Prediction Intervals for Electricity Load Forecasting Using Neural Networks

Mashud Rana, Irena Koprinska, Abbas Khosravi, and Vassilios G. Agelidis

Abstract—Most of the research in time series is concerned with point forecasting. In this paper we focus on interval forecasting and its application for electricity load prediction. We extend the LUBE method, a neural network-based method for computing prediction intervals. The extended method, called LUBEX, includes an advanced feature selector and an ensemble of neural networks. Its performance is evaluated using Australian electricity load data for one year. The results showed that LUBEX is able to generate high quality prediction intervals, using a very small number of previous lag variables and having acceptable training time requirements. The use of ensemble is shown to be critical for the accuracy of the results.

I. INTRODUCTION

We consider the task of predicting the future electricity load (demand) from a time series of previous electricity loads. In particular, we study a time series of electricity loads measured every 30 minutes and our goal is to make predictions for the next value. Forecasting with lead times up to an hour is classified as Very Short Term Load Forecasting (VSTLF) and is a key task in the operation of electricity markets, such as the Australian national electricity market.

Electricity markets balance supply and demand - the market operator forecasts the demand, suppliers make bids indicating the amount of electricity they can generate and its price, the bids are matched against the forecast and the best ones are accepted, so that the demand is met and the cost is minimized. Predicting the load accurately is essential; over-predicting increases the cost while under-predicting compromises the reliability of supply.

The electricity load series data is complex. It is highly non-linear and non-stationary, with several cycles that are nested into each other, e.g. daily, weekly and yearly. In addition, it includes random components resulting from large industrial loads with irregular hours of operation, special events, holidays, extreme weather and sudden weather change. It is also noisy due to malfunctioning of data collection devices. All these factors combined together make the task of building accurate prediction models challenging.

The most successful methods for VSTLF reported in the literature are based on Holt-Winters exponential smoothing and autoregressive integrated moving average [1-3], linear regression [4-6] and Neural Networks (NNs) [4-10]. These methods, however, focus on point forecasting, i.e. at time $t$ the task is to predict the load value for time $t + h$, where $h$ is the forecasting horizon. In this paper we focus on interval forecasting, i.e. at time $t$ the task is to predict an interval of values for time $t + h$ with a certain probability, called confidence level.

More specifically, an interval forecast, or Prediction Interval (PI), consists of an upper and lower bound, between which the future value is expected to lie with a certain prescribed probability [11]. Predicting an interval of values with a certain probability instead of just predicting a single value gives more information about the variability of the target variable and the associated uncertainty. Hence, interval forecasts are more useful than point forecasts for risk management and quantification of the certainty of the point forecasts. They are especially suitable for applications requiring balancing of demand and supply, e.g. electricity and financial markets, stock manufacturing and inventory management [11].

A new method for constructing PIs using NNs was proposed in [12]. It is called LUBE and its main idea is to directly generate the lower and upper bounds of PIs using a feedforward NN with two outputs. LUBE uses the point forecasts for the training data and a specially designed cost function that is minimized in order to generate high quality PIs for new data. The LUBE method was compared with other NN-based methods for PI construction such as the delta [13], Bayesinan [14] and bootstrap methods [15]. LUBE was shown to generate high quality and valid PIs, typically superior to those generated by the other methods. The comparative study also showed that LUBE was simpler to implement and more flexible as it does not make prior assumptions about the data distribution.

In this paper we study the effectiveness of the LUBE method for interval forecasts for electricity load prediction. We also extend the LUBE method in two ways: 1) by incorporating feature selection and 2) by using an ensemble of LUBE NNs instead of a single NN in order to improve the reliability of the constructed PIs. The contribution of this paper can be summarized as follows:

1. We present a new method for interval forecasting, called LUBEX, an extension of the LUBE method. The extension is two-fold. Firstly, it uses an advanced feature selection method to identify a small set of...
informative variables. Good feature selection is crucial for the accuracy of prediction methods. We applied a novel estimator of mutual information based on $k$-nearest neighbors [16]. Secondly, we construct an ensemble of LUBE NNs instead of a single NN in order to reduce the sensitivity of LUBE to the random initialization of weights, the random perturbation during training and the network architecture, and improve the stability of the results.

2. We conduct a comprehensive evaluation of the proposed LUBEX method using 30-minute data from the Australian Energy Market Operator (AEMO) [17] for one year. The results show that the proposed method is able to generate high quality interval forecasts.

The rest of the paper is organized as follows. A problem statement is given in Section II. Section III presents measures for evaluating the quality of PIs. Sections IV and V describe the proposed LUBEX method. Section VI summarizes the experimental setup. Section VII presents and discusses the results. Section VIII concludes the paper.

II. PROBLEM STATEMENT

Given is a time series containing $n$ observations: $X_1, X_2, ..., X_n$. In a standard point forecasting scenario, the goal is to forecast $X_{n+h}$, the value of the series $h$ steps ahead, using the data up to time $n$. In an interval prediction scenario, our goal is to forecast the PI of $X_{n+h}$, i.e. its lower and upper bounds, $L(X_{n+h})$ and $U(X_{n+h})$. In this paper we consider a forecasting horizon $h = 1$, so our task is to predict $L(X_{n+1})$ and $U(X_{n+1})$, the PI for a 1-step ahead point forecast.

Given a confidence level $\mu = (1-\alpha)100\%$, a valid PI for $X_{n+1}$ will satisfy the following condition: $P(L(X_{n+1}) \leq X_{n+1} \leq U(X_{n+1})) \geq 1-\alpha$, i.e. the probability that the interval will contain $X_{n+1}$ will be equal or greater than the prescribed confidence level.

III. EVALUATING THE QUALITY OF PIS

The two most important properties of PI are coverage probability and width. A good PI will have high coverage probability and small width. While measures based on the coverage probability have been widely used to assess the quality of PIs, the width has been mostly ignored [12]. Following [12, 18], we evaluate the quality of both properties by using the three measures described below: PICP, PINAW and CWC.

Given a dataset of $N$ examples, the Prediction Interval Coverage Probability (PICP) is the probability that the target value $X_i$ of the $i$-th example will fall between the upper limit $U_i$ and lower limit $L_i$ of the prediction interval $PI_i$, averaged over all $i$. PICP is calculated empirically by counting the number of target values that fall within these limits:

$$PICP = \frac{1}{N} \sum_{i=1}^{N} c_i \cdot 100\%,$$

where

$$c_i = \begin{cases} 
1, & \text{if } X_i \in [L_i, U_i] \\
0, & \text{otherwise}
\end{cases}$$

A required nominal confidence level $\mu = (1-\alpha)100\%$ is set in advance. PIs with PICP smaller than $\mu$ are not reliable.

PICP alone is not enough to measure the quality of a PI. The width of the interval is related to its coverage probability – wider PIs have high coverage probability and vice versa. Thus, it is possible to achieve the required minimum PICP by simply widening the interval. However, too wide intervals are useless in practice. Hence, when assessing the quality of the PI we need to consider also their width. To do this, we use the Prediction Interval Normalized Averaged Width (PINAW). It measures the average width of the PIs, for all points in the dataset, normalized by the range of the target values $R$:

$$PINAW = \frac{1}{N} \frac{1}{R} \sum_{i=1}^{N} (U_i - L_i)$$

We also use the Coverage Width-based Criterion (CWC) [12] which combines PICP and PINAW:

$$CWC = PINAW \left(1 + \gamma e^{-\eta (PICP - \mu)} \right),$$

where $\eta$ and $\gamma$ are the parameters determining the contribution of PICP and NMPIW.

The two main principles of CWC are: 1) If the coverage probability is above the confidence threshold, i.e. $PICP \geq \mu$, then CWC should depend only on the PI’s width. This is achieved by setting $\gamma = 0$ to eliminate the exponential term. CWC becomes equal to the width PINAW and has a low value; 2) If the coverage probability is below the confidence threshold, i.e. the PIs are not reliable, CWC will have a high value, regardless of the PI’s width measured by PINAW. This is achieved by using a high value for $\eta$ in the exponential term and by setting $\gamma = 1$ to consider this term. As the exponential term becomes much higher than PINAW, the influence of PINAW is lost. CWC will be high because of the high value of the exponential term. In this way, the CWC balances the usefulness of the constructed PIs (narrow width) with their correctness (acceptable coverage probability). A detailed analysis of the CWC evaluation function is given in [12].

IV. THE PROPOSED LUBEX METHOD

We propose a new method for computing PI, called LUBEX, which is an extension of the LUBE method [12]. We firstly briefly describe the original LUBE method and then our proposed extension.
A. The LUBE Method

The LUBE method computes PIs using feedforward NNs. It was shown to produce high quality PIs at a reasonable computational cost, outperforming other methods such as the delta [13], Bayesian [14] and bootstrap methods [15]. The main idea is to train a NN to predict the PIs. The network learns to compute PIs on a training set and is then used to predict PIs on a new set of examples.

The architecture of the LUBE NN is a standard multi-layer perceptron as shown in Fig. 1. There are \( p \) input neurons, corresponding to the input variables of each example, and two output neurons corresponding to the lower and upper bounds of the PI for this example. There are one or more hidden layers and the number of neurons in them is set experimentally.

The LUBE NN is trained to minimize the CWC cost function. As CWC is not differentiable, the backpropagation algorithm cannot be used to do this. Instead, the LUBE NN uses the simulated annealing algorithm. The simulated annealing combines hill-climbing and random walk strategies. The latter is used to escape bad local minima.

Fig. 2 gives the flow chart of the LUBE NN training. After initialization of the parameters, including the starting temperature \( T_0 \), which is set to a high value, the starting solution (NN weights) \( w_0 \) and the starting cost function \( CWC_0 \), the NN is used to learn PIs for the training data. The two target values \( U_{t+1} \) and \( L_{t+1} \) of the prediction interval \( PI_{t+1} \) are both set to the target point forecast \( X_{t+1} \).

The cost function \( CWC_{new} \) is then computed and minimized. If there is an improvement, i.e. \( CWC_{new} \) is less than the previous best \( CWC \), \( CWC_{new} \) and \( w_{new} \) are accepted (hill-climbing step). Otherwise, the Metropolis criterion is used to test for acceptance (random walk step). The acceptance depends on the temperature and the badness of the new solution. The temperature is reduced as the algorithm progresses. Hence, bad solutions are more likely to be accepted at the beginning of the training. If the stopping criterion is not satisfied, a new set of weights is obtained via small perturbation and the training continues.

When the training is completed, the trained NN is used to compute the PIs for the testing data. More details about the parameter setting and our implementation of the simulated annealing are presented in Section VI.

B. The LUBEX Method

The LUBEX method is summarized in Fig. 3. We extend the LUBE method in two ways: 1) by adding a feature selection step and 2) by using an ensemble of LUBE NNs instead of a single NN.

1. Split data into training, validation and testing sets: \( D_{train} \), \( D_{valid} \) and \( D_{test} \)
2. Conduct feature selection using \( D_{train} \) and mutual information.
3. Train an ensemble of LUBE NNs to learn the prediction intervals for \( D_{train} \). Use \( D_{valid} \) to select the architecture of the LUBE NNs.
4. Test the performance of the ensemble on \( D_{test} \)
The first extension, feature selection, is needed to reduce the data dimensionality for faster training of the prediction models and faster online use of these models, and also for improving the prediction accuracy by removing the irrelevant and redundant features. We used mutual information for feature selection as described in Section V.

The second extension, using an ensemble, is necessary to improve the reliability of the generated PIs by reducing the sensitivity of the LUBE NN to the network architecture, the random initialization of the weights at the beginning and random perturbation of weights during training.

LUBEX forms an ensemble of LUBE NNs using the following method. It considers NNs with one hidden layer and builds 30 NN architectures $A_{1}, ..., A_{30}$ by varying the number of hidden neurons in them from 1 to 30. Using each architecture $A_{j}$, $j=1, ..., 30$, it builds an ensemble $E_{j}$ of $m$ LUBE NNs (we used $m=100$). The ensemble members of $E_{j}$ have the same architecture $A_{j}$ but are initialized to different random weights and are subject to different random perturbations during training. Each of them is trained on the training set. To predict the PI for the new example $i$, the ensemble $E_{j}$ combines the predictions of its members by taking the median of their lower and upper bounds: $\text{PI}_{i} = \text{median}(L_{i,j}, U_{i,j})$, median $(U_{i,j}, U_{i,j})$. The $m$ ensembles are evaluated on the validation set, the best one is selected (the one with smallest CWC) and used to predict the test data. This method reduces the variance and improves the stability of the generated PIs.

V. FEATURE SELECTION FOR PIS CONSTRUCTION

We applied Mutual Information (MI) as a feature selection method. MI is well known feature selector in classification tasks [19, 20], but is relatively new for time series applications. It is a very useful method as it captures both linear and nonlinear dependencies.

MI measures the dependence between two variables. If the two variables are independent, MI is zero; if they are dependent, MI has a positive value corresponding to the strength of the dependency. While MI has been very popular for nominal variables, its computation for continuous variables is more difficult as it requires making assumptions about the data distribution. In this work we apply the approach for MI estimation based on $k$-nearest neighbor distances [16]. It was shown to be efficient and more reliable than the traditional methods.

The MI between two random continuous variables $X$ and $Y$ with dimensionality $N$ is estimated as:

$$MI(X,Y) = \psi(k) - \frac{1}{k} - \frac{1}{N} \sum_{i=1}^{N} \left( \psi(n_x) + \psi(n_y) \right) + \psi(N),$$

where $\psi(x)$ is the digamma function $\psi(x) = \Gamma'(x)/\Gamma(x)$, $k$ is the number of nearest neighbors (we used $k=6$); $n_x(i)$ is the number of points $x_j$ with a distance to $x_i$ satisfying $\|x_i - x_j\| \leq \epsilon_x(i)/2$ and $n_y(i)$ is the number of points $y_j$ satisfying $\|y_i - y_j\| \leq \epsilon_y(i)/2$, where $\epsilon_x(i)/2$ is the distance between $x_i$ and its $k$-th neighbor in the $X$ subspace and $\epsilon_y(i)/2$ is the distance between $y_i$ and its $k$-th neighbor in the $Y$ subspace.

As described in Section VI, in this paper we use half-hourly data for 12 months, called case studies (C1-C12). To conduct feature selection using MI, we first apply a 1-week sliding window consisting of 336 half-hourly lag variables, for the training data for each case study. We then compute the MI score between each of the lag variables from the sliding window and the class variable and finally rank the lag variables in decreasing order based on the MI score.

![Fig. 4. MI scores and ranking of features for the twelve case studies](image)

Fig. 4 shows the normalized MI scores for each variable in ranked order, for each case study, from C1 to C12. There are two important observations: 1) the graphs for all 12 case studies are very similar and 2) the MI score sharply drops at rank 10 and after than only very slowly decreases. Based on this analysis, for each case study, we select the 10 highly ranked variables. They are listed in Table I.

<table>
<thead>
<tr>
<th>Case study</th>
<th>Selected features to predict $\text{PI}_{i,t}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Jan)</td>
<td>$X_{i,t}$ to $X_{i,t}, XD_{i,t}$ to $XD_{i,t}, XW_{i,t}$ to $XW_{i,t}$</td>
</tr>
<tr>
<td>2 (Feb)</td>
<td>$X_{i,t}$ to $X_{i,t}, XD_{i,t}$ to $XD_{i,t}, XD_{t-1}$, $XD_{t-1}$, $XW_{t-1}$ to $XW_{t-1}$</td>
</tr>
<tr>
<td>3 (Mar)</td>
<td>$X_{i,t}$ to $X_{i,t}, XD_{1,t}$ to $XD_{1,t}, XD_{t-1}$, $XW_{t-1}$, $XW_{t-1}$ to $XW_{t-1}$</td>
</tr>
<tr>
<td>4 (Apr)</td>
<td>$X_{i,t}$ to $X_{i,t}, XD_{t-1}$ to $XD_{t-1}, XD_{t-1}$, $XD_{t-1}$, $XW_{t-1}$ to $XW_{t-1}$</td>
</tr>
<tr>
<td>5 (May)</td>
<td>$X_{i,t}$ to $X_{i,t}, XD_{t-1}$ to $XD_{t-1}, XD_{t-1}$, $XD_{t-1}$, $XW_{t-1}$, $XW_{t-1}$ to $XW_{t-1}$</td>
</tr>
<tr>
<td>6 (June)</td>
<td>$X_{i,t}$ to $X_{i,t}, XD_{t-1}$ to $XD_{t-1}$, $XD_{t-1}$, $XW_{t-1}$, $XW_{t-1}$ to $XW_{t-1}$</td>
</tr>
<tr>
<td>7 (July)</td>
<td>$X_{i,t}$ to $X_{i,t}, XD_{t-1}$ to $XD_{t-1}$, $XD_{t-1}$, $XW_{t-1}$, $XW_{t-1}$ to $XW_{t-1}$</td>
</tr>
<tr>
<td>8 (Aug)</td>
<td>$X_{i,t}$ to $X_{i,t}, XD_{t-1}$ to $XD_{t-1}$, $XD_{t-1}$, $XW_{t-1}$, $XW_{t-1}$ to $XW_{t-1}$</td>
</tr>
<tr>
<td>9 (Sep)</td>
<td>$X_{i,t}$ to $X_{i,t}, XD_{t-1}$ to $XD_{t-1}$, $XD_{t-1}$, $XW_{t-1}$, $XW_{t-1}$ to $XW_{t-1}$</td>
</tr>
<tr>
<td>10 (Oct)</td>
<td>$X_{i,t}$ to $X_{i,t}, XD_{t-1}$ to $XD_{t-1}$, $XD_{t-1}$, $XW_{t-1}$, $XW_{t-1}$ to $XW_{t-1}$</td>
</tr>
<tr>
<td>11 (Nov)</td>
<td>$X_{i,t}$ to $X_{i,t}, XD_{t-1}$ to $XD_{t-1}$, $XD_{t-1}$, $XW_{t-1}$, $XW_{t-1}$ to $XW_{t-1}$</td>
</tr>
<tr>
<td>12 (Dec)</td>
<td>$X_{i,t}$ to $X_{i,t}, XD_{t-1}$ to $XD_{t-1}$, $XD_{t-1}$, $XW_{t-1}$, $XW_{t-1}$ to $XW_{t-1}$</td>
</tr>
</tbody>
</table>

where:

$X_{t}$: load on forecast day at time $t$; $X_{DN}$: load $N$ days before at time $t$; $X_{W}$: load 1 week before at time $t$.

We can see that in all case studies the majority of the selected variables correspond to the loads from: the previous few lags, the previous day around the prediction time and the previous week around the prediction time. This selection is
consistent with the daily and weekly cycles of the electricity load data.

VI. EXPERIMENTAL SETUP

A. Data

We use electricity load data for the state of New South Wales in Australia for the year 2010. The data is provided by the Australian Energy Market Operator (AEMO) and is publicly available [17]. The data is sampled every 30 minutes and recorded for each month. Thus there are 12 case studies in total, one for each month, and each of them contains between 1344 samples (for February) and 1488 samples (for the 31 days long months).

For each case study, the available data set is divided into three non-overlapped subsets: training $D_{\text{train}}$, validation $D_{\text{valid}}$ and testing $D_{\text{test}}$. The split was 50%-30%-20% respectively. The training set is used for feature selection and training of the NN models, the validation set is used to select the best NN architecture and the testing set is used to evaluate the performance of the proposed method for PI construction.

B. Implementation Details and Parameters

Here we describe additional implementation details and parameter settings to facilitate the replication of our results.

The parameters of the simulated annealing algorithm used for training of the NNs were set as follows: starting temperature $T_0=5$; temperature schedule: geometric, $T_{\text{new}}=r T_{\text{old}}$, $r=0.8$; constant in the acceptance criterion $k=1$ and regularization factor $\lambda=0.9$. The NN parameter perturbation was performed by generating random numbers with zero mean and unit variance. The training was terminated when one of the following conditions was met: a maximum number of epochs (1000) was reached, no further improvement was observed for a certain number of epochs (100) or a very low temperature was reached ($1.10^{-8}$).

In the CWC cost function we used $\eta=50$ and nominal confidence level $\mu=0.9$. Note that the high value of $\eta$ will highly penalize the PIs with a coverage probability below the nominal confidence level.

VII. RESULTS AND DISCUSSION

In this section we first discuss the convergence of the LUBEX method on training data and then the quality of the generated PIs for the testing data and the stability of the results.

A. Convergence on Training Data

We first examine the convergence behavior of LUBEX on the training data. The results showed good convergence in all 12 case studies. Two typical convergence graphs, for case study 2 and 6, are shown in Figs. 5 and 6. The CMC graphs use natural logarithmic scale. From these figures we can see than the initial value of the CWC cost function is high (e.g. $e^{40} \approx 2.35 \times 10^{17}$); this means that the initial PICP was unsatisfactory. However, the CWC value decreases sharply in the first 20-50 epochs and then gradually plateaus at about epoch 100, reaching a very low value (e.g. $e^2 \approx 7.38$). The corresponding temperature plots are also presented; they show the gradual decrease in temperature, following the geometric schedule, for reaching convergence. As the temperature decreases, the algorithm becomes more hill-climbing and less random walk. When the temperature is below 2, only states with improved CWC are accepted and CWC continuously decreases.

Fig. 5. Converge on training data for case study 2 (February): CWC and temperature plots

Fig. 6. Converge on training data for case study 6 (June): CWC and temperature plots

The training time of LUBEX was 40-60 minutes; this included the training of the 100 ensemble members and the aggregation of their outputs. The training time of a single ensemble member varied from 20s to 1 min, with
convergence typically reached at 500 epochs, although the CWC value was already very low after 100 epochs. Hence, we conclude that LUBEX has acceptable training time requirements.

B. Quality of PIs for Testing Data

To evaluate the quality of the constructed PIs for the testing data, we use the three performance indices introduced in Section III: PICP, PINAW and CWC. The results are shown in Table II.

To further improve the reliability of the results the ensemble of LUBE networks was created and evaluated five times, i.e. steps 3 and 4 from Fig. 3 were repeated five times. This further reduces the variability due to the different random initializations of the NN weights and perturbation during training. Thus, the PICP, PINAW and CWC results in Table II are the average values over the five runs.

<table>
<thead>
<tr>
<th>Case Study</th>
<th>PICP [%]</th>
<th>PINAW</th>
<th>CWC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Jan)</td>
<td>99.91</td>
<td>12.55</td>
<td>12.55</td>
</tr>
<tr>
<td>2 (Feb)</td>
<td>100.00</td>
<td>17.04</td>
<td>17.04</td>
</tr>
<tr>
<td>3 (Mar)</td>
<td>99.91</td>
<td>12.40</td>
<td>12.40</td>
</tr>
<tr>
<td>4 (Apr)</td>
<td>91.40</td>
<td>10.33</td>
<td>10.33</td>
</tr>
<tr>
<td>5 (May)</td>
<td>97.22</td>
<td>9.69</td>
<td>9.69</td>
</tr>
<tr>
<td>6 (June)</td>
<td>91.95</td>
<td>8.84</td>
<td>8.84</td>
</tr>
<tr>
<td>7 (July)</td>
<td>99.04</td>
<td>10.83</td>
<td>10.83</td>
</tr>
<tr>
<td>8 (Aug)</td>
<td>99.65</td>
<td>14.06</td>
<td>14.06</td>
</tr>
<tr>
<td>9 (Sep)</td>
<td>97.83</td>
<td>12.13</td>
<td>12.13</td>
</tr>
<tr>
<td>10 (Oct)</td>
<td>99.30</td>
<td>13.25</td>
<td>13.25</td>
</tr>
<tr>
<td>11 (Nov)</td>
<td>99.82</td>
<td>10.23</td>
<td>10.23</td>
</tr>
<tr>
<td>12 (Dec)</td>
<td>93.30</td>
<td>11.01</td>
<td>11.01</td>
</tr>
</tbody>
</table>

Table II shows that the coverage probability PICP is higher than the nominal confidence level of 90%, for all 12 case studies. This means that the constructed PIs are valid. The mean PICP and standard deviation over all case studies were 97.44 ± 3.29 indicating that in most of the case studies, the constructed PIs considerably outperformed the prescribed confidence level – e.g. in 7 cases PICP was above 99% and only in 3 cases it was between 91.4% and 93.3%. Thus, the LUBEX method, using MI as a feature selector and minimizing the CWC cost function is able to construct PIs with a high coverage probability.

By examining the second column of Table II, PINAW, we can see that the constructed PIs are narrow; their mean width was 11.86 ± 2.24. For visual comparison Fig. 7 shows the widths for the PIs with the largest and smallest width, case 2 (PINAW=17.04) and case 6 (PINAW=8.84), respectively. Their coverage probabilities are different – 100% for the first one and 91.95% for the second one, confirming that wider intervals are typically associated with higher coverage. If the coverage probability is similar, the width shows the uncertainty in data – higher width is associated with higher uncertainty and vice versa.

An examination of the CWC column of Table II shows that the CWC values are the same as the PINAW values for all case studies. This is as expected and follows from the definition of the CWC function for the case when the coverage probability is higher than the prescribed confidence level (Section III).

C. Variability of PICP on Testing Data

Fig. 8 shows a box plot of the PICP results for the five runs of each case study. The edges of the box are the 25th and 75th percentiles, the mark in the middle is the median, and the ends of the whiskers are the minimum and maximum of the five values, that are not considered outliers. Outliers are shown with crosses. We can see that the variations are small. The standard deviations varied from 0 for case study 2 to 1.22 for case study 12.

Thus, the LUBEX method with MI is able to compute stable PIs.

D. Effect of the ensemble

To evaluate the effect of the ensemble, we compare LUBEX with ensemble and LUBEX with a single NN. All other conditions are the same.
The results for LUBEX with a single NN are shown in Table III. We can see that the coverage probability PICP is unsatisfactory. It is greater than the nominal confidence level of 90% only in one case (case study 9), and it is only marginally greater (90.23%). Thus, LUBEX without an ensemble generated invalid PIs in 11 out of 12 case studies. We can also observe that the PIs computed by LUBEX with a single NN are slightly wider than those computed by LUBEX with an ensemble. However, the width is irrelevant when the constructed intervals are invalid. This is reflected by the high values of the cost function CWC as it includes a heavy penalty for invalid PIs.

**Table III. PICP for Testing Data Using LUBEX with MI and Single NN for Confidence Level 90%**

<table>
<thead>
<tr>
<th>Study</th>
<th>PICP [%]</th>
<th>PINAW</th>
<th>CWC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Jan)</td>
<td>87.13</td>
<td>14.51</td>
<td>322.12</td>
</tr>
<tr>
<td>2 (Feb)</td>
<td>79.41</td>
<td>23.79</td>
<td>122.90 x 10^12</td>
</tr>
<tr>
<td>3 (Mar)</td>
<td>78.43</td>
<td>14.44</td>
<td>237.99 x 10^6</td>
</tr>
<tr>
<td>4 (Apr)</td>
<td>75.02</td>
<td>10.53</td>
<td>90.457 x 10^7</td>
</tr>
<tr>
<td>5 (May)</td>
<td>74.35</td>
<td>8.12</td>
<td>934.11 x 10^6</td>
</tr>
<tr>
<td>6 (June)</td>
<td>80.63</td>
<td>8.02</td>
<td>8630.73</td>
</tr>
<tr>
<td>7 (July)</td>
<td>89.48</td>
<td>10.57</td>
<td>71.27</td>
</tr>
<tr>
<td>8 (Aug)</td>
<td>79.65</td>
<td>9.75</td>
<td>109.32 x 10^9</td>
</tr>
<tr>
<td>9 (Sept)</td>
<td>90.23</td>
<td>11.23</td>
<td>35.35</td>
</tr>
<tr>
<td>10 (Oct)</td>
<td>88.70</td>
<td>13.91</td>
<td>1391.60</td>
</tr>
<tr>
<td>11 (Nov)</td>
<td>88.24</td>
<td>14.80</td>
<td>637.38</td>
</tr>
<tr>
<td>12 (Dec)</td>
<td>83.83</td>
<td>14.89</td>
<td>1879.33</td>
</tr>
<tr>
<td>mean</td>
<td>82.92</td>
<td>12.88</td>
<td>102.51 x 10^11</td>
</tr>
<tr>
<td>st.dev</td>
<td>5.73</td>
<td>4.29</td>
<td>354.75 x 10^11</td>
</tr>
</tbody>
</table>

Fig. 9 shows a box plot of the PICP results for the five runs of each case study using LUBEX with a single NN. By comparing Fig. 9 with Fig. 8, and noting the different scales of the y axes, we can see that when using a single NN the PICP variations are much bigger. The smallest standard deviation was 3.78 (case study 1) and the three biggest were 26.93 (case study 2), 21.26 (case study 8) and 16.04 (case study 5).

We found that the use of ensemble was very important. LUBEX with a single NN was not able to compute PIs with satisfactory coverage probability in most of the cases, and its performance varied considerably from one run to another. Thus, both extensions, the use of MI as a feature selector and the use of an ensemble of NNs, were highly beneficial.

In conclusion, our results show that the ensemble component in LUBEX is very important for constructing valid and stable PIs.

**VIII. CONCLUSION**

We considered the task of interval forecasting for electricity load prediction. We adopted and extended the LUBE method, a NN-based method for estimating the lower and upper bound of PIs. The extended method, called LUBEX, includes a feature selection technique based on a novel estimate of MI and an ensemble of NNs to guard against fluctuations in performance.

We conducted a comprehensive evaluation using Australian electricity load data for one year, sampled every 30 minutes. The results showed that LUBEX was able to produce high quality PIs – i.e. PIs with a high coverage probability (97.44%) that satisfy the minimum confidence level, and were very stable over multiple runs. These results were achieved using only 10 variables, previous electricity loads, selected by the MI algorithm. The training time was acceptable – it took on average 50 minutes to train the ensemble and produce the combined prediction. Hence, we conclude that LUBEX is a promising method for interval forecasting for electricity load prediction.

Future work will investigate the performance of LUBEX with other feature selection methods and its extension for more than one step ahead forecasting. We also plan to apply LUBEX to other domains such as prediction of solar power produced by photovoltaic plants.

**ACKNOWLEDGMENT**

Mashud Rana is supported by an Endeavour Postgraduate Award and NICTA Research Project Award (NRPA).

**REFERENCES**


