

This is the author-created version of the following work:

Kabir, HM Dipu, Mondal, Subrota Kumar, Khanam, Sadia, Khosravi, Abbas, Rahman, Shafin, Chalak Qazani, Mohammad Reza, Alizadehsani, Roohallah, Asadi, Houshyar, Mohamed, Shady, Nahavandi, Saeid, and Acharya, U. Rajendra (2023) *Uncertainty aware neural network from similarity and sensitivity. Applied Soft Computing*, 149 (Part A) .

Access to this file is available from:

<https://researchonline.jcu.edu.au/86721/>

© 2023 Elsevier B.V. All rights reserved.

Please refer to the original source for the final version of this work:

<https://doi.org/10.1016/j.asoc.2023.111027>

Uncertainty Aware Neural Network from Similarity and Sensitivity

H M Dipu Kabir^a, Subrota Kumar Mondal^b, Sadia Khanam^c, Abbas Khosravi^d,
Shafin Rahman^e, Mohammad Reza Chalak Qazani^d, Roohallah Alizadehsani^d,
Houshyar Asadi^d, Shady Mohamed^d, Saeid Nahavandi^f, U Rajendra Acharya^{g,h,i}
^a Independent researcher. ^bFIE, Macau University of Science and Technology, Macao.
^cDhaka Dental College, Dhaka, Bangladesh.
^dIISRI, Deakin University, Australia. ^eNorth South University, Dhaka, Bangladesh.
^fHarvard University, Allston, MA, USA. ^gECE, Ngee Ann Polytechnic, Singapore.
^hBME, SST, SUSS University, Singapore. ⁱBIME Asia University, Taiwan.
hmdkabir@connect.ust.hk

Abstract

Recent uncertainty quantification approaches lack transparency. Algorithms often perform poorly in an input domain and the reason for poor performance remains unknown. Therefore, we present a neural network training method that considers similar samples with sensitivity awareness in this paper. In the proposed NN training method for UQ, first, we train a shallow NN for the point prediction. Then, we train another NN to predict absolute differences between targets and predictions. In the next step, we select each sample in the training set one by one and compute both prediction and error sensitivities. Then we select similar samples with sensitivity consideration and save indexes of similar samples. The ranges of an input parameter become narrower when the output is highly sensitive to that parameter. After that, we construct initial uncertainty bounds (UB) by considering the distribution of sensitivity-aware similar samples. Prediction intervals (PIs) from initial uncertainty bounds are larger and cover more samples than required. Therefore, we train bound correction NN. As following all the steps for finding UB for each sample requires a lot of computation and memory access, we train a UB computation NN. The UB computation NN takes an input sample and provides an uncertainty bound. The UB computation NN is the final product of the proposed approach. We have achieved superior performance in most situations through the proposed transparent method. Scripts of the proposed method are available in the following GitHub repository: github.com/dipuk0506/UQ

Keywords: Uncertainty Bound, Probabilistic Forecast, Neural Network, Prediction Interval, Uncertainty Quantification, Heteroscedastic Uncertainty.

1. Introduction

Whenever any prediction system receives an unexpectedly high prediction error, people try to investigate the reason for that failure. NNs consist of weights and bi-

ases. Values of those parameters are determined during the training. Therefore, the training procedure and the sample distribution play a vital role in NNs heteroscedastic performance [1, 2]. The knowledge of sample distribution also helps individuals in the investigation. Moreover, NNs training needs to be robust to achieve competitive performance on rare and critical samples, and a high overall performance [3, 4, 5].

Traditional regressive models can find the regression mean of a quantity for any input combination within the range [6, 7]. They also provide an overall statistical error, such as the mean-square-error (MSE), the root-mean-square-error (RMSE), etc [8]. Traditional point prediction models, with an overall statistical error, cannot represent the level of heteroscedastic uncertainty. The level of uncertainty can be high in one input domain and low in another input domain [9, 10]. Prediction intervals with a coverage probability can indicate the level of heteroscedastic uncertainty. Regions with narrower intervals have lower uncertainty. Regions with wider interval have lower uncertainty [11]. Uncertainty quantification is becoming popular in various fields [12, 13]. Neural networks are also getting popularity due to their optimal performances [14]. There exist several popular approaches for constructing prediction intervals. Bayesian neural networks are getting popular due to their applications in deep learning. However, Bayesian regressive neural networks struggle to maintain a coverage probability, close to the expected coverage probability. The cost-function-based prediction intervals can provide a narrow interval and the coverage probability becomes very close to the expected coverage probability [15]. However, there are different cost functions proposed by different researchers, and there exist debates on their acceptability. A group of researchers prescribed reducing failure distance and some other researchers prescribed bringing the target near the mid-interval. There is also debate on penalizing high coverage probability. Therefore, we propose a method that trains NNs based on similar samples instead of a cost function.

Although neural networks have brought promising results in numerous fields, they are prone to overfit in highly uncertain datasets [16, 17]. Predictions from neural networks mainly face two types of uncertainties: (i) aleatoric uncertainty and (ii) epistemic uncertainty. Epistemic uncertainty is known as the modeling error [18]. NNs of optimal size can result in a much lower epistemic uncertainty. However, the presence of a high aleatoric uncertainty can potentially bring a high epistemic uncertainty as the NN may fail to evaluate the exact regression mean. Moreover, optimal size NN depends on both application and data. Shallow NNs with a few neurons can potentially exhibit the underfit issue due to insufficient parameters. Deeper NNs can potentially exhibit overfitting errors [19]. Trial and error is the only way to achieve the optimal size. Both the aleatoric and epistemic uncertainty may stay combinedly in many situations. The third type of uncertainty reveals, while someone tries to present the uncertainty. That is representational uncertainty.

Fig. 1 presents rough sketches presenting aleatoric and epistemic uncertainties and their combined uncertainties. Fig. 1(a) presents the distribution of targets. Fig. 1(b) presents a rough interval prediction of the expected value. When the sample density is high with aleatoric variances, the NN mostly exhibits aleatoric uncertainties. In regions lacking samples, the prediction system exhibits epistemic uncertainty due to the lack of training examples. Regions with insufficient samples can potentially exhibit combined aleatoric and epistemic uncertainty. In Fig. 1 (b), rough diagrams are drawn

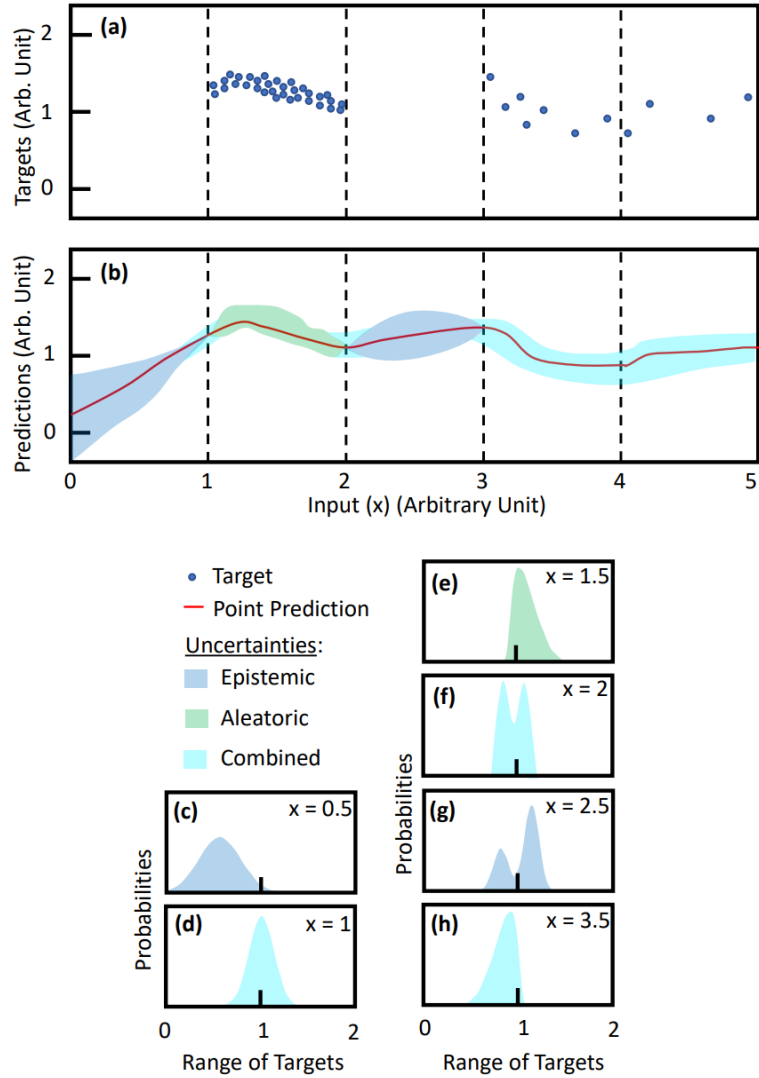


Figure 1: Rough sketches for the visualization of the distributions of uncertainties. Sub-sketch (a) presents the distribution of targets. Sub-sketch (b) presents aleatoric, epistemic, and combined uncertainties due to insufficient data as intervals. Sub-sketches (c) to (h) present probable probability distributions of uncertainties for several different values of inputs.

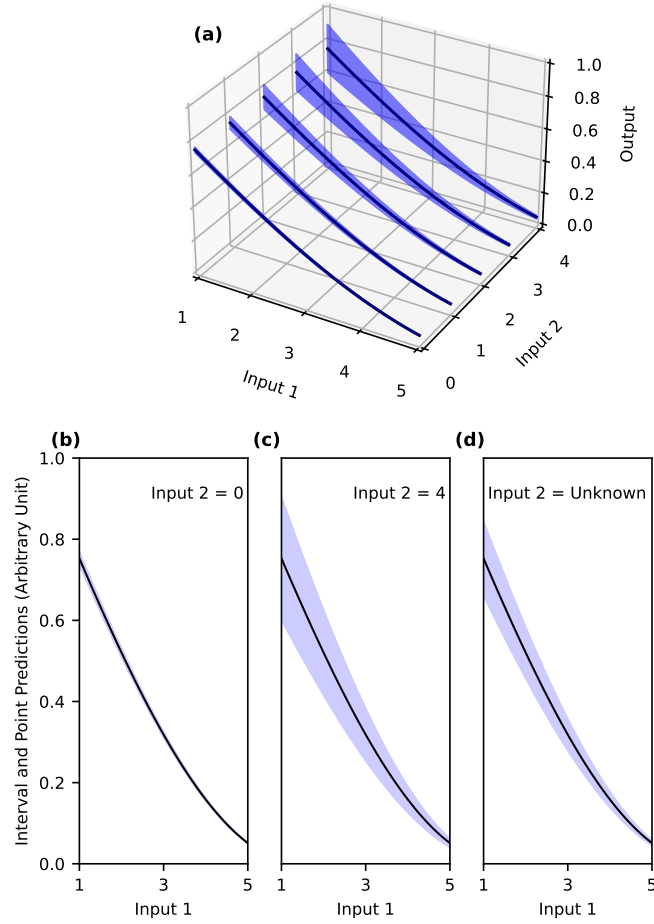


Figure 2: A parameter may not affect the prediction but may cause uncertainties. This figure shows rough sketches to visualize the influences of such parameters in uncertainty quantification. In subplot (a), *Input 1* is responsible for both prediction and uncertainty. *Input 2* is responsible for uncertainties. Subplots (b) and (c) present uncertainties for two different values of *Input 2*. Subplot (d) presents the level of quantified uncertainty when *Input 2* is unknown.

for visualization. Combined uncertainty can also occur at the edge of two regions. There is an edge at $x=1$, joining a no-sample region and a high-density region. That edge has an uncertainty partially caused by the aleatoric variation in the high-density region and partially caused by the lack of example samples in the no-sample region. As the width of the intervals represents the level of uncertainty, it is possible to understand the level of heteroscedastic uncertainties and a highly probable region of outcome from prediction intervals. However, representational uncertainty exists while presenting uncertainties as intervals. Intervals do not represent multimodality and skewness of uncertainty distributions.

Fig. 1(c)-(h) are presenting probability densities at different locations. Probability densities can provide a more detailed visualization of uncertainty. In Fig. 1(c)-(d), probability distributions are purely Gaussian. In Fig. 1(e) and Fig. 1(h), probability distributions are skewed Gaussian. In Fig. 1(e)-(g), probability distributions are multimodal. An interval contains three numbers, the upper bound, the lower bound, and the coverage probability. The prediction interval-based representation of uncertainties cannot express the skewness or multimodality, resulting in representation uncertainties. Moreover, real-world quantities are dependent on numerous factors [20, 21]. Drawing probability distribution over one variable is possible while i) other variables contain a certain set of values, or ii) other variables are unknown. Therefore, avoiding representational uncertainty is almost impossible in current representation approaches.

In a real-world scenario, many unknown parameters may cause uncertainty in the value of a quantity. The temperature in a location on a sunny day may face uncertainty depending on the presence of a cloud at that location. Fig. 2 presents a system containing two inputs. The first input is responsible for both prediction and uncertainty and the second input is partially responsible for uncertainty. A 3D plot in Fig. 2(a) shows that the point prediction increases with the decrease of the first input. The magnitude of uncertainty varies based on the second input. The user receives more specific uncertainty information when the second input is known. Fig. 2(b) presents a situation when the second input is zero. The uncertainty becomes very low in such a situation. Fig. 2(c) presents a situation when the second input has its highest value. The uncertainty is much higher than Fig. 2(b). Fig. 2(d) presents uncertainty over the first input when the second input is unknown. That input may have any value between zero to four. Based on the value of the second input, there exist different levels of uncertainties. It is also possible to quantify uncertainty in absence of the second input by averaging the effect of different values of the second input. However, knowing a parameter related to uncertainty makes us more certain about the predicted value and makes intervals statistically narrower with expected coverage probability. When the first input is zero, modeling uncertainty without the information of the first input makes the interval wider than expected. When the first input is four, modeling uncertainty without the information of the first input results in intervals of low statistical coverage probability. Cost function-based NN training methods automatically consider similarity, sensitivity, and error sensitivity when they are trained through a reward-based system. However, they have the following limitations: 1) there is no widely accepted cost function, and 2) we cannot observe similar samples. Similarity-based NN training methods do not face the debate of cost function. However, they do not consider error sensitivity and that results in an inferior performance. We are considering the error sensitivity for the first time

which is not considered in similarity-based prediction systems.

This manuscript is the upgraded version of our previous conference paper [22]. We have improved the paper with the consideration of error sensitivity. Consideration of the error sensitivity has improved the performance. Besides performance improvement, we have shared many resources for future researchers. The scripts of the initial paper were developed on the Matlab software. We develop Python codes for the proposed method. As Python is more popular in data science, we expect that more researchers will use our code. We have also shared scripts of several other approaches. That may also help future researchers. Contributions to the paper are as follows:

1. Consideration of similarity, sensitivity, and error sensitivity for the first time.
2. Users can see similar samples during the investigation.
3. We have achieved superior performance in most situations.

The paper also contains the following information to help readers.

1. Comparison among the proposed method and multiple Bayesian and direct interval construction methods.
2. Explanation of aleatoric and epistemic uncertainty with sketches.
3. Sketches explaining conditional uncertainties.
4. Sketches explaining the reason for considering error sensitivity.
5. Sketches explaining the reason for computing absolute error while computing error sensitivity.
6. Link to scripts of the proposed method and other investigated methods.

The rest of the paper is organized as follows. Section 2 presents a short overview of uncertainty-aware machine learning. Section 3 presents the framework of the proposed method. Section 4 presents investigated datasets. Section 5 presents results with relevant texts, figures, and tables. Section 6 is the concluding section.

2. Research on Uncertainty Aware Machine Learning

Researchers have developed numerous uncertainty-aware machine learning algorithms in recent decades [23, 21, 20]. There are two overlapping research terms in machine learning and uncertainty quantification: i) Uncertainty in ML [24] and ii) ML for UQ [25]. The first term represents the uncertainty in ML models. As deep learning models contain a large number of parameters and often receive a large input, such as high-resolution images and they are prone to overfit [26, 27]. Both overfitting and underfitting cause epistemic uncertainty. The second term is popularly known as the NN-based uncertainty quantification. Many quantities in nature have partially random and partially deterministic portions. NN is applied to quantify uncertainties in those quantities.

2.1. Bayesian Techniques

Bayesian techniques are popular approaches in Deep Uncertainty quantification. In a Bayesian inference, the user holds a prior belief about the previous state. The probability of a hypothesis is determined considering prior beliefs and the available observations. Bayes's theorem is applied to update the probability distribution for upcoming

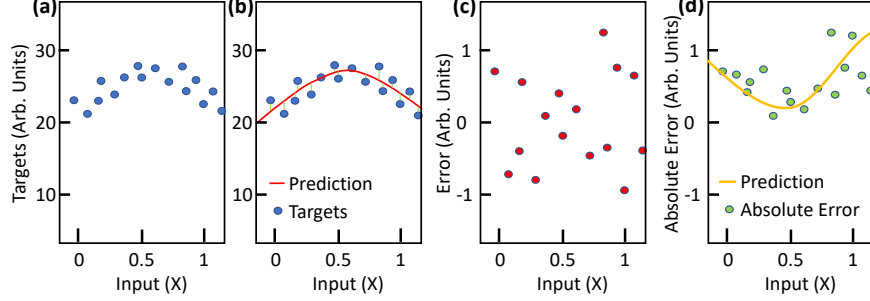


Figure 3: The reason for the development of error prediction NN on absolute error. An input parameter can be responsible for prediction or uncertainty, or both. Subplot (a) visualizes a target distribution over a range of one input. (b) visualizes targets with predictions. (c) presents prediction errors with rough values for visualization. Subplot (e) presents the mean absolute error. Uncertainty is high near 0 and 1. Uncertainty is low near 0.5. Prediction of error cannot represent the level of uncertainty where prediction of absolute error can predict that.

information. In Bayesian statistics, when we observe an event x_E , the probability of happening a hypothesis y_H simultaneously is as follows [28]:

$$P(y_H|x_E) = \frac{P(x_E|y_H)}{P(x_E)} P(y_H) \quad (1)$$

where, $P(\cdot)$ is the probability function. There exist several popular variants of Bayesian Learning. Such as Variational Inference [29], Bayesian Active Learning [30, 31], Dropout as Bayesian Approximation [32, 33], Laplacian Approximations [34, 35], etc.

2.2. Cost Function-based Direct Interval

Cost function-based direct interval construction neural network training is popularly used for uncertainty quantification in natural quantities. Well known examples are uncertainty quantification of wind power generations, electricity demand, electricity price, etc. The lower-upper bound estimation (LUBE) method is the very first and a highly cited cost function-based uncertainty quantification method [36]. The LUBE method constructs prediction-interval-based uncertainty quantification system. The LUBE method considers a combination of prediction interval coverage probability (PICP) and prediction interval normalized average width (PINAW).

The PI normalized average width can be obtained as [37]:

$$PINAW = \frac{1}{R \times n} \sum_{j=1}^n (\bar{y}_j - \underline{y}_j), \quad (2)$$

where, $\bar{y}_j$ is the upper bound for j^{th} sample, $\underline{y}_j$ is the lower bound for j^{th} sample, R is the range of targets, n is the number of samples. $PINAW$ provides a scalar value of the average width. PICP is described as the following equation [37]:

$$PICP = \frac{1}{n} \sum_{j=1}^n c_j \quad (3)$$

where,

$$c_j = \begin{cases} 1, & t_j \in [\underline{y}_j, \bar{y}_j] \\ 0, & t_j \notin [\underline{y}_j, \bar{y}_j], \end{cases}$$

where, t_j is the target for j^{th} sample. They developed the coverage width criterion CWC by combining PICP and PINAW as follows [36]:

$$CWC = PINAW\{1 + \gamma(PICP)e^{\eta(1-PICP)}\}, \quad (4)$$

where, $\gamma(PICP)$ is a penalty for low coverage ($PICP < 1 - \alpha$). $\gamma(PICP)$ is represented by the following equation.

$$\gamma(PICP) = \begin{cases} 1, & PICP < 1 - \alpha \\ 0, & PICP \geq 1 - \alpha, \end{cases}$$

where, $\eta = 50$ is a hyperparameter. α is the target non-coverage probability [38]. After the proposal of the LUBE method, several researchers have proposed improved cost functions [37, 39, 40] and more efficient training methods. Cost-function-based uncertainty quantification NN training methods have recently received a lot of attention in recent years due to their eye-catching performances. However, their performances vary depending on considerations of parameters. A group of researchers concentrates on reducing the average PINAW, while another group aims to optimize PINAW and PICP in critical samples. Different cost functions are developed to achieve good coverage in critical situations. There exist confusion about the relative effectiveness of different cost functions.

2.3. Effectiveness of NNs in Prediction and UQs

Neural networks are the most extensively used predictors in recent decades. According to our literature search, researchers have been modeling engineering, scientific and economic quantities through mathematical equations for centuries. Developing a mathematical model requires observation of input-output relationships for different combinations of other inputs, mathematical knowledge of expressing relationships, and selection of parameters. Therefore, mathematical model development of complex systems requires skilled manpower and a long time for development.

Many relationships between the input and outputs can be confusing to humans. The human brain can search over a limited number of events. To find a statistical relationship humans or machines may require to search over a large number of data. Such as the complex relationship between the temperature and the presence can potentially vary depending on several other factors. It is difficult for a skilled human to derive a mathematical relationship between cloud and temperature. Neural networks can easily find those statistical relationships through reward-oriented training.

Another popular and time-saving technique for the development of any prediction system is the lookup table. The lookup table method does not require extensive work from specialized humans. The prediction system interpolates the probable value from nearby samples. However, the lookup table or similar samples requires large memory.

Data loading time from memories also significantly increases. The memory requirement drastically increases with the number of input parameters. Moreover, datasets often lack an adequate number of samples in rare and critical domains. Neural network model parameters require less memory than the dataset. Training of NNs does not need extensive human work and the trained NN contains a much smaller chunk of data. Therefore, NNs are efficient in terms of both run-time memory and less human involvement.

2.4. *Similarity-based Uncertainty Quantification*

Several researchers have demonstrated similarity-based prediction systems [41, 42, 43, 44]. Researchers can also obtain the level of uncertainty from similar samples [45]. However, such approaches did not get enough attention due to a high execution time. Model-based approaches require a large time during the model training but the execution is fast. Recent advancement in neural networks (NNs) has brought state-of-the-art performances in numerous fields. However, NN often fails with high confidence in critical situations. The NN is a black box predictor. It is difficult to find the reason for any poor performance. A potential solution can be efficient data pre-processing. It is possible to find aleatoric uncertainty and sample density from similar samples. An individual can train NNs on prediction and uncertainty values obtained from the similarity-oriented method. It is also possible to evaluate the statistical performance of trained models to compare performances.

3. The Proposed Framework

The proposed uncertainty quantification framework is computationally extensive during the data preparation for the NN training. However, the execution is faster, as a single input combination is applied to the trained shallow NN to obtain outputs. Besides the uncertainty-bound NN, we also train a sample density NN to help users. Sample density near a sample indicates the experience of the UQ NN near that sample. A very low sample density may cause a high epistemic uncertainty. Therefore, we facilitate users by providing a sample density neural network. In this paper, we propose a method containing the following steps for uncertainty quantification. We often receive data in a different format than the format required to train neural networks. Such as the original dataset may contain a string of samples taken at a certain interval with start and end times. The dataset for proposed NN training requires inputs(X) and output(Y) matrix. When the dataset is not originally in a proper format, the user may need some data pre-processing before applying the proposed steps.

3.1. *Find Similar Samples:*

Finding similar events is the most computationally extensive and critical part of the proposed system. The proposed method considers both error and prediction sensitivities while choosing similar samples. The following steps are followed to find similar samples from the dataset.

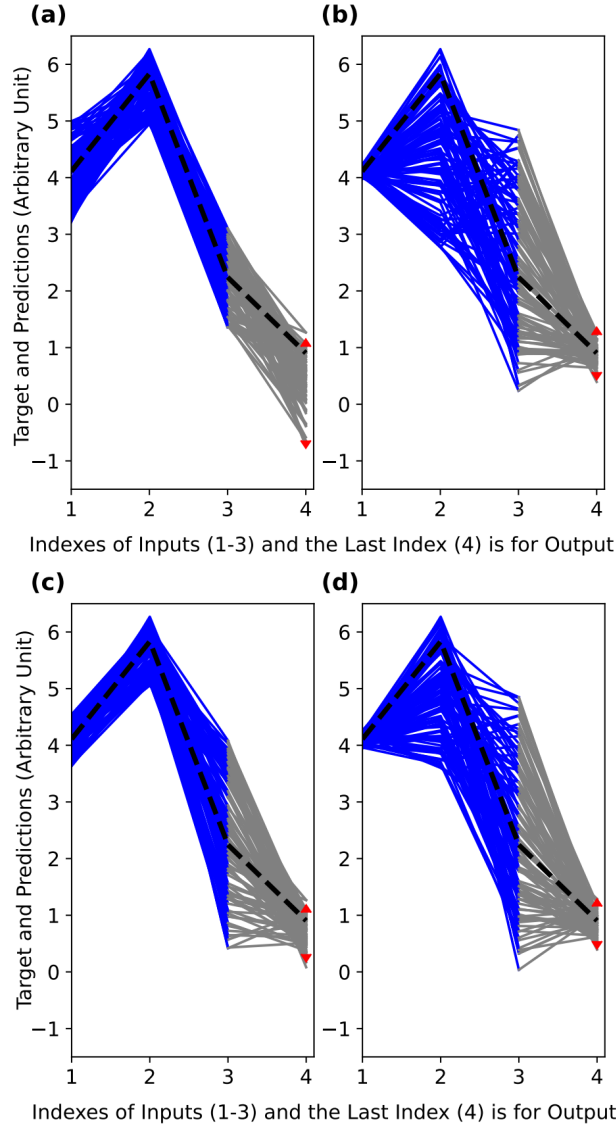


Figure 4: Similar samples with the example sample and derived rough prediction interval. Similar samples and prediction intervals are plotted while (a) only similarity, (b) similarity and prediction sensitivity, (c)-(d) similarity, prediction sensitivity and error-sensitivity. In (c), prediction sensitivity and error-sensitivity have equal weights. In (d), we adjust relative weights of prediction sensitivity and error-sensitivity based on relative variances.

3.1.1. Train a Shallow NN for Prediction:

We train an initial point prediction neural network and apply that neural network for obtaining both the prediction sensitivity and the error sensitivity.

3.1.2. Compute Absolute Errors and Save:

The initially trained NN may have both aleatoric and epistemic uncertainty. However, that point prediction neural network cannot indicate the level of uncertainty. That neural network is applied to the entire training dataset to obtain the absolute error. Through that process, we obtain a new dataset containing input combinations(X) and the absolute differences ($\|NN_p(X) - T\|$) between targets of the original dataset (T) and predictions from the point prediction NN ($NN_p(X)$).

Fig. 3 presents the reason for training NN for absolute errors instead of just errors. Subplot (a) visualizes a target distribution over a range of one input. Subplot (b) visualizes targets with predictions. Subplot (c) presents prediction errors with rough values for visualization. Subplot (e) presents the mean absolute error.

An input parameter can be responsible for prediction or uncertainty, or both. Theoretically, the NN training brings the true-regression-mean. In a perfect situation, NN trained on the error signal should always predict a value very close to zero. Absolute errors are always positive. In less uncertain regions, absolute errors have small values. In a highly uncertain region, absolute errors have both small and high values and the true regression mean of absolute error value represents an indication of the level of uncertainty. In Fig. 3 uncertainty is high near 0 and 1. Uncertainty is low near 0.5. Therefore, prediction of error cannot represent the level of uncertainty where prediction of absolute error can predict that.

3.1.3. Train a Shallow NN for Absolute Errors:

The proposed method considers both the prediction sensitivity and the error sensitivity. The error contains both aleatoric and epistemic uncertainties. Moreover, one individual sample may not represent the actual level of uncertainty near that sample. A region of high uncertainty has a high statistical absolute error. Therefore, we are training a NN with absolute errors to achieve the regression mean of absolute errors.

3.1.4. Compute Sensitivity and Relative Weights:

We compute both prediction and error sensitivities from trained neural networks. Prediction sensitivity of i^{th} input near j^{th} sample can be computed as:

$$S_p(i)|_{X \rightarrow X_j} = \frac{\partial}{\partial x_i} NN_p(X)|_{X \rightarrow X_j} \quad (5)$$

Consideration of sensitivity brings a superior performance compared to similarity alone [22]. The proposed work applies both prediction and error sensitivity to achieve superior performances.

3.1.5. Find Sensitivity Aware Similar Samples:

Construction of prediction intervals from the sensitivity-aware selection of samples brings narrower intervals with expected coverage. Fig. 4 presents the importance

Algorithm 1: Obtaining Indexes of Similar Samples

Input $\leftarrow$ Training dataset

Parameters and functions:

$N \leftarrow$ Number of samples
 $n \leftarrow$ Number of input parameters
 $Abs() \leftarrow$ Absolute values
 $Var() \leftarrow$ Variance
 $R_{Var} \leftarrow$ Variance ratio
 $X \leftarrow$ Inputs ($n \times N$ matrix)
 $T \leftarrow$ Targets ($1 \times N$ matrix)
 $i \leftarrow$ Index of the corresponding sample
 $j \leftarrow$ Index of the compared sample
 $R \leftarrow$ Range of inputs ($1 \times n$ matrix)
 $N_S \leftarrow$ Number of close samples to select
 $NN_p \leftarrow$ Point prediction NN
 $AE \leftarrow$ Absolute errors ($1 \times N$ matrix)
 $NN_e \leftarrow$ Absolute error prediction NN
 $Dev_{Index} \leftarrow [Deviation\ Index]$ ($2 \times N$ matrix)
 $Dev(i, j) \leftarrow$ Deviation between two inputs ($1 \times n$ matrix)
 $S_p \leftarrow$ Prediction sensitivity ($1 \times n$ matrix)
 $S_e \leftarrow$ Error sensitivity ($1 \times n$ matrix)
 $S_{en} \leftarrow$ Combined sensitivity ($1 \times n$ matrix)
 $N_c \leftarrow$ Number of similar samples considered
 $I_{SS} \leftarrow$ Indexes of selected similar samples ($N_c \times N$ matrix)
 $S_{Th} \leftarrow$ Similarity threshold ($1 \times n$ matrix)

Execution:

Train NN_p
 $AE \leftarrow Abs(NN_p(X) - T)$
 $R_{Var} \leftarrow \frac{Var(AE)}{Var(T)}$
Train NN_e
for $i \leftarrow 1$ **to** N **do**
 Compute prediction sensitivity S_p near sample i
 Compute error sensitivity S_e near sample i
 $S_{en} \leftarrow S_e \times R_{Var} + S_p \times (1 - R_{Var})$
 for $j \leftarrow 1$ **to** N **do**
 Compute $Dev(i, j)$
 $Dev_{Index}(j) \leftarrow [max(S_{en}.Dev(i, j)./R) \quad j]$
 end
 Sort ascending Dev_{Index} based on deviation column
 $I_{SS}(i) \leftarrow$ Indexes (j) from First N_c row from Dev_{Index} matrix
 $\triangleright$ Top N_c matched indexes
 $S_{Th}(i) \leftarrow$ Maximum $S_{en}.Dev(i, j)./R$ among selected indexes
end
Output $\leftarrow I_{SS}, S_{Th}$

of sensitivity consideration with a proper ratio between error and prediction sensitivities. Fig. 4(a) presents a prediction interval construction technique that considers only similarity. Similar samples are selected based on value differences among similar parameters. As a result, selection intervals for all inputs are the same in Fig. 4(a). Consideration of the similarity alone usually provides a wider interval. Some of the input parameters can be random to output. A random input parameter can make the interval wider as many very similar samples may have quite different values at the random parameter. Also, a slightly less similar sample may have a closer value near the parameter which is random to output. Therefore, some closely related samples are discarded and some less closely related samples are considered. That causes output distribution more spread, causing a wider interval.

The algorithm in Fig. 4(b) considers both similarity and the prediction sensitivity. The algorithm with only prediction sensitivity consideration is available in [46]. The first two inputs are responsible for the prediction and uncertainty respectively. The third input is random. The algorithm allows lower deviation in highly sensitive samples. Allowing very high deviation to an input parameter that is random to output keep similar samples under consideration. Therefore, interval in Fig. 4(b) is narrower than the interval in Fig. 4(a).

Fig. 4(c) presents similar samples and constructed prediction intervals while considering prediction and error-similarities equally. Fig. 4(d) presents similar samples and constructed prediction intervals while considering prediction and error-similarities based on their variation ratio, mentioned in Algorithm 1. Algorithm 1 is divided into two parts, besides inputs and outputs. The first part introduces the parameters and functions used in the algorithm. The second part presents the steps in execution. Approaches with sensitivity consideration, shown in Fig. 4(b)-(d). Consideration of sensitivity bring narrow intervals while Fig. 4(d) becomes slightly narrower than Fig. 4(b) and Fig. 4(c). The probable reason for superior performance is a better trade-off between prediction and error sensitivity.

The sensitivity of input varies based on the current value of parameters. For example, the effect of a slight time change on electricity demand depends on the current time and the current electricity demand value. The electricity demand changes rapidly during the evening. The electricity demand does not change rapidly during midnight in residential areas. However, the pattern may change in the future due to electric vehicle charging and other smart loads.

3.2. Sample Density NN Training:

While searching for similar samples according to Algorithm 1, we track the sensitivity-aware deviation of selected samples. The highest value deviation among selected samples is recorded and using that deviation, we obtain the sample density. The sample density provides a rough idea about the experience of NN near the sample. We train a NN for sample density prediction so that users do not require to search similar samples for finding sample densities. Users can compute a rough sample density near a sample by providing the sample input combination to the sample density neural network.

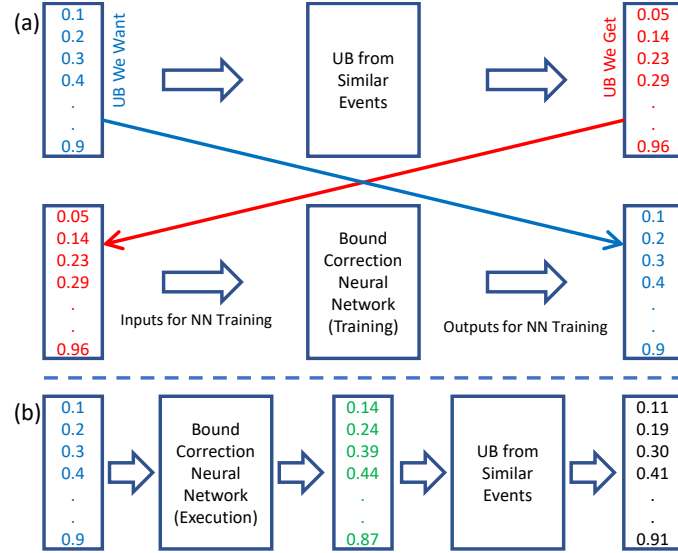


Figure 5: The bound correction NN training and execution. Although similar events have very close values of highly sensitive parameters, similar events are not the exactly same event. Therefore, prediction intervals from similar events are slightly wider than actual and covers more samples than expected. Subplot (a) visualize the process of bound correction NN training. UB we want is varied from 0.01 to 0.99 with 0.01 step. Subplot (b) shows how bound correction NN helps us with modified UB values.

3.3. Plotting Similar Events Near a Sample:

Although the algorithm extracts similar samples for all samples in the dataset in the first subsection of this section. We have provided a separate script for observing similar samples for a specific sample in the GitHub repository for better visualization. The name of the script is *A3- Plotting Similar Events.ipynb*. The user of the proposed prediction system can easily observe similar samples of a training sample with the help of that script.

3.4. Bound Correction NN Training

Intervals obtained from the distribution of similar samples are wider than required. As similar samples are not the same sample, outputs of similar samples are more spread. Fig. 5(a) presents the situation of distribution spreading and bound correction neural network training process. We select bound by sorting outputs of similar samples and by applying the initial UB value. We evaluate practical statistical bound-value for each UB value by investigating them on the entire training dataset. When we want to get an uncertainty lower bound of 0.10, we get an uncertainty lower bound of about 0.05. Lower bounds are lowered due to the extra spread. Upper bounds shifts upwards due to the same reason. If we want to get a lower bound of 0.05, we have to provide 0.10 to similarity count system. Therefore, we develop a bound correction NN which converts 0.05 to 0.10. Fig. 5(b) presents how we apply the bound correction NN to get optimal UB values.

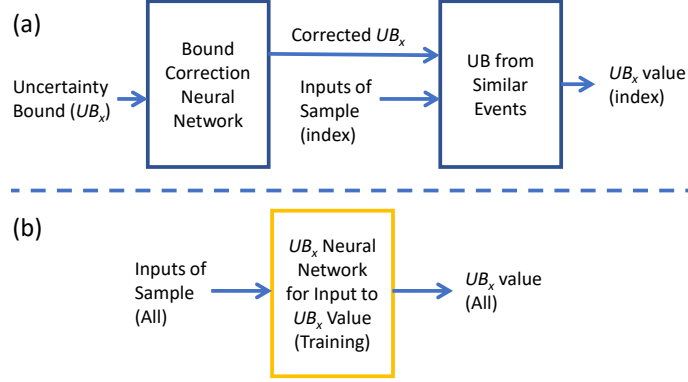


Figure 6: Reason for the direct NN for UB. (a) presents the proposed detailed method of computing uncertainty bound. The detailed method consists of bound correction, finding similar events from the entire test data, and finding the bound from distributions. The process is computationally extensive, and the system requires access to the entire training data for computing the UB. Therefore, we compute UB of certain cumulative probability (UB_{CP}) through the detailed method and use those input-output combinations for training a direct NN. (b) shows the structure of the direct NN of a certain cumulative probability UB_x .

3.5. Direct NN Training for UQ

Obtaining PI from similar samples requires the dataset. Computing predictions after searching the entire data is not feasible. Therefore, we train a bound prediction NN based on the bound-corrected uncertainty bounds. Fig. 6(a) presents the reason for training a NN for direct uncertainty quantification. The traditional process requires finding similar samples, bound correction NN, and obtaining the bound from the distribution of samples. That process is computation extensive and memory hungry. Therefore, we follow these steps during the training and obtain uncertainty bound for the entire training dataset. We train NNs for training input combination and obtained uncertainty bounds as example outputs. The direct NN with input-output combination is available in Fig. 6(b). During the execution, a single input pattern and an uncertainty bound are provided as the input of NN. The user of the NN can also change the value of uncertainty bound (x) and train another NN. Through this approach, the user can obtain multiple uncertainty bounds and intervals.

Although it is possible to train a NN containing both the varying UB and default input combinations, we investigate NNs with a single UB (UB_{CP}). We observe high epistemic uncertainty while developing a single NN for the prediction of different UB values. Therefore, we train different NN for different UB values and that significantly lowers the epistemic uncertainty.

Table 1 presents comparative merits and demerits of different uncertainty quantification methods. Point prediction with the root-mean-square-error (RMSE) or any variance information provides prediction intervals of constant width and does not reflect heteroscedasticity. Delta, Bayesian, mean-variance-estimation (MVE), and Bootstrap methods consider heteroscedasticity, however, fail to capture asymmetry due to the assumption of Gaussian distribution. Lower-upper-bound estimation (LUBE)

Table 1: Uncertainty estimation approaches and their characteristics.

Approach	Merits				
	Heteroscedasticity	Asymmetry	Observe Similar Samples	Considers Sensitivity	Assumption on Distribution
Point Prediction with RMSE	No	No	No	Yes	Yes
Delta Method	Yes	No	No	Yes	Yes
Bayesian Method	Yes	No	No	Yes	Yes
MVE Method	Yes	No	No	Yes	Yes
Bootstrap Method	Yes	No	No	Yes	Yes
LUBE Method	Yes	Yes	No	Yes	No
Similarity-based Methods	Yes	Yes	Yes	No	No
Proposed Method	Yes	Yes	Yes	Yes	No

method brings a smart prediction interval with the consideration of asymmetry and heteroscedasticity. The LUBE method does not follow any assumption on the distribution. However, the LUBE method is a cost function-based NN training method. Therefore, the LUBE method cannot track similar samples. Common similarity-based methods often perform poorly, as they do not consider sensitivity. The proposed method considers both similarity and sensitivity to construct optimal uncertainty bounds.

4. Datasets Investigated

We investigate our proposed algorithm on both synthetic and real-world datasets. At first, we apply the proposed method to Dataset-1 and Dataset-3 of the synthetic data repository [47]. Fig. 7 visualizes input-output relationships in investigated synthetic datasets. Dataset-1 has one input and homoscedastic uncertainty over the input range. Fig. 7(a) presents input-output relationship in Dataset-1. As there exists one input in Dataset-1, one figure can explain the input-output relationship. Fig. 7(b)-(d) present input-output relationships in Dataset-3. In this dataset, the first input determines prediction, the second input determines uncertainty, and the third input is random to output. A more detailed description of synthetic datasets is available in the paper [47].

Besides the initial investigation of the proposed method on synthetic datasets, we apply the method to Global Health Observatory (GHO) data (2000-2015) [48]. That dataset was collected from the World Health Organization (WHO) and United Nations (UN) websites and shared in the Kaggle platform [48]. That dataset contains a Comma-Separated Values (CSV) file. The CSV file contains a matrix. The matrix contains twenty-two columns. Two columns out of twenty-two columns contain texts: one contains country names and the other one contains the developing status of the country. The other twenty columns contain numeric values. A correlation matrix between numeric values is presented as Fig. 8. The dataset contains 2938 entries. However, some

entries lack one or more quantities. Discarding incomplete entries, there exist 1649 entries on the dataset. The correlation matrix provides a rough correlation between two parameters. Such as life expectancy has slightly increased over years during this period. Although most of the relationships are close to neutral, we can observe several high positive statistical correlations between parameters. The Gross Domestic Product (GDP) has a good statistical correlation with the percentage of expenditure. The number of infant deaths is also highly correlated to the number of deaths under-five years. The dataset provides the Income Composition of Resources (ICOR). ICOR is an index computed through education, life expectancy, per capita income, etc. ICOR has high positive statistical correlations with life expectancy, Body mass index (BMI), and schooling. This index is used to rank countries. Adult mortality has a statistically positive correlation to the number of AIDS patients. Several parameters are negatively correlated to each other. Life expectancy negative correlation with adult mortality, AIDS, and thinness at early ages. However, the correlation matrix is unable to grab periodic relationships. Fig. 7(a) shows a periodic relationship between input and output. NN can capture such periodic relationships where the correlation matrix may exhibit a near-zero correlation value.

We also investigate the performance of the proposed method on the electricity demand and wind power datasets. We download wind power generation and electricity demand samples from August 2012 to August 2022 from the UK-grid website ¹. For both the Wind Power and Electricity Demand datasets, we consider four recent samples, time of day, and the day number of the year as inputs. We write a separate script for the data preparation. We also upload the data preparation script to the GitHub repository.

Among our investigated datasets, only Dataset-1 has a constant standard deviation. Dataset-3 seems to have a constant standard deviation when they are observed in terms of the first input in Fig 7(b). However, Dataset-3 seems to have variable standard deviation or heteroscedastic uncertainty over the second input, shown in Fig 7(c). In Fig. 7(c), when the value of the second input stays within the range 1 to 3 inclusive, the upper range of output remains constant, and the lower range varies between -1 to about -1.5 based on the values of the second input. Besides synthetic datasets, we apply the algorithm to electricity demand, wind power, and life expectancy datasets. These three real-world datasets also have heteroscedastic uncertainties. In the electricity dataset, both the demand and the level of uncertainty are low during midnight, and they become high during the afternoon. Uncertainty in wind-power generation is also high when the wind speed is medium and varies over time. The wind-power generation becomes more certain when wind speed is either lower than the cut-in speed or higher than the rated speed. The life expectancy dataset also contains a complex relationship between input combination and uncertainty in targets. As those complex heteroscedastic relationships cannot be expressed in simple equations, we train NNs to predict them.

¹www.gridwatch.templar.co.uk/

5. Results

We have applied different approaches to datasets. We applied Bayesian NNs from two different sources. Also, we apply the direct interval construction method with three different cost functions. We also apply the proposed approach with and without error sensitivity consideration. We train NNs and investigate performances multiple times. As BNN models do not train a high performing prediction interval, we train BNNs for five times ($N_{Trial} = 5$) and obtain statistical results. For cost-function-based direct interval methods and similarity based data preparation methods we train NNs for 100 times and obtain statistical results.

5.1. Bayesian Neural Networks

We follow the scripts of Blundell et al. [49, 50] and Lee et al. [51] as examples of Bayesian NN to quantify uncertainties. Scripts of Blundell et al. [49, 50] is popularly known as BNN in Pyro. Scripts of Lee et al. [51] is popularly known as the TorchBNN module.

5.1.1. Bayesian Neural Networks in Pyro

The Pyro NN assumes a gaussian distribution of probability and provides regression mean and standard deviation. Using the mean value and standard deviation value, it is possible to draw intervals using the following relationship:

$$PI = [\mu - z\sigma, \mu + z\sigma] \quad (6)$$

where, μ is the median and σ is the standard deviation of the probability distribution. z is a parameter which determines the statistical coverage probability of interval. z becomes 1.15, 1.64, and 1.96 for 75%, 90%, and 95% coverage probabilities respectively. The investigated Pyro neural network contains two hidden layers. Each layer contains five hundred neurons. Pyro NNs are trained over twenty-five iterations. The customized Adam optimizer for pyro is applied for the optimization. The learning rate is set to 1e-3. According to our simulations, the Pyro UQ method overestimates the uncertainty. The intervals becomes much wider than required in the Pyro method. The first segment of Table 2 presents the performance of the Pyro BNNs in investigated datasets. The first three rows of the table present the performance of Pyro intervals on Dataset-1. We construct intervals of 95%, 90%, and 80% overage probabilities. Constructed PIs are wider than required and covers more sample. As a result PICP values are much larger than required.

The default Pyro model cannot find the complex relationship between inputs and output with a reasonable statistical error. Fig. 9(a) presents targets with Pyro mean and standard deviation curves. Neural network training methods often treat high-frequency changes as noise. Some training methods can capture 3-4 ups and downs over the range of one input but fails to be fitted for more than 5 ups and downs. However, human eyes can easily detect those patterns. Such as traditional NNs with proper training can capture trends of Dataset-7 [47], however, they fail to capture Dataset-8 of the synthetic dataset [47].

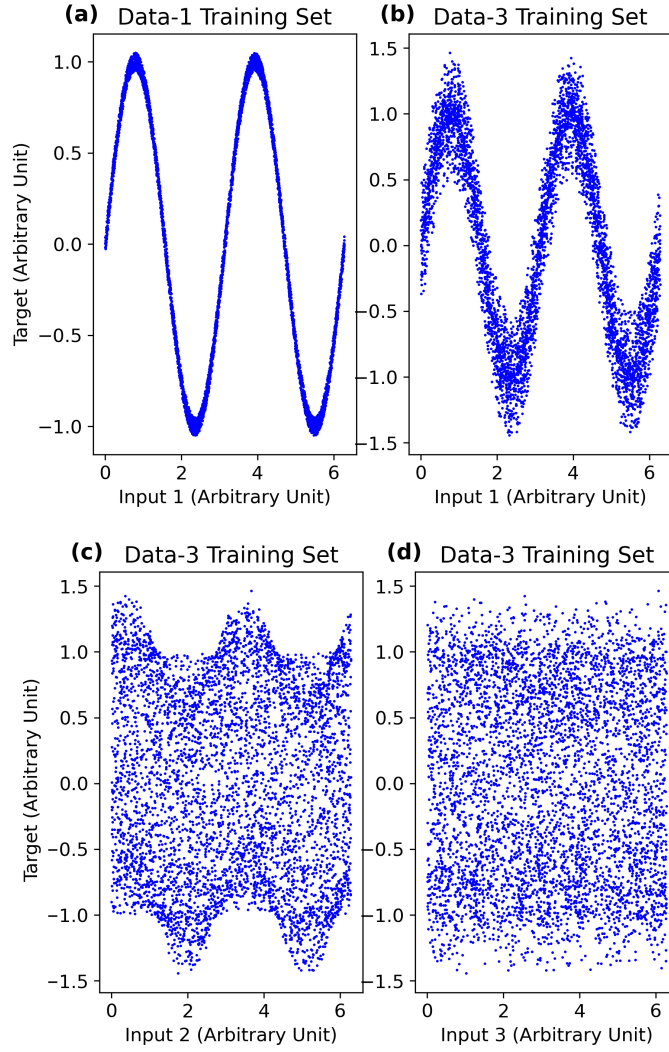


Figure 7: Relationship between target and inputs on Dataset-1, and Dataset-3[47]. The first input highly influences the point prediction. The second input highly influences the level of uncertainty. The target is not related to the third input.

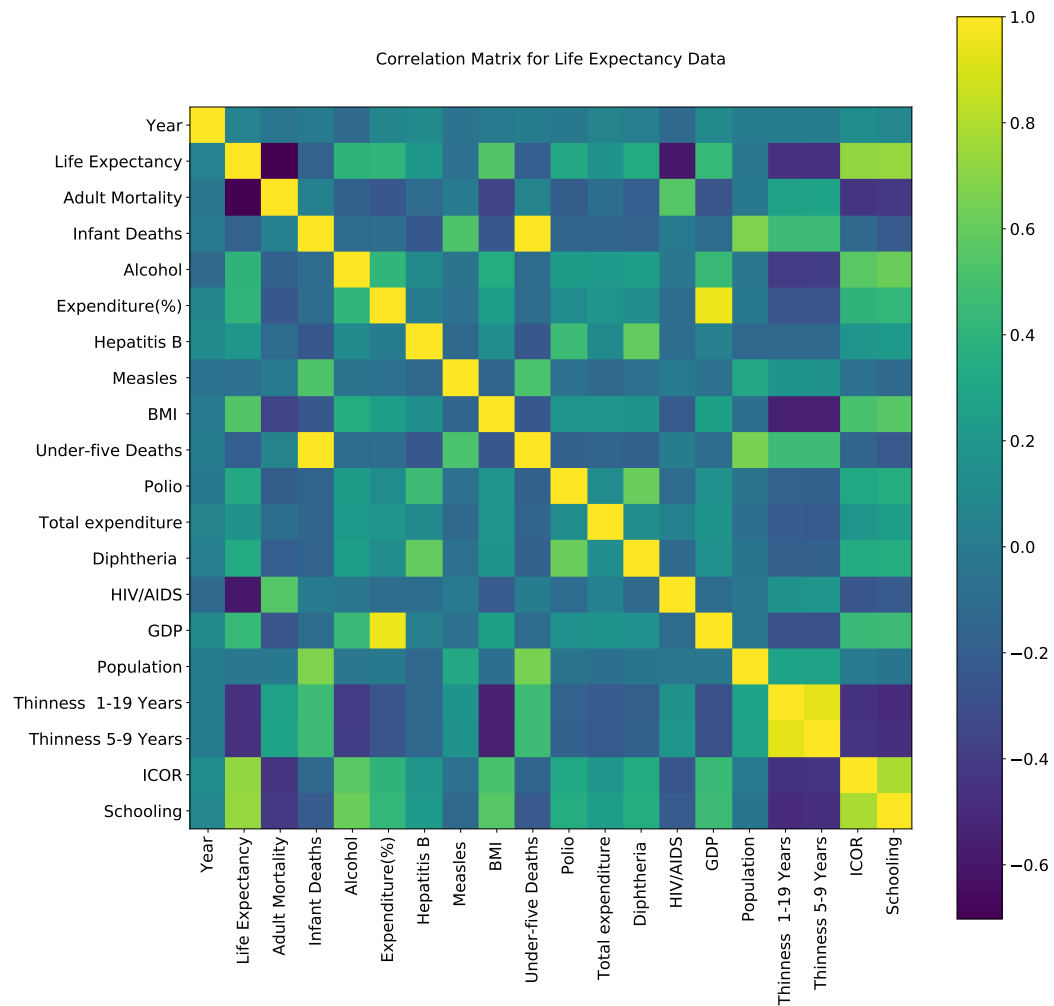


Figure 8: Correlation matrix on Global Health Observatory (GHO) data (2000-2015) [48]. The Dataset contains twenty numeric parameters and two non-numeric parameters. This figure shows a correlation matrix among numeric parameters. Non-numeric parameters are containing country names and development statuses.

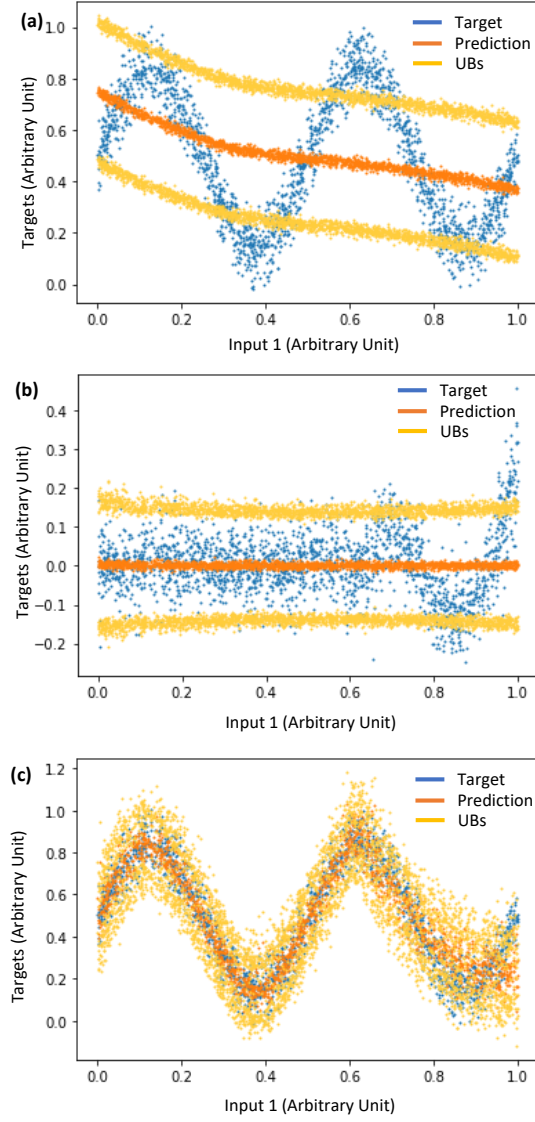


Figure 9: Point prediction and prediction intervals with Bayesian NN. (a) Pyro Bayesian NN. (b) Prediction and interval values while applying Pyro BNN to the error signal. (c) Prediction and interval values with targets after adjusting point prediction.

5.1.2. Pyro Model with Shallow Point Prediction NN

The Pyro model is unable to capture 3-4 ups and downs of the output for an input range. Therefore, the PINAW becomes several times larger than required and PICP becomes larger than required. Therefore, we apply a point prediction model to predict the regression mean of the dataset. Then we compute the error signal of the traditional error prediction model. Finally, we train a Pyro model to compute the uncertainty of the error signal. The prediction NN contains two hidden layers. Both layers contain one thousand neurons. Traditional NN training also applies the Adam optimizer with a learning rate of 0.05. Traditional NNs are trained over one thousand epochs. The training and network details of the pyro NN are the same as described in the previous subsection. Fig. 9(b) presents the error signal. It also presents the regression mean and standard deviation interval of the error signal from the Pyro model. Fig. 9(c) presents the targets with point prediction-adjusted Pyro outputs. Observing Fig. 9(a) and Fig. 9(c), it is obvious to us that Pyro with the help of a point prediction neural network provides superior performance compared to Pyro alone.

The second segment of Table 2 presents the performance of predicted intervals on different datasets. PIs become sharper and yet covers the required percentage of samples.

5.1.3. Bayesian Neural Networks in TorchBNN Module

The TorchBNN module is developed by Lee et al. [51]. The TorchBNN module does not predict an interval or regression mean with standard deviation. Instead, it predicts a single value. That predicted value changes from time to time. The bayesian NN in the TorchBNN module is developed with two hidden layers. Neurons are declared with a prior μ of zero and a prior σ of 0.1. Each layer contains five hundred neurons. We apply a cost containing both MSE loss and Keras variational autoencoder loss. The learning rate is set to 0.01. We train the TorchBNN BNNs over ten thousand iterations. Fig. 10(a)-(b) presents targets and predictions while applying the TorchBNN module on (a) Dataset-1 and (b) Dataset-3 [47]. At first, the TorchBNN neural networks are trained on these datasets. Then, NNs are applied to the test data subsets. Targets and predictions are shown in Fig. 10.

5.2. Distribution Free Direct Intervals

Khosravi et. al. first proposed the distribution-free direct interval method, known as the lower-upper bound estimation (LUBE) method [36]. Bayesian NNs have assumptions on the distribution of uncertainties. The direct interval construction method trains NNs directly with the help of a reward-based cost function.

Direct interval training in this paper consists of initial NN training with rough targets, followed by a final training. The direct NN consists of two hidden layers, each layer containing one thousand neurons. The initial training consists of one thousand epochs while the final training consists of five thousand epochs. Both training uses the Adam optimizer. The initial training has a learning rate of $5e-2$ and the final training has a learning rate of $5e-5$. The equation for generating a rough target is as follows:

$$t_{j,rough} = t_j \pm \Delta t \quad (7)$$

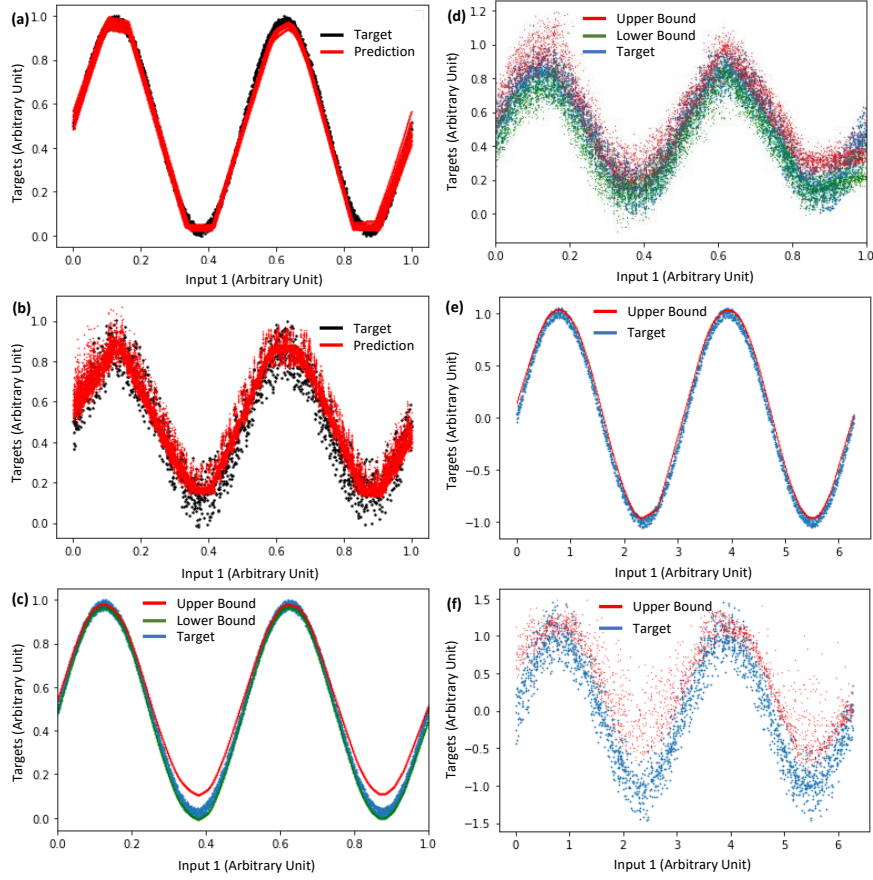


Figure 10: (a)-(b) Targets and predictions while applying the TorchBNN module on (a) Dataset-1 and (b) Dataset. The TorchBNN module provides a point prediction value instead of providing an interval prediction. However, as the parameters of the model are not like parameters of traditional neural networks, the prediction is slightly different from time to time, reflecting the uncertainty. (c)-(d) Targets with uncertainty bounds of 95% coverage probability (c) on Dataset-1 (d) on Dataset-3 while applying the direct interval construction method with the optimal cost function. (e)-(f) Targets with uncertainty upper bounds of 90% coverage probability while applying the proposed similarity, sensitivity, and error sensitivity-based method. The method constructs an uncertainty bound of a certain cumulative probability value. These figures show denormalized values. Therefore, the ranges in both axis are different from the normalized range (0 to 1).

$t_{j,rough}$ contains two rough targets one is $t_j + \Delta t$ and another one is $t_j - \Delta t$. We kept $\Delta t = R/4$ based on observations. Future user of our script can adjust value of margin (Δt) based on the level of uncertainty on their dataset.

5.2.1. LUBE Method

The LUBE method train NNs following a cost function, known as Coverage Width Criteria (CWC). The CWC is expressed as Eq. (4).

The first segment of Table 3 presents the performance of the LUBE method on investigated Datasets. The LUBE cost function is designed to prevent lower coverage with a high penalty $\gamma(PICP)$. Therefore, most of the LUBE NNs construct an interval that is wide enough to cover the required number of samples. As $\gamma(PICP)$ becomes zero for $PICP > 1 - \alpha$, the CWC becomes equal to PINAW for $PICP > 1 - \alpha$. The term $1 - \alpha$ is also known as the Prediction Interval Nominal Coverage (PINC).

The prediction interval normalized failure distance (PINAfD) is a parameter described in the optimal cost function subsection. Although PINAW generally becomes smaller for lower PINC, PINAFD may increase or decrease based on PINC. Most failure distances increase while intervals become narrower. However, more samples become uncovered for narrower PIs. As a result, the average failure distance often decreases.

5.2.2. Deviation from the Mid Interval Method

L G Marin et al. proposed deviation from the mid-interval method [39]. They consider a smooth PI failure penalty. Besides PINAW and PI failure penalties, they consider the nearness of mid-intervals from targets. Their cost function is as follows:

$$\min_{NN} \beta_1 PINAW + \beta_2 \|e\|^2 + \exp[-\eta(PICP - (1 - \alpha))] \quad (8)$$

where, β_1 , and β_2 are hyper parameters, $\|e\|$ is defined by the following equation:

$$\|e\| = \sqrt{\sum_{j=1}^n \left| t_j - \frac{\bar{y}_j + y_j}{2} \right|^2}. \quad (9)$$

The second segment of Table 3 presents performances of prediction intervals developed by deviation from the mid-interval method. As the deviation from the mid-interval method has a slightly lower penalty for less coverage and more reward for narrowing PIs, PIs are slightly narrower. Also, PICP frequently becomes lower than PINC. Although the Table 3 presents average values of 100 iterations. PICP is lower than PINC on average in most of the combinations.

5.2.3. Optimal Cost Function Method

The optimal cost function is developed by combining the philosophies of several recently proposed cost functions [37]. The cost function is as follows:

$$\min_{NN} PINAW + \rho \cdot PINAFD + \beta \cdot (1 - \alpha + \delta - PICP)^2 \quad (10)$$

Table 2: PI Optimization Performance for Bayesian Neural Network-based methods.

Bayesian Method: Pyro [49, 50]								
Data	$1 - \alpha$	N_{Trial}	μ_{PINAW}	μ_{PICP}	σ_{PICP}	μ_{PINAFD}	μ_{CWC}	μ_{CWFDC}
Data-1	0.95	5	1.72	100%	0%	0	1.72	2.848
	0.90	5	1.41	100%	0%	0	1.41	5.923
	0.80	5	1.11	100%	0%	0	1.11	19.16
Data-3	0.95	5	1.86	100%	0%	0	1.86	2.988
	0.90	5	1.55	100%	0%	0	1.55	6.063
	0.80	5	1.31	100%	0%	0	1.31	19.36
Life Exp.	0.95	5	2.02	100%	0%	0	2.02	3.148
	0.90	5	1.69	100%	0%	0	1.69	6.203
	0.80	5	1.42	100%	0%	0	1.42	19.47
Elec. Demand	0.95	5	1.21	100%	0%	0	1.21	2.338
	0.90	5	1.01	100%	0%	0	1.01	5.523
	0.80	5	0.85	100%	0%	0	0.85	18.9
Wind Power	0.95	5	1.38	100%	0%	0	1.38	2.508
	0.90	5	1.15	100%	0%	0	1.15	5.663
	0.80	5	0.97	100%	0%	0	0.97	19.02
Bayesian Method: Pyro with Point Prediction NN								
Data	$1 - \alpha$	N_{Trial}	μ_{PINAW}	μ_{PICP}	σ_{PICP}	μ_{PINAFD}	μ_{CWC}	μ_{CWFDC}
Data-1	0.95	5	0.15	100%	0%	0	0.15	1.278
	0.90	5	0.12	100%	0%	0	0.12	4.633
	0.80	5	0.11	96.2%	3.2%	0.01	0.11	11.672
Data-3	0.95	5	0.32	100%	0%	0	0.32	1.448
	0.90	5	0.27	98.1%	1.6%	0.012	0.27	3.17
	0.80	5	0.23	93.1%	4.3%	0.061	0.23	7.612
Life Exp.	0.95	5	0.71	100%	0%	0	0.71	1.838
	0.90	5	0.59	95.7%	2.1%	0.039	0.59	1.981
	0.80	5	0.50	90.9%	3.8%	0.058	0.50	5.459
Elec. Demand	0.95	5	0.24	100%	0%	0	0.24	1.368
	0.90	5	0.20	100%	0%	0	0.20	4.713
	0.80	5	0.17	93.5%	4.9%	0.033	0.17	8.016
Wind Power	0.95	5	0.62	100%	0%	0	0.62	1.748
	0.90	5	0.52	97.1%	1.1%	0.011	0.52	2.709
	0.80	5	0.44	95.0%	1.8%	0.035	0.44	10.275
Bayesian Method: TorchBNN Module [51]								
Data	$1 - \alpha$	N_{Trial}	μ_{PINAW}	μ_{PICP}	σ_{PICP}	μ_{PINAFD}	μ_{CWC}	μ_{CWFDC}
Data-1	0.95	5	0.05	81.3%	4.2%	0.006	47.244	9.786
	0.90	5	0.04	76.7%	4.5%	0.012	30.951	9.574
	0.80	5	0.03	62.5%	4.9%	0.011	189.351	17.154
Data-3	0.95	5	0.14	69.1%	6.1%	0.010	5.891e4	34.341
	0.90	5	0.11	50.3%	6.8%	0.015	4.593e7	80.927
	0.80	5	0.10	41.9%	5.9%	0.019	1.876e3	76.560
Life Exp.	0.95	5	0.43	67.3%	6.2%	0.052	4.45e5	39.542
	0.90	5	0.39	56.1%	6.9%	0.050	8.961e6	59.608
	0.80	5	0.38	51.9%	7.2%	0.055	4.804e5	42.776
Elec. Demand	0.95	5	0.15	91.0%	1.1%	0.011	1.258	1.064
	0.90	5	0.14	82.6%	3.1%	0.012	5.803	3.273
	0.80	5	0.13	71.8%	3.7%	0.009	7.974	4.371
Wind Power	0.95	5	0.21	84.3%	3.0%	0.018	44.437	6.223
	0.90	5	0.19	69.1%	3.9%	0.020	6.563e3	23.108
	0.80	5	0.18	62.7%	4.3%	0.022	1.028e3	16.947

where, ρ is the failure distance resistance parameter. ρ becomes equal to one when providing an equal concentration to PINAW and PINAFD. β is the PI failure penalty factor, α is the target non-coverage probability, and δ is the PICP margin. Usually, $\delta = \alpha/20$ that brings PICP > PINC statistically. The prediction interval normalized average failure distance (PINAFD) is expressed as follows:

$$PINAFD = \frac{\sum_{j=1}^n (1 - c_j) \times \min(|t_j - \bar{y}_j|, |y_j - t_j|)}{R \times \left\{ \sum_{j=1}^n (1 - c_j) \right\} + \epsilon}. \quad (11)$$

where,

$$c_j = \begin{cases} 1, & t_j \in [\underline{y}_j, \bar{y}_j] \\ 0, & t_j \notin [\underline{y}_j, \bar{y}_j], \end{cases}$$

where, t_j is the target for j^{th} sample. R is the range of targets, $\epsilon \approx (1e - 10)$ is a very small number added to avoid the divided by zero error, $\bar{y}_j$ and $\underline{y}_j$ are the predicted upper bound and lower bound respectively for j^{th} sample. This cost function is also known as the coverage width failure distance criteria (CWFDC). A publicly available script is also available with execution details of optimal cost-function-based training on the GitHub repository of this paper. The script contains an extra bound swap penalty parameter to ensure that there is no bound swap. The script also performs an initial training with rough targets before the cost function based training. Rough targets are generated by adding a margin with the original target. Initial training with rough targets increases convergence of NN training and reduces training time significantly [46].

Fig. 10 (c)-(d) present targets with uncertainty bounds of 95% coverage probability while applying the direct interval construction method with the optimal cost function [37]. Fig. 10(c) and Fig. 10(d) presents targets with intervals for Dataset-1 and Dataset-3 respectively. The third segment of Table 3 presents the performance of the optimal cost function on investigated datasets. The optimal cost function provides superior performance on average compared to Bayesian and other cost-function-based training methods.

5.3. Proposed Method

The direct interval construction method with optimal cost function provides a sharp interval with nearly accurate coverage. However, there is still debate on the selection of appropriate cost functions. Some researchers think that only narrowing PIs should be the goal. Some other researchers think about getting a small failure distance. Some other researchers are trying to bring the mid-interval near the target. Another approach is training NNs for uncertainty-bound values. However, cost-function-based training NNs for uncertainty-bound values brings poor NNs. Neural networks with such training often predict a constant which brings statistical average UB instead of a heteroscedastic uncertainty bound. Therefore, we propose a similarity-based method to find UB and train NNs with obtained bounds.

The initial point prediction NN training consists of six hundred epochs. The error prediction NN also requires the same number of epochs. Both NNs contain two hidden layers and each layer contains five hundred neurons. The learning rate is 0.05 for both

Table 3: PI Optimization Performance for Cost function-based Direct Interval Construction methods.

LUBE Method [36]								
Data	$1 - \alpha$	N_{Trial}	μ_{PINAW}	μ_{PICP}	σ_{PICP}	μ_{PINAfD}	μ_{CWC}	μ_{CWFDC}
Data-1	0.95	100	0.072	96.2%	2.2%	0.01	0.072	0.127
	0.90	100	0.070	90.9%	3.1%	0.02	0.070	0.098
	0.80	100	0.069	82.1%	5.3%	0.01	0.069	0.140
Data-3	0.95	100	0.177	95.5%	1.6%	0.02	0.177	0.200
	0.90	100	0.178	90.7%	2.3%	0.03	0.178	0.210
	0.80	100	0.177	81.6%	4.2%	0.01	0.177	0.205
Life Exp.	0.95	100	0.475	95.9%	2.1%	0.09	0.475	0.586
	0.90	100	0.462	89.9%	3.3%	0.10	0.948	0.580
	0.80	100	0.431	83.3%	5.2%	0.08	0.431	0.776
Elec. De-mand	0.95	100	0.171	95.3%	1.0%	0.02	0.171	0.191
	0.90	100	0.163	90.7%	1.9%	0.03	0.163	0.195
	0.80	100	0.160	81.2%	2.1%	0.02	0.160	0.182
Wind Power	0.95	100	0.331	96.0%	1.6%	0.06	0.331	0.419
	0.90	100	0.312	90.6%	2.9%	0.05	0.312	0.363
	0.80	100	0.300	80.0%	6.5%	0.05	0.300	0.400
Deviation from Mid Interval method [39]								
Data	$1 - \alpha$	N_{Trial}	μ_{PINAW}	μ_{PICP}	σ_{PICP}	μ_{PINAfD}	μ_{CWC}	μ_{CWFDC}
Data-1	0.95	100	0.071	95.0%	1.9%	0.015	0.071	0.089
	0.90	100	0.069	89.8%	2.6%	0.014	0.145	0.108
	0.80	100	0.068	79.5%	4.3%	0.011	0.155	0.192
Data-3	0.95	100	0.182	94.8%	1.3%	0.031	0.383	0.223
	0.90	100	0.180	91.6%	2.1%	0.033	0.180	0.274
	0.80	100	0.176	80.3%	3.3%	0.027	0.176	0.228
Life Exp.	0.95	100	0.470	94.2%	1.8%	0.085	1.171	0.610
	0.90	100	0.463	89.7%	3.3%	0.113	1.001	0.608
	0.80	100	0.429	78.1%	5.0%	0.101	1.538	0.951
Elec. De-mand	0.95	100	0.175	95.1%	0.9%	0.029	0.175	0.205
	0.90	100	0.168	90.3%	1.6%	0.025	0.168	0.195
	0.80	100	0.167	80.7%	1.9%	0.021	0.167	0.193
Wind Power	0.95	100	0.331	94.8%	1.5%	0.046	0.697	0.387
	0.90	100	0.312	89.6%	2.7%	0.039	0.693	0.392
	0.80	100	0.300	78.2%	4.2%	0.038	1.038	0.730
Optimal Cost Function [37]								
Data	$1 - \alpha$	N_{Trial}	μ_{PINAW}	μ_{PICP}	σ_{PICP}	μ_{PINAfD}	μ_{CWC}	μ_{CWFDC}
Data-1	0.95	100	0.069	95.1%	1.1%	0.005	0.069	0.075
	0.90	100	0.068	90.5%	1.8%	0.008	0.068	0.076
	0.80	100	0.067	80.9%	2.7%	0.006	0.067	0.074
Data-3	0.95	100	0.179	95.0%	0.8%	0.020	0.179	0.202
	0.90	100	0.175	90.3%	1.4%	0.013	0.175	0.190
	0.80	100	0.172	80.2%	2.6%	0.017	0.172	0.221
Life Exp.	0.95	100	0.466	93.8%	1.4%	0.078	1.315	0.649
	0.90	100	0.461	90.0%	2.6%	0.081	0.461	0.555
	0.80	100	0.440	79.9%	3.8%	0.084	0.903	0.585
Elec. De-mand	0.95	100	0.173	95.1%	0.8%	0.016	0.173	0.190
	0.90	100	0.167	90.0%	1.5%	0.018	0.167	0.198
	0.80	100	0.162	80.2%	1.8%	0.017	0.162	0.211
Wind Power	0.95	100	0.327	96.1%	1.1%	0.024	0.327	0.387
	0.90	100	0.310	90.9%	2.4%	0.019	0.310	0.337
	0.80	100	0.307	81.6%	2.9%	0.020	0.307	0.345

Table 4: PI Optimization Performance for Similarity-based Approaches.

UB from Similarity and Sensitivity [22]								
Data	$1 - \alpha$	N_{Trial}	μ_{PINAW}	μ_{PICP}	σ_{PICP}	μ_{PINAFD}	μ_{CWC}	μ_{CWFDC}
Data-1	0.95	100	0.063	95.0%	1.9%	0.004	0.063	0.070
	0.90	100	0.062	90.1%	2.1%	0.005	0.062	0.075
	0.80	100	0.060	79.8%	2.6%	0.004	0.126	0.136
Data-3	0.95	100	0.191	95.0%	0.5%	0.019	0.191	0.213
	0.90	100	0.185	90.2%	0.8%	0.016	0.185	0.206
	0.80	100	0.180	80.3%	1.8%	0.012	0.180	0.217
Life Exp.	0.95	100	0.461	94.8%	1.7%	0.081	0.970	0.552
	0.90	100	0.457	90.9%	2.3%	0.077	0.457	0.542
	0.80	100	0.451	80.3%	4.2%	0.074	0.451	0.550
Elec. Demand	0.95	100	0.171	95.0%	0.7%	0.019	0.171	0.193
	0.90	100	0.167	90.1%	1.6%	0.017	0.167	0.192
	0.80	100	0.164	80.2%	2.2%	0.016	0.164	0.212
Wind Power	0.95	100	0.327	95.2%	0.9%	0.024	0.327	0.351
	0.90	100	0.311	90.3%	1.3%	0.019	0.311	0.332
	0.80	100	0.308	80.1%	2.0%	0.017	0.308	0.366
UB from Similarity, Sensitivity, and Error Sensitivity								
Data	$1 - \alpha$	N_{Trial}	μ_{PINAW}	μ_{PICP}	σ_{PICP}	μ_{PINAFD}	μ_{CWC}	μ_{CWFDC}
Data-1	0.95	100	0.062	95.0%	1.8%	0.004	0.062	0.069
	0.90	100	0.061	90.1%	2.0%	0.004	0.061	0.073
	0.80	100	0.059	80.2%	2.2%	0.003	0.059	0.094
Data-3	0.95	100	0.190	95.0%	0.6%	0.018	0.190	0.211
	0.90	100	0.185	90.0%	0.7%	0.015	0.185	0.213
	0.80	100	0.178	80.2%	1.3%	0.013	0.178	0.223
Life Exp.	0.95	100	0.455	95.1%	1.7%	0.083	0.455	0.539
	0.90	100	0.453	90.1%	2.2%	0.079	0.453	0.540
	0.80	100	0.450	80.2%	2.9%	0.077	0.450	0.559
Elec. Demand	0.95	100	0.169	95.0%	0.7%	0.019	0.169	0.191
	0.90	100	0.165	89.9%	1.5%	0.018	0.338	0.201
	0.80	100	0.162	80.0%	2.0%	0.016	0.162	0.228
Wind Power	0.95	100	0.320	95.1%	1.0%	0.023	0.320	0.344
	0.90	100	0.297	89.8%	1.2%	0.020	0.625	0.342
	0.80	100	0.289	80.0%	2.0%	0.017	0.289	0.356

situations. The sample density NN contains two hidden layers and each layer contains five hundred neurons. The learning rate is also 0.05 for the sample density NN training. The bound correction NN also has the same model and training attributes. The final NN size and training epoch varies based on the complexity of the data. Training scripts are available at the GitHub repository.

Fig. 10 presents targets with uncertainty upper bounds of intervals of 90% coverage probability while applying the proposed similarity, sensitivity, and error sensitivity-based method. The method constructs an uncertainty bound of a certain cumulative probability value. Fig. 10(a) presents targets of Dataset-1 with an uncertainty upper bound of 95% cumulative probability. Fig. 10(b) presents targets of Dataset-3 with an uncertainty upper bound of 95% cumulative probability. This figure shows denormalized values. Therefore, the ranges in both axis are different from the normalized range (0 to 1). Table 4 presents both the proposed method and our recently proposed similarity and sensitivity-based method [22]. Our recently proposed paper [22] considers similarity and prediction sensitivity. The method proposed in this paper considers similarity, prediction sensitivity, and error sensitivity. Earlier in this paper, Fig. 2 explained the effect of discarding one parameter which only contributes to uncertainty.

Among cost-function-based methods, the optimal cost function method provides superior performance in Dataset-1 and wind power data. LUBE and optimal cost function methods perform equally for Dataset-3 and life expectancy data. The LUBE method provides superior performance on the electricity demand dataset. Among similarity-based methods, both methods perform equally on the electricity demand dataset. The proposed method provides superior performance on the other four datasets.

The proposed method statistically outperforms other investigated methods. All five datasets are investigated for three different nominal coverage probabilities. We compute CWC and CWFDC values for each combination. We present optimal values among the same type of training with bold-faced numbers. Among BNN methods Pyro BNN with point prediction support provides optimal results in all situations. Therefore, all values of cost functions are bold-faced in this segment. We present the optimum value of all methods with both bold and italic numbers.

The proposed method provides overall superior performance on Dataset-1, Life expectancy data, and Windpower generation data. The optimal cost function method provides overall superior performance on Dataset-3. The LUBE method provides overall superior performance on the Electricity demand dataset. Therefore, the proposed method is providing superior performance statistically and bringing transparency.

5.4. Discussion

This paper is the improved version of one of our previous papers [22]. In the previous paper, we proposed a neural network-based uncertainty quantification from similarity and sensitivity. In this paper, we introduce error sensitivity. We visualize the need for considering error sensitivity in Fig. 3. Prediction sensitivity mostly considers parameters that are related to point prediction. Another input parameter can potentially increase or decrease the error by keeping the same regression mean. We visualize the effect of considering error sensitivity in Fig. 4. The prediction interval becomes wider when we consider only similarities, as shown in Fig. 4(a). With the consideration of prediction sensitivity, the intervals become narrower. While considering the error sensitivity, we need to prioritize sensitivities based on their relative magnitude. Otherwise, intervals can potentially become wider. Fig. 4(d) is an example of a proper tradeoff between prediction sensitivity and error sensitivity. Table 4 presents a comparative performance between the current method and [22]. We have investigated several datasets. The proposed method provides superior or equal performance for all the datasets.

Acknowledgment

This research was partially supported by the Australian Research Council’s Discovery Projects funding scheme (project DP190102181).

6. Conclusion

In this paper, we have proposed a similarity-based uncertainty quantification neural network training method. According to our observation, cost-function-based methods and similarity-based methods outperforms Bayesian methods. Bayesian methods are

formulated based on strong assumptions and often fail to provide optimal coverage. Cost-function-based methods provide superior results in several investigated datasets. Similarity-based methods are better in other datasets. The proposed method has provided statistically superior performances in three out of five datasets. Besides the statistical numeric improvement on the quality of prediction interval, the proposed process has the provision to plot similar samples and to observe sample density near a sample. The plot of the distribution of similar samples can potentially convey a more detailed idea of the distribution of uncertainties. We have proposed the method with publicly available codes and with execution details as Jupyter Notebook files. Several scripts of Bayesian NN and direct interval construction are also provided with execution details to help future researchers. We have also explained several phenomena and reasons for performing specific actions with relevant sketches and diagrams. Future researchers can potentially apply the proposed method to new datasets. Future researchers can also propose a more robust method and compare results with the help of shared scripts.

Although trained models in the proposed method statistically provide superior performance, the training is computationally extensive. We reduce the execution time by training NN for the prediction. Future researchers may apply a less computation extensive data selection technique to find sensitivity-aware similar samples. All the investigated methods, including the proposed method exhibit poor performance when the number of training samples is small. With a small number of training samples, prediction intervals become wide and fail to cover the target more frequently.

References

1. Theisen, R., Wang, H., Varshney, L.R., Xiong, C., Socher, R.. Evaluating state-of-the-art classification models against bayes optimality. *Advances in Neural Information Processing Systems* 2021;34:9367–9377.
2. Fan, F.L., Wang, D., Guo, H., Zhu, Q., Yan, P., Wang, G., Yu, H.. On a sparse shortcut topology of artificial neural networks. *IEEE Transactions on Artificial Intelligence* 2021;3(4):595–608.
3. Koochali, A., Dengel, A., Ahmed, S.. If you like it, gan it—probabilistic multivariate times series forecast with gan. *Engineering Proceedings* 2021;5(1):40.
4. Lakshminarayanan, B., Pritzel, A., Blundell, C.. Simple and scalable predictive uncertainty estimation using deep ensembles. In: *Advances in neural information processing systems*. 2017:6402–6413.
5. Kabir, H.. Reduction of class activation uncertainty with background information. *arXiv preprint arXiv:230503238* 2023;.
6. Morala, P., Cifuentes, J.A., Lillo, R.E., Ucar, I.. Towards a mathematical framework to inform neural network modelling via polynomial regression. *Neural Networks* 2021;142:57–72.
7. Mir, M., Kabir, H.D., Nasirzadeh, F., Khosravi, A.. Neural network-based interval forecasting of construction material prices. *Journal of Building Engineering* 2021;39:102288.

8. Kavousi-Fard, A., Dabbaghjamanesh, M., Jin, T., Su, W., Roustaei, M.. An evolutionary deep learning-based anomaly detection model for securing vehicles. *IEEE Transactions on Intelligent Transportation Systems* 2020;22(7):4478–4486.
9. Dong, H., Gong, Q., Zhu, M.. Optimal energy management of automated grids considering the social and technical objectives with electric vehicles. *International Journal of Electrical Power & Energy Systems* 2021;130:106910.
10. Zare, A., Shoeibi, A., Bajestani, N.S., Moridian, P., Alizadehsani, R., Hallaji, M., Khosravi, A.. Accurate prediction using triangular type-2 fuzzy linear regression: Simplifying complex t2f calculations. *IEEE Systems, Man, and Cybernetics Magazine* 2022;8(3):51–60.
11. Mobtahej, M., Esapour, K., Tajalli, S.Z., Mohammadi, M.. Effective demand response and gans for optimal constraint unit commitment in solar-tidal based microgrids. *IET Renewable Power Generation* 2021;.
12. Ikidid, A., El Fazziki, A., Sadgal, M.. Multi-agent and fuzzy inference-based framework for traffic light optimization. *International Journal of Interactive Multimedia and Artificial Intelligence* 2021;.
13. Gomez Vargas, A.E., Obregón, N., Alvarez Pomar, C.L.. Uncertainty model for quantitative precipitation estimation using weather radars. *International Journal of Interactive Multimedia and Artificial Intelligence* 2016;.
14. Harish, B., Roopa, C.. Automated ecg analysis for localizing thrombus in culprit artery using rule based information fuzzy network. *International Journal of Interactive Multimedia and Artificial Intelligence* 2020;.
15. Elder, B., Arnold, M., Murthi, A., Navrátil, J.. Learning prediction intervals for model performance. In: *Proceedings of the AAAI Conference on Artificial Intelligence*; vol. 35. 2021:7305–7313.
16. Goel, A., Tung, C., Eliopoulos, N., Hu, X., Thiruvathukal, G.K., Davis, J.C., Lu, Y.H.. Directed acyclic graph-based neural networks for tunable low-power computer vision. In: *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*. 2022:1–6.
17. Zilly, J., Achille, A., Censi, A., Frazzoli, E.. On plasticity, invariance, and mutually frozen weights in sequential task learning. *Advances in Neural Information Processing Systems* 2021;34:12386–12399.
18. Malinin, A., Gales, M.. Predictive uncertainty estimation via prior networks. In: *Advances in Neural Information Processing Systems*. 2018:7047–7058.
19. Ma, P., Wu, Y., Li, Y., Guo, L., Jiang, H., Zhu, X., Wu, X.. Hw-forest: Deep forest with hashing screening and window screening. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 2022;.

20. ITO, M., HE, F., OKI, E.. Backup resource allocation of virtual machines for probabilistic protection under capacity uncertainty. *IEICE Transactions on Communications* 2022;:2021EBP3144.
21. Siljak, H., Macaluso, I., Marchetti, N.. Artificial intelligence for dynamical systems in wireless communications: Modeling for the future. *IEEE Systems, Man, and Cybernetics Magazine* 2021;7(4):13–23.
22. Kabir, H.D., Khosravi, A., Nahavandi, D., Nahavandi, S.. Uncertainty quantification neural network from similarity and sensitivity. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE; 2020:1–8.
23. Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U.R., et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion* 2021;76:243–297.
24. Gal, Y., Ghahramani, Z.. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: *international conference on machine learning*. PMLR; 2016:1050–1059.
25. Kabir, H.D., Khosravi, A., Hosen, M.A., Nahavandi, S.. Neural network-based uncertainty quantification: A survey of methodologies and applications. *IEEE Access* 2018;.
26. Ajlan, I.K., Rahim, H.B.A., Ismael, A.J., et al. Text recognition from images. *Texas Journal of Engineering and Technology* 2022;10:10–18.
27. Kamal, M.S., Northcote, A., Chowdhury, L., Dey, N., Crespo, R.G., Herrera-Viedma, E.. Alzheimer’s patient analysis using image and gene expression data and explainable-ai to present associated genes. *IEEE Transactions on Instrumentation and Measurement* 2021;70:1–7.
28. Bayes, T.. Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London* 1763;1(53):370–418.
29. Campbell, T., Beronov, B.. Sparse variational inference: Bayesian coresets from scratch. *Advances in Neural Information Processing Systems* 2019;32.
30. Zhou, X., Liu, H., Pourpanah, F., Zeng, T., Wang, X.. A survey on epistemic (model) uncertainty in supervised learning: Recent advances and applications. *Neurocomputing* 2021;.
31. Olivier, A., Shields, M.D., Graham-Brady, L.. Bayesian neural networks for uncertainty quantification in data-driven materials modeling. *Computer Methods in Applied Mechanics and Engineering* 2021;386:114079.

32. Serpell, C., Araya, I., Valle, C., Allende, H.. Probabilistic forecasting using monte carlo dropout neural networks. In: *Iberoamerican Congress on Pattern Recognition*. Springer; 2019:387–397.
33. Posch, K., Pilz, J.. Correlated parameters to accurately measure uncertainty in deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 2020;32(3):1037–1051.
34. Jospin, L.V., Laga, H., Boussaid, F., Buntine, W., Bennamoun, M.. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine* 2022;17(2):29–48.
35. Upadhyay, U., Chen, Y., Akata, Z.. Robustness via uncertainty-aware cycle consistency. *Advances in Neural Information Processing Systems* 2021;34:28261–28273.
36. Khosravi, A., Nahavandi, S., Creighton, D., Atiya, A.F.. Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE transactions on neural networks* 2010;22(3):337–346.
37. Kabir, H.D., Khosravi, A., Kavousi-Fard, A., Nahavandi, S., Srinivasan, D.. Optimal uncertainty-guided neural network training. *Applied Soft Computing* 2021;99:106878.
38. Khosravi, A., Nahavandi, S., Creighton, D., Atiya, A.F.. Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Transactions on neural networks* 2011;22(9):1341–1356.
39. Marín, L.G., Valencia, F., Sáez, D.. Prediction interval based on type-2 fuzzy systems for wind power generation and loads in microgrid control design. In: *Fuzzy Systems (FUZZ-IEEE), 2016 IEEE International Conference on*. IEEE; 2016:328–335.
40. Zhang, G., Wu, Y., Wong, K.P., Xu, Z., Dong, Z.Y., Iu, H.H.C.. An advanced approach for construction of optimal wind power prediction intervals. *IEEE Transactions on Power Systems* 2015;30(5):2706–2715.
41. Burnstein, E., Stotland, E., Zander, A.. Similarity to a model and self-evaluation. *The Journal of Abnormal and Social Psychology* 1961;62(2):257.
42. Hiza, M., Duncan, A.. A correlation for the prediction of interaction energy parameters for mixtures of small molecules. *AIChE Journal* 1970;16(5):733–738.
43. Yao, H., Tang, X., Wei, H., Zheng, G., Li, Z.. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In: *Proceedings of the AAAI conference on artificial intelligence*; vol. 33. 2019:5668–5675.
44. Khan, W., Aziz, A.. Natural convection flow of a nanofluid over a vertical plate with uniform surface heat flux. *International Journal of Thermal Sciences* 2011;50(7):1207–1214.

45. VanDecar, J., Crosson, R.. Determination of teleseismic relative phase arrival times using multi-channel cross-correlation and least squares. *Bulletin of the Seismological Society of America* 1990;80(1):150–169.
46. Kabir, H.D., Khosravi, A., Nahavandi, S., Srinivasan, D.. Neural network training for uncertainty quantification over time-range. *IEEE Transactions on Emerging Topics in Computational Intelligence* 2020;5(5):768–779.
47. Kabir, H.D., Abdar, M., Khosravi, A., Nahavandi, D., Mondal, S.K., Khanam, S., Mohamed, S., Srinivasan, D., Nahavandi, S., Suganthan, P.N.. Synthetic datasets for numeric uncertainty quantification: Proposing datasets for future researchers. *IEEE Systems, Man, and Cybernetics Magazine* 2023;9(2):39–48.
48. Jarshi, K.. Life expectancy (who): Statistical analysis on factors influencing life expectancy. 2018. URL www.kaggle.com/datasets/kumarajarshi/life-expectancy-who.
49. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.. Weight uncertainty in neural network. In: *International conference on machine learning*. PMLR; 2015:1613–1622.
50. Uber Technologies, I.. Bayesian regression tutorial. 2018. URL pyro.ai/examples/bayesian_regression.html.
51. Lee, S., Kim, H., Lee, J.. Graddiv: Adversarial robustness of randomized neural networks via gradient diversity regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2022;.