Robust Prediction Intervals from Three Neural Networks Trained by the MSE Loss

Anonymous Author(s)

Abstract

We propose a novel method to learn prediction values, lower and upper bounds of a prediction interval from three independent neural networks (NNs) using mean squared error (MSE) loss for uncertainty quantification in regression tasks. Our method requires no distributional assumption on data, does not introduce unusual hyperparameters to the NNs and to the training, and can effectively identify out-of-distribution (OOD) samples and reasonably quantify their uncertainty. Benchmark experiments show that our method outperforms current state-of-the-art with respect to predictive uncertainty quality, robustness, and OOD samples identification.

1 Introduction

Neural networks (NNs) are widely used in prediction tasks due to their unrivaled performance and flexibility in modeling complex unknown functions of the data. Although NNs provide accurate predictions, quantifying the uncertainty of their predictions is a challenge. Uncertainty quantification (UQ) is crucial for many real-world applications—such as self-driving cars, autonomous experimental and operational controls, and in medical and life related areas—as well as in some auxiliary ways—such as to accelerate exploration in reinforcement learning and for active learning. Furthermore, precise and calibrated UQ are useful for interpreting confidence, capturing domain shift of out-of-distribution (OOD) conditions, and more importantly realizing 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 [16], to assumption-based variational inference [9, 6], and to empirical ensemble approaches [15, 19, 2]. These methods require either high computational demands or strong assumptions or large memory costs to store the ensemble of models. Another set of UQ methods in regression is calculating prediction intervals (PIs)—which 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. PIs directly communicate uncertainty which provides more understandable information for decision making [18, 22]. Maximum likelihood estimation [24] is a well-known approach for building PIs by using two NNs, where one predicts the value and the other predicts the variance. This technique imposes a Gaussian assumption on model errors and may cause problems in producing bounds for asymmetric distributions. The quality-driven (QD) approach proposed in [18] requires no distributional assumption by defining a sophisticated loss function. But QD is unable to generate point estimates, has a fragile training process, and likely underestimates the uncertainty on OOD samples, Built on QD, the PIVEN method in [22] adds the capability to calculate point estimates and the PI method in [21] further improves the training stability of QD by integrating a penalty function to the loss.

Recently developed PI methods [18, 22, 21] tend to design sophisticated loss functions to obtain a well-calibrated PI. Although these work has achieved promising results, their performance is sensitive to the unusual hyperparameters introduced into their customized loss functions. Since these hyperparameters are not commonly used, we have very little knowledge and experience about how to properly choosing them. In practice, these hyperparameters usually need fine tuning [21] to achieve the desired performance, which makes these methods less practical and less robust when deployed.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

In this work, we develop PI3NN (prediction interval based on three neural networks)—a novel method for calculating PIs. Different from current PI methods [18, 22, 21] that design sophisticated loss functions to obtain a well-calibrated PI, our PI3NN approach only uses the standard loss functions, such as mean squared error (MSE), for training. Specifically, PI3NN uses a combination of the three separately trained NNs to learn the mean prediction (i.e., point estimation), and the lower and upper bounds of the PI. PI3NN not only has the nice properties as the state-of-the-art PI methods have—such as requiring no distributional assumption and producing tight PI bounds, PI3NN also embraces some exclusive advantages. For instance, it introduces no extra hyperparameters which enables a robust training; and it is also able to capture domain shift and reasonably quantify uncertainty on OOD samples. This work makes the following contributions:

1. We use three independently trained NNs to construct the mean prediction and the PI without assuming data distribution, which makes PI3NN flexible for a wide range of regression tasks.
2. We use the standard loss functions such as MSE for training the three NNs without introducing unusual hyperparameters in the loss, which enables a robust and fast training.
3. We use a suitable initialization scheme to initialize the NNs' training to identify OOD samples and reasonably quantify their uncertainty.

2 Related work

Quantifying uncertainty of regression models has long been an active area of research. Early and recent work was nicely summarized and reviewed in these three survey papers [11, 26, 1]. The studies on UQ for regression can be generally categorized into two groups—PI approaches and non-PI approaches. The non-PI approaches use a distribution to quantify uncertainty, which can be further divided into Bayesian [16] and non-Bayesian methods. Bayesian methods—including Markov chain Monte Carlo [17] and Hamiltonian Monte Carlo [23]—place priors on NN weights and then infer a posterior distribution from the training data. Non-Bayesian methods includes evidential regression [2] that places priors directly over the likelihood function and some ensemble learning methods that do not use priors. For example, the deep evidential regression (DER) method proposed in [2] placed evidential priors over the Gaussian likelihood function and training the NN to infer the hyperparameters of the evidential distribution. Gal and Ghahramani [6] proposed using Monte Carlo dropout (MC-dropout) to estimate predictive uncertainty by using Dropout (which can be interpreted as ensemble model combination) at test time. Deep ensembles [15] employed a combination of ensembles of NNs learning and adversarial training to quantify uncertainty with a Gaussian distributional assumption on the data. Pearce et al. [19] proposed an anchored ensembling by using the randomized MAP sampling to increase the diversity of NN training in the ensemble.

PI approaches directly communicate uncertainty by offering a lower and upper bound for a prediction. The most common techniques to construct the PI are the delta method (also known as analytical method) [25, 10], methods that directly predict the variance (maximum likelihood method and ensemble method) [4, 8] and quantile regression method [13, 14]. Most recent PI methods are developed on the high-quality principle—a PI should be as narrow as possible, whilst capturing a specified portion of data. Khosravi et al. [12] developed the Lower Upper Bound Estimation method, incorporating the high-quality principle directly into the NN loss function for the first time. Inspired by [12], the QD approach in [18] defined a loss function that can generate a high-quality PI and is able to optimize the loss using stochastic gradient descent as well. Built on QD, the PIVEN method in [22] adds an extra term in the loss to enable the calculation of point estimates and the PI method in [21] further integrates a penalty function to the loss to improves the training stability of QD.

Both PI and non-PI approaches have their strengths and weaknesses. Non-PI approaches—specifically optimize the accuracy—can produce more accurate value predictions but they may overestimate the uncertainty with a wide PI bound. On the other hand, PI approaches—specifically optimize PIs based on the high-quality principle—can produce tight uncertainty bounds, but they tend to result in less accurate point estimates and suffer from underestimation of uncertainty on OOD samples.

3 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 $\mathcal{D}_{\text{train}} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$, in order to approximate a target function

$f(\boldsymbol{x})$. The training of $f_{\boldsymbol{\omega}}$ is conducted by solving the following optimization problem:

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

where $J(\boldsymbol{\omega})$ is the standard mean squared error (MSE) loss with $\ell_i(\boldsymbol{\omega}) = \|y_i - f_{\boldsymbol{\omega}}(\boldsymbol{x}_i)\|_2^2$. Since the data of the output of $f(\boldsymbol{x})$ is usually polluted by some random noise, the relation between $\{\boldsymbol{x}_i\}_{i=1}^N$ and $\{y_i\}_{i=1}^N$ is described by a regression formulation $y_i = f(\boldsymbol{x}_i) + \varepsilon$, where ε denotes the random noise. Thus, the output of a regression relation is a random variable.

However, the neural network $f_{\boldsymbol{\omega}}(\boldsymbol{x})$ is a function map, in which one input only maps to one deterministic output. The use of the MSE loss cannot quantify the uncertainty in the data either. For example, when the noise ε follows a Gaussian distribution $\mathcal{N}(0, \sigma^2)$, the MSE loss $J(\boldsymbol{\omega})$ will go to zero (assuming $f_{\boldsymbol{\omega}}$ is sufficiently expressive) during the training, regardless of how big σ is. Therefore, to quantify the uncertainty in f_ω, we can either assume y follows a specific type of probability distributions then find the best parameters, or construct PIs without imposing any a priori assumption on the output's distribution. In this paper, we choose the second route. The QD method [18] defined an exclusive loss function with several new hyperparameters. It is observed in numerical experiments in Section 5 that the performance of the QD method is not robust against variation of the hyperparameters. This challenge motivated us to develop a more robust PI method for quantify uncertainties of NN-based regression model.

4 Our PI3NN method

The objective of a PI method is to construct two function, denoted by $U(\boldsymbol{x})$ and $L(\boldsymbol{x})$, to represent the upper bound and the lower bound of the PI, respectively. The state-of-the-art approach, e.g., QD, define $U(\boldsymbol{x})$ and $L(\boldsymbol{x})$ as two neural networks, but train networks jointly by defining a new loss function with extra hyperparameters. In this work, we would like to answer the following questions:

- **Q1:** Can we produce reliable PIs without introducing unusual hyperparameters 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?

The key idea of the PI3NN method is to construct the PI by training three neural networks separately using the standard MSE loss and using root-find methods to define the upper bound $U(\boldsymbol{x})$ and lower bound $L(\boldsymbol{x})$. We denote the three neural networks by $f_{\boldsymbol{\omega}}(\boldsymbol{x})$, $u_{\boldsymbol{\theta}}(\boldsymbol{x})$, $l_{\boldsymbol{\xi}}(\boldsymbol{x})$, respectively. In this work, we use fully connected ReLU architecture for the three networks. Specifically, the PI3NN method constructs the PI in three steps.

Step 1: train $f_{\boldsymbol{\omega}}(\boldsymbol{x})$ to approximate the mean of $f(\boldsymbol{x})$. This completely follows the standard NN-based regression process using the MSE loss in Eq. (1). The trained $f_{\boldsymbol{\omega}}(\boldsymbol{x})$ will serve two purposes. The first is to provide a baseline to generate data for training $u_{\boldsymbol{\theta}}(\boldsymbol{x})$, $l_{\boldsymbol{\xi}}(\boldsymbol{x})$ in **Step 2**; the second is to provide a point estimation of $\mathbb{E}[f]$. In this step, we use the well established regularization techniques, e.g., the conventional L_1 and L_2 penalties, to avoid over-fitting.

Step 2: train $u_{\boldsymbol{\theta}}(\boldsymbol{x})$, $l_{\boldsymbol{\xi}}(\boldsymbol{x})$ to learn the uncertainty profile. To proceed, we use the trained $f_{\boldsymbol{\omega}}(\boldsymbol{x})$ as a baseline to generate two separate datasets, denoted by $\mathcal{D}_{\text{upper}}$ and $\mathcal{D}_{\text{lower}}$, respectively, 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 use $\mathcal{D}_{\text{upper}}$ to train $u_{\boldsymbol{\theta}}(\boldsymbol{x})$, and use $\mathcal{D}_{\text{lower}}$ to train $l_{\boldsymbol{\xi}}(\boldsymbol{x})$. To make sure the outputs of $u_{\boldsymbol{\theta}}(\boldsymbol{x})$, $l_{\boldsymbol{\xi}}(\boldsymbol{x})$ are positive, we add the operation $\sqrt{(\cdot)^2}$ to the output layer of both networks. The two NNs are trained *separately* using the standard MSE loss, i.e.,

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

Unlike the sophisticated losses in [15, 2, 21], the three NNs of PI3NN are trained using the standard MSE without introducing unusual hyperparameters, which promises a more robust NN train.

Step 3: construct the PI via root-finding methods. Note that $u_{\boldsymbol{\theta}}(\boldsymbol{x}), l_{\boldsymbol{\xi}}(\boldsymbol{x})$ do not directly represent the upper and lower bounds of the PI. Instead, they only approximate the difference between the data and $f_{\boldsymbol{\omega}}$ described by the datasets $\mathcal{D}_{\text{upper}}$ and $\mathcal{D}_{\text{lower}}$. In this work, we define the upper and lower bounds of the PI as

$$U(\boldsymbol{x}) = f_{\boldsymbol{\omega}}(\boldsymbol{x}) + \alpha u_{\boldsymbol{\theta}}(\boldsymbol{x}), \qquad L(\boldsymbol{x}) = f_{\boldsymbol{\omega}}(\boldsymbol{x}) - \beta v_{\boldsymbol{\xi}}(\boldsymbol{x}), \tag{4}$$

where α and β are two unknown scalars. For a given quantile $\gamma \in [0, 1]$, we use the bisection method [20] to determine the value of α and β by finding the roots of the following equations:

$$\begin{aligned} Q_{\text{upper}}(\alpha) &= \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}_{\text{upper}}} \mathbf{1}_{y_i > U(\boldsymbol{x}_i)}(\boldsymbol{x}_i, y_i) - N(1-\gamma)/2 = 0, \\ Q_{\text{lower}}(\beta) &= \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}_{\text{lower}}} \mathbf{1}_{y_i < L(\boldsymbol{x}_i)}(\boldsymbol{x}_i, y_i) - N(1-\gamma)/2 = 0, \end{aligned} \tag{5}$$

where N is the number of samples in $\mathcal{D}_{\text{train}}$ and $\mathbf{1}(\cdot)$ is the indicator function, defined by

$$\mathbf{1}_{y_i > U(\boldsymbol{x}_i)}(\boldsymbol{x}_i, y_i) = \begin{cases} 1, & \text{if } y_i > U(\boldsymbol{x}_i), \\ 0, & \text{otherwise,} \end{cases} \quad \text{and} \quad \mathbf{1}_{y_i < L(\boldsymbol{x}_i)}(\boldsymbol{x}_i, y_i) = \begin{cases} 1, & \text{if } y_i < L(\boldsymbol{x}_i), \\ 0, & \text{otherwise,} \end{cases}$$

which count how many train samples are outside the interval $[L(\boldsymbol{x}), U(\boldsymbol{x})]$. When the root-finding problems in Eq. (5) are exactly solved[1] (i.e., $Q_{\text{upper}}(\alpha_r) = Q_{\text{lower}}(\beta_r) = 0$), the number of training samples falling in $[L(\boldsymbol{x}), U(\boldsymbol{x})] = [f_{\boldsymbol{\omega}} - \beta_r l_{\boldsymbol{\xi}}, f_{\boldsymbol{\omega}} + \alpha_r u_{\boldsymbol{\theta}}]$ will be exactly $N\gamma$. In this way, our prediction interval method can produce an accurate uncertainty bound that precisely encloses a specified portion of data with a narrow interval width. Moreover, our prediction interval calculation does not impose any distributional assumptions to enable a general application.

Training our model once produces a prediction value and a PI that captures aleatoric uncertainty. Training PI3NN multiple times with different initialization contains model diversity, and the variance of the predictions can be used as an estimate of epistemic uncertainty. Like the studies in [18, 22], we perform an ensemble of PI3NN models and summarizes the PI results as the uncertainty estimate.

4.1 Identifying out-of-distribution (OOD) samples

When using the trained model $f_{\boldsymbol{\omega}}(\boldsymbol{x})$ to make predictions for $\boldsymbol{x} \notin \mathcal{D}_{\text{train}}$, it is required that the UQ method can accurately identify the OOD samples and reasonably quantify their uncertainty, i.e., for $\boldsymbol{x} \notin \mathcal{D}_{\text{train}}$, the PI's width increases with the distance between $\boldsymbol{x}$ and $\mathcal{D}_{\text{train}}$. We achieve OOD identification by properly initializing one parameter of $u_{\boldsymbol{\theta}}$ and $l_{\boldsymbol{\xi}}$. Specifically, we add a few more operations in **Step 2** before training $u_{\boldsymbol{\theta}}$ and $l_{\boldsymbol{\xi}}$.

- Define $u_{\boldsymbol{\theta}}$ and $l_{\boldsymbol{\xi}}$ as fully-connected ReLU networks, and initialize their weights and biases using the default option.
- Compute the mean outputs $\mu_{\text{upper}} = \sum_{i=1}^{N} u_{\boldsymbol{\theta}}(\boldsymbol{x}_i)/N$ and $\mu_{\text{lower}} = \sum_{i=1}^{N} l_{\boldsymbol{\xi}}(\boldsymbol{x}_i)/N$ using the training set $\mathcal{D}_{\text{train}}$ and initial weights and biases.
- *Modify the biases of the output layers of $u_{\boldsymbol{\theta}}$ and $l_{\boldsymbol{\xi}}$ to $c\,\mu_{\text{upper}}$ and $c\,\mu_{\text{lower}}$, where c is a relatively big number (e.g., we use $c = 10$ in our tests).*
- Follow the rest of **Step 2** to train $u_{\boldsymbol{\theta}}$ and $l_{\boldsymbol{\theta}}$ using the MSE loss.

The key ingredient is the modification of the bias of the output layer. It is known that a ReLU network provides a piecewise linear function. The weights and biases of hidden layers defines how the input space is partitioned into a set of linear regions [3]; the weights of the output layer determines how those linear regions are combined; and the bias of the output layer acts as a shifting parameter. The weights and biases are usually initialized with some standard distribution, e.g., uniform or Gaussian. Setting the biases to $\mu_{\text{upper}} + c\,\sigma_{\text{upper}}$ and $\mu_{\text{lower}} + c\,\sigma_{\text{lowr}}$ with a big value for c will significantly increase the output of the initial $u_{\boldsymbol{\theta}}$ and $l_{\boldsymbol{\xi}}$. During the training, the loss in Eq. (3) will encourage

[1] Since α and β can vary continuously, it is easy to achieve $Q_{\text{upper}}(\alpha_r) = Q_{\text{lower}}(\beta_r) = 0$.

the decrease of $u_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $l_{\boldsymbol{\xi}}(\boldsymbol{x})$ only for in-distribution samples (i.e., $\boldsymbol{x}_i \in \mathcal{D}_{\text{train}}$), not for OOD samples. Therefore, after training, $u_{\boldsymbol{\theta}}(\boldsymbol{x}) + l_{\boldsymbol{\xi}}(\boldsymbol{x})$ will be bigger in the OOD region than in the in-distribution region. Additionally, due to the continuity of the ReLU network, the $u_{\boldsymbol{\theta}}(\boldsymbol{x}) + l_{\boldsymbol{\xi}}(\boldsymbol{x})$ will increase with the distance between $\boldsymbol{x}$ and $\mathcal{D}_{\text{train}}$. Thus, we define the following *confidence score*

$$\Gamma(\boldsymbol{x}) = \min \left\{ \frac{\sum_{i=1}^{N} (u_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + l_{\boldsymbol{\xi}}(\boldsymbol{x}_i))/N}{u_{\boldsymbol{\theta}}(\boldsymbol{x}) + l_{\boldsymbol{\xi}}(\boldsymbol{x})}, 1.0 \right\}, \tag{6}$$

where $\boldsymbol{x}_i \in \mathcal{D}_{\text{train}}$. Basically, if $\boldsymbol{x}$ is an in-distribution sample, then the confidence score should be around one. As $\boldsymbol{x}$ is move away from $\mathcal{D}_{\text{train}}$, $u_{\boldsymbol{\theta}}(\boldsymbol{x}) + l_{\boldsymbol{\xi}}(\boldsymbol{x})$ is getting bigger so the confidence score become smaller. Note that we could incorporate the scalars α and β for the PI into Eq. (6), but it is not necessary because the confidence score is defined in a relative sense.

5 Experiments

We use three examples to demonstrate our method and compare its performance to two top-performing baselines—QD [18] and DER [2] at https://github.com/TeaPearce/Deep_Learning_Prediction_Intervals and https://github.com/aamini/evidential-deep-learning). We use the hyperparameters suggested by the authors for both methods. We perform all the experiments on a single Ubuntu 20.04 workstation with 64GB DDR4 RAM. All models are implemented with package environment with Python 3.8.3 and Google TensorFlow 2.4.1. The code for reproducing all the experiments are included in the supplementary material.

5.1 Evaluation metrics

We use three evaluation metrics to assess predictive accuracy. Root mean squared error (RMSE) measures the accuracy of prediction value; prediction interval coverage probability (PICP) and mean prediction interval width (MPIW) assess the quality of predictive uncertainty. PICP represents the ratio of dataset samples that fall within their respective PIs, and MPIW calculates the average width of the PIs for the samples. Taking the training set $\mathcal{D}_{\text{train}} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$ as an example, we use k_i to indicate whether the output y_i is located in the PI, i.e.,

$$k_i = \begin{cases} 1, & \text{if } L(\boldsymbol{x}_i) \leq y_i \leq U(\boldsymbol{x}_i), \\ 0, & \text{otherwise} \end{cases} \qquad \text{and} \qquad C = \sum_{i=1}^{N} k_i, \tag{7}$$

where $L(\boldsymbol{x})$ and $U(\boldsymbol{x})$ are the upper and lower bounds of the PIs. Then, PICP and MPIW are calculated by the following formula:

$$\text{PICP} = \frac{C}{N} \quad \text{and} \quad \text{MPIW} = \frac{1}{N} \sum_{i=1}^{N} U(\boldsymbol{x}_i) - L(\boldsymbol{x}_i). \tag{8}$$

A high-quality UQ method should produce PIs such that the PICP is close to the defined confidence level γ (commonly set to 0.95) while having MPIW as small as possible. Unlike the QD method [18] that incorporates the PICP and MPIW into its loss, our PI3NN method aims to achieve high-quality PICP and MPIW without explicitly incorporating these two criteria into the loss.

Another metric to evaluate the quality of predictive uncertainty is to assess its generalization to domain shift, also referred to as out-of-distribution (OOD) samples. A sound, well-calibrated UQ method should provide a high predictive uncertainty on OOD data. Besides evaluation of uncertainty calibration quality, we additionally assess a method's robustness from two aspects—assumptions on data distribution and sensitivity to hyperparameters. A robust UQ method should impose no assumptions on data and should be insensitive to hyperparameters. In assessment, we choose QD and DER as baselines because they are top-performers and have demonstrated comparable and even superior performance to the state-of-the-art such as Deep ensembles [15] and MC-dropout [6] in terms of RMSE, PICP and MPIW. Moreover, DER also demonstrated top performance for quantifying uncertainty on OOD samples and QD introduced less hyperparameters in its loss but still obtained competitive results compared to its followup methods in [22, 21]. Detailed information about experiment setup and more experimental results are available in Appendix.

5.2 A one-dimensional non-Gaussian dataset

We first qualitatively compare the performance of our PI3NN method against the QD and DER baselines on a one-dimensional non-Gaussian cubic regression dataset (Figure 1). Following [7, 15, 2], we train models on $y = x^3 + \varepsilon$ within $[-4, 4]$ and test within $[-7, 7]$. 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 PI3NN method and QD capture about 95% of training data with tight bounds in $[-4, 4]$, while DER produces an unnecessarily wide lower bound in $[-4, 4]$ due to its Gaussian assumption on the noise's distribution. Additionally, our PI3NN method and DER produce a reasonably wide predictive uncertainty in the OOD region $[-7, -4] \cup [4, 7]$, while QD results in a very narrow (overconfident) uncertainty bound on the OOD samples.

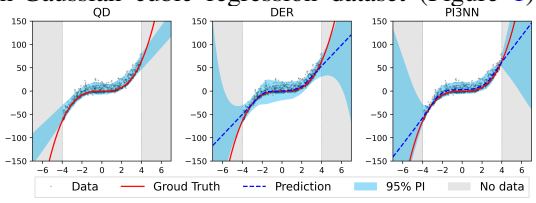

Figure 1: 95% PI for $y = x^3 + \varepsilon$ with asymmetric noise ε. Our PI3NN method enables precise prediction within the training regime and reasonably wide uncertainty estimates in regions with on training data. In contrast, QD [18] underestimate the uncertainty in OOD region and DER [2] overestimate the uncertainty in training regime.

5.3 UCI regression benchmarks

In our second experiment, we compare our PI3NN method to QD and DER for predictive uncertainty estimation on ten real world datasets (the UCI datasets [5]) widely used as regression benchmarks and in UQ methods inter-comparison studies.

Data splitting. To ensure fair comparison, we randomly split each dataset into a training set with 90% samples and a test set with 10% samples. We create five pairs of training and test sets by changing random seeds to test sensitivity of the methods with respect to different data splitting.

Setup for PI3NN: The three NNs used in PI3NN have the same network architecture and hyper-parameters configurations: one-hidden layer with ReLU nonlinearity, containing 100 neurons for all datasets. The Adam optimizer is used with learning rate being 0.01 and the maximum epochs being 50,000. Conventional L1 and L2 regularization are implemented with both penalties set to 0.02. Our method does not introduce unusual hyperparameters that require fine tuning, and one model setup works for all of the experiments. Meanwhile, any traditional NN training techniques that proved to be effective are applicable to our method.

Setup for QD: The hyperparameters of QD used in this study are taken from the authors. They provided full details of the first two datasets (*boston* and *concrete*) in their code. However, there is no suggested hyperparameters (e.g., a total number of epochs, learning rate, learning decay rate, initialization variance) for the remaining 8 datasets. We apply the same hyperparameters from the *boston* case to the 8 datasets, and assign 800 as the maximum epochs (except for the *MSD* case that has a maximum of 20 epochs). All of the NNs contain one hidden layer with 50 neurons, except for two larger datasets—*protein* and *MSD*—which have 100 hidden neurons. QD-soft loss functions and ReLU activation functions are used across all experiments. The batch size is set to 100 for all cases. The QD method related parameters such as the softening variable s is set to 160.0. We use the same weight parameter λ (Eq. (15) in [18]) as in the QD paper regarding the UCI datasets, where $\lambda = 4.0$ for *naval*, 6.0 for *yacht*, 30.0 for *wine*, 40.0 for *protein*, and 15.0 for rest of the datasets.

Setup for DER: We use the model setup and hyperparameters presented in the DER paper for our comparison study. All of the NNs have one hidden layer with 50 neurons with ReLU as activation functions. The output layer is modified with 4 neurons to produce 4 parameters of the evidential distribution. The total number of epochs is set to 40 by default. Since the PICP and MPIW calculations are not included in the DER paper, we compute the PICP and MPIW using the following formula

$$U_{\text{DER}} = \mu_{\text{DER}} + 1.96\sigma_{\text{DER}} \quad L_{\text{DER}} = \mu_{\text{DER}} - 1.96\sigma_{\text{DER}},$$

where μ_{DER} and σ_{DER} are produced by the DER method. The value 1.96 corresponds the 95% of the PI under the Gaussian assumption.

Results: We aim to produce a tight 95% PI for each dataset. We have five pairs of training and test datasets. For each pair, we perform ensemble training with five different NN initializations (i.e., five different random seeds), so we have a total of 25 runs for each dataset. The results shown in Table

1 are the mean and the standard deviation value of PICP, MPIW and RMSE computed using the results of the 25 runs. We have the following observations. First, our PI3NN method significantly outperforms DER, because MPIW generated by DER are too wide such that PICP are very close to 1.0 for eight out of the ten datasets. Although DER produces the most accurate point estimate (indicated by RMSE) for three datasets, its generated PIs are so wide that provide little information about the uncertainty. Second, PI3NN significantly outperforms QD with respect to RMSE, because the loss function of QD does not encourage the optimizer to produce an accurate point estimation. Third, PI3NN only marginally outperforms QD in terms of PICP and MPIW in Table 1.

QD uses different hyperparameter values in its loss function for different datasets to produce the results in Table 1, and those hyperparameters were carefully fine-tuned, as indicated in [18]. In contrast, PI3NN only uses the standard MSE loss without introducing unusual hyperparameters. Thus, to make a fair comparison, we test the sensitivity of QD to its two loss associated hyperparameters (softening variable s and weight parameter λ) with the permutation of two arrays. The s array ranges from 100.0 to 200.0 with an incremental of 10.0, and the λ array starts with 5.0 and ends at 25.0 with an incremental of 5.0. In total, we conducted 55 experiments for each dataset using the combination of the two hyperparameter samples while keeping the other parameters fixed. The box-plots of PICP is given in Figure 2 to illustrate the sensitivity of PICP with respect to the two hyperparameters in QD's loss function. The box-plots of MPIW and RMSE are given in the Appendix. We observe that PICP is very sensitive to the two QD hyperparameters. For example, in the boston case, the difference between the biggest and the smallest PICP is about 28% of the target 0.95 PICP value. The PI3NN method does not have this issue, because it only uses the standard MSE loss. The combination of Table 1 and Figure 2 demonstrates the superior robustness of our PI3NN method compared with QD.

Dataset	PICP			MPIW			RMSE		
	DER	QD	PI3NN	DER	QD	PI3NN	DER	QD	PI3NN
Boston	**0.87±0.03**	0.82±0.05	0.84±0.05	1.26±0.18	0.95±0.08	0.93±0.11	2.99±0.28	3.19±0.29	**0.33±0.04**
Concrete	1.00±0.00	0.86±0.04	**0.91±0.02**	1249±191	0.87±0.06	1.18±0.21	5.14±0.33	5.93±0.51	**0.32±0.03**
Energy	0.98±0.00	0.89±0.04	**0.95±0.02**	101±89	0.73±0.07	0.94±0.29	2.39±0.35	3.29±0.44	**0.20±0.04**
Kin8nm	1.00±0.00	0.91±0.01	**0.94±0.00**	3915±2.46	2.14±0.07	1.36±0.06	**0.06±0.00**	0.17±0.01	0.36±0.01
Naval	1.00±0.00	0.96±0.02	**0.94±0.01**	3920±0.00	5.99±6.72	0.51±0.17	**0.01±0.01**	0.01±0.01	0.14±0.02
Power	1.00±0.00	0.94±0.01	**0.95±0.00**	3220±362	0.86±0.01	0.99±0.17	2.96±0.07	4.18±0.13	**0.24±0.01**
Protein	0.99±0.00	0.93±0.00	**0.95±0.00**	285±209	2.47±0.02	2.80±0.04	4.26±0.11	5.48±0.06	**0.77±0.01**
Wine	0.98±0.01	0.91±0.02	**0.94±0.02**	4.30±0.22	2.10±0.08	2.84±0.26	**0.55±0.04**	0.64±0.04	0.76±0.05
Yacht	0.84±0.02	0.88±0.06	**0.92±0.04**	0.45±0.10	0.25±0.05	0.23±0.07	0.45±0.10	2.19±1.01	**0.06±0.01**
MSD	0.98±0.00	0.92±0.01	**0.95±0.00**	9.16±2.49	3.82±4.74	1.71±0.02	9.01±0.30	27.40±4.65	**0.50±0.01**

Table 1: Our PI3NN method outperforms DER, because MPIW generated by DER is too wide such that PICP is very close to 1.0 for eight out of the ten datasets. PI3NN outperforms QD for the RMSE metric, because the loss function of QD does not encourage the optimizer to produce an accurate point estimate. PI3NN marginally outperforms QD with respect to PICP and MPIW, but the QD results are based on fine-tuned hyperparameters for each dataset. The sensitivity of QD's performance to its hyperparameters is given in Figure 2.

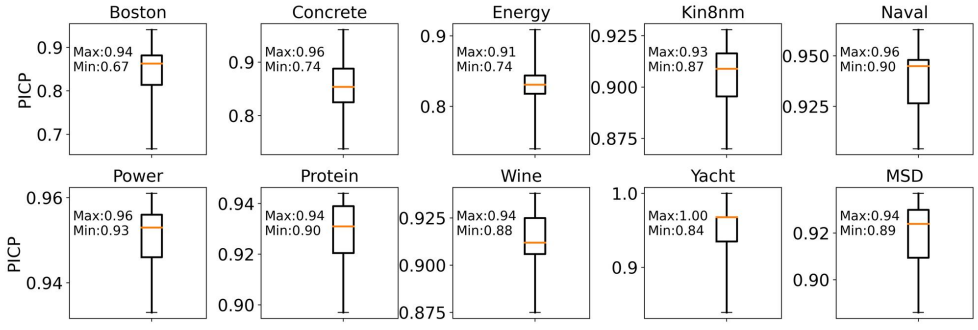

Figure 2: Illustration of the high sensitivity of PICP with respect to the two hyperparameters in QD's loss function. For example, in *boston* case, the difference between the biggest and the smallest PICP is about 28% of the target 0.95 PICP value. In comparison, PI3NN does not have this issue because it only uses standard MSE loss. The combination of Table 1 and Figure 2 shows the superior robustness of PI3NN to QD.

5.4 Test on OOD identification

We use a 10-dimensional cubic function $f(\boldsymbol{x}) = \frac{1}{10}(x_1^3 + \cdots + x_{10}^3)$ to demonstrate the effectiveness of the proposed strategy in Section 4.1 in OOD identification. The training data for the input $\boldsymbol{x}$ is

generated by drawing 5,000 samples from the standard Gaussian distribution $\mathcal{N}(0,1)$; the training data for the output is then obtained by $y = f(\boldsymbol{x}) + \varepsilon$ with ε also following $\mathcal{N}(0,1)$. We define a test set with 10,000 OOD samples. The samples of $\boldsymbol{x}$ are drawn from $\mathcal{N}(0,25)$, which creates a significant distribution shift from the training set. We use a single hidden layer ReLU architecture with 200 hidden neurons for $f_{\boldsymbol{\omega}}(\boldsymbol{x})$, $u_{\boldsymbol{\theta}}(\boldsymbol{x})$, $l_{\boldsymbol{\xi}}(\boldsymbol{x})$. We set the constant c in Eq. (6) to 10 in this test. We use Adam to train the three NNs with the learning rate of 0.01.

The results are shown in Figure 3. As expected, the confidence score for the training samples (red dots) are close to 1. For the test samples (blue dots), the confidence score significantly decreases with the increase of the distance between $\boldsymbol{x}$ and the mean of the training samples. The well separation between the red and blue clouds shows effectiveness of our OOD identification approach in Section 4.1. In Figure 3 (b), we use exactly the same setting as that for generating Figure 3 (a) but do not modify the biases of the output layers of $u_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $l_{\boldsymbol{\xi}}(\boldsymbol{x})$. In this case, we can see that the confidence score provides many over-confident predictions, i.e., many OOD samples (blue dots) are given pretty high confidence scores. This demonstrates that the modification of the biases plays a critical role in OOD identification for the PI3NN method. For comparison, we also run QD and DER for the same problem by replacing $u_{\boldsymbol{\theta}} + l_{\boldsymbol{\xi}}$ with the width of the 95% PIs they produced. Both QD and DER suffer from the same problem as PI3NN without OOD detection, i.e., producing many over-confidence PIs.

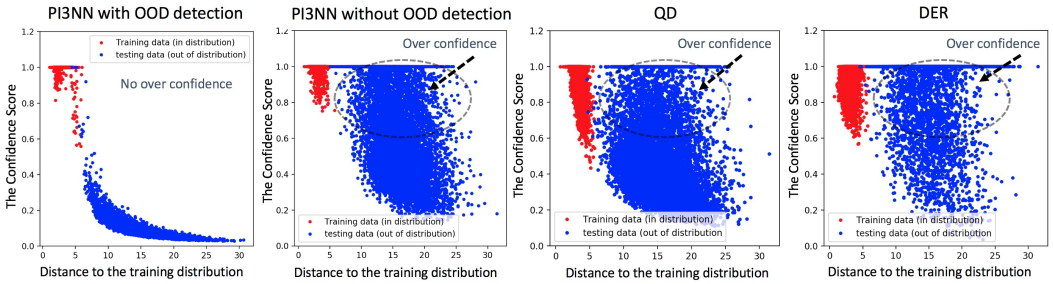

Figure 3: Comparison of OOD identification. The red dots are in distribution samples, and the blue dots are OOD samples. The horizontal axis measures the distance between a sample and the training distribution (the mean of the $\boldsymbol{x}$ values in the training set). The OOD samples have bigger distances (farther from the training distribution) than the in-distribution samples. PI3NN with the OOD detection (described in Section 4.1) can successfully separate the in-distribution and OOD samples by giving low confidence to OOD samples, while PI3NN without OOD detection, QD and DER produce many over-confident predictions, i.e., giving high confidence scores to OOD samples.

6 Conclusion and discussion

This work proposes a different route from the state-of-the-art methods [15, 2, 21] on how to design robust PI methods for uncertainty quantification. Instead of designing sophisticated loss functions, we only use the standard MSE loss in the PI3NN method. Both the architecture and the training process for the three neural networks used in PI3NN are standard, such that choosing the commonly seen hyperparameters, e.g., the learning rate, the number of hidden neurons, the number of layers, is almost a textbook problem that can be easily conducted without fine tuning by non-expert users. Additionally, the OOD identification capability of PI3NN is also simply designed by a small modification of the bias of the output layer, which is demonstrated to be very effective in Figure 3.

The limitations of the PI3NN method include: (1) For a target function with multiple outputs, each output needs to have its own PI and OOD confidence score. The PI and the confidence score cannot oversee all the outputs. For example, this could make it challenging to apply PI3NN to NN models having image as outputs, (e.g., autoencoders). (2) The effectiveness of the OOD detection approach depends on that there are sufficiently many piecewise linear regions (of ReLU networks) in the OOD area. So far, this is achieved by the standard random initialization (ensure uniform distributed piecewise linear regions at the beginning of training) and L_1/L_2 regularization (ensure the linear regions not collapse together around the training set). However, there is no guarantee of uniformly distributed piecewise linear regions after training. Improvement of this requires significant theoretical work on how to manipulate the piecewise linear function defined by the ReLU network. This work is for purely research purpose and will have no negative social impact.

References

[1] Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V., and Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges.

[2] Amini, A., Schwarting, W., Soleimany, A., and Rus, D. (2020). Deep evidential regression. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14927–14937. Curran Associates, Inc.

[3] Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. (2016). Understanding Deep Neural Networks with Rectified Linear Units. *arXiv e-prints*, page arXiv:1611.01491.

[4] Carney, J., Cunningham, P., and Bhagwan, U. (1999). Confidence and prediction intervals for neural network ensembles. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*, volume 2, pages 1215–1218 vol.2.

[5] Dua, D. and Graff, C. (2017). UCI machine learning repository.

[6] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA. PMLR.

[7] Hernández-Lobato, J. M. and Adams, R. P. (2015). Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1861–1869. JMLR.org.

[8] Heskes, T. (1996). Practical confidence and prediction intervals. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'96, page 176–182, Cambridge, MA, USA. MIT Press.

[9] Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14(4):1303–1347.

[10] Hwang, J. T. G. and Ding, A. A. (1997). Prediction intervals for artificial neural networks. *Journal of the American Statistical Association*, 92(438):748–757.

[11] Jospin, L. V., Buntine, W. L., Boussaïd, F., Laga, H., and Bennamoun, M. (2020). Hands-on bayesian neural networks - a tutorial for deep learning users. *CoRR*, abs/2007.06823.

[12] Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. (2011). Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE Transactions on Neural Networks*, 22(3):337–346.

[13] Koenker, R. and Bassett, G. (1978). Regression quantiles. *Econometrica*, 46(1):33–50.

[14] Koenker, R. and Hallock, K. F. (2001). Quantile regression. *Journal of Economic Perspectives*, 15(4):143–156.

[15] Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6405–6416, Red Hook, NY, USA. Curran Associates Inc.

[16] MacKay, D. J. C. (1992). A practical bayesian framework for backpropagation networks. *Neural Comput.*, 4(3):448–472.

[17] Neal, R. M. (2012). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

[18] Pearce, T., Brintrup, A., Zaki, M., and Neely, A. (2018). High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4075–4084. PMLR.

[19] Pearce, T., Leibfried, F., and Brintrup, A. (2020). Uncertainty in neural networks: Approximately bayesian ensembling. In Chiappa, S. and Calandra, R., editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 234–244. PMLR.

[20] Quarteroni, A., Sacco, R., and Saleri, F. (2006). *Numerical Mathematics (Texts in Applied Mathematics)*. Springer-Verlag, Berlin, Heidelberg.

[21] S. Salem, T., Langseth, H., and Ramampiaro, H. (2020). Prediction intervals: Split normal mixture from quality-driven deep ensembles. In Peters, J. and Sontag, D., editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 1179–1187. PMLR.

[22] Simhayev, E., Katz, G., and Rokach, L. (2021). Piven: A deep neural network for prediction intervals with specific value prediction.

[23] Springenberg, J. T., Klein, A., Falkner, S., and Hutter, F. (2016). Bayesian optimization with robust bayesian neural networks. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

[24] Touretzky, D. S., Leen, T. K., Nix, D. A., and Weigend, A. S. (1995). Learning local error bars for nonlinear regression.

[25] VlEAUX, R. D. D., Schumi, J., Schweinsberg, J., and Ungar, L. H. (1998). Prediction intervals for neural networks via nonlinear regression. *Technometrics*, 40(4):273–282.

[26] Wang, H. and Yeung, D.-Y. (2021). A survey on bayesian deep learning.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See supplemental material

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the Conclusion and discussion section
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the Conclusion and discussion section
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] All code, data and the instructions for reproducing the results are included in the supplemental materials
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] All training details are described in the manuscript and also can be found in the provided code, code instructions are provided either in the supplemental material or in the comment section within the code
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Included in the main text and supplemental material
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Described in the main text, see Section 5 Experiments
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Yes, the code we took for comparison and the UCI data sets are properly cited in the main text
   (b) Did you mention the license of the assets? [Yes] Included in the supplemental material
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Included in the supplemental material
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] Included in the supplemental material
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]