boldsymbol{Z},\boldsymbol{\theta}}^{(0)}\} = \text{AG-FNN}(\boldsymbol{x}, \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}),
$$

where $\boldsymbol{x}$ represents the input covariates, while $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ represents the mean vector and covariance matrix for the total set of parameters in the neural network.

A.2 Posterior Moments for parameters and hidden units

The posterior moments for the parameters $\boldsymbol{\theta}$ and the hidden units $\boldsymbol{Z}$ are obtained following

$$
\begin{aligned}
f(\boldsymbol{Z}|\boldsymbol{y}) &= \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}),\\
\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}} &= \boldsymbol{\mu}_{\boldsymbol{Z}} + \boldsymbol{J}_{\boldsymbol{Z}}\left(\boldsymbol{\mu}_{\boldsymbol{Z}^+|\boldsymbol{y}} - \boldsymbol{\mu}_{\boldsymbol{Z}^+}\right),\\
\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}} &= \boldsymbol{\Sigma}_{\boldsymbol{Z}} + \boldsymbol{J}_{\boldsymbol{Z}}\left(\boldsymbol{\Sigma}_{\boldsymbol{Z}^+|\boldsymbol{y}} - \boldsymbol{\Sigma}_{\boldsymbol{Z}^+}\right)\boldsymbol{J}_{\boldsymbol{Z}}^\intercal,\\
\boldsymbol{J}_{\boldsymbol{Z}} &= \boldsymbol{\Sigma}_{\boldsymbol{Z}\boldsymbol{Z}^+}\boldsymbol{\Sigma}_{\boldsymbol{Z}^+}^{-1},\\
f(\boldsymbol{\theta}|\boldsymbol{y}) &= \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\boldsymbol{\theta}|\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}|\boldsymbol{y}}),\\
\boldsymbol{\mu}_{\boldsymbol{\theta}|\boldsymbol{y}} &= \boldsymbol{\mu}_{\boldsymbol{\theta}} + \boldsymbol{J}_{\boldsymbol{\theta}}\left(\boldsymbol{\mu}_{\boldsymbol{Z}^+|\boldsymbol{y}} - \boldsymbol{\mu}_{\boldsymbol{Z}^+}\right)\\
\boldsymbol{\Sigma}_{\boldsymbol{\theta}|\boldsymbol{y}} &= \boldsymbol{\Sigma}_{\boldsymbol{\theta}} + \boldsymbol{J}_{\boldsymbol{\theta}}\left(\boldsymbol{\Sigma}_{\boldsymbol{Z}^+|\boldsymbol{y}} - \boldsymbol{\Sigma}_{\boldsymbol{Z}^+}\right)\boldsymbol{J}_{\boldsymbol{\theta}}^\intercal\\
\boldsymbol{J}_{\boldsymbol{\theta}} &= \boldsymbol{\Sigma}_{\boldsymbol{\theta}\boldsymbol{Z}^+}\boldsymbol{\Sigma}_{\boldsymbol{Z}^+}^{-1},
\end{aligned}
$$

where the short-hand notations for the parameters and hidden units in the j^{th} and the subsequent layer are $\{\boldsymbol{\theta}^+, \boldsymbol{Z}^+\} \equiv \{\boldsymbol{\theta}^{(j+1)}, \boldsymbol{Z}^{(j+1)}\}$ and $\{\boldsymbol{\theta}, \boldsymbol{Z}\} \equiv \{\boldsymbol{\theta}^{(j)}, \boldsymbol{Z}^{(j)}\}$. The details regarding these statements can be found in [9].

B Appendix B

B.1 PDF of V^2 only depends on the mean parameter μ_{V^2}

Proof. Given that V is zero-mean Gaussian, the moments of V can be derived using a Gaussian moment generating function so that

$$\begin{aligned} \mu_V &= \mathbb{E}[V] = 0, \\ \sigma_V^2 &= \mathbb{E}[(V-\mu_V)^2] = \mathbb{E}[V^2] - \mathbb{E}[V]^2, \\ &= \mathbb{E}[V^2], \end{aligned} \tag{15}$$

$$\mathbb{E}[V^4] = 3\mathbb{E}[V^2]^2, \tag{16}$$

where using the 4^{th} central moment from Equation (16) we can define,

$$\text{var}(V^2) = \mathbb{E}[(V^4)] - \mathbb{E}[V^2]^2 = 2\mathbb{E}[V^2]^2. \tag{17}$$

With the approximation that $V^2 \sim \mathcal{N}(v^2; \mu_{V^2}, \sigma_{V^2}^2)$ is a Gaussian random variable, the PDF for V^2 can be fully defined by its mean and variance,

$$\begin{aligned} \mu_{V^2} =& \mathbb{E}[V^2], \\ \sigma_{V^2}^2 =& \text{var}(V^2) = 2\mathbb{E}[V^2]^2, \end{aligned}$$

where by using Equation (17), the variance $\text{var}(V^2)$ can also be expressed in terms of the expected value $\mathbb{E}[V^2]$. Hence, the PDF of V^2 only depends on the unknown parameter μ_{V^2} so that

$$\begin{aligned} f(v^2|\mu_{V^2}, \sigma_{V^2}^2) &\equiv f(v^2|\mu_{V^2}), \\ &= \mathcal{N}(v^2; \mu_{V^2}, 2\mu_{V^2}^2). \end{aligned} \tag{18}$$

□

B.2 Prior predictive PDF for V^2 and V

Proof. Let us consider that the expected value μ_{V^2} is described by the random variable $\overline{V^2} : \overline{v^2} \in (0, \infty)$ for which

$$f(\overline{v^2}) \sim \mathcal{N}(\overline{v^2}; \mu^{\overline{V^2}}, (\sigma^{\overline{V^2}})^2). \tag{19}$$

Using Equations (18) and (19), we can rewrite the PDF of V^2 as

$$f(v^2|\overline{v^2}) = \mathcal{N}(v^2; \overline{v^2}, 2(\overline{v^2})^2). \tag{20}$$

The Gaussian random variables V^2 and $\overline{V^2}$ can be represented by

$$V^2 = \overline{V^2} + \sqrt{2}\,\overline{V^2}\epsilon, \quad \epsilon \sim \mathcal{N}(0,1) \tag{21}$$

$$\overline{V^2} = \mu^{\overline{V^2}} + \sigma^{\overline{V^2}}\zeta, \quad \zeta \sim \mathcal{N}(0,1). \tag{22}$$

Using Equations (21)) and (22), the mean and variance of the prior predictive PDF of V^2 are given by

$$\begin{aligned}
\mathbb{E}[V^2] &= \mathbb{E}[\overline{V^2} + \sqrt{2}\mathbb{E}[\overline{V^2}\,\epsilon], \quad (\overline{V^2}\,\epsilon \to 0)\\
&= \mu_{\overline{V^2}}, \qquad (23)\\
\mathrm{var}(V^2) &= \mathrm{var}(\overline{V^2}) + 2\ \mathrm{var}(\overline{V^2}\ \epsilon),\\
&= \sigma^2_{\overline{V^2}} + 2(\mathrm{var}(\overline{V^2}) \cdot \mathrm{var}(\epsilon) \to 1 + \mathrm{var}(\epsilon) \to 1 \cdot \mathbb{E}[\overline{V^2}]^2),\\
&= 3\sigma^2_{\overline{V^2}} + 2\mu^2_{\overline{V^2}}, \qquad (24)
\end{aligned}$$

where using the GMA Equations in Section A.1,

$$\mathrm{var}(\overline{V^2}\ \epsilon) = \mathrm{var}(\overline{V^2}) \cdot \mathrm{var}(\epsilon) + \mathrm{var}(\epsilon) \cdot \mathbb{E}[\overline{V^2}]^2.$$

Using Equations (15) and (23), the error variance term is given by

$$\sigma^2_V = \mu_{\overline{V^2}}. \qquad (25)$$

□

B.3 Posterior moments for $\overline{V^2}$

Proof. The joint PDF $f(\overline{v^2}, v^2|y)$ is

$$f(\overline{v^2}, v^2|y) = \mathcal{N}\left(\begin{pmatrix} \overline{v^2} \\ v^2 \end{pmatrix}; \boldsymbol{\mu}_{\overline{V^2},V^2}, \boldsymbol{\Sigma}_{\overline{V^2},V^2}\right), \qquad (26)$$

having a mean vector $\boldsymbol{\mu}_{\overline{V^2},V^2}$ and a covariance matrix $\boldsymbol{\Sigma}_{\overline{V^2},V^2}$ defined as

$$\begin{aligned}
\boldsymbol{\mu}_{\overline{V^2},V^2} &= \begin{bmatrix} \mu^{\overline{V^2}} & \mu^{V^2} \end{bmatrix}^\intercal,\\
\boldsymbol{\Sigma}_{\overline{V^2},V^2} &= \begin{bmatrix} \sigma^2_{\overline{V^2}} & \mathrm{cov}(\overline{V^2}, V^2) \\ \mathrm{cov}(V^2, \overline{V^2}) & \sigma^2_{V^2} \end{bmatrix},
\end{aligned}$$

The covariance term between V^2 and $\overline{V^2}$ in Equation (26) is obtained using the GMA equations so that

$$\begin{aligned}
\mathrm{cov}(V^2, \overline{V^2}) &= \mathrm{cov}(V^2, \overline{V^2}),\\
&= \mathrm{cov}(\overline{V^2} + \sqrt{2}\ \overline{V^2}\epsilon, \overline{V^2}),\\
&= \mathrm{var}(\overline{V^2}) + \sqrt{2}\mathrm{cov}(\overline{V^2}\epsilon, \overline{V^2}),\\
&= \mathrm{var}(\overline{V^2}) + \sqrt{2}(\mathrm{cov}(\overline{V^2}, \overline{V^2})\mathbb{E}[\epsilon] \to 0\\
&+ \mathrm{cov}(\epsilon, \overline{V^2}) \to 0\, \mathbb{E}[\overline{V^2}]),\\
&= \sigma^2_{\overline{V^2}}.
\end{aligned}$$

Using the Gaussian conditional properties and Equation (26), the PDF $f(\overline{v^2}|v^2, y)$ is

$$f(\overline{v^2}|v^2, y) = \mathcal{N}(\overline{v^2}, \mu^2_{\overline{V^2}|V^2}, \sigma^2_{\overline{V^2}|V^2}), \qquad (27)$$

where the conditional mean and variance are

$$\mu^2_{\overline{V^2}|V^2} = \mu_{\overline{V^2}} + k(v^2 - \mu_{V^2}), \tag{28}$$

$$\sigma^2_{\overline{V^2}|V^2} = \sigma^2_{\overline{V^2}} - k^2\sigma^2_{V^2},$$

$$k = \frac{\text{cov}(V^2, \overline{V^2})}{\sigma^2_{V^2}},$$

$$= \frac{\sigma^2_{\overline{V^2}}}{\sigma^2_{V^2}}. \tag{29}$$

Using Equation (27), (29) is rewritten as

$$f(\overline{v^2}|y) = \int \mathcal{N}(\mu^2_{\overline{V^2}|V^2}, \sigma^2_{\overline{V^2}|V^2}) \cdot \mathcal{N}(\mu_{V^2|y}, \sigma^2_{V^2|y}) dv^2,$$
$$= \mathcal{N}(\overline{v^2}; \mu_{\overline{V^2}|y}, \sigma^2_{\overline{V^2}}),$$

where using the Gaussian distribution properties for random mean and constant variance, the posterior mean and variance of $\overline{V^2}$ are

$$\mu_{\overline{V^2}} = \mathbb{E}\left[\mu_{\overline{V^2}} + k(V^2|y - \mu_{V^2})\right],$$
$$= \mu_{\overline{V^2}} + k(\mu_{V^2|y} - \mu_{V^2}),$$
$$\sigma^2_{\overline{V^2}} = \sigma^2_{\overline{V^2}} - k^2\sigma^2_{V^2} + k^2\text{var}(V^2|y),$$
$$= \sigma^2_{\overline{V^2}} + k^2(\sigma^2_{V^2} - \sigma^2_{V^2}). \tag{30}$$

□

C Appendix C

C.1 Hyperparameters for the approximate inference methods

In this section we present the hyperparameters for the approximate inference methods used for comparison in this study that are: PBP [12], MC-dropout [5, 20], deterministic NN [26], ensemble of neural networks [16], DVI [34], PBP-MV [31], VMG [17], SWAG [18], the two subspace inference methods PCA+ESS and PCA+VI [14], and TAGI [9].

The code for PBP is provided in [13]. PBP tunes its hyperparameter, namely the precision (λ), automatically for which a gamma hyper-prior is considered with the scale and inverse scale parameters, $\alpha^\lambda_0 = \beta^\lambda_0 = 6$. The batch size used is $\texttt{B} = 1$.

The code for MC-dropout is provided in [19]. The optimal values for the hyperparameters namely, the dropout rate d and the precision parameter τ are identified for each data split by performing grid-search over a range of (d, τ) pairs. The batch size used is $\texttt{B} = 128$ and the Adam optimizer used is with the default learning rate built-in Keras [2]. The number of forward iterations (T) used for obtaining predictive uncertainty are 10^4.

For the Ensembles method, we implement an ensemble of 5 neural networks trained with random initialization of weights and bias having a two-headed output layer that provides the mean and error variance. We use a softplus activation function as suggested in [16]. The batch size used is $\texttt{B} = 128$, the learning rate is 0.01, and the epsilon parameter for adversarial training is set to 1%. We also implemented a deterministic neural network with only one model having the same set of

hyperparameters as implemented for the Ensembles.
For DVI, the set of hyperparameters for the toy problem are provided in the code by [33], while for the regression benchmark they are provided by [34]. We have not implemented PBP-MV and VMG but directly used the results presented in the article by [31].
The code for the subspace inference (SI) methods and stochastic weight averaging Gaussian (SWAG) are provided in [15]. The hyperparameters involved in the code are the learning rates for stochastic gradient descent (SGD) and SWAG, the starting epoch for SWAG, the number of Monte Carlo (MC) samples drawn in forward pass, the maximum number of SWAG models, the weight decay rate, and the temperature. The values for the set of hyperparameters are different for each dataset including the batch size and the maximum number of epochs used for training which are taken directly from the original code without modification. However, the number of MC samples and the maximum number of SWAG models are fixed to 30 and 20, respectively for the small UCI regression datasets.
The implementation code for TAGI is provided in the Github repository by [8]. The original work identifies the optimal error variance σ_V^2 using a 5 fold cross-validation. The batch size used is $\mathtt{B} = 10$, and the prior covariance for bias is initialized using $0.01 \cdot \boldsymbol{I}$, and for the weights using the Xavier's approach.

D Appendix D

D.1 Initialization of the neural network's parameters

Using He's approach [11], the prior covariance matrix for the weights $\boldsymbol{W}^{(j)}$ and bias $\boldsymbol{B}^{(j)}$, in any hidden layer j are given by $\boldsymbol{\Sigma}_{\boldsymbol{W}}^{(j)} = \boldsymbol{\Sigma}_{\boldsymbol{B}}^{(j)} = \frac{1}{n_{j-1}} \cdot \boldsymbol{I}$, where n_{j-1} represents the number of hidden units in the previous layer $j-1$. We modify He's approach by introducing a scaling factor α [26] for the weights such that the new prior covariance matrix is $\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{W}}^{(j)} = \frac{\alpha}{n_{j-1}} \cdot \boldsymbol{I}$ that are initialized according to the data. Also, a different scaling factor β is considered for the weights connected to the 2^{nd} output node providing the error-variance. The best pair of values for these two hyperparameters (α and β) are identified using a validation set and a grid-search procedure. The list of hyperparameter values over which the grid-search is carried out are

$$\begin{aligned} \text{patience} &: \{3, 5, 10\}, \\ \alpha &: \{0.1, 0.5, 1\}, \\ \beta &: \{0.1, 0.01, 0.001\}, \end{aligned}$$

where patience is the hyperparameter for the early-stopping procedure. Table 2 shows the optimal values for the hyperparameters used for each dataset. Figures 6 and 7 shows the learning curves for TAGI-V using both the original and the modified He's approach for parameter initialization. The results shows that the modified He's approach provides better predictive accuracy in most datasets. Figure 8 shows the optimal epoch for each dataset obtained using early stopping procedure.

Table 2: Optimized set of hyperparameters identified using grid-search procedure. The parameters α and β, and patience are associated with the modified He's approach and early-stopping procedure, respectively. The grid-search is carried out using a validation set obtained from the original training set by a $80-20$ split ratio. The total computational time (in sec) required for the grid-search procedure is also provided.

Datasets	α	β	patience	Total Time (in sec)
Boston	0.5	0.01	5	591.63
Concrete	0.5	0.01	5	1225.83
Energy	0.5	0.01	5	420.30
Kin8nm	1	0.01	5	3562.74
Naval	0.5	0.01	3	19570.24
Power	0.5	0.001	10	4805.74
Protein	0.5	0.1	10	23299.31
Wine	0.1	0.01	10	353.71
Yacht	0.1	0.1	5	648.60

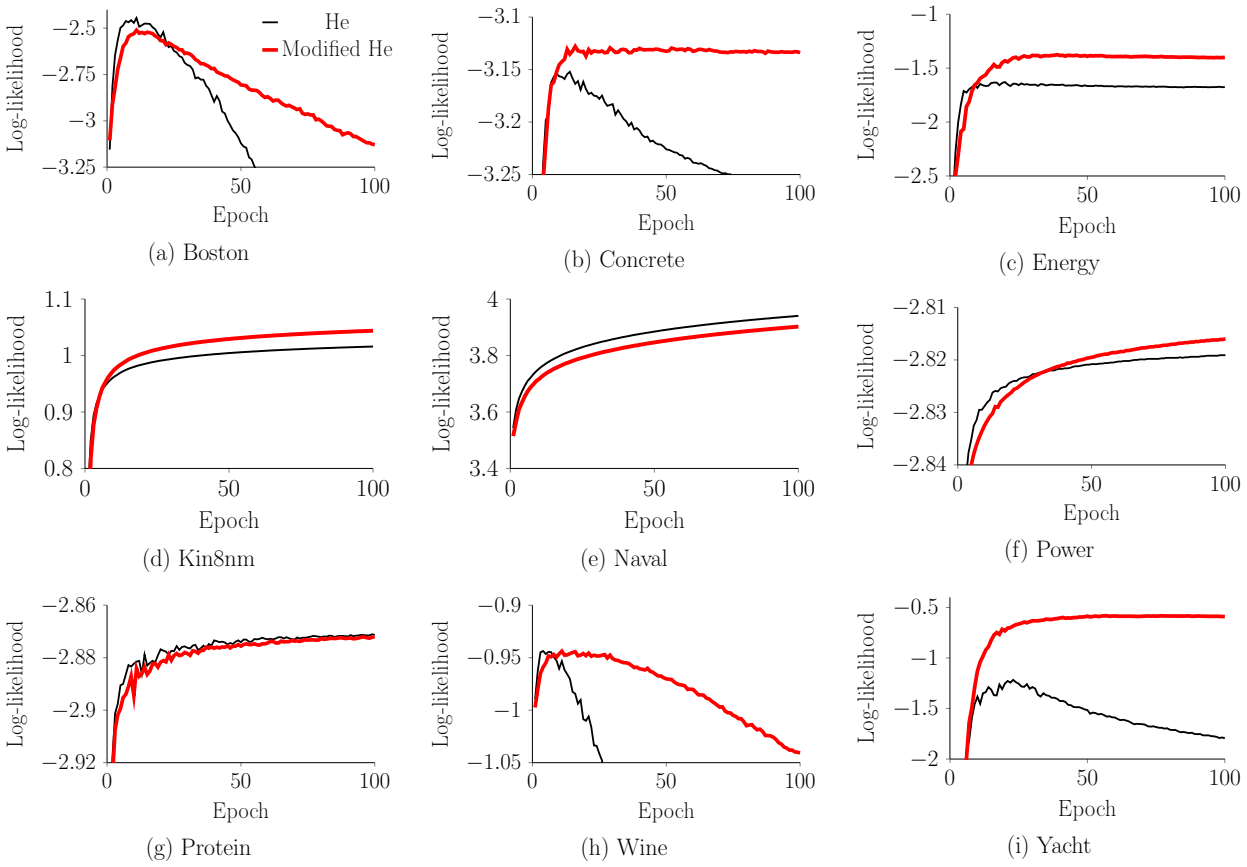


Figure 6: The learning curves for test log-likelihood showing the comparative performance between the original and modified He's approach for parameter initialization. The black and red solid line represents the performance using the original and modified He's approach, respectively. In the original He's approach [11], we set the scaling factors, $\alpha = \beta = 1$, but for the modified He's approach we tune the scaling factor [26] for each dataset using a grid-search procedure over possible set of hyperparameter values.

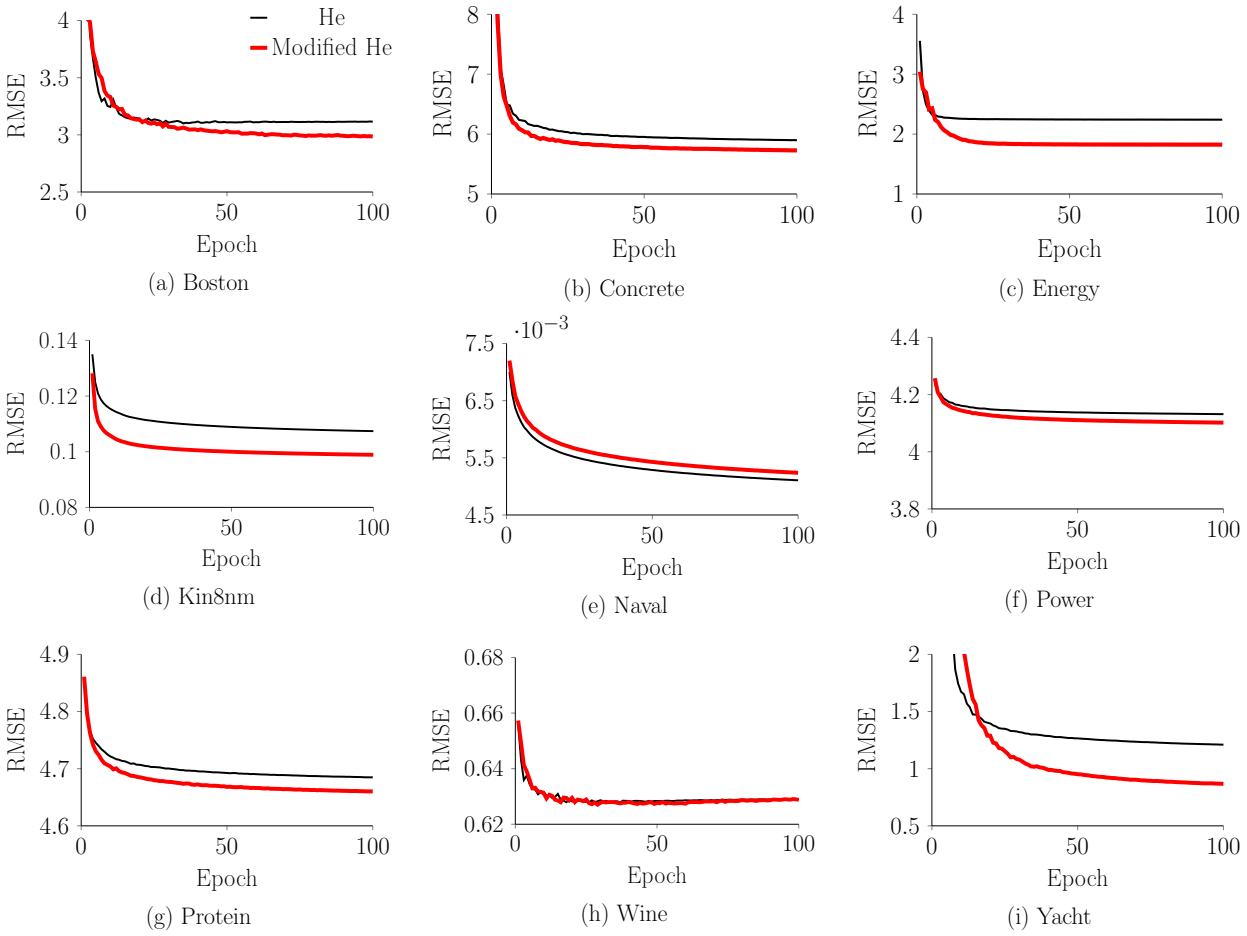


Figure 7: The learning curves for test RMSE showing the performance using the original and modified He's approach for parameter initialization. The black and red solid line represents the performance using the original and modified He's approach, respectively. In the original He's approach [11], we set the scaling factors, $\alpha = \beta = 1$, but for the modified He's approach, we tune the scaling factor [26] for each dataset using a grid-search procedure over possible set of hyperparameter values.

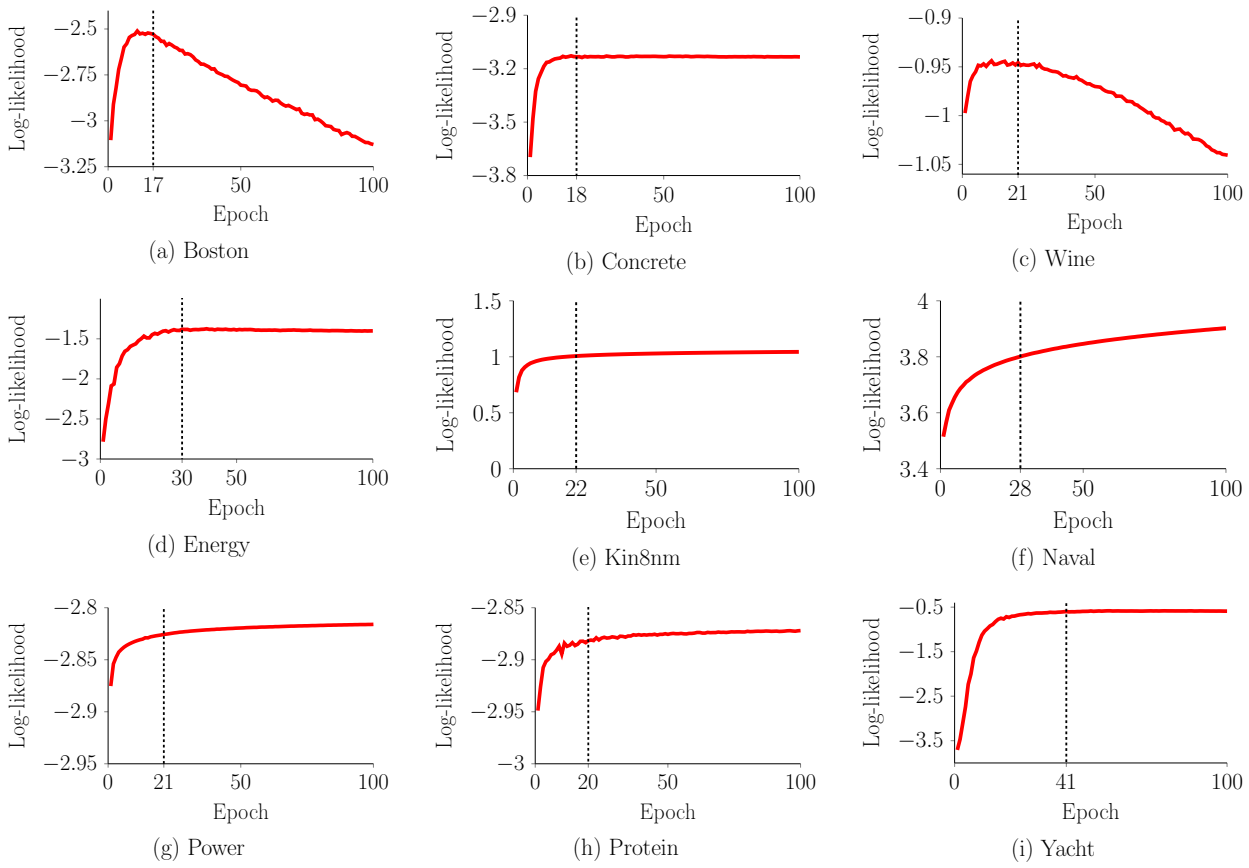


Figure 8: The learning curves for TAGI-V under epoch setting showing the test log-likelihood for the small UCI regression datasets. The optimal epoch is highlighted by the black dotted line found using early-stopping procedure.

E Appendix E

E.1 Plots showing the maximum test log-likelihood and minimum test RMSE under the epoch and time setting

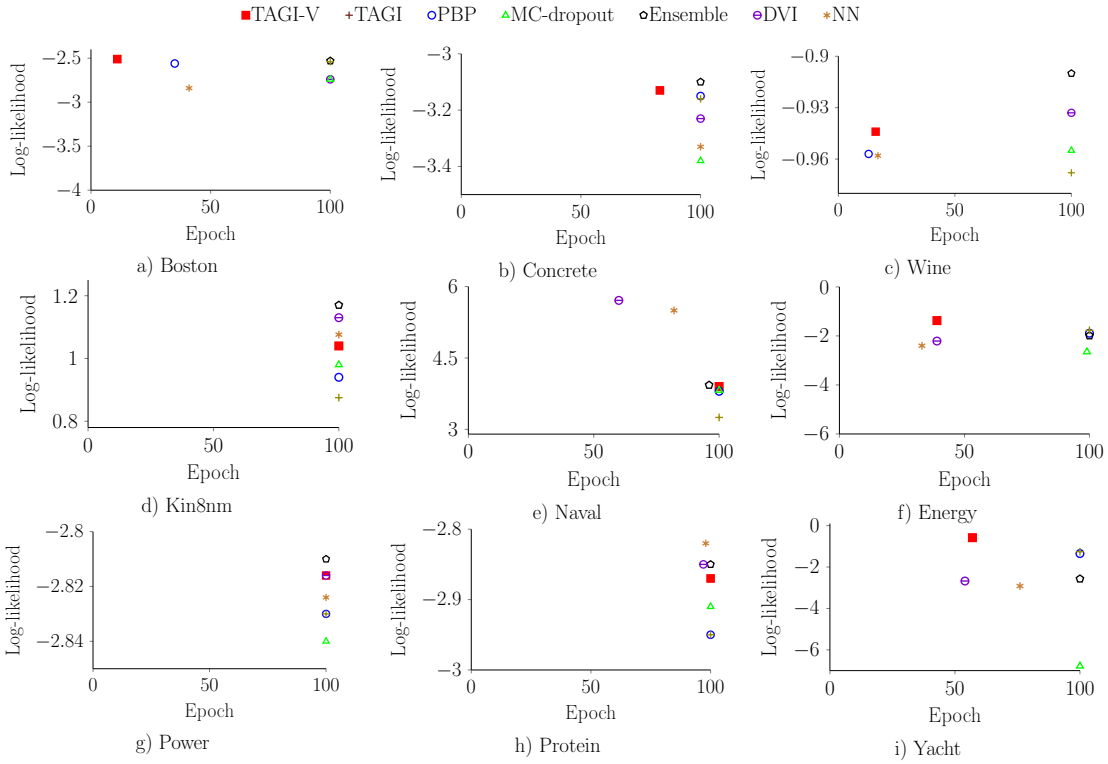


Figure 9: Comparison for the maximum test log-likelihood obtained with the existing baselines under the epoch setting for the datasets a) Boston, b) Concrete, c) Wine, d) Kin8nm, e) Naval, f) Energy, g) Power, h) Protein, and i) Yacht. The horizontal axis shows the total number of epochs, i.e., 100 and the vertical axis shows the test log-likelihood values. The marked points are : TAGI-V (■), PBP (○) [12], MC-dropout (△) [5], DVI (⊖) [34], deterministic NN (✱) [26], Ensemble (⬠) [16], and TAGI (+) [9].

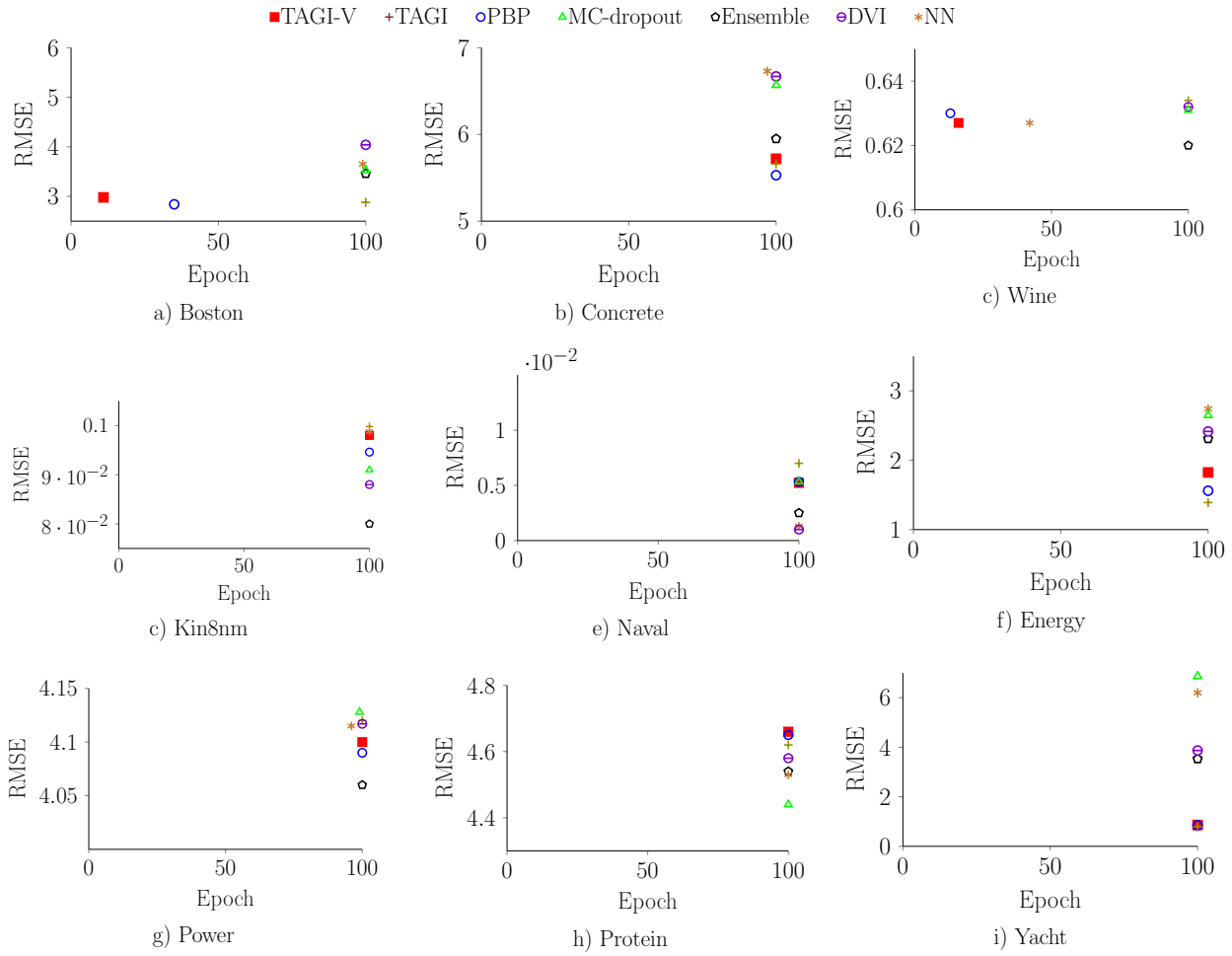


Figure 10: Comparison for the minimum test root mean square error (RMSE) values obtained with the existing baselines under the epoch setting for the datasets a) Boston, b) Concrete, c) Wine, d) Kin8nm, e) Naval, f) Energy, g) Power, h) Protein, and i) Yacht. The horizontal axis shows the total number of epochs, i.e., 100 and the vertical axis shows the test RMSE values. The marked points are : TAGI-V (■), PBP (○) [12], MC-dropout (△) [5], DVI (⊖) [34], deterministic NN (✱) [26], Ensemble (⬠) [16], and TAGI (+) [9].

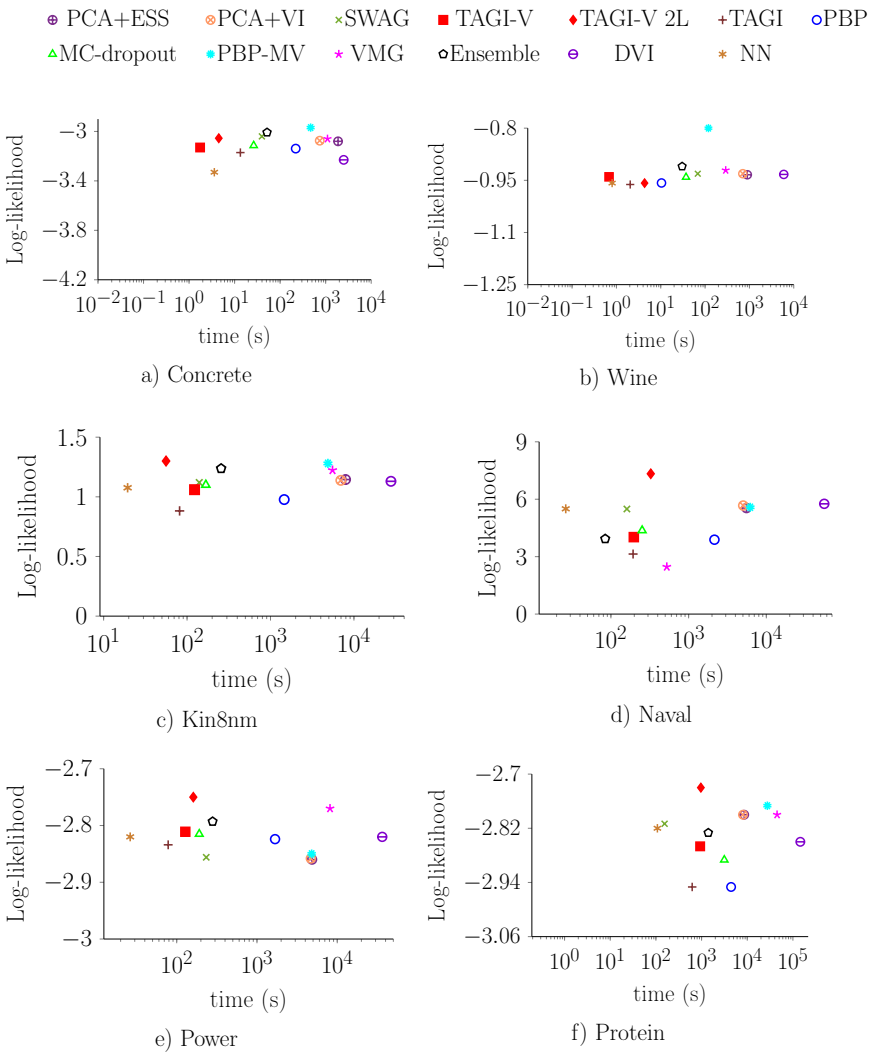


Figure 11: Comparison for the maximum test log-likelihood values obtained with the existing baselines under the time setting for the datasets a) Concrete, b) Wine, c) Kin8nm, d) Naval, e) Power, and f) Protein. The horizontal axis represents the training time (s) in log scale (base 10) and the vertical axis represents either the test log-likelihood values in linear scale. The marked points are : TAGI-V (■), PBP (○) [12], MC-dropout (△) [5], DVI (⊖) [34], deterministic NN (✱) [26], Ensemble (⬠) [16], TAGI (✚) [9], TAGI-V 2L (♦) that represents a TAGI-V network of two layers and 100 hidden nodes, PBP-MV (●) [31], VMG (★) [17], SWAG (×) [18], the two subspace inference methods, i.e., PCA+ESS (⊕) , and PCA+VI (⊗) [14].

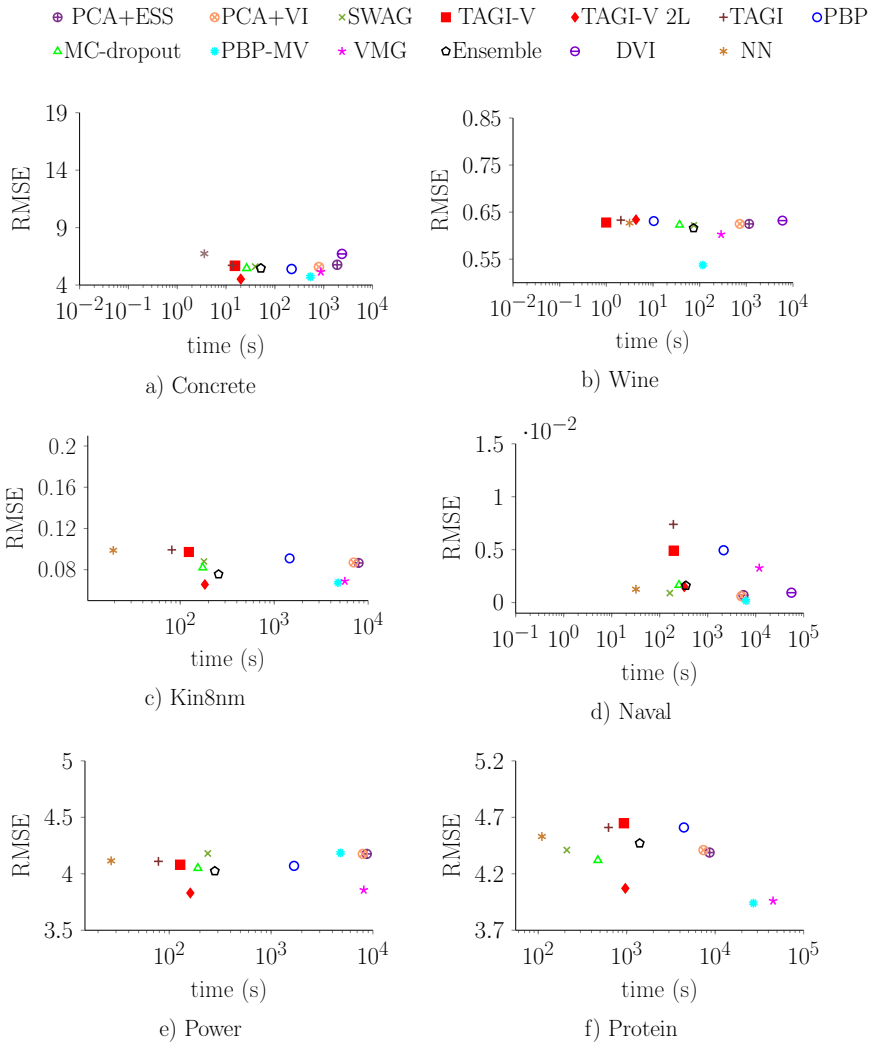


Figure 12: Comparison for the minimum test root mean square error (RMSE) values obtained with the existing baselines under the time setting for the datasets a) Concrete, b) Wine, c) Kin8nm, d) Naval, e) Power, and f) Protein. The horizontal axis represents the training time (s) in log scale (base 10) and the vertical axis represents the test RMSE values in linear scale. The marked points are : TAGI-V (■), PBP (○) [12], MC-dropout (△) [5], DVI (⊖) [34], deterministic NN (✱) [26], Ensemble (⬠) [16], TAGI (✚) [9], TAGI-V 2L (♦) that represents a TAGI-V network of two layers and 100 hidden nodes, PBP-MV (●) [31], VMG (★) [17], SWAG (✖) [18], the two subspace inference methods, i.e., PCA+ESS (⊕) , and PCA+VI (⊗) [14].

F Appendix F

F.1 Learning curves showing test log-likelihood and test RMSE under the epoch and time setting.

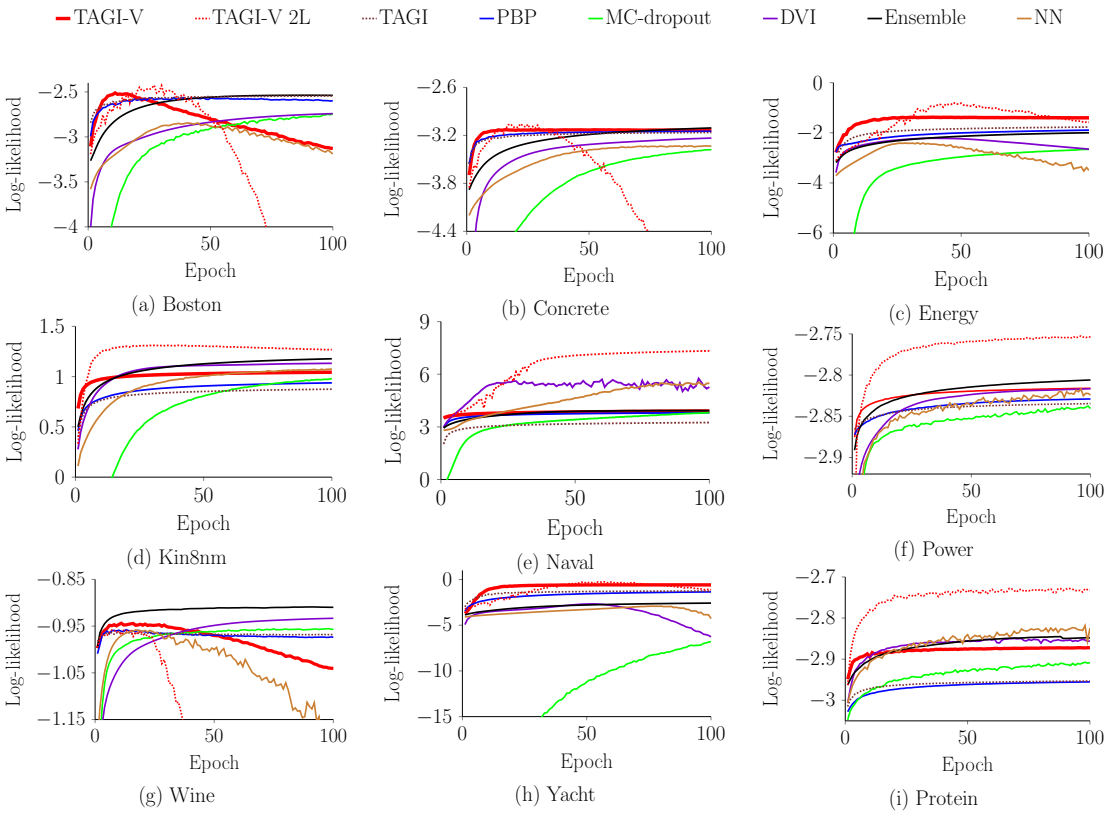


Figure 13: Learning curves showing the test log-likelihood under the epoch setting. The horizontal axis shows the number of epochs and the vertical axis shows the test loglikelihood. The colored line plots are : TAGI-V (red solid line), PBP (blue solid line) [12], MC-dropout (green solid line) [5], DVI (purple solid line) [34], deterministic NN (yellow solid line) [26], Ensemble (black solid line) [16], TAGI (brown dotted line) [9], and TAGI-V 2L (red dotted line) that represents a TAGI-V network of two layers and 100 hidden nodes.

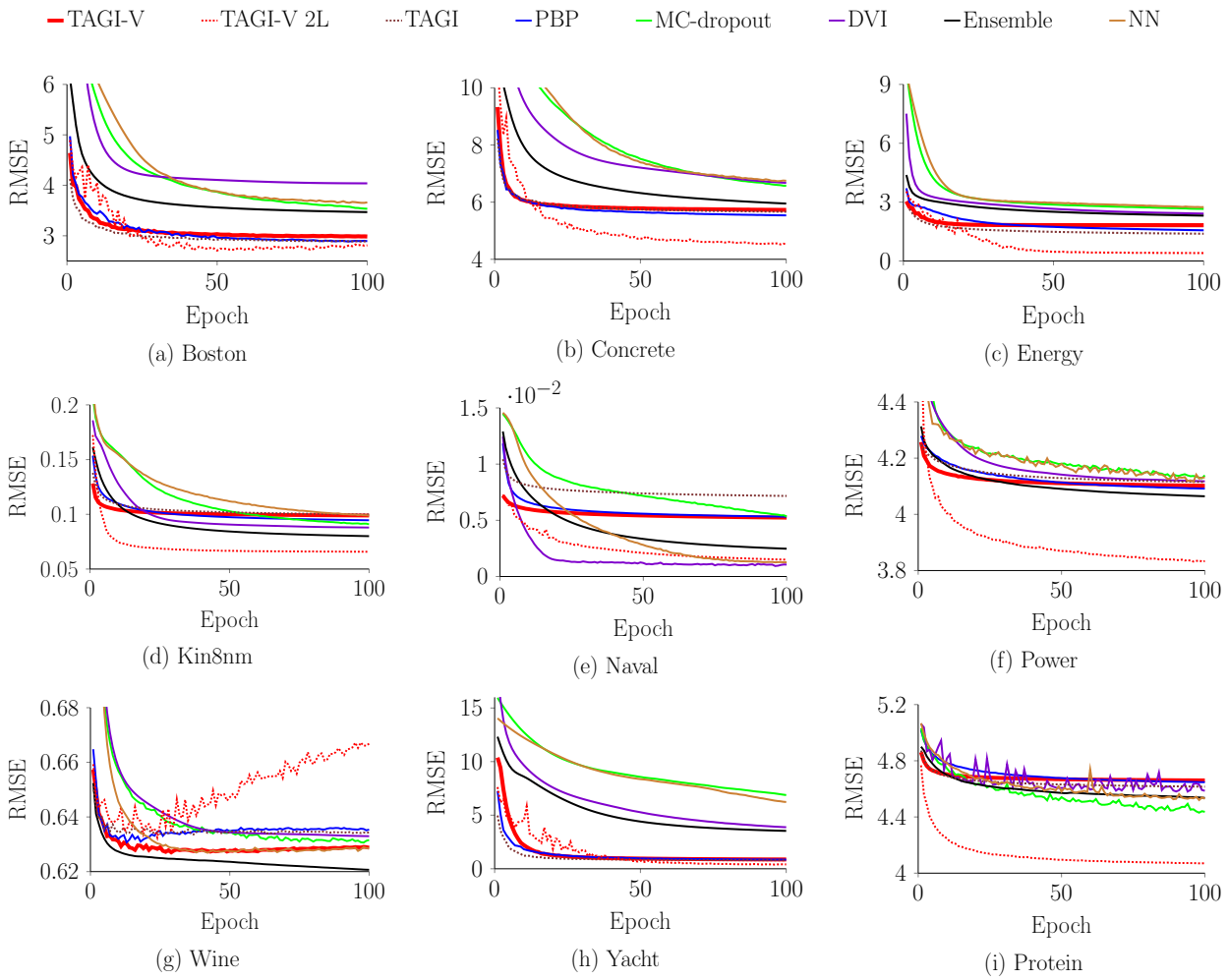


Figure 14: Learning curves showing the test RMSE under the epoch setting. The horizontal axis shows the number of epochs and the vertical axis shows the test RMSE. The colored line plots are : TAGI-V (red solid line), PBP (blue solid line) [12], MC-dropout (green solid line) [5], DVI (purple solid line) [34], deterministic NN (yellow solid line) [26], Ensemble (black solid line) [16], TAGI (brown dotted line) [9], and TAGI-V 2L (red dotted line) that represents a TAGI-V network of two layers and 100 hidden nodes.

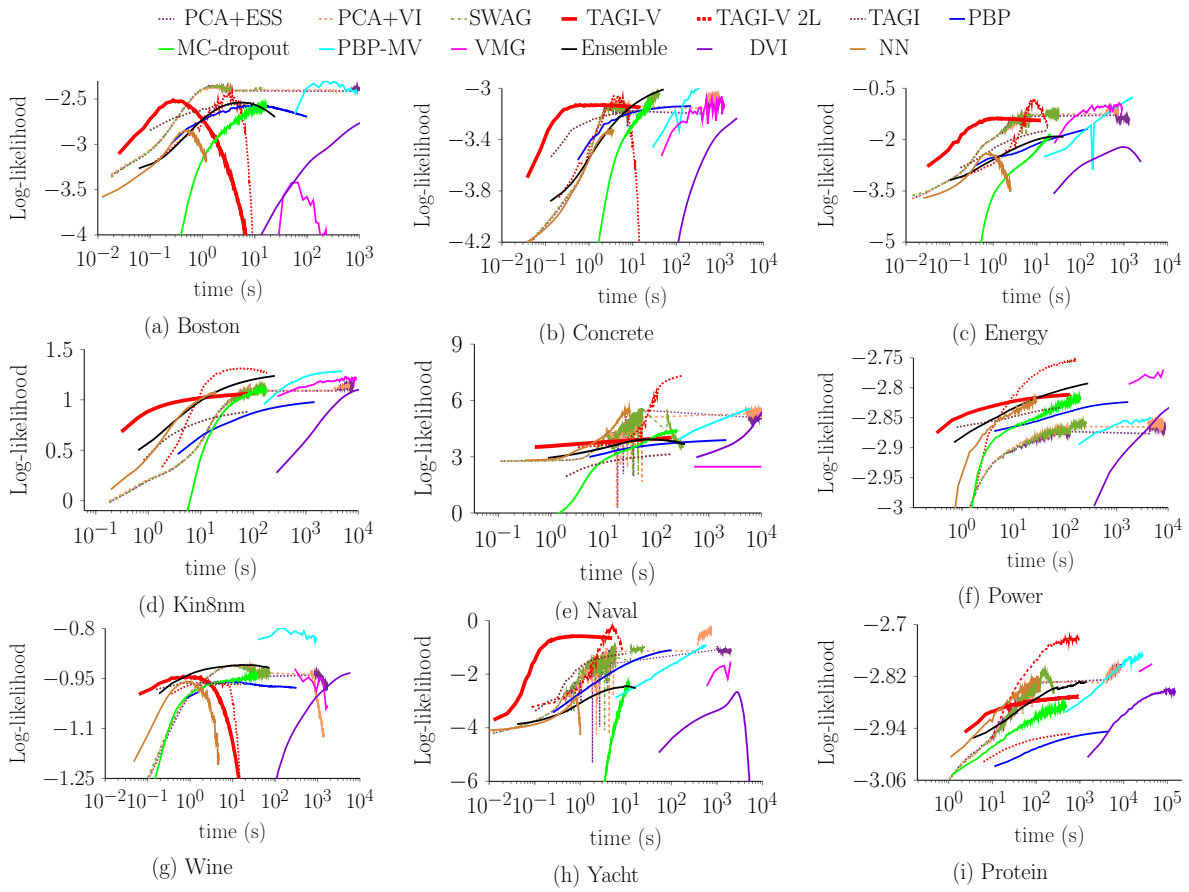


Figure 15: Learning curves showing the test log-likelihood under the time setting. The horizontal axis represents training time (s) in log scale (base 10) and the vertical axis represents the test log-likelihood in linear scale. The colored line plots are : TAGI-V (red solid line), PBP (blue solid line) [12], MC-dropout (green solid line) [5], DVI (purple solid line) [34], deterministic NN (yellow solid line) [26], Ensembles (black solid line) [16], TAGI (brown dotted line) [9], TAGI-V 2L (red dotted line) that represents a TAGI-V network of two layers and 100 hidden nodes, PBP-MV (cyan solid line) [31], VMG (magenta solid line) [17], SWAG (dark green dash-dotted line) [18], and the two subspace inference methods, PCA + ESS (violet dotted line) and PCA + VI (orange dashed line) [14]. The learning curves for PBP-MV and VMG are obtained directly from the original article [31].

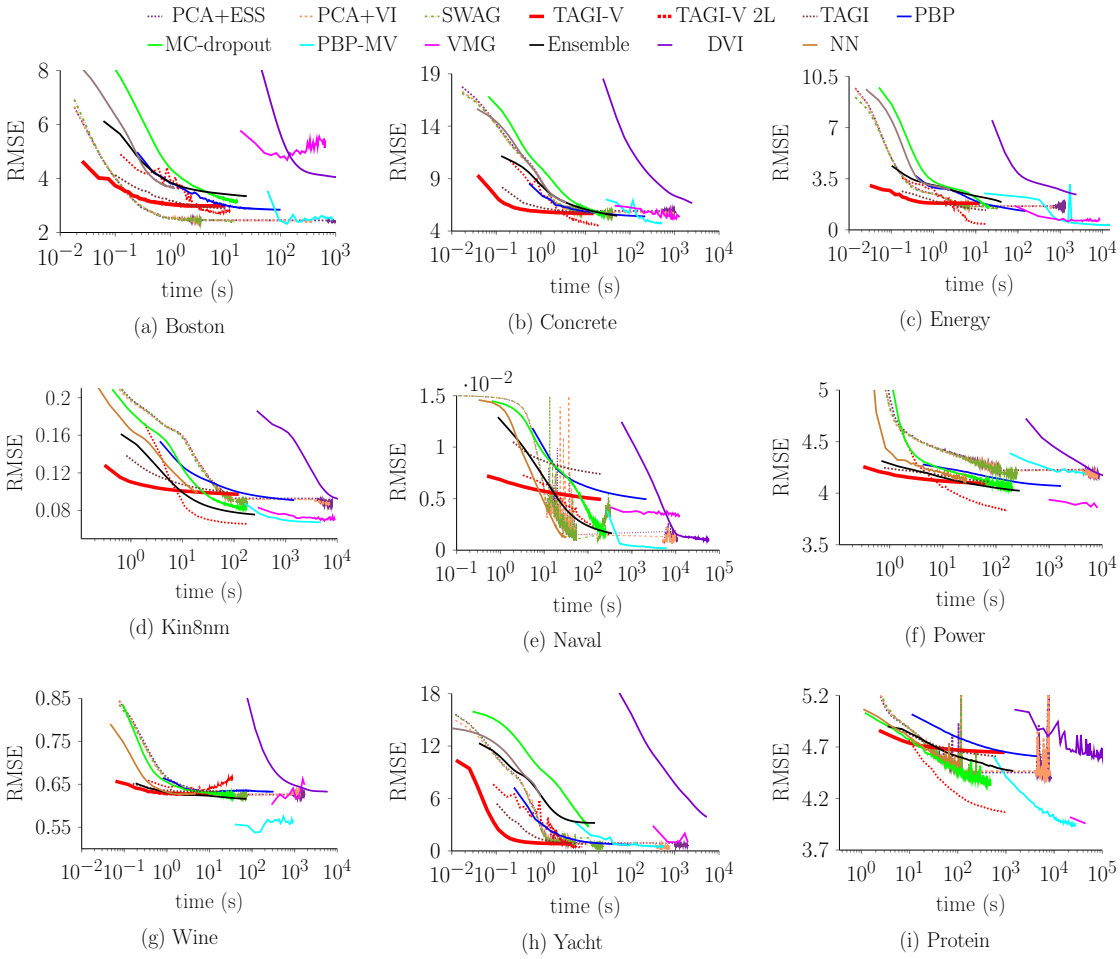


Figure 16: Learning curves showing the test RMSE under the time setting. The horizontal axis represents training time (s) in log scale (base 10) and the vertical axis represents the test RMSE in linear scale. The colored line plots are : TAGI-V (red solid line), PBP (blue solid line) [12], MC-dropout (green solid line) [5], DVI (purple solid line) [34], deterministic NN (yellow solid line) [26], Ensembles (black solid line) [16], TAGI (brown dotted line) [9], TAGI-V 2L (red dotted line) that represents a TAGI-V network of two layers and 100 hidden nodes, PBP-MV (cyan solid line) [31], VMG (magenta solid line) [17], SWAG (dark green dash-dotted line) [18], and the two subspace inference methods, PCA + ESS (violet dotted line) and PCA + VI (orange dashed line) [14]. The learning curves for PBP-MV and VMG are obtained directly from the original article [31].

G Appendix G

G.1 Comparison for computational time between the approximate inference methods.

Table 3: Comparison between the approximate inference methods for average training time (s) per epoch (Rank legend: **first**). All the experiments are carried out using 12 cores 3GHz CPU.

Datasets	TAGI-V	TAGI	MC-Dropout	Deep Ensembles	PBP	PBP-MV	VMG	DVI	SWAG	PCA+ESS	PCA+VI	NN
Boston	0.025	0.099	0.041	0.061	0.25	37	18.6	12.86	0.05	3.5	1.22	**0.012**
Concrete	0.038	0.134	0.066	0.129	0.55	28.57	35.71	24.91	0.04	1.03	0.95	**0.037**
Energy	0.031	0.177	0.051	0.102	0.375	14.7	18.38	24.26	0.032	1.3	0.7	**0.025**
Kin8nm	0.309	0.814	0.434	0.642	3.65	158.73	222.22	277.31	0.22	8	7	**0.194**
Naval	0.493	1.934	0.631	0.881	5.375	271.26	271.26	579.689	0.32	11	10	**0.320**
Power	0.32	0.783	0.496	0.701	4.20	181.82	981.82	363.816	0.25	8.7	8.1	**0.28**
Protein	2.327	6.23	1.193	3.506	11.075	556	21296.00	1498.41	1.2	44	42	**1.10**
Wine	0.062	0.157	0.093	0.185	0.80	39.68	166.67	59.53	0.076	1.7	1.4	**0.047**
Yacht	0.012	0.102	0.0295	0.041	0.250	5.56	327.78	53.01	0.025	2	0.75	**0.010**

H Appendix H

H.1 Hyperparameter tuning and computational time for large UCI regression datasets

The hyperparameters that needs to be learned are the parameters α and β associated with the modified He's approach and patience for the early-stopping procedure. The list of hyperparameter values over which the grid-search is carried out are as follows:

$$\begin{aligned} \text{patience} &: \{3, 5, 10\}, \\ \alpha &: \{0.1, 0.5\}, \\ \beta &: \{0.1, 0.01, 0.001, 0.0001\}, \end{aligned}$$

where patience is the hyperparameter for the early-stopping procedure. Note that any combination of hyperparameters causing numerical overflow errors are omitted from the grid-search procedure. Table 4 shows the optimal values for the hyperparameters used for each dataset as well as the total average training time (s), and the average optimal epoch identified using the early stopping procedure. A batch size of 10 is used, except for Pol for which we used a batch size of 1 to avoid numerical errors. Moreover, the computations are done using the NVIDIA Tesla P40 GPU.

Table 4: Optimized set of hyperparameters obtained using the grid-search procedure as well as the total training time (in s) and the average optimal epoch identified using the early stopping procedure for each of the large UCI datasets. The parameters α and β, and patience are associated with the modified He's approach and early-stopping procedure, respectively. The grid-search is carried out using a validation set obtained from the original training set by a $80-20$ split ratio.

Datasets	α	β	Patience	Total Training Time (s)	Optimal Epoch
Elevators	0.1	0.1	10	5070	30
KeggD	0.1	10^{-4}	3	7940	21
KeggU	0.1	0.1	10	10521	26
Pol	0.5	10^{-3}	3	2328	22
Skillcraft	0.1	0.1	3	177	10

References

[1] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to MCMC for machine learning. *Machine learning*, 50(1):5–43, 2003.

[2] François Chollet. Keras: Deep learning for humans. `https://github.com/fchollet/keras`, 2015.

[3] Bhargob Deka, Luong Ha Nguyen, Saeid Amiri, and James-A Goulet. The Gaussian multiplicative approximation for state-space models. *Structural Control and Health Monitoring*, 29(3):e2904, 2022.

[4] Bhargob Deka, Luong Ha Nguyen, and James-A Goulet. TAGI-V. `https://github.com/bhargobdeka/TAGI-V`, 2022.

[5] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059. PMLR, 2016.

[6] Andrew Gelman, John B Carlin, Hall S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[7] James-A Goulet. *Probabilistic machine learning for civil engineers*. MIT Press, 2020.

[8] James-A Goulet, Luong Ha Nguyen, and Saeid Amiri. TAGI. `https://github.com/CivML-PolyMtl/TAGI`, 2021.

[9] 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.

[10] Arjun K Gupta and Daya K Nagar. *Matrix variate distributions*. Chapman and Hall/CRC, 2018.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[12] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869. PMLR, 2015.

[13] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic-backpropagation. `https://github.com/HIPS/Probabilistic-Backpropagation`, 2016.

[14] Pavel Izmailov, Wesley J Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Subspace inference for Bayesian deep learning. In *Uncertainty in Artificial Intelligence*, pages 1169–1179. PMLR, 2020.

[15] Pavel Izmailov, Wesley J Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Subspace inference for Bayesian deep learning. `https://github.com/wjmaddox/drbayes`, 2021.

[16] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6405–6416, 2017.

[17] Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix Gaussian posteriors. In *International Conference on Machine Learning*, pages 1708–1716. PMLR, 2016.

[18] 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*, pages 13153–13164, 2019.

[19] Jishnu Mukhoti, Pontus Stenetorp, and Yarin Gal. Dropoutuncertaintyexps. `https://github.com/yaringal/DropoutUncertaintyExps`, 2018.

[20] Jishnu Mukhoti, Pontus Stenetorp, and Yarin Gal. On the importance of strong baselines in bayesian deep learning. *arXiv e-prints*, pages arXiv–1811, 2018.

[21] Iain Murray, Ryan Adams, and David MacKay. Elliptical slice sampling. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 541–548. JMLR Workshop and Conference Proceedings, 2010.

[22] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[23] Luong-Ha Nguyen and James-A Goulet. Analytically tractable hidden-states inference in Bayesian neural networks. *Journal of Machine Learning Research*, 23(50):1–33, 2022.

[24] Luong-Ha Nguyen and James-A. Goulet. cuTAGI: a CUDA library for Bayesian neural networks with tractable approximate Gaussian inference. `https://github.com/lhnguyen102/cuTAGI`, 2022.

[25] Roland Orre, Anders Lansner, Andrew Bate, and Marie Lindquist. Bayesian neural networks with confidence estimations applied to data mining. *Computational Statistics & Data Analysis*, 34(4):473–493, 2000.

[26] Chintala Soumith Paszke Adam, Gross Sam and Chanan Gregory. Pytorch. `https://github.com/pytorch/pytorch`, 2021.

[27] Herbert E Rauch, F Tung, and Charlotte T Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

[28] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep Bayesian bandits showdown: An empirical comparison of Bayesian deep networks for thompson sampling. *arXiv e-prints*, pages arXiv–1802, 2018.

[29] Herbert E Robbins. An empirical Bayes approach to statistics. In *Breakthroughs in Statistics*, pages 388–394. Springer, 1992.

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

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

[32] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378. PMLR, 2016.

[33] Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E Turner, José Miguel Hernández-Lobato, and Alexander L Gaunt. Deterministic variational inference. `https://github.com/microsoft/deterministic-variational-inference`, 2018.

[34] 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. *arXiv e-prints*, pages arXiv–1810, 2018.