A PREDICTION INTERVAL METHOD FOR UNCERTAINTY QUANTIFICATION OF REGRESSION MODELS

Pei Zhang & Siyan Liu & Dan Lu
Computational Sciences and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, TN 37830
{zhangp1,lius1,lud1}@ornl.gov

Guannan Zhang
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37830
{zhangg}@ornl.gov

Ramanan Sankaran
Computational Sciences and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, TN 37830
{sankaranr}@ornl.gov

ABSTRACT

This paper considers calculation of prediction intervals (PIs) by neural networks (NNs) for quantifying uncertainty in regression tasks, so as to provide fast, accurate and robust emulators to accelerate scientific simulations. We propose a novel method to learn lower and upper bounds of the PI using independent NNs without defining an exclusive loss. Our method requires no distributional assumption, does not introduce extra hyper-parameters, and can effectively identify out-of-distribution samples and quantify their uncertainty. We demonstrate advantages of our method using a benchmark problem and two real-world scientific applications.

1 INTRODUCTION

Accelerating simulators allows researchers to try out many ideas and has shown promise to improve scientific discovery and enable real-time prediction-based experimental control and engineering operation. Neural networks (NNs) have shown impressive success in speeding up simulations by forming a fast emulator to learn the underlying mechanism. However, to successfully learn the simulations, the emulator needs to be not only fast and accurate but more importantly capable of quantifying uncertainty. Thus, when we deploy the emulator we can interpret confidence, capture domain shift of out-of-distribution (OOD) conditions, and recognize when the model is likely to fail.

A diverse set of approaches have been developed to quantify uncertainties of NN models, ranging from fully Bayesian NNs (MacKay, 1992), to assumption-based variational inference (Hoffman et al., 2013; Gal & Ghahramani, 2016), and to empirical ensemble approaches (Lakshminarayanan et al., 2017; Pearce et al., 2020; Amini et al., 2020). These methods require either high computational demands or strong assumptions or large memory costs to store the ensemble of models. When quantifying uncertainty of NNs used to build emulators for simulations, *we focus on evaluating model outputs' uncertainty of a regression task.* Building prediction intervals (PIs) is a widely used approach for this purpose. PIs provide a lower and upper bound for an NN's output, such that the value of the prediction falls between the bounds for some target percentage (e.g., 95%) of the unseen data. Maximum likelihood estimation (Touretzky et al., 1995) is a well-known approach for building PIs by using two NNs, where one predicts the value and the other predicts the variance. Deep ensemble (DE) method (Lakshminarayanan et al., 2017) is another way being used for generating PIs. DE uses a single NN to output the predicted mean and variance simultaneously. Both techniques impose a Gaussian assumption on model errors and may cause problems in producing bounds for the asymmetric distributions.

Recently, a distribution-free, quality-driven (QD) approach (Pearce et al., 2018) was proposed to calculate PIs for NNs. QD derives the loss function based on the axiom that high-quality PIs should be as narrow as possible while capturing a specified portion of data. It uses two hyper-parameters in the loss to meet the axiom. The method demonstrates promising performance, but a hyper-parameter fine tuning may be needed to achieve the desired performance.

In this work, we develop a novel method for calculating the PIs. We use NNs to learn the lower and upper bounds of the PI directly without assuming a specific data distribution which makes it flexible and suitable for a wide range of problems. Additionally, we use standard loss functions such as mean squared error (MSE) without introducing extra hyper-parameters, which enables a stable and fast learning. Furthermore, our method is capable of capturing domain shift of OOD, which allows for a reasonable uncertainty quantification of the unseen data and future conditions. These three contributions of our work can facilitate simulations by providing fast, accurate and robust emulators.

2 Background

We consider the following regression task: learn a function $y = f_{\boldsymbol{\omega}}(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$, parameterized by a vector $\boldsymbol{\omega}$, using a given dataset $\mathcal{D}_{\text{train}} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$. The training of $f_{\boldsymbol{\omega}}$ is done by solving the following optimization problem:

$$\min_{\boldsymbol{\omega}} J(\boldsymbol{\omega}) \quad \text{with} \ \ J(\boldsymbol{\omega}) = \frac{1}{N}\sum_{i=1}^{N} L_i(\boldsymbol{\omega}), \tag{1}$$

where $L_i(\boldsymbol{\omega})$ is the loss function. In this work, we define $L_i(\boldsymbol{\omega})$ as the sum of the squared error, i.e., $L_i(\boldsymbol{\omega}) = \|y_i - f_{\boldsymbol{\omega}}(\boldsymbol{x}_i)\|_2^2$, which encourages the model to learn the correct answer in the ℓ_2 norm. Generally, the goal of regression is to predict y with a point estimate. Here, we estimate not only y but also its uncertainty to quantify our confidence in the prediction.

3 Our method

In this work, we would like to answer the following questions:

- ***Q1:*** *Can we produce reliable PIs without introducing sensitive hyper-parameters into training?*
- ***Q2:*** *Can we compute PIs without imposing any assumption on data distribution?*
- ***Q3:*** *Can we reasonably estimate the uncertainty on out-of-distribution samples?*

Our key idea is to learn $f_{\boldsymbol{\omega}}(\boldsymbol{x})$, the upper and lower bounds of the PI separately using three *independent* neural networks. The specific procedure of our method is described below.

Learning the uncertainty profiles. We first train the model $f_{\boldsymbol{\omega}}(\boldsymbol{x})$ by solving the problem in Eq. (1) using the MSE loss. This is the standard regression problem, where the trained $f_{\boldsymbol{\omega}}(\boldsymbol{x})$ provides the average approximation of the training data in $\mathcal{D}_{\text{train}}$. Next, use the trained $f_{\boldsymbol{\omega}}(\boldsymbol{x})$ to generate two separate datasets $\mathcal{D}_{\text{upper}}$ and $\mathcal{D}_{\text{lower}}$ for learning the upper and lower uncertainty profiles, i.e.,

$$\begin{aligned} \mathcal{D}_{\text{upper}} &= \big\{(\boldsymbol{x}_i, y_i - f_{\boldsymbol{\omega}}(\boldsymbol{x}_i)) \;\big|\; y_i \geq f_{\boldsymbol{\omega}}(\boldsymbol{x}_i), i = 1, \ldots, N\big\}, \\ \mathcal{D}_{\text{lower}} &= \big\{(\boldsymbol{x}_i, f_{\boldsymbol{\omega}}(\boldsymbol{x}_i) - y_i) \;\big|\; y_i < f_{\boldsymbol{\omega}}(\boldsymbol{x}_i), i = 1, \ldots, N\big\}, \end{aligned} \tag{2}$$

where $\mathcal{D}_{\text{upper}}$ and $\mathcal{D}_{\text{lower}}$ includes data points above and below $f_{\boldsymbol{\omega}}(\boldsymbol{x})$, respectively. The number of data points in $\mathcal{D}_{\text{upper}}$ and $\mathcal{D}_{\text{lower}}$ should be comparable when the MSE loss for training $f_{\boldsymbol{\omega}}(\boldsymbol{x})$ achieves a sufficiently small value.

Next we train another two models, denoted by $u_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $v_{\boldsymbol{\xi}}(\boldsymbol{x})$, to learn the uncertainty profiles described by $\mathcal{D}_{\text{upper}}$ and $\mathcal{D}_{\text{lower}}$. $u_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $v_{\boldsymbol{\xi}}(\boldsymbol{x})$ are trained *separately* using the MSE loss, i.e.,

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \sum_{\boldsymbol{x}_i \in \mathcal{D}_{\text{upper}}} (y_i - f_{\boldsymbol{\omega}}(\boldsymbol{x}_i) - u_{\boldsymbol{\theta}}(\boldsymbol{x}_i))^2, \quad \boldsymbol{\xi} = \arg\min_{\boldsymbol{\xi}} \sum_{\boldsymbol{x}_i \in \mathcal{D}_{\text{lower}}} (f_{\boldsymbol{\omega}}(\boldsymbol{x}_i) - y_i - v_{\boldsymbol{\xi}}(\boldsymbol{x}_i))^2. \tag{3}$$

The idea of training $u_{\boldsymbol{\theta}}$ and $v_{\boldsymbol{\xi}}$ separately makes it possible to use the standard MSE loss without defining more sophisticated losses as in (Lakshminarayanan et al., 2017; Amini et al., 2020).

Optimizing PIs via root-finding methods. We now describe how to use the trained $u_{\boldsymbol{\theta}}$ and $v_{\boldsymbol{\xi}}$ to calculate PIs for a given quantile $\gamma \in [0, 1]$. We define the upper and lower bounds of the PI to be

$f_{\boldsymbol{\omega}} + \alpha u_{\boldsymbol{\theta}}$ and $f_{\boldsymbol{\omega}} - \beta v_{\boldsymbol{\xi}}$, respectively, where α and β are unknown parameters. We determine their values by finding the roots of the following functions

$$\begin{aligned} Q_{\text{upper}}(\alpha) &= \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}_{\text{upper}}} \mathbb{1}_{y_i \geq f_{\boldsymbol{\omega}}(\boldsymbol{x}_i) + \alpha u_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}(\boldsymbol{x}_i, y_i) - N(1-\gamma)/2, \\ Q_{\text{lower}}(\beta) &= \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}_{\text{lower}}} \mathbb{1}_{y_i \leq f_{\boldsymbol{\omega}}(\boldsymbol{x}_i) - \beta v_{\boldsymbol{\xi}}(\boldsymbol{x}_i)}(\boldsymbol{x}_i, y_i) - N(1-\gamma)/2, \end{aligned} \tag{4}$$

where N is the number of samples in $\mathcal{D}_{\text{train}}$ and $\mathbb{1}(\cdot)$ is the indicator function. In this work, we use the bisection method (Quarteroni et al., 2006) to find the roots of Q_{upper} and Q_{lower}, denoted by α_r and β_r, respectively. The PI for the quantile γ is then defined by $[f_{\boldsymbol{\omega}} - \beta_r v_{\boldsymbol{\xi}}, f_{\boldsymbol{\omega}} + \alpha_r u_{\boldsymbol{\theta}}]$. When the root finding problems in Eq. (4) are exactly solved, i.e., $Q_{\text{upper}}(\alpha_r) = Q_{\text{lower}}(\beta_r) = 0$, the number of samples falling in $[f_{\boldsymbol{\omega}} - \beta_r v_{\boldsymbol{\xi}}, f_{\boldsymbol{\omega}} + \alpha_r u_{\boldsymbol{\theta}}]$ is exactly $N\gamma$. We emphasize that our strategy does not impose any assumption on the distribution of $y_i \in \mathcal{D}_{\text{train}}$.

Identifying out-of-distribution samples. When using the trained model $f_{\boldsymbol{\omega}}(\boldsymbol{x})$ to make predictions for $\boldsymbol{x} \notin \mathcal{D}_{\text{train}}$, it is highly desirable that the uncertainty, i.e., indicated by the width of the PI, increases with the distance between $\boldsymbol{x}$ and $\mathcal{D}_{\text{train}}$. Since the uncertainty profiles $u_{\boldsymbol{\theta}}$ and $v_{\boldsymbol{\xi}}$ are trained independently, it is easy to achieve OOD identification by proper initialization. Specifically, we initialize $\boldsymbol{\theta}$ and $\boldsymbol{\xi}$ such that $u_{\boldsymbol{\theta}}$ and $v_{\boldsymbol{\xi}}$ are significantly bigger than $|y_i - f_{\boldsymbol{\omega}}(\boldsymbol{x}_i)|$, where $f_{\boldsymbol{\omega}}$ is already trained. This can be achieved by setting the bias of the output layer of $u_{\boldsymbol{\theta}}$ and $v_{\boldsymbol{\xi}}$ to a big value. During the training process, the losses in Eq. (3) will encourage the decrease of $u_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $v_{\boldsymbol{\xi}}(\boldsymbol{x})$ only for in-distribution samples (i.e., $\boldsymbol{x}_i \in \mathcal{D}_{\text{train}}$), not for OOD samples. Thus, after training, the PI will be significantly wider for OOD samples than for in-distribution samples (see Figure 1), which provides an effective OOD detection approach.

4 Experiments

We use three examples to demonstrate our method. We compare our method to QD (Pearce et al., 2018) and DE (Lakshminarayanan et al., 2017) in calculation of PIs for non-Gaussian data and OOD samples, and in terms of training stability and accuracy. Detailed information about experiment setup and more experimental results are available in Appendix.

4.1 Calculating PIs based on non-Gaussian data

We demonstrate our method in calculating PIs on a non-Gaussian synthetic dataset (Figure 1). We train models on $y = x^3 + \varepsilon$ within $[-4, 4]$ and test within $[-6, 6]$. The noise ε is defined by $\varepsilon = s(\zeta)\zeta$, where $\zeta \sim \mathcal{N}(0, 1)$, $s(\zeta) = 10$ for $\zeta \geq 0$ and $s(\zeta) = 2$ for $\zeta < 0$. For such asymmetric noise, the 95% PIs produced by our method and QD capture about 95% of training data with *tight* bounds in $[-4, 4]$, while DE produces an unnecessarily wide lower bound in $[-4, 4]$ due to its Gaussian assumption on the noise's distribution. Additionally, our method and DE produce reasonably wide PIs in the OOD region $[-7, -4] \cup [4, 7]$, while QD produces too narrow (overconfident) PIs that underestimate the uncertainty on OOD samples.

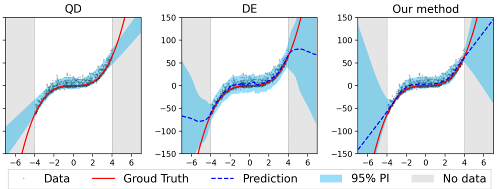

Figure 1: Comparison of 95% PI for $y = x^3 + \varepsilon$ with asymmetric noise ε. Our method and QD outperform DE by producing tighter bounds on in-distribution data, as both methods do not impose assumptions on the distribution of ε. Our method and DE outperform QD in OOD region by providing more reasonable (wider) PIs.

4.2 Learning an OOD-aware antoencoder-based combustion model

We use our method to build an OOD-aware autoencoder-based combustion model for fast prediction of reacting flows inside combustion engines. Autoencoder NNs have been applied for reducing chemical kinetics (Zhang et al., 2021; Mirgolbabaei et al., 2014), but it involves the risk that non-physical solutions may be produced for combustion states far away from training data. Figure 2 presents such an example. It shows the elemental mass conservation errors of two test sets, i.e., in-distribution (ID)

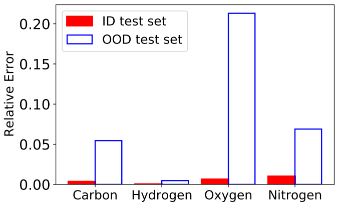

Figure 2: Elemental mass conservation errors. An elemental error of 20% is expected to cause large errors in temperature prediction thus false extinction/ignition events in engine simulations. With OOD-aware model, non-physical solutions with conservation laws violated can be detected before causing further damages.

and OOD. Both sets have not been seen during training. The maximum error in ID test set is less than 1%, while it shoots up to 20% in the OOD set. Such a large elemental error in reduced chemistry models is expected to cause significant errors in energy conservation/temperature prediction thus leading to non-physical solutions in engine simulations. Identifying OOD samples is critical to detecting non-physical solutions before they lead to poor engine designs. In Figure 3, we evaluate our method's capability in identification of OOD samples in the two test sets by comparing it to DE.

A sound PI method should produce a large predictive uncertainty when the predictive error is high and a small uncertainty when the error is low, showing a close correspondence between the uncertainty and the error, and it should also provide a large uncertainty on OOD samples and a small uncertainty on ID. In Figure 3, our method shows a strong correlation between the uncertainty and the error with a large number of samples clustered along the diagonal line. Additionally, our method can clearly separate OOD samples from ID samples with different uncertainty and error magnitudes. In comparison, DE with ensemble size of one cannot distinguish OOD samples from ID. Although DE with ensemble size of ten shows an improved separation of OOD from ID, it does not produce as good uncertainty-error correlation as our method. Furthermore, our method is easy to train, while the training of DE is tedious due to the multiple objectives involved in its loss function.

4.3 Learning an Earth system land model with accurate PIs

We demonstrate our method in calculation of accurate and reliable PIs using the Earth System Land Model (ELM) developed in U.S. Department of Energy. ELM simulates global carbon flux and a single model simulation can take more than one day. We use our method to build an emulator for accelerating the ELM simulation of ten quantities with calculation of their PIs to quantify uncertainty. We have 1000 ELM simulation samples with 500 samples for training and 500 samples for testing. We compare our method to QD using two metrics, prediction interval coverage probability (PICP) and mean prediction interval width (MPIW) whose definition is included in Appendix. A well calibrated PI should be as narrow as possible whilst capturing a specified portion of data. So, when it is asked to output a 90% PI, an accurate uncertainty estimation method should produce a small MPIW value while ensuring its PICP around 90%. Table 1 summarizes the PICP and MPIW results for the testing set. Our method outperforms QD[1] on all ten model outputs, where the PICP is closer to the 90% target and MPIW is on average 60% narrower. Additionally, our method does not introduce extra hyper-parameters into training as we just use the MSE loss. In contrast, QD introduces two sensitive hyper-parameters, shown in Figure 4, which suggests hyper-parameter fine tuning is required for QD.

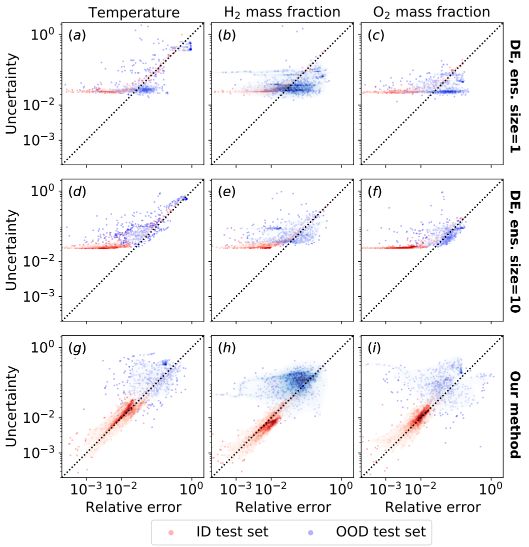

Figure 3: Predictive uncertainty (the width of PI) vs. predictive error of the ID (red) and OOD (blue) test sets from our method and the DE method. DE with a single run $((a) - (c))$ fails to capture the difference in uncertainty for ID and OOD samples. DE with 10 runs $((d) - (f))$ shows improved but still limited separation of OOD samples from ID samples and it fails to produce the uncertainty-error correlation. Our method $((g) - (i))$ shows a strong correlation between the uncertainty and the error and clearly demonstrates that OOD and ID have different uncertainty magnitudes.

[1] The QD results in Table 1 is after hyper-parameter fine tuning shown in Appendix.

	Output 1		Output 2		Output 3		Output 4		Output 5	
	PICP	MPIW	PICP	MPIW	PICP	MPIW	PICP	MPIW	PICP	MPIW
QD	94.6%	2.09	99.2%	2.24	100%	3.09	99.5%	1.76	99.6%	2.33
Our method	90.8%	1.96	90.0%	0.67	91.4%	0.57	91.6%	0.72	90.0%	0.62

	Output 6		Output 7		Output 8		Output 9		Output 10	
	PICP	MPIW	PICP	MPIW	PICP	MPIW	PICP	MPIW	PICP	MPIW
QD	98.8%	2.48	99.6%	2.15	98.6%	1.94	99.6%	2.12	99.6%	1.75
Our method	90.2%	0.59	90.0%	1.41	90.4%	0.81	90.2%	0.72	90.0%	0.67

Table 1: Comparison on the PICP and MPIW for the 90% PI target. Our method outperforms QD by providing PICP closer to the 90% target and MPIW on average 60% narrower.

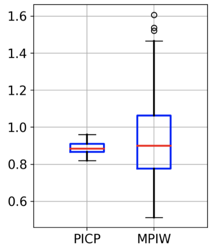

Figure 4: PICP and MPIW values of 100 QD runs with 100 samples of its hyper-parameters. Our method does not need hyper-parameter fine tuning but QD does. PICP and MPIW provided by QD are sensitive to the hyper-parameters.

5 Conclusions

In this work, we proposed a PI method for uncertainty quantification of regression models which requires no data distribution assumption, introduces no sensitive hyper-parameters and can effectively identify OOD samples. We demonstrated the advantages of our method in a toy regression task with non-Gaussian noise and two real-world scientific applications, in comparison with two state-of-the-art approaches, QD and DE. In the non-Gaussian regression example, our method captures the asymmetric noise with desired tight bounds in ID region and wider PIs in OOD region, while DE produces unnecessarily wide lower bounds in ID region and QD fails to produce as wide PIs for OOD as the other two methods. The autoencoder combustion model example suggests that our method provides better uncertainties than DE by showing a strong uncertainty-error correlation and a clear separation of OOD samples from ID. In the ELM model example, our method outperforms QD by producing PIs closer to the desired coverage proportion and reducing the average PI width by around 60%.

Acknowledgments

This material is based upon work supported in part by the U.S. Department of Energy, Office of Science, Offices of Advanced Scientific Computing Research, under the contract ERKJ352, and by the AI Initiative at the Oak Ridge National Laboratory (ORNL). ORNL is operated by UT-Battelle, LLC, for the U.S. Department of Energy under Contract DE-AC05-00OR22725.

REFERENCES

Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 14927–14937. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/aab085461de182608ee9f607f3f7d18f-Paper.pdf.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/gal16.html.

Evatt R Hawkes, Ramanan Sankaran, James C Sutherland, and Jacqueline H Chen. Scalar mixing in direct numerical simulations of temporally evolving plane jet flames with skeletal CO/H_2 kinetics. *Proceedings of the Combustion Institute*, 31(1):1633–1640, 2007.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(4):1303–1347, 2013. URL http://jmlr.org/papers/v14/hoffman13a.html.

Laurent Valentin Jospin, Wray L. Buntine, Farid Boussad, Hamid Laga, and Mohammed Bennamoun. Hands-on bayesian neural networks - a tutorial for deep learning users. *CoRR*, abs/2007.06823, 2020. URL https://arxiv.org/abs/2007.06823.

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*, NIPS'17, pp. 64056416, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Comput.*, 4(3):448472, May 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.3.448. URL https://doi.org/10.1162/neco.1992.4.3.448.

Hessam Mirgolbabaei, Tarek Echekki, and Nejib Smaoui. A nonlinear principal component analysis approach for turbulent combustion composition space. *International Journal of Hydrogen Energy*, 39(9):4622–4633, 2014. ISSN 0360-3199.

Tim Pearce, Alexandra Brintrup, Mohamed Zaki, and Andy Neely. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4075–4084, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/pearce18a.html.

Tim Pearce, Felix Leibfried, and Alexandra Brintrup. Uncertainty in neural networks: Approximately bayesian ensembling. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 234–244. PMLR, 26–28 Aug 2020. URL http://proceedings.mlr.press/v108/pearce20a.html.

Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics (Texts in Applied Mathematics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 3540346589.

D. S. Touretzky, T. K. Leen, David A. Nix, and A. S. Weigend. Learning local error bars for nonlinear regression, 1995.

Hao Wang and Dit-Yan Yeung. A survey on bayesian deep learning, 2021.

Pei Zhang, Ramanan Sankaran, and Evatt R Hawkes. A priori examination of reduced chemistry models derived from canonical stirred reactors using three-dimensional direct numerical simulation datasets. In *AIAA Scitech 2021 Forum*, pp. 1784, 2021.

A Appendix

A.1 related work

A large history of uncertainty estimation methods has been proposed to quantify NN uncertainty, and most of them are developed based on Bayesian theory. Jospin et al. (2020); Wang & Yeung (2021) summarized Bayesian methods for deep learning, including Markov chain Monte Carlo (MCMC) and variational inference (VI). Bayesian NNs, which learn a distribution over weights, require some modifications to the training procedure and are usually computationally expensive. To address these challenges, ensemble approaches (Pearce et al., 2020; Lakshminarayanan et al., 2017; Amini et al., 2020) have been proposed to provide a simple-to-implement and readily parallelizable alternative for NN uncertainty quantification.

Prediction intervals (PIs) are another way of quantifying NN uncertainty specifically for regression tasks. PIs directly communicate model outputs uncertainty by offering a lower and upper bound for their predictions, and assure that the actual data will fall between the bounds with the desired probability. These features make PIs understandable and informative for decision making. The calculation of PIs, which works on the quantity of target variables directly, is straightforward and rather simple compared to the Bayesian NNs, which deal with the large amount of NN weights. Our method falls into this category.

A.2 Setup for the toy problem in Section 4.1

The training set consist of 1000 random samples of $y = x^3 + \varepsilon$ from $[-4, 4]$. $f_{\boldsymbol{\omega}}(\boldsymbol{x})$ is defined as a fully connected neural network with 1 hidden layers and 200 hidden neurons. $u_{\boldsymbol{\theta}}$ and $v_{\boldsymbol{\xi}}$ share the same network architecture as $f_{\boldsymbol{\omega}}$. To make sure u and v are strictly positive, we add operation $\sqrt{(\cdot)^2}$ to the outputs of the neural networks. The same approach is used for the other two real-world application problems in Section 4.2 and 4.3. The three neural networks are trained separately by stochastic gradient descent with learning rate being 0.01.

For the QD method (Pearce et al., 2018), we use the implementation at `https://github.com/TeaPearce/Deep_Learning_Prediction_Intervals`. For the DE method (Lakshminarayanan et al., 2017), we use the implementation at `https://github.com/vvanirudh/deep-ensembles-uncertainty`. We used the default parameters suggested by the authors of the papers for both baseline methods.

A.3 Setup for the Earth system land model in Section 4.3

Energy Exascale Earth System Land Model (ELM) developed in U.S. Department of Energy is a complex land-surface model that simulates terrestrial water, energy, and biogeochemical processes in a global scale. ELM is an important tool for improving our understanding of ecosystem responses to climate change but is very computationally expensive. In this work, we use our method to build a fast-to-evaluate emulator to accelerate the ELM simulation. We are considering ten model outputs that are related to carbon fluxes.

The definition of PICP and MPIW. For a testing dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ with N samples, we denote by $l_i^{\rm PI}$ and $u_i^{\rm PI}$ the lower and upper bounds of an PI at $\boldsymbol{x}_i$. Then a PI with confidence $\gamma \in [0, 1]$ should capture the desired portion of data, i.e., $\mathbb{P}(l_i^{\rm PI} \le y_i \le u_i^{\rm PI}) \ge 1 - \gamma$. For each data y_i, we can define a binary value k_i as

$$k_i = \begin{cases} 1, & \text{if } l_i^{\rm PI} \le y_i \le u_i^{\rm PI}, \\ 0, & \text{otherwise} \end{cases}$$

to indicate if y_i is included in the PI. Then the total number of data points included in the PI can be represented by

$$c = \sum_{i=1}^{N} k_i,$$

The PICP and MPIW are defined by

$$\text{PICP} = \frac{c}{N} \quad \text{and} \quad \text{MPIW} = \frac{1}{N}\sum_{i=1}^{N} u_i^{\text{PI}} - l_i^{\text{PI}}.$$

The desired PI should provide PICP that is close to γ with smallest value of MPIW.

Our method consists of three fully-connected neural networks for average, upper bound, and lower bound approximation, respectively. All networks share the same structure with two hidden layers, and 100 hidden neurons. The input layer contains 8 neurons for the 8 input features with linear activation. Rectified linear unit (ReLU) activation functions are implemented for hidden layers. The input layer contains 8 neurons for the 8 input features with linear activation. Adam optimizer is used for updating the gradients with a 0.02 learning rate initially, and it starts to decay exponentially after 50 iterations with a 0.9 decay rate. The entire data set contains 1,000 samples with 8 inputs and 10 outputs. We split it into 500 training and 500 testing points randomly, and standardize them before the training processes. For a single experiment, the three neural networks are trained sequentially to reach 300 iterations before the upper/lower bound optimization. The optimization target quantile is 0.9 in our experiments. The models are implemented with Tensorflow 2.4.1.

Hyper-parameter tuning for QD. The QD method defines a new loss using the PICP and MPIW, i.e.,

$$L_{\text{QD}} = \frac{1}{c}\sum_{i=1}^{N}(u_i^{\text{PI}} - l_i^{\text{PI}})k_i + \frac{\lambda N}{\gamma(1-\gamma)} \max\left\{0, (1-\gamma) - \frac{1}{N}\sum_{i=1}^{N}\sigma((u_i^{\text{PI}} - y_i)s)\sigma((y_i - l_i^{\text{PI}})s)\right\},$$

where σ is the sigmoid function. The two sensitive hyper-parameters are the weight λ and the smoothness parameter s of the sigmoid function. They are tuned based on our 50%/50% train/testing data split, and the grid search method is used. The softening factor (s = 160.0) is used to avoid weight shrinking with the low function during training. Lambda (λ=15) is selected to control the importance of coverage versus width for the loss function.

A.4 Setup and more results for the Combustion problem in Section 4.2

Chemical kinetic models are crucial for computational fluid dynamics (CFD) modeling of reacting flows in combustion systems. To describe the chemical kinetics of typical hydrocarbon fuels, detailed mechanisms consisting of hundreds of species and Arrhenius reaction steps are required. The large number of chemical species represent the large number of transport equations need to be solved in CFD simulations. Reduced chemical models are needed for fast and accurate CFD simulations. In this work, an OOD-aware autoencoder reduced chemistry model is developed for syngas CO/H_2 combustion. The original chemical mechanism (Hawkes et al., 2007) has 11 species, i.e., H_2, O_2, O, OH, H_2O, H, HO_2, CO, CO_2, HCO and N_2. The reduced autoencoder model has two variables. The training set contains around 1.14 million samples with 12 input features, which are the 11 species mass fractions and temperature, from zero-dimensional perfectly stirred reactors. The ID test set contains 0.34 million samples from the same distribution with the training set. The OOD test set consists of 1.9 million samples from three-dimensional direct numerical simulations in (Hawkes et al., 2007).

All NN models in the combustion problem are implemented with Tensorflow 2.4.0. The autoencoder model is defined as a fully connected NN with five layers, i.e., input layer, hidden layer in the encoder part with 12 neurons, latent layer with 2 neurons, hidden layer in the decoder part with 12 neurons, and output layer. Hyperbolic tangent activation function, Tanh, is used. When combining the autoencoder model with the DE method, we use an additional NN for feature variances. The variance NN takes the same input as the autoencoder and outputs variances for the 12 features with three dense hidden layers. Every hidden layer has 60 neurons with Tanh activation function. Softplus activation function is used for the output layer to guarantee a positive variance. The two NNs are trained together with the loss function,

$$L_{DE} = \frac{1}{N}\sum_{i=1}^{N} L_{DE,i}(\boldsymbol{\omega}),$$

where $L_{DE,i}$ is the loss function for the ith sample as,

$$L_{DE,i} = \frac{1}{2}\sum_{k=1}^{N_x}\log\sigma_{i,k}^2 + \frac{1}{2}\sum_{k=1}^{N_x}\frac{(x_{i,k} - \mu_{i,k})^2 + \varepsilon_{ele}}{\sigma_{i,k}^2}.$$

In the above equation, $N_x = 12$ is the number of input features, $\mu_{i,k}$ and $\sigma_{i,k}^2$ are the predicted mean and variance of the kth feature of the ith sample, $x_{i,k}$, respectively, and ε_{ele} is a penalty for violation of elemental mass conservation. In the training process, a competition between the two terms in the loss function is observed, which leads to an undesirable accuracy deterioration of the autoencoder model. A decreasing stepwise learning rate with 6000 epochs is specified in the training, where 0.01, 0.005 and 0.001 are used for the first 3000 epochs and 5×10^{-4} is used for the rest.

When combining the autoencoder model with our method in Section 3, we use two sets of 12 variance NNs for 12 output features, respectively. Each pair of NNs output the lower and upper standard deviation of a single feature, where 1 dense hidden layer with 1200 neurons and Tanh activation function is employed. Different from the DE method, the variance NNs in our method are trained independently from the autoencoder model hence provide an easy and nonintrusive method for uncertainty. In the combustion problem, $\gamma = 0.6826$ is specified to get the reported PIs. The learning rate is specified as 5×10^{-4} for the training of the NNs.

In Section 4.2, results for three variables, i.e., temperature and mass fractions of H_2 and O_2, from the DE method and our method have been reported (Figure 3). Results for all the 12 variables are presented here in Figures A.1, A.2 and A.3 for DE with ensemble size of one, DE with ensemble size of ten, and our method, respectively. Consistent conclusions with Figure 3 can be made that our method provides superior performance in identification of the OOD samples and in producing the correlation between the predictive uncertainty and the predictive error.

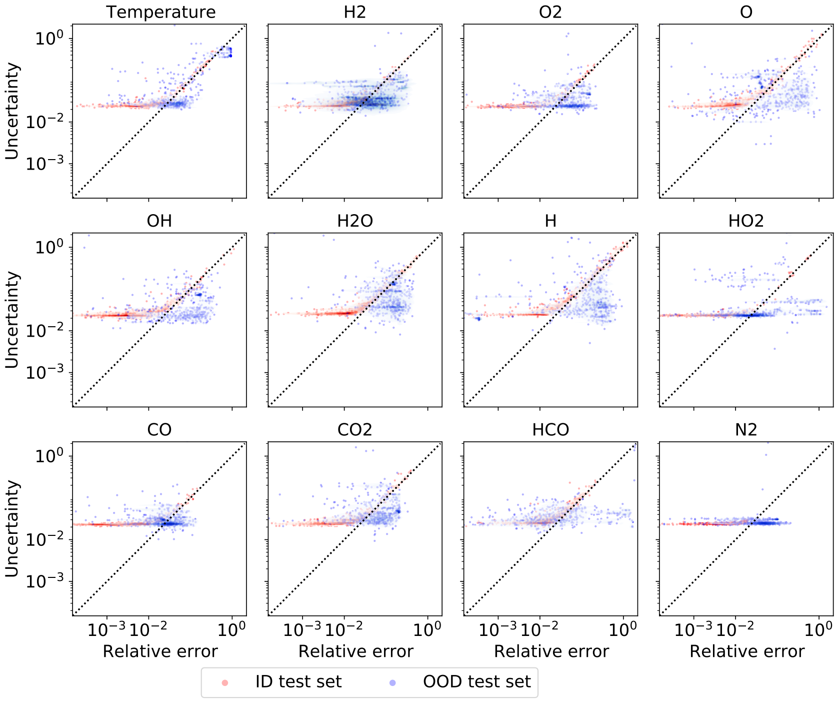

Figure A.1: Predictive uncertainty vs. predictive error for all output features of ID (red) and OOD (blue) test sets from the DE method with ensemble size of 1.

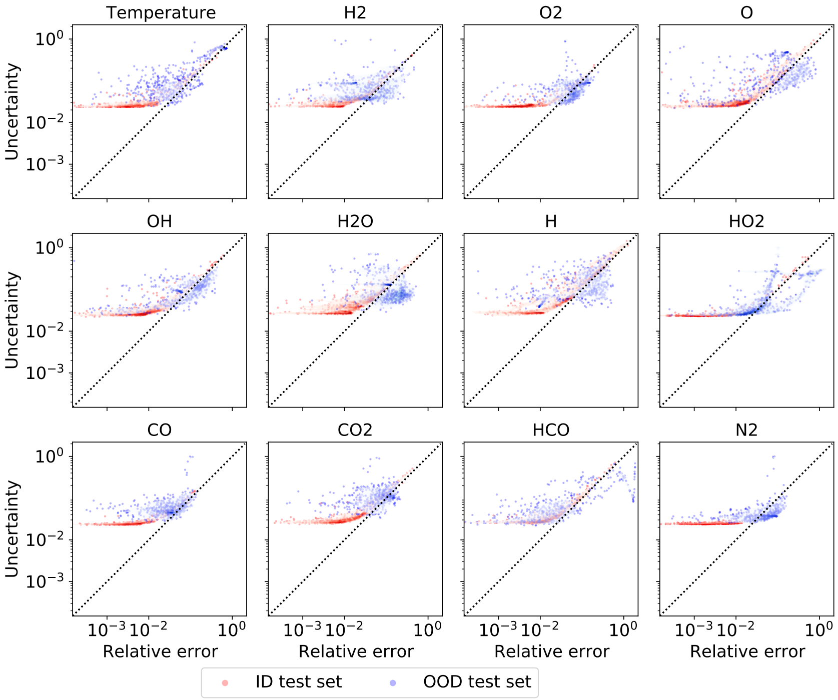

Figure A.2: Predictive uncertainty vs. predictive error for all output features of ID (red) and OOD (blue) test sets from the DE method with ensemble size of 10.

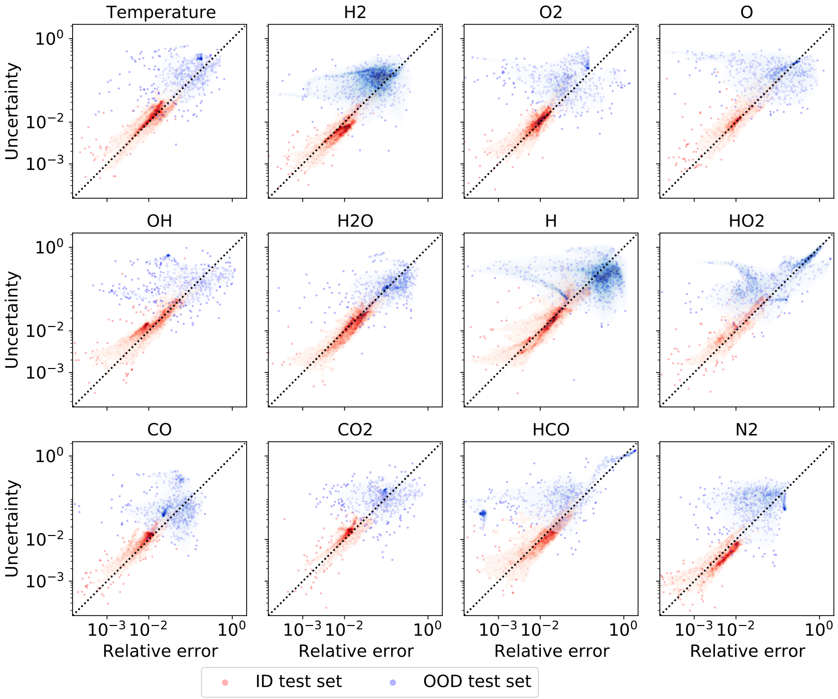

Figure A.3: Predictive uncertainty vs. predictive error for all output features of ID (red) and OOD (blue) test sets from our method.