Uncertainty Quantification Neural Network from Similarity and Sensitivity

H M Dipu Kabir, Abbas Khosravi, Darius Nahavandi, Saeid Nahavandi

Institute for Intelligent Systems Research and Innovation (IISRI), Deakin University, Australia
{dkabir, abbas.khosravi, darius.nahavandi, saeid.nahavandi}@deakin.edu.au

***Abstract*—Uncertainty quantification (UQ) from similar events brings transparency. However, the presence of an irrelevant event may degrade the performance of similarity-based algorithms. This paper presents a UQ technique from similarity and sensitivity. A traditional neural network (NN) for the point prediction is trained at first to obtain the sensitivity of different input parameters at different points. The relative range of each input parameter is set based on sensitivity. When the sensitivity of one parameter is very high, a small deviation in that parameter may result in a large deviation in output. While selecting similar events, we allow a small deviation in highly sensitive parameters and a large deviation in less sensitive parameters. Uncertainty bounds are computed based on similar events. Similar events contain exact matches and slightly different samples. Therefore, we train a NN for bound correction. The bound-corrected uncertainty bounds (UB) provide a fair and domain-independent uncertainty bound. Finally, we train NNs to compute UB directly. The end-user need to run the final NN to obtain UB, instead of following the entire process. The code of the proposed method is also uploaded to Github. Also, users need to run only the fifth script to train a NN of a different UB.**

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

I. Introduction

The trustworthiness is the most important characteristic of any prediction system. The traditional prediction system is a point-prediction-type prediction system where the performance is measured from the statistical error. Commonly used statistical error values are the root-mean-square-error (RMSE), sum-squared-error(SSE), mean-absolute-error(MAE), etc. Point prediction has come to saturation through numerous amounts of research [1]–[3]. The best prediction system has zero epistemic uncertainty but the prediction is unable to remove the aleatoric uncertainty. Although uncertainty bounds cannot reduce the aleatoric uncertainty, properly trained multiple uncertainty bounds can indicate the level of aleatoric uncertainty.

We can never be fully certain about uncertainties but we can quantify uncertainties with the help of uncertainty bounds [4]. There are many approaches to the design of prediction interval-based uncertainty quantification systems [5]. Traditional methods are the Delta Method, Mean-Variance Estimation Method, Bayesian Method, Bootstrap Method, etc. All of these Neural Network training methods have an assumption of the Gaussian distribution. Other regressions based uncertainty quantification methods are popular in the quantification of epistemic uncertainty in deep learning [6]. The regression-based Bayesian method or dropout method also has a strong assumption on the distribution of unknown targets. The cost-function-based direct neural network training for uncertainty quantification has no assumption on the probability distribution of unknown targets. As a result, they construct better prediction intervals. However, there is another debated issue among direct NN-based uncertainty quantification. There are several versions of the cost function [7], [8]. Some researcher thinks that considering the average width of the interval is enough but some other researchers think that considering the failure distance is also required. Some researchers penalize only low coverage, where others penalize both high and low coverage. Therefore, we are proposing an uncertainty quantification technique without any assumption.

The improvement in the point prediction has come to the saturation. No matter, how good the point prediction system is, there is certain statistical error value of the prediction system [9]–[14]. A five-minutes ahead prediction system for electricity demand usually has a much lower error than a thirty-minutes ahead wind power generation prediction system. The value of the error for a perfectly trained point prediction system is the aleatoric uncertainty. The outcome of the same consequence can be slightly different due to the aleatoric uncertainty. The level of uncertainty varies from quantity to quantity and based on situations for the same quantity. The electricity demand in the morning can be more predictable than in the evening. The uncertainty is asymmetrically heteroscedastic and cannot be explained with the point prediction system. [15]–[18].

Traditional NN-based UQ techniques consider the Gaussian probability distribution of targets. In the early 2010s, Khosravi et al. propose NN based direct interval construction method [5]. The NN is directly trained to quantify the uncertainty through a cost function. Khosravi et al. consider coverage probability and the width in the LUBE cost function [19]. The LUBE method penalizes only low coverages. Later, Wan et al. propose to penalize both high and low coverages [5]. Marin et al consider deviation from the mid interval [20]. The performance criteria for the uncertainty quantification are still a debating issue. There is no ground truth for the measurement of the performance of quantified uncertainty. Therefore, the proposed system trains NNs, considering similarity and sensitivity. We count sensitivity-based similar occurrences to construct the uncertainty bound. The proposed system also indicates the sample density near the input pattern. The user can get an idea of the strength of the prediction from the sample density. The example code is available at the following link: [21].

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

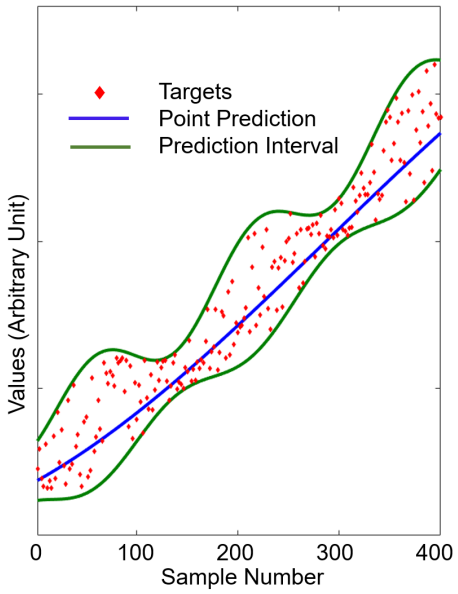

Fig. 1. The effectiveness of the interval forecast over the point prediction. Targets can be slightly different for the same input combination. The blue line, representing the point prediction cannot represent the heteroscedastic (time-varying/ parameter-varying) uncertainty. PIs represented by green lines can represent the level of uncertainty in a certain situation.

II. Theoretical Background

All-natural events contain a certain amount of uncertainty. The level of uncertainty can be different based on the system or the time or any other input parameters [22]–[24]. Humans cannot avoid uncertainty but a good estimation of the uncertainty enables better management and a greater profit. Many real-world events are entirely random. An individual household may run the washing machine or not at a certain time. A cloud user may require multiple cloud instances or may not require a cloud instance at a time. A perfect example of such a random situation is rolling a six-sided dice. while rolling a six-sided the probability of getting any side is equal. The probability of getting any side between one to six inclusive is equal. However, the average of outcomes of rolling a thousand dice has a high probability of getting a number close to 3.5, which is the average of possible outcomes [8]. However, with a rolling of five dice, there is a high probability of getting an average between three to four. The usage of electricity is point predictable with a very large number of users but the usage is interval predictable with a medium number of users.

Fig. 1 presents the interval representation of heteroscedastic uncertainty. The intervals are wider near the sample 80 and the intervals are narrower near the sample 130. A point prediction with overall statistical error cannot represent the heteroscedastic uncertainty. However, PIs represented by green lines can capture the uncertainty.

In general modeling techniques, the target (t_i) deviates with the prediction (y_i) by a small error value(ϵ_i) due to uncertainties. Therefore, the target can be presented as [5]:

$$t_i = y_i + \epsilon_i \tag{1}$$

where t_i is the measured target, and i (where $i = 1, 2, \cdots n$) is the sample number. ϵ_i is the error signal with the zero expectation. The error term makes a difference between the target t_i and its true regression mean, $\hat{y}_i$. Usually, it is considered that the term, ϵ_i in (1) is uniformly distributed. In practice, a model is applied to evaluate the true regression mean, $\hat{y}_i$ and the total variance can be defined as follows:

$$t_i - \hat{y}_i = [y_i - \hat{y}_i] + \epsilon_i \tag{2}$$

The variance at the first term on the right-hand side of (2) ($[y_i - \hat{y}_i]$) is considered in the PI construction. Therefore, the uncertainty of the prediction, $\hat{y}_i$, and the true regression, y_i are counted in PI depending on the calculation of components of the normalized probability distribution $P(y_i \mid \hat{y}_i)$. In contrast to PIs, PIs try to quantify irregularities linked to the discrepancy between the predicted values, y_i and the measured values, t_i. This associates to the normalized probability distribution $P(t_i \mid \hat{y}_i)$. When the two terms in the right-hand side of the Eqn. (2) are independent of each other, the total variation associated with the model outcome is expressed as:

$$\sigma_i^2 = \sigma_{\hat{y}_i}^2 + \sigma_{\hat{\epsilon}_i}^2 \tag{3}$$

The term $\sigma_{\hat{y}_i}^2$ introduces from the parameter estimation errors and the model misspecification ($[y_i - \hat{y}_i]$) known as the epistemic uncertainty, $\sigma_{\hat{\epsilon}_i}^2$ is the measure of the noise variance known as the aleatory uncertainty. Traditional PI construction techniques are based on the Gaussian approximation of uncertainty. The Gaussian distribution of assumed in Eq. 3 may not be true for all input combination [5]. Therefore, there was extensive research on the NN based direct interval construction, which can provide a smart interval for any arbitrary distribution of errors.

A. Cost function based Uncertainty Quantification

The cost function based UQ is getting popular due to its state of the art performance. The cost functions are designed to tune several statistical parameters. The two most extensively applied parameters are Prediction interval coverage probability (PICP), and Prediction interval normalized average width (PINAW). PICP and PINAW are defined as follows:

$$PICP = \frac{1}{n}\sum_{j=1}^{n} c_j \tag{4}$$

where, n is the total number of samples, c_j is the coverage of the j^{th} sample, defined as follows:

$$c_j = \begin{cases} 1, t_j \in [\underline{y}_j, \overline{y}_j] \\ 0, t_j \notin [\underline{y}_j, \overline{y}_j]. \end{cases} \tag{5}$$

where, t_j, $\underline{y}_j$, and $\overline{y}_j$ are the target, the lower bound, and the upper bound respectively for the j^{th} sample.

$$PINAW = \frac{1}{n \times R}\sum_{j=1}^{n}(\underline{y}_j - \overline{y}_j). \tag{6}$$

where, R is the range of targets.

The very first cost function based NN training for UQ is known as the lower upper bound estimation (LUBE) method.

In this method, the NN training aims to minimize the following cost function [5]:

$$CWC = PINAW + \gamma^{(\alpha,PICP)} e^{\eta(PINC-PICP)} \quad (7)$$

where, PI nominal coverage ($PINC$) is the expected value of the PICP after the NN training.

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

The LUBE method aims to reduce the PINAW when the required PICP is achieved. Several improvements to the LUBE method have proposed by several other researchers [8], [25], [26]. L G Marns consider the deviation from the mid interval [20]; G Zhang consider the deviation information criteria [27] etc.

B. Proposed Uncertainty Bounds

Uncertainty bounds or probability bounds are popular techniques of representing uncertainties in various fields [5], [28]. The uncertainty bound of a certain probability is the value corresponding to that cumulative probability. PI consists of two uncertainty bounds, the upper bound and the lower bound. The proposed NN training system consists of five steps, i) finding error and sensitivity based similarities and saving them, ii) training sample density NN (optional step), iii) visualization of similar samples (optional step), iv) training bound correction NN, v) training NN for the UB.

C. Error based Similarity

A previous work computes similarities based on correlation [29]. However, finding similarities based on correlation does not consider the relative amplitude of different input parameters. Therefore, the correlation-based similarity finding is effective for time-series data. Inputs of a prediction or uncertainty quantification system may have different types of inputs. In the forecast of temperature, inputs can be several previous temperatures, humidity, wind speed, day-time, etc. Different input parameters may have different ranges. Moreover, correlation compares the shapes of the curves. As a result, two different input patterns may have different amplitude but their shapes can be the same. However, the relationship between the inputs and the output can be nonlinear. The magnitude of the output may not increase in the same proportion as the output. Therefore, we consider the normalized absolute deviation based similarity.

The process of obtaining similar samples is presented as Algorithm 1. Each sample in the training data is compared with other samples. In the comparison between the two samples, the normalized deviations for all input values are computed at first. Samples are sorted based on the maximum of normalized deviations. Indexes of the top N_S close matches are saved as the index of similar samples (I_{SS}). The maximum normalized deviation for the most deviant sample among the selected sample is saved as the similarity threshold ($S_{Th}(i)$). $S_{Th}(i)$ is later applied for the estimation of the sample density.

Algorithm 1: Obtaining normalized absolute deviation based similar samples

Input ← Training Data
N ← Number of samples in the training data
n ← Number of input parameters
i ← Index of the corresponding sample
j ← Index of the sample compared
R ← Range matrix ($1 \times n$ matrix)
N_S ← Number of similar samples considered
Dev_{Index} ← [Deviation Index] ($2 \times N$ matrix)
$Dev(i,j)$ ← Deviation between input parameters
I_{SS} ← Index of similar samples ($N_S \times N$ matrix)
S_{Th} ← Similarity Threshold ($1 \times N$ matrix)

for $i \leftarrow 1$ *to* N **do**
  for $j \leftarrow 1$ *to* N **do**
    Compute $Dev(i,j)$
    $Dev_{Index}(j) \leftarrow$ [max($Dev(i,j)./R$) j]
  Sort ascending Dev_{Index} based on first column
  $I_{SS}(i) \leftarrow$ First N_S row and Index column of Dev_{Index} matrix ▷ Top N_S matched indexes
  $S_{Th}(i) \leftarrow$ Maximum $Dev(i,j)./R$ among selected indexes
Output ← I_{SS}, S_{Th}

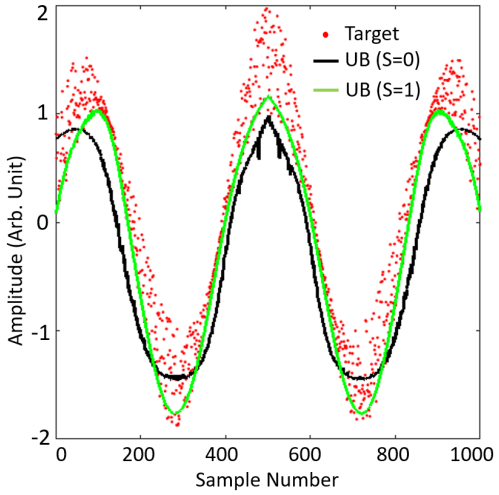

Fig. 2. Targets and uncertainty lower bound of 5% cumulative probability for the toy example. Targets are represented by red dots. The figure presents bounds without considering sensitivity as a black line, and bounds with the consideration of the sensitivity as a green line. Consideration of the sensitivity improves the accuracy and domain-independence of bounds.

D. Consideration of Input Sensitivities

Most neural networks have a large number of inputs. Some inputs may not influence the output. Inputs may have different sensitivities to outputs. As deviation at a less sensitive input causes a smaller change in the output, we allow a larger deviation for a less sensitive input.

The inclusion of a random variable as input excludes many closely matches and includes more distinct samples when the sensitivity is not considered. Because two distinct samples may get almost the same random values and two similar samples may get different random values. That may degrade the performance of the proposed UB construction system.

To investigate the importance of sensitivity we develop a toy example. The example has three inputs and one output. The output is directly related to the first input. The sign and amplitude of the uncertainty of the output are related to the second input. The third input is random to the output. We run codes with the toy data and plot target distribution and uncertainty bound of 5% cumulative probability. Uncertainty bounds are drawn for both inclusion and exclusion of sensitivity. Fig. 2 presents targets and bounds for the toy example. Code for the generation of toy dataset and the toy dataset is uploaded to Github [21].

Algorithm 2 presents the sensitivity-based similarity finding technique. The algorithm trains a point prediction NN in line-13. The point prediction NN helps to achieve sensitivity in line-16. The sensitivity is element-wise multiplied when we compute the similarity in line-19. The similarity threshold is also modified while considering sensitivity.

Algorithm 2: Obtaining normalized absolute deviation and Sensitivity based similar samples

1 Input ← Training Data
2 N ← Number of samples in the training data
3 n ← Number of input parameters
4 i ← Index of the corresponding sample
5 j ← Index of the sample compared
6 R ← Range matrix ($1 \times n$ matrix)
7 N_S ← Number of similar samples considered
8 Dev_{Index} ← [Deviation Index] ($2 \times N$ matrix)
9 $Dev(i,j)$ ← Deviation between input parameters
10 I_{SS} ← Index of similar samples ($N_S \times N$ matrix)
11 S_{Th} ← Similarity Threshold ($1 \times N$ matrix)
12
13 Train the point-prediction neural network (NN_p) for the data
14
15 **for** $i \leftarrow$ *1 to* N **do**
16 Find Normalized Sensitivity (S_i) of all input variables near input combination i with the help of NN_p.
17 **for** $j \leftarrow$ *1 to* N **do**
18 Compute $Dev(i,j)$
19 $Dev_{Index}(j) \leftarrow$ [max(S_i.*$Dev(i,j)$./R) j]
20 Sort ascending Dev_{Index} based on first column
21 $I_{SS}(i) \leftarrow$ First N_S row and Index column of Dev_{Index} matrix ▷ Top N_S matched indexes
22 $S_{Th}(i) \leftarrow$ Maximum S_i.*$Dev(i,j)$./R among selected indexes
23 Output ← I_{SS}, S_{Th}

E. Sample Density Neural Network

The training dataset may lack some patterns. Therefore, we aimed to provide the sample density near the input pattern. When the sample density is high, the prediction of the NN is more reliable. The sample density is the density of samples near the input pattern. Algorithm 3 presents the process of sample density NN training. The number of selected samples is divided by the similarity threshold to find the sample density near the corresponding pattern. Using the same process, the sample density is computed for all input patterns of the training data. A NN is trained for the sample density. Although it is possible to compute the sample density from the training data, the NN implementation is faster than computing the sample density from the dataset.

Algorithm 3: Neural Network training for sample density

1 Input ← Training Data, S_{Th}, N_S
2 S_D ← Sample density
3
4 $S_D \leftarrow N_S./S_{Th}$ ▷ Density of samples in nearby regions
5 ▷ While considering top N_S matches, S_{Th} is the maximum deviation among considered matches.
6 NN_{input} ← Training Data (Inputs)
7 $NN_{output} \leftarrow S_D$
8 Train NN
9 Output ← NN for S_D

F. Bound Correction

The error based similar pattern selection method selects both of the exact patterns and slightly different patterns. The highest possible difference between original and selected patterns is the similarity threshold (S_{Th}). As similar patterns are spread on either side of the original pattern, the examples outputs are spread towards both maxima and minima.

Fig. 3 presents an example input pattern as a thick blue line. One hundred similar samples are presented by thin dark grey lines. Corresponding outputs of similar samples are represented by extended light grey lines. These output examples form a rough probability distribution of the output, presented a green line. The target of the input pattern is presented as a blue diamond sign. We can also observe from Fig. 3 that matched patterns are slightly different than the original pattern. As similar patterns slightly deviate from the original pattern and the output distribution is formed with both closely matched and slightly different patterns, the probability distribution of output is slightly spread than the real distribution. Uncertainty bounds change their position due to the excess spread of the probability density function. If we are choosing an uncertainty upper bound of 95% cumulative probability, it becomes a point that corresponds to 96% to 99% cumulative probability. Therefore, we train a bound-correction neural network.

When the user wants to get the 95% UB bound value from the probability distribution function (PDF) of a similar event, he gets an output corresponds to 96%-99% UB value. The user may try to get about 92-93% UB value to get, 95% UB value. Therefore, we generate a table between the given value and the achieved value and train an inverse NN which corrects the bound. Algorithm 4 presents the bound correction NN training process. We iterate over a range of UB values and apply them to the probability density function to receive the transformed UB value. Finally, we train a NN with transformed UB value as input, (what we want to get) and provided UB value as output (what we need to provide, to get what we want).

With that bound correction NN, the user can find expected UB from the PDF. Say one user needs 95% UB. The user provides 95% to the bound correction NN (NN_{BC}). NN_{BC} return 92-93%, as it was provided to PDF to get 95% UB. The user applies that transformed value to the PDF to receive the UB value. Through the process, a more accurate UB value is achieved. However, the bound correction value changes with

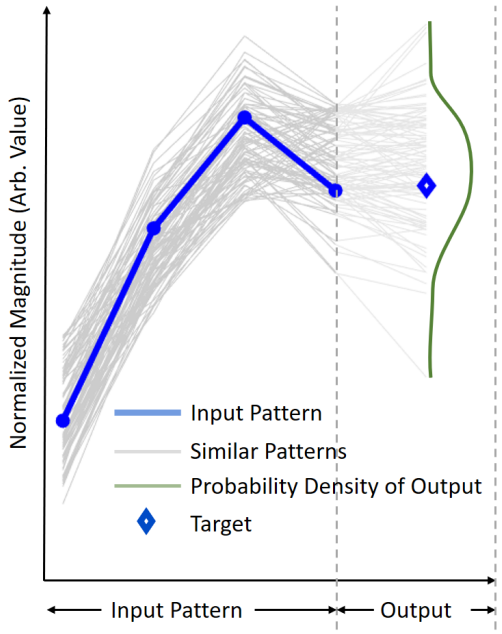

Fig. 3. Probability density of outputs from normalized error based similar patterns. Similar patterns are drawn with the original pattern. Outputs of similar patterns form the probability distribution. As the similar patterns are slightly different the probability distribution is also slightly spread compared to the original.

data and the number of similar samples considered. The NN needs to be retrained when the number of considered close matches is changed.

Algorithm 4: Algorithm for bound correction

Input ← Training Data, I_{SS}
PDF ← Output Distribution ($1 \times N$ matrix)

for $UB \leftarrow$ *1 to 99* **do**
  $UB_{Assumed}(UB) \leftarrow UB$
  for $i \leftarrow$ *1 to* N **do**
    Find matched indexes of the i^{th} sample
    PDF ← Output distribution from matched indexes
    PDF ← Sort(PDF)
    Find UB^{th} Percentile value (UBP) from PDF
    Check $UBP > Output(i)$
  P_G ← Percentage of times $UBP > Output(j)$
  $UB_{Found}(UB) = P_G$
NN_{BC} ← Bound Correction NN
NN_{BC} Input ← UB_{Found}
NN_{BC} Output ← $UB_{Assumed}$
Train NN_{BC}
Output ← NN_{BC}

G. NN Training for UB

The proposed NN training for UB consists of two major steps: 1) finding UB values (targets) for each sample from the PDF 2) train NN with input combinations and corresponding targets. Algorithm 5 presents the process of training NNs for the uncertainty bound. The UB is corrected at first using the bound correction. That corrected UB value is then applied to PDF to obtain targets (T_i). The NN (NN_{UB}) is trained considering obtained T as example outputs and input patterns of samples as example input.

Algorithm 5: Training NN for the Uncertainty Bound

Input ← Training Data, UB (a value from 0 to 1), NN_{BC}, I_{SS}
T_i ← Target for i^{th} sample for the NN training
NN_{UB} ← NN for UB

$UB_{Correct}$ ← Correct UB with NN_{BC}
for $i \leftarrow$ *1 to* N **do**
  Find matched indexes of the i^{th} sample
  PDF ← Output distribution from matched indexes
  PDF ← Sort(PDF)
  $T_i \leftarrow UB^{th}_{Correct}$ Percentile value (UBP) from PDF
$NN_{UB-input}$ ← Training Data (Inputs), UB
$NN_{UB-output} \leftarrow T$
Train NN_{UB}
Output ← NN_{UB}

III. Results

This paper proposes a novel algorithm for uncertainty quantification and applies the algorithm to a toy dataset, the electricity demand, and the wind power data. The toy dataset is generated with a program. The details of the generation are described in subsection $II - D$. Electricity demand and wind power data during 1^{st} August 2012 to 1^{st} August 2019 are downloaded from the UK Gridwatch website [30]. We consider four recent samples and the time of the day as input. The randomized dataset is split to 60% training 20% testing and 20% cross-validation datasets. As each NN is trained for a certain UB, UB is not an input to the NN.

A. Uncertainty Bounds for Electricity Demand

The uncertainty of the electricity demand has been increasing due to the growing population and the large scale installation of novel appliances. Such as the large scale installation of the electric vehicles (EVs) has made the overall demand more uncertain. A more uncertain electricity demand results in the less efficient management of a power grid.

There are several electricity generation sources to meet the demand of a grid. Some sources are renewable generation sources. Renewable generation sources require a high installation cost but the running cost is almost zero. Fossil-fuel based power plants require low installation cost but the running cost is high. Moreover, the governments of developed countries are trying to save fossil fuel for future generations and encouraging grid management to use renewable resources.

Fig. 4 presents UBs with targets for the electricity demand of the UK grid on 16th July 2019. Bounds are constructed for the 5-minutes ahead prediction. Targets remain between 15% to 85% uncertainty bounds for about 70% of the time. Fig. 5 presents UBs with targets for the electricity demand of the UK grid on 16th July 2019. Bounds are constructed for the 30-minutes ahead prediction. UBs are much wider than the UBs of 5-minutes ahead prediction. The uncertainty becomes higher for the long term prediction.

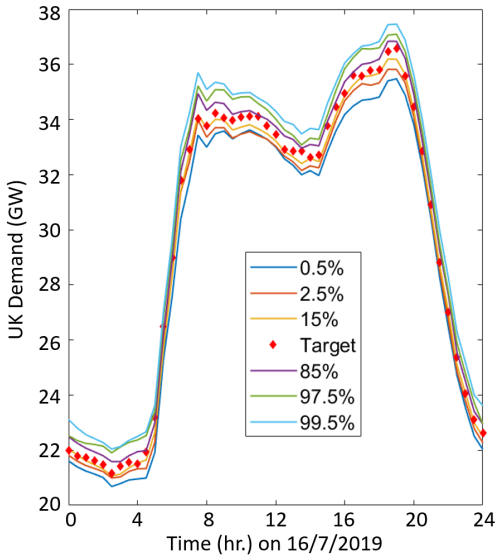

Fig. 4. Uncertainty bounds with targets for the electricity demand of the UK grid on 16th July 2019. Bounds are constructed for the 5-minutes ahead prediction.

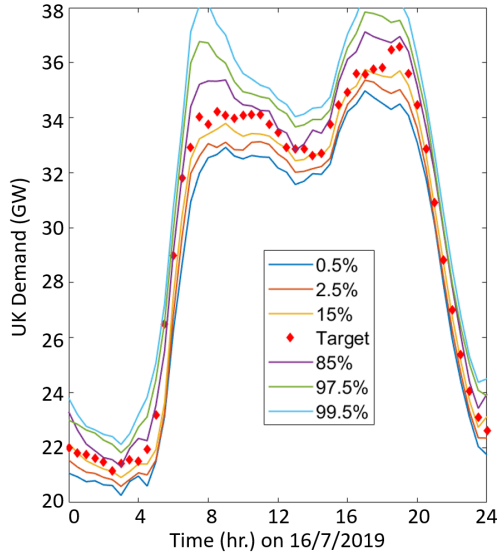

Fig. 5. Uncertainty bounds with targets for the electricity demand of the UK grid on 16th July 2019. Bounds are constructed for the 30-minutes ahead prediction.

B. Uncertainty Bounds for Wind Power Generation

Large scale renewable generation has made the grid more unstable. Among the renewable generations, wind power is the most uncertain power generation source. Therefore, we construct uncertainty bounds for wind power generation.

Fig. 6 presents UBs with targets for the total wind power generation of the UK grid on 16th July 2019. Bounds are constructed for the 5-minutes ahead prediction. Targets remain between 15% to 85% uncertainty bounds for about 70% of the time. Fig. 7 presents UBs with targets for the total wind power generation of the UK grid on 16th July 2019. Bounds are constructed for the 30-minutes ahead prediction. About 10-20% of the electricity demand is met by wind power generation sources in the UK grid.

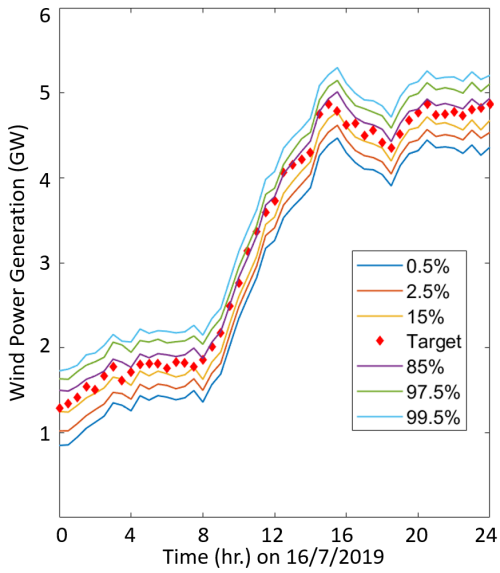

Fig. 6. Uncertainty bounds with targets for the total wind power generation of the UK grid on 16th July 2019. Bounds are constructed for the 5-minutes ahead prediction.

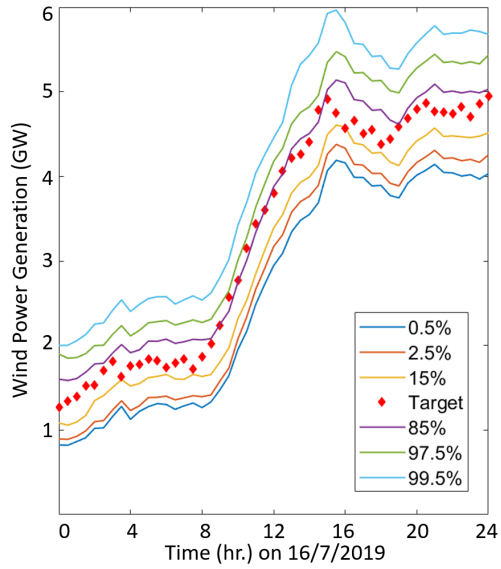

Fig. 7. Uncertainty bounds with targets for the total wind power generation of the UK grid on 16th July 2019. Bounds are constructed for the 30-minutes ahead prediction.

Table I presents the accuracy of uncertainty bounds. NNs are trained for different uncertainty bounds and different time-ahead predictions. Then NNs are applied to compute UBs for the entire dataset and UBs are compared to original targets. The first segment of the table presents the performance of UB construction NNs for the prediction of electricity demand. The second segment of the table presents the performance of UB construction NNs for the prediction of the wind power generation. According to the table, short-term predictions are more accurate than long term predictions.

TABLE I. PERFORMANCE OF CONSTRUCTED UBS

Statistical Performance of Uncertainty Bounds on the Electricity Demand Data											
Planned UB Values	1%	2.5%	5%	10%	15%	50%	85%	90%	95%	97.5%	99%
Performance of 30-min Ahead UBs	0.90%	2.46%	5.11%	10.17%	14.94%	50.03%	85.01%	89.96%	95.00%	97.47%	99.12%
Performance of 10-min Ahead UBs	1.00%	2.55%	4.96%	10.00%	15.09%	49.97%	84.94%	90.01%	94.98%	97.51%	99.11%
Performance of 5-min Ahead UBs	1.01%	2.50%	4.99%	10.00%	14.99%	50.01%	85.02%	90.01%	95.01%	97.52%	98.98%
Statistical Performance of Uncertainty Bounds on the Wind Power Generation Data											
Planned UB Values	1%	2.5%	5%	10%	15%	50%	85%	90%	95%	97.5%	99%
Performance of 30-min Ahead UBs	0.93%	2.52%	5.02%	10.03%	14.98%	50.13%	85.09%	89.90%	94.96%	97.47%	99.05%
Performance of 10-min Ahead UBs	1.02%	2.53%	4.96%	10.01%	15.07%	49.93%	84.94%	90.01%	94.96%	97.50%	99.03%
Performance of 5-min Ahead UBs	1.00%	2.50%	4.99%	10.00%	14.99%	50.00%	85.02%	90.02%	95.00%	97.50%	99.00%

C. Performance on Intervals

The level of uncertainty is determined by the gap between two intervals. Therefore, we construct prediction intervals of different coverage probabilities. We train uncertainty bound computing NNs for 0.025%, 0.05%, 0.1%, 0.2%, 0.8%, 0.9%, 0.95%, and 0.975% cumulative probability. Uncertainty bounds are applied to compute prediction intervals of different nominal coverage probability. We also construct intervals through popular direct interval construction methods. Table II presents the performance of intervals constructed through the current model and the popular direct interval construction methods. Performance on the 5-minutes ahead forecast is presented for the electricity demand and wind power data. The value of η is set to 1000 while computing the CWC. The CWC is presented as Eqn. 7 The PINAFD is computed considering ρ =1, β =1000, and $\delta = \alpha$ /50. Readers are referred to the paper [8] for the information of these parameters.

PINAW and PINAFD values are significantly smaller in the proposed UB computation method. The proposed algorithm achieves a 40% decrement in PINAW for the toy dataset. About 10% improvement is observed for the electricity demand and windpower datasets. The proposed uncertainty bound construction system also maintains a good PICP.

TABLE II. PERFORMANCE OF INTERVALS

LUBE Method [19]						
Data	PINC	PICP	PINAW	PINAFD	CWC	CWFDC
Toy	95%	95.10%	26.31%	11.26%	0.263	0.376
	90%	90.16%	22.45%	13.02%	0.225	0.355
	80%	80.32%	21.36%	17.11%	0.214	0.385
Demand	95%	95.06%	2.07%	0.51%	0.021	0.026
	90%	90.09%	2.01%	0.74%	0.020	0.029
	80%	80.14%	1.90%	0.77%	0.019	0.034
Wind	95%	95.14%	5.42%	1.23%	0.054	0.067
	90%	90.12%	4.61%	2.08%	0.046	0.068
	80%	80.15%	3.23%	2.74%	0.032	0.066
Mid-interval Deviation Consideration by L. G. Marin [20]						
Data	PINC	PICP	PINAW	PINAFD	CWC	CWFDC
Toy	95%	94.95%	24.06%	3.21%	1.889	0.275
	90%	89.99%	22.00%	4.61%	1.325	0.271
	80%	80.18%	21.2%	4.92%	0.212	0.266
Demand	95%	95.16%	3.03%	0.39%	0.030	0.035
	90%	90.19%	2.78%	0.34%	0.028	0.074
	80%	80.25%	2.72%	0.33%	0.027	0.033
Wind	95%	95.21%	9.02%	1.09%	0.090	0.102
	90%	90.22%	7.36%	1.56%	0.074	0.089
	80%	80.30%	6.51%	1.83%	0.065	0.084
Optimal LUBE Method [31]						
Data	PINC	PICP	PINAW	PINAFD	CWC	CWFDC
Toy	95%	94.98%	20.89%	2.74%	1.430	0.238
	90%	90.17%	19.05%	2.96%	0.191	0.194
	80%	80.49%	18.32%	3.57%	0.183	0.220
Demand	95%	95.10%	2.34%	0.31%	0.023	0.027
	90%	90.21%	2.29%	0.23%	0.023	0.025
	80%	80.40%	2.27%	0.20%	0.023	0.025
Wind	95%	95.09%	7.40%	0.53%	0.074	0.079
	90%	90.18%	6.01%	0.58%	0.060	0.066
	80%	80.41%	4.31%	0.67%	0.043	0.050
Proposed Method						
Data	PINC	PICP	PINAW	PINAFD	CWC	CWFDC
Toy	95%	94.96%	15.85%	0.53%	1.65	0.166
	90%	88.92%	14.63%	0.71%	4.9E4	0.317
	80%	80.65%	12.99%	1.10%	0.1299	0.147
Demand	95%	95.02%	2.00%	1.22%	0.020	0.033
	90%	89.88%	1.87%	1.27%	3.34	0.042
	80%	80.22%	1.82%	1.30%	0.018	0.034
Wind	95%	94.90%	5.03%	1.32%	2.7686	0.068
	90%	90.03%	4.61%	1.49%	0.046	0.064
	80%	80.70%	4.22%	1.57%	0.042	0.067

IV. CONCLUSION

The purpose of writing this paper is to present a novel method of constructing uncertainty bounds from similarity & sensitivity and to share the code. The proposed method constructs uncertainty bounds without any assumption on the distribution and the quality criteria. The number of similar samples can be changed based on the preference of the user. One segment of the NN training code also shows similar samples of a certain sample. Therefore, the proposed method is traceable for any prediction failure.

REFERENCES

[1] L. Tarisciotti, G. L. Calzo, A. Gaeta, P. Zanchetta, F. Valencia, and D. Sáez, "A distributed model predictive control strategy for back-to-back converters," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 9, pp. 5867–5878, 2016.

[2] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, PhD thesis, University of Cambridge, 2016.

[3] H. Quan, D. Srinivasan, and A. Khosravi, "Integration of renewable generation uncertainties into stochastic unit commitment considering reserve and risk: A comparative study," *Energy*, vol. 103, pp. 735–745, 2016.

[4] A. Malinin and M. Gales, "Predictive uncertainty estimation via prior networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 7047–7058.

[5] H. D. Kabir, A. Khosravi, M. A. Hosen, and S. Nahavandi, "Neural network-based uncertainty quantification: A survey of methodologies and applications," *IEEE Access*, 2018.

[6] T. Pearce, M. Zaki, A. Brintrup, and A. Neely, "High-quality prediction intervals for deep learning: A distribution-free, ensembled approach," *arXiv preprint arXiv:1802.07167*, 2018.

[7] H. A. E.-W. Khalifa, P. Kumar, and F. Smarandache, "On optimizing neutrosophic complex programming using lexicographic order," *Neutrosophic Sets and Systems*, vol. 32, no. 1, p. 21, 2020.

[8] H. D. Kabir, A. Khosravi, S. Nahavandi, and A. Kavousi-Fard, "Partial adversarial training for neural network-based uncertainty quantification," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2019.

[9] S. Ai, A. Chakravorty, and C. Rong, "Household power demand prediction using evolutionary ensemble neural network pool with multiple network structures," *Sensors*, vol. 19, no. 3, p. 721, 2019.

[10] C. Serpell, I. Araya, C. Valle, and H. Allende, "Probabilistic forecasting using monte carlo dropout neural networks," in *Iberoamerican Congress on Pattern Recognition*. Springer, 2019, pp. 387–397.

[11] A. Koohestani, M. Abdar, A. Khosravi, S. Nahavandi, and M. Koohestani, "Integration of ensemble and evolutionary machine learning algorithms for monitoring diver behavior using physiological signals," *IEEE access*, vol. 7, pp. 98 971–98 992, 2019.

[12] A. Seretis, X. Zhang, and C. D. Sarris, "Uncertainty quantification of radio propagation models using artificial neural networks," in *2019 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting*. IEEE, 2019, pp. 229–230.

[13] H. Asadi, C. P. Lim, A. Mohammadi, S. Mohamed, S. Nahavandi, and L. Shanmugam, "A genetic algorithm–based nonlinear scaling method for optimal motion cueing algorithm in driving simulator," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 232, no. 8, pp. 1025–1038, 2018.

[14] M. R. C. Qazani, H. Asadi, S. Khoo, and S. Nahavandi, "A linear time-varying model predictive control-based motion cueing algorithm for hexapod simulation-based motion platform," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[15] C. Bennett, M. Moghimi, M. Hossain, J. Lu, and R. A. Stewart, "Applicability of load forecasting techniques for customer energy storage control systems," in *Power and Energy Engineering Conference (APPEEC), 2015 IEEE PES Asia-Pacific*. IEEE, 2015, pp. 1–5.

[16] M. Hadjicharalambous, M. M. Polycarpou, and C. G. Panayiotou, "Neural network-based construction of online prediction intervals," *Neural Computing and Applications*, pp. 1–19, 2019.

[17] M. S. Nashwan, S. Shahid, and X. Wang, "Uncertainty in estimated trends using gridded rainfall data: A case study of bangladesh," *Water*, vol. 11, no. 2, p. 349, 2019.

[18] T. Davig and A. Foerster, "Uncertainty and fiscal cliffs," *Journal of Money, Credit and Banking*, vol. 51, no. 7, pp. 1857–1887, 2019.

[19] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Lower upper bound estimation method for construction of neural network-based prediction intervals," *IEEE transactions on neural networks*, vol. 22, no. 3, pp. 337–346, 2010.

[20] L. G. Marín, F. Valencia, and D. Sáez, "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, pp. 328–335.

[21] H. M. D. Kabir, "Nn-based-uncertainty-bound-with-examples." [Online]. Available: https://github.com/dipuk0506/NN-based-Uncertainty-Bound-with-Examples

[22] H. Asadi, S. Mohamed, C. P. Lim, S. Nahavandi, and E. Nalivaiko, "Semicircular canal modeling in human perception," *Reviews in the Neurosciences*, vol. 28, no. 5, pp. 537–549, 2017.

[23] A. Mohammadi, H. Asadi, S. Mohamed, K. Nelson, and S. Nahavandi, "Openga, a c++ genetic algorithm library," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 2051–2056.

[24] M. R. C. Qazani, H. Asadi, and S. Nahavandi, "High-fidelity hexarot simulation-based motion platform using fuzzy incremental controller and model predictive control-based motion cueing algorithm," *IEEE Systems Journal*, 2019.

[25] N. Cruz, L. G. Marín, and D. Sáez, "Prediction intervals with lstm networks trained by joint supervision," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[26] J. Ji, Y. Sun, F. Kong, and Q. Miao, "A construction approach to prediction intervals based on bootstrap and deep belief network," *IEEE Access*, vol. 7, pp. 124 185–124 195, 2019.

[27] G. Zhang, Y. Wu, K. P. Wong, Z. Xu, Z. Y. Dong, and H. H.-C. Iu, "An advanced approach for construction of optimal wind power prediction intervals," *IEEE Transactions on Power Systems*, vol. 30, no. 5, pp. 2706–2715, 2015.

[28] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in neural information processing systems*, 2017, pp. 6402–6413.

[29] H. D. Kabir, M. A. Hosen, S. Nahavandi, and A. Khosravi, "Prediction interval with examples of similar pattern and prediction strength," in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2017, pp. 1–4.

[30] Gridwatch, "Uk national grid status," Aug. 2019. [Online]. Available: www.gridwatch.templar.co.uk/

[31] H. M. D. Kabir, A. Khosravi, A. Kavousi-Fard, S. Nahavandi, and D. Srinivasan, "Optimal uncertainty-guided neural network training," *arXiv preprint arXiv:1912.12761*, 2019.