Neural Network Prediction Interval Based on Joint Supervision

Nicolás Cruz
Department of Electrical Engineering
University of Chile
Santiago, Chile
nicolascruz2187@gmail.com

Luis G. Marín
Department of Electrical Engineering
University of Chile
Santiago, Chile
luis.marin@ing.uchile.cl

Doris Sáez
Department of Electrical Engineering
University of Chile
Santiago, Chile
dsaez@ing.uchile.cl

***Abstract*—In this paper, a new prediction interval model based on a joint supervision loss function for capturing the uncertainties associated with the modeled phenomenon is described. This model provides the upper and lower bounds of the predicted values in accordance with the desired coverage probability, as well as their expected values. A benchmark problem is used to evaluate the proposed method, and a comparison with the neural network covariance method is performed. Additionally, the proposed method was applied to forecast the residential demand from a town in UK, considering the prediction interval performance for one-day ahead. The results show that the method is able to generate an interval with narrower width than the covariance method, and maintains the coverage probability. The information provided by the prediction interval could be used in the design of microgrid energy management systems.**

Keywords—Prediction interval, joint supervision, neural network.

I. Introduction

Prediction models are fundamental tools for estimating future states of time dependent variables. They rely on approximating the underlying distribution of a certain function in order to generate an output that is as close to the real value as possible. Several methods for performing predictions are available and used, such as linear models, fuzzy systems, and neural networks for non-linear predictions [1]. Machine learning has become popular during the last decade in industry, academia, medicine, and several other areas where their quality of universal approximators allows modeling complex systems based only on a set of samples [2],[3].

However, while the predictions generated by a neural network can be very accurate, given enough data and correctly selected hyperparameters, these predictions do not take the uncertainty of the data into account. Indeed, in its supervised form, the current paradigm of neural network training treats each point of the training dataset as a completely accurate representation of the distribution generated by the modeled function. However, this assumption is violated when noise is present in the system, meaning that the accuracy of the model drops as the uncertainty increases. Since an estimation of the reliability of the prediction is fundamental for several applications (such as load forecasting), prediction intervals have been proposed [4].

A prediction interval consists of an upper and lower value that defines a range in which the estimated random variable is expected to belong, given a coverage probability. This means that the model is able to predict the uncertainty of an observation yet to be made. To be useful, prediction intervals are expected to maximize the amount of data inside the bounds, as well as to be as narrow as possible [5].

Several methods have been proposed to construct the prediction interval. Most of them require two different models, the first one aimed at constructing the crisp prediction, and the second one designed to generate the corresponding prediction interval. In most cases certain assumptions on the distribution of the data are made[6]. Such algorithms include the covariance method that uses the error between the prediction of the model and the real data to generate an interval under the assumption of normally distributed noise with zero mean. Given the complexity of the modeled systems, these assumptions are difficult to validate and do not hold true for certain systems, leading to a loss in the accuracy of the interval.

In response to these shortcomings, models that are capable of generating prediction intervals without any assumptions on the distribution of the data have been proposed. Such models include techniques such as that presented in [7] where complex pipelines are required to obtain the prediction interval, resulting in implementation of a convoluted algorithm. Other methods include lower upper bound estimation (LUBE) [6] in which a two-output neural network is used to generate the interval with the parameters tuned by a special algorithm achieving state of the art results, and [8] where genetic algorithms are used to tune the parameters of a two output neural network. While both methods use neural networks to generate the prediction interval, the first one still requires a different model to achieve the crisp prediction, and the second one is subject to the shortcomings of evolutionary algorithms applied to the training of neural networks, mainly, the very high computational complexity for high dimensionality problems, resulting in slow parameter tuning.

In this work we propose a novel prediction interval based on a three-output neural network trained by backpropagation using a combination of the classical loss function as well as a new interval loss function, which is also proposed in this paper, that measures the quality of the generated interval. This model retains the simplicity and efficiency of the backpropagation trained neural networks while achieving state of the art results in interval generation and model prediction. Furthermore, this method allows unifying the crisp prediction and the interval generation under a single model trained by a single algorithm (backpropagation), greatly simplifying the application process. This method is also fast to train and benefits from all the

978-1-5090-6014-6/18/$31.00 ©2018 IEEE

Authorized licensed use limited to: National Library of Technology. Downloaded on June 11,2021 at 22:24:13 UTC from IEEE Xplore. Restrictions apply.

development made for backpropagation techniques, such as generalization techniques and parallelized training.

The main contribution of this work consists of the development of a new differentiable loss function for interval prediction generation. This function is consistent with backpropagation training and is able to create accurate prediction intervals. In this paper we evaluate the benefits of the intervals generated with this loss function, and offer a real case study.

The paper is organized as follows: The neural network structure used to generate the crisp and interval predictions is introduced in Section II. The new loss function used to train the proposed neural network prediction interval is provided in Section III. The results obtained on a state dependent noise benchmark series, and a real case study by constructing intervals for load forecasting are presented in Section IV. In both cases, the proposed method is compared to a standard prediction interval method. The conclusions of this study are presented in the last section.

II. Neural network prediction Interval modeling

A neural network is defined by a set of adjustable parameters that determine the output of the system $\hat{y}(k)$ to a certain input $z_i(k)$ at the time instant k [9]. These parameters consist of a set of hidden and output weights ($w_{j,i}^h$, $w_{j,l}^o$), and a set of hidden and output biases $\left(b_j^h, b_l^o\right)$. Mathematically, the neural network output is defined as:

$$\hat{y}_l(k) = \sum_{j=1}^{L} w_{j,l}^o \left(\tanh\left(\sum_{i=1}^{p} w_{j,i}^h\, z_i(k) + b_j^h\right)\right) + b_l^o \qquad (1)$$

where l is the number of output units, L is the number of neurons in the hidden layer and p is the number of inputs to the network. The parameters of the neural network are trained by minimizing the root mean square error ($RMSE$) using backpropagation in a paradigm defined as supervised learning [10]. In this paper a method capable of generating a neural network prediction interval, as well as a crisp prediction is proposed. Therefore, a neural network with three outputs is used with $l = \{upper, crisp, lower\}$. Thus, $\hat{y}_{upper}(k)$ represents the upper bound of the interval at time k; $\hat{y}_{crisp}(k)$ represents the estimated value at time k; and $\hat{y}_{lower}(k)$ corresponds to the lower bound of the interval at time k.

To obtain the interval as well as a crisp prediction, the parameters $\hat{\theta} = \{w_{j,i}^h,\ b_j^h, w_{j,l}^o, b_l^o\}$ $l = \{upper, crisp, lower\}$ are tuned through backpropagation method. This training stage is carried out after data is acquired and some model structure has been selected [3]. In this work, a new total loss function is used as the metric for the training process of the three-output neural network. The total loss function (L_{total}) proposed is based on the joint supervision approach as presented in [11],[12].

In this approach, the tuning of the neural network parameters is based on the supervision of more than one loss function. Several loss functions can define multiple objectives for the network. In this study, by combining two loss functions, the network can be trained to generate an interval that meets the desired coverage probability as well as minimizing the width of such an interval. In the next section, the proposed method for developing the prediction interval is presented.

III. Proposed Method for Developing the prediction interval

The training task is basically an estimation problem according to some metric. The parameters $\hat{\theta}$ are tuned by propagating the error $e_l(k) = y(k) - \hat{y}_l(k)$ through the network in order to minimize the difference between the model outputs $\hat{y}_l(k)$ at the time instant k and the real value $y(k)$ at the time instant k of the dataset. Then, the classical loss function as training metric are defined as follows:

$$L_s^l = \frac{1}{N}\sum_{k=1}^{N} e_l^2(k) \qquad l = \{upper, crisp, lower\} \qquad (2)$$

where N is the amount of available training data. To calculate the change in the parameters of the neural network, the gradient is calculated as the derivative of the classical loss function (L_s^l). According to (2), the gradients are defined as follows:

$$grad_s^l = -\frac{2}{N}\sum_{i=1}^{N} e_l(k) \quad l = \{upper, crisp, lower\} \qquad (3)$$

This classical loss function (see eq. (2)) minimizes the error between the predicted values $\hat{y}_l(k)$ and real values $y(k)$. However, it is impossible to define an interval using this metric alone since there are no terms that represent the interval. To address this issue, a second metric, called interval loss function (L_i), is introduced here. The interval loss function for the upper (L_i^{upper}) and the lower (L_i^{lower}) bounds of interval are defined as follows:

$$L_i^{upper} = \frac{1}{N}\sum_{k=1}^{N} E_{upper}\left(e_{upper}(k)\right) \qquad (4)$$

$$L_i^{lower} = \frac{1}{N}\sum_{k=1}^{N} E_{lower}\left(e_{lower}(k)\right) \qquad (5)$$

where:

$$E_{upper}\left(e_{upper}(k)\right) = \begin{cases} \left(e_{upper}(k)\right)^2 \ if\ e_{upper}(k) > 0 \\ 0\ otherwise \end{cases} \qquad (6)$$

$$E_{lower}(e_{lower}(k)) = \begin{cases} (e_{lower}(k))^2\ if\ e_{lower}(k) < 0 \\ 0\ otherwise \end{cases} \qquad (7)$$

According to (4) and (6), for each point of data $y(k)$ with a larger value than the upper bound $\hat{y}_{upper}(k)$, the cost introduced by the L_i^{upper} function increases quadratically in accordance with the difference between the real data $y(k)$ and the predicted upper value of the interval $\hat{y}_{upper}(k)$. Data points $y(k)$ with a lower value than the upper bound $\hat{y}_{upper}(k)$ introduce no cost. Likewise, as presented in (5) and (7), any data point $y(k)$ with a lower value than the corresponding lower bound $\hat{y}_{lower}(k)$

Authorized licensed use limited to: National Library of Technology. Downloaded on June 11,2021 at 22:24:13 UTC from IEEE Xplore. Restrictions apply.

introduces a cost. When the upper and lower interval loss functions are combined, the result is a penalization for each data point $y(k)$ that lies outside the prediction interval. The three outputs of the neural network use the same target data $y(k)$ during the training process. The introduction of different total loss functions for each output results in a different gradient for each of them. This, in turn, is the key factor for the prediction interval generation.

To obtain the gradients of the new interval loss function (L_i), the function must be differentiable in the whole space. Since the function is well defined in R^*, a proof of differentiability in 0 is provided for the upper interval loss function (L_i^{upper}):

$$\frac{\delta E_{upper}(0)}{\delta\left(e_{upper}\right)} = \lim_{h\to 0^+}\frac{E_{upper}(h) - E_{upper}(0)}{h} = 0 \qquad (8)$$

$$\frac{\delta E_{upper}(0)}{\delta\left(e_{upper}\right)} = \lim_{h\to 0^-}\frac{E_{upper}(h) - E_{upper}(0)}{h} = 0 \qquad (9)$$

The development is equivalent for the lower interval loss function (L_i^{lower}). Hence:

$$grad_i^{upper} = \frac{1}{N}\sum_{i=1}^{N}\frac{\delta E_{upper}\left(e_{upper}(k)\right)}{\delta\left(\hat{y}_{upper}(k)\right)} \qquad (10)$$

$$grad_i^{lower} = \frac{1}{N}\sum_{i=1}^{N}\frac{\delta E_{lower}(e_{lower}(k))}{\delta\left(\hat{y}_{upper}(k)\right)} \qquad (11)$$

where:

$$\frac{\delta E_{upper}\left(e_{upper}(k)\right)}{\delta\left(\hat{y}_{upper}(k)\right)} = \begin{cases} -2\ e_{upper}(k)\ if\ e_{upper}(k) > 0 \\ 0\ otherwise \end{cases} \qquad (12)$$

$$\frac{\delta E_{lower}(e_{lower}(k))}{\delta(\hat{y}_{lower}(k))} = \begin{cases} -2\ e_{lower}(k)\ if\ e_{lower}(k) < 0 \\ 0\ otherwise \end{cases} \qquad (13)$$

The total loss function (L_{total}) for the upper and lower bounds of the interval are defined as the weighted sum of the classic loss function (see eq. (2)) and, the upper and lower interval loss function (see eqs. (4) and (5)):

$$L_{total}^{upper} = L_s^{upper} + \lambda L_i^{upper} \qquad (14)$$

$$L_{total}^{lower} = L_s^{lower} + \lambda L_i^{lower} \qquad (15)$$

where λ is the weighting factor. The crisp prediction output is obtained by using only the classical loss function (see eq. (2)). The gradients for the total interval loss functions are defined as:

$$grad_{total}^{upper} = grad_s^{upper} + \lambda grad_i^{upper} \qquad (16)$$

$$grad_{tot}^{lower} = grad_s^{lower} + \lambda grad_i^{lower} \qquad (17)$$

Since the total loss functions and gradients for all outputs are well defined, the classical backpropagation method can be used to tune the network's parameters $\hat{\theta}$.

A. Identification of the Parameters

The aim of the identification method is to obtain the parameters $\hat{\theta}$ of the neural network prediction interval such that the interval will be as narrow as possible, and so that it contains the largest possible amount of measured data [13]. The minimization of the interval width is achieved through the term L_s^l (eq. see (2)) by a minimization of the error between the interval bounds $(\hat{y}_{lower}(k), \hat{y}_{upper}(k))$ and the real data. On the other hand, the minimization of the number of points outside the interval is achieved through the terms L_i^{upper} and L_i^{lower} (see eqs. (4) and (5)) which introduces a penalty to any data point outside the interval. Note that while the classical loss function (L_s^l) tends to reduce the width of the interval, the interval loss function (L_i) tends to increase it, introducing a tradeoff between the coverage probability and the interval width that can be regulated by modifying the parameter λ (see eqs. (14) and (15)).

In this paper, the use of a neural network prediction interval model is proposed for forecasting the output at future steps ahead (N_p). Therefore, the parameters of the neural network are tuned for N_p steps ahead. The prediction at future steps ahead is considered as a function of the output until the previous step, real and/or forecasted, depending on the number of steps ahead.

To evaluate the quality of the interval, the prediction interval coverage probability $(PICP)$, and the prediction interval normalized average width $(PINAW)$ are used as metrics for N_p step ahead:

$$PINAW = \frac{1}{NR}\sum_{k=1}^{N}\hat{y}_{upper}(k+h) - \hat{y}_{lower}(k+h) \qquad \forall h = 1,\cdots,N_P \qquad (18)$$

$$PICP = \frac{1}{N}\sum_{k=1}^{N}\delta_{k+h} \qquad \forall h = 1,\cdots,N_P \qquad (19)$$

with $\delta_{k+h} = 1$ if $y(k+h)\epsilon[\hat{y}_{lower}(k+h), \hat{y}_{upper}(k+h)]$ otherwise $\delta_{k+h} = 0$. Let R denote the range given by the maximum and minimum values of measurements [6],[14].

Fig. 1 presents the methodology proposed to obtain the prediction intervals. First the neural network prediction interval is trained using backpropagation with the total loss functions, and the set of parameters $\hat{\theta}(k+h)$ is found.

The algorithm presented in Fig. 1 iterates, gradually increasing the interval width until the desired coverage probability $(PICP \geq \mu)$ is achieved. This gradual increase of the interval width is reached by modifying parameter λ in each iteration.

Once the desired $PICP$ is reached, the algorithm continues to iterate while keeping the value of λ constant. This is done to compensate for the random weight initialization, which can give different results for the same total loss functions (L_{total}) and λ parameter. In each iteration the $PINAW$ of the resulting neural network prediction interval is compared to the value of

Authorized licensed use limited to: National Library of Technology. Downloaded on June 11,2021 at 22:24:13 UTC from IEEE Xplore. Restrictions apply.

the best$PINAW$ overall, and if the $PINAW$ of the current iteration is lower than the best$PINAW,$ the network is saved, and the $bestPINAW$ is set as the current $PINAW$. Finally, a neural network prediction interval with a desired coverage probability and a minimum interval width is obtained.

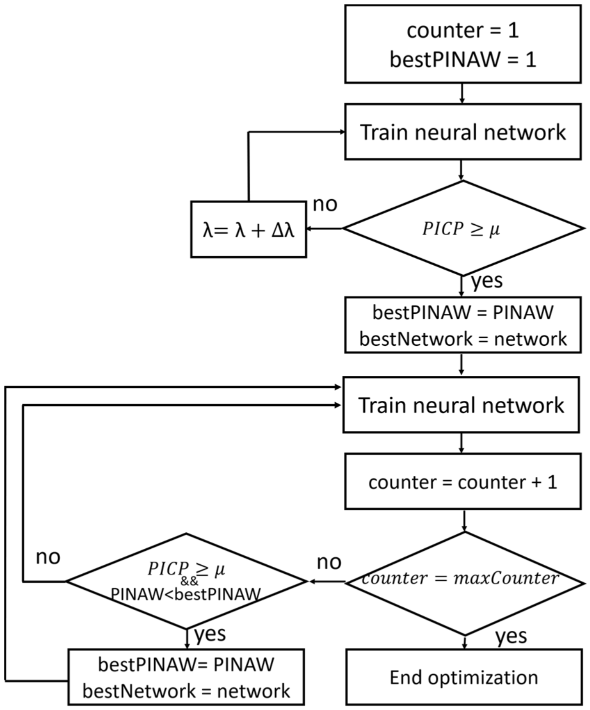

Fig. 1. Methodology for Developing Prediction Intervals.

IV. Experiment and results

In this work, comparisons were made between the proposed method based on joint supervision, and the covariance method based on the work developed in [15]. In the covariance method the prediction interval is generated by considering the difference between the observations $y(k)$ and the values predicted by the model $\hat{y}(k)$. This method is based on the assumption that the underlying noise possesses a normal distribution with zero mean and variance σ^2.

The neural network in (1) can be written as:

$$\hat{y}_l(k) = \sum_{j=1}^{L} w_{j,l}^{o}\, \bar{Z}_j(k) + b_l^{o} \tag{20}$$

where $\bar{Z}_j(k)$ is the output of the hidden layer of the neural network:

$$\bar{Z}_j(k) = \tanh\left(\sum_{i=1}^{P} w_{j,i}^{h}\, z_i(k) + b_j^{h}\right) \tag{21}$$

Given this, we can write the upper and lower interval predictions following the definition presented in [15]:

$$\hat{y}_{upper}(k) = \sum_{j=1}^{L} w_j^{o}\, \bar{Z}_j^{*}(k) + b^{o} + t_\alpha\, I^{NN} \tag{22}$$

$$\hat{y}_{lower}(k) = \sum_{j=1}^{L} w_j^{o}\, \bar{Z}_j^{*}(k) + b^{o} - t_\alpha\, I^{NN} \tag{23}$$

where

$$I^{NN} = \sigma_e \sum_{j=1}^{L} \left(1 + \bar{Z}_j^{*T}(k) \left(\bar{Z}_j^{T}(k) \bar{Z}_j(k)\right)^{-1} \bar{Z}_j^{*}(k)\right)^{\frac{1}{2}} \tag{24}$$

$\bar{Z}_j^{*}(k)$ is obtained with new input data used to predict the future observation, and $\bar{Z}_j(k)$ is a matrix obtained with the data used in the training process, where parameters of the neural network were found. The error variance is σ_e, and t_α is a tunable parameter which is found by running multiple iterations with increasingly larger t_α values until the desired prediction interval coverage probability $(PICP)$ is achieved on experimental data [16]. Similar to the proposed method implemented for joint supervision, several training iterations were run for the covariance neural network in order to minimize the error caused by the random initialization of the parameters, and the parameters of the neural network were tuned with N_p steps ahead.

A. Benchmark

The proposed method was evaluated based on the modified Chen series [17]. This modified database has state dependent noise:

$$\begin{aligned} y(k) = {} & \left(0.8 - 0.5\exp\left(-y^2(k-1)\right)\right)y(k-1) - \\ & \left(0.3 + 0.9\exp\left(-y^2(k-1)\right)\right)y(k-2) + \\ & u(k-1) + 0.2\,u(k-2) + \\ & 0.1\,u(k-1)u(k-2) + e(k) \end{aligned} \tag{25}$$

where the system noise, $e(k) = 0.5\exp(-y(k-1)^2)\,\beta(k)$, depends on the previous state of the output model, and $\beta(k)$ is white noise. Given the properties of the exponential function, the noise will be greater when $y(k-1)$ is close to zero.

The training, validation, and test sets are separated in proportions of 55, 25, and 20 percent respectively. The validation set is used in the selection of model structure stage. Fig. 2. shows 400 points of the input $(u(k))$, the noise and the output value $(y(k))$ of the modified series. The regressors $u(k-1), u(k-2), y(k-1)$ and $y(k-2)$ were selected based on eq. (25), and the hidden neuron numbers were chosen as the ones that minimize the RMSE in the validation dataset. After that, the neural network on which the covariance method is based used 10 neurons in the hidden layer, while the network trained with the proposed method had 14 neurons in the hidden layer. The neural networks were trained on the training set, and early stopping was used to prevent overfitting.

Fig. 3. shows the behavior of the $PICP$ and $PINAW$ when the parameter λ is increased (see eqs. (14) and (15)). While the $PINAW$ corresponding to the interval width increases at a constant rate as the parameter λ becomes larger, the $PICP$ which measures the coverage probability tends to stabilize at the desired coverage probability.

Authorized licensed use limited to: National Library of Technology. Downloaded on June 11,2021 at 22:24:13 UTC from IEEE Xplore. Restrictions apply.

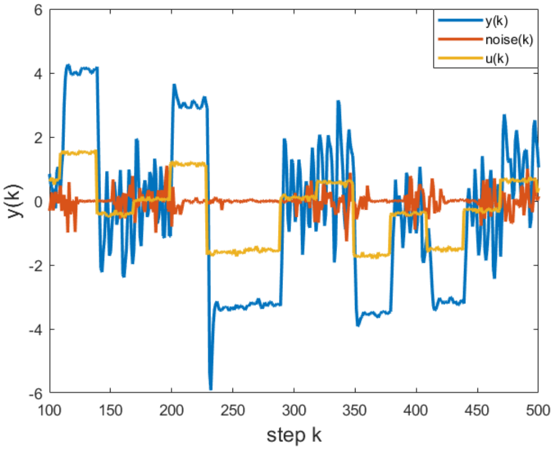

Fig. 2. Modified Chen Series.

In this study, the desired $PICP$ was defined at 90%. For the case presented in Fig. 3, the $PINAW$ with which the desired coverage probability is achieved corresponds to around 20%. The method proposed possesses a diminishing returns nature, meaning that the best results in terms of the $PICP$ and $PINAW$ tradeoff will be obtained before the $PICP$ curve stabilizes.

Table I shows the root mean square error *(RMSE)* and the mean absolute error *(MAE)* as metrics for evaluating the quality of the model when a single trajectory is provided as the output. Additionally, the prediction interval normalized average width ($PINAW$) and the prediction interval coverage probability ($PICP$) are included as the performance of the interval modeling. In this paper, test data set is used for evaluating the models at different prediction horizons. Finally, the training time is included in the table.

The $RMSE$ and MAE shown in Table I, reveal that the proposed method has lower values than the covariance method. The reason for this is that using a total loss function integrates the uncertainty into the model, and since the joint supervision network shares parameters for the interval and the crisp predictions, the network is able to extract more information from the same data. Supervision by total loss functions with multiple objectives can lead the solution away from local minima, and also improve generalization of the algorithm as reported in [12]. However, the prediction error increases for a larger prediction horizon, because the models face different levels of uncertainty at different steps ahead.

As can be seen in Table I, while both methods are able to maintain the coverage probability at about 90% (the desired $PICP$), the joint supervision method has the benefit of providing a narrower interval (see $PINAW$) for all the prediction steps ahead, which in turn, provides more information about the uncertainty phenomena modeled. This difference becomes apparent when comparing Fig. 4. with Fig. 5. These figures correspond to the sixteen-step ahead prediction interval of both models, tuned with 90% coverage probability, using a rolling horizon strategy. While the covariance methods maintain a constant width for the interval, the proposed method achieves a narrower interval in states with little noise, and an interval length of similar width to that of the covariance method in states with high noise.

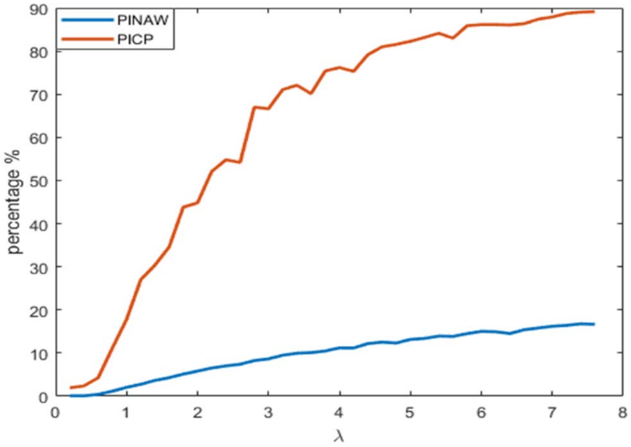

Fig. 3. PICP and PINAW According to λ.

TABLE I. PERFORMANCE METRICS

Prediction Horizon	Performance indexes	Neural Network Models	
		Covariance	Joint Supervision
One step ahead	RMSE (kW)	0.2675	0.2583
	MAE (kW)	0.1733	0.1601
	PINAW (%)	7.12	6.57
	PICP (%)	88.94	89.54
	Training time (sec)	42	72
Eight steps ahead	RMSE (kW)	0.6319	0.6227
	MAE (kW)	0.4586	0.4549
	PINAW (%)	18.47	15.75
	PICP (%)	89.35	91.71
	Training time (sec)	46	84
Sixteen steps ahead	RMSE (kW)	0.6470	0.6554
	MAE (kW)	0.4702	0.4855
	PINAW (%)	19.35	17.47
	PICP (%)	89.91	90.37
	Training time (sec)	31	69

This can be explained by the covariance method assuming normally distributed noise, since joint supervision does not make any assumptions on the distribution of the noise, which makes it an ideal candidate to model state dependent noise such as that found in the modified Chen series.

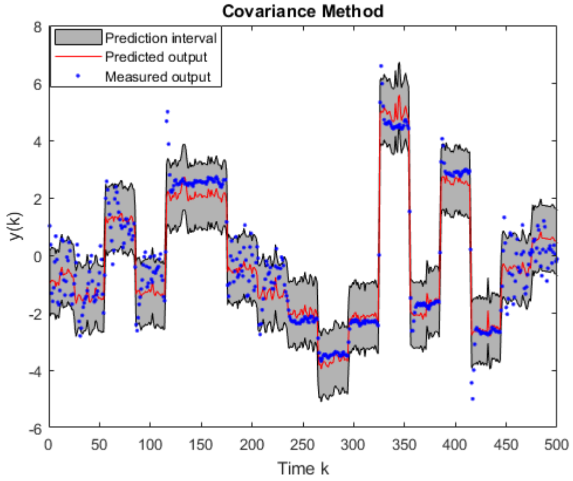

Fig. 4. Sixteen-step-ahead Prediction Interval based on Covariance.

Authorized licensed use limited to: National Library of Technology. Downloaded on June 11,2021 at 22:24:13 UTC from IEEE Xplore. Restrictions apply.

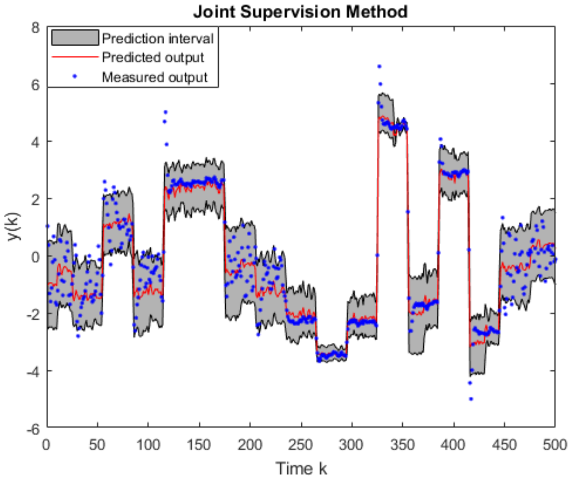

Fig. 5. Sixteen-step-ahead Prediction Interval based on Joint Supervision.

Finally, the parameter numbers to tune are $L(p+1)+l(L+1)$, where L is the number of neurons in the hidden layer, p is the input number, and l are the output units. Thus, 61 and 115 parameters were found for the covariance method and the proposed method, respectively. Therefore, as shown in Table I, the training time for the joint supervision method is higher than that of the covariance method. Additionally, iterations were added in the proposed method when the desired *PICP* was reached, keeping the λ value constant.

B. *Application for Load Forecasting*

Load forecasting is an important issue in the operation of microgrids. In order to use robust model predictive control, it is desirable to predict a reasonable interval in which the following data point will fall [18]. In this work, the proposed method was used to represent the future load uncertainty of 20 residential dwellings from a town in UK, for various prediction horizons.

The available load data was collected for the summer season of the year 2008, in which a period of 93 days was used, divided into 52 days of training, 23 days for validation, and 18 days for collecting test data. The maximum electric load is 29.54 KW with a sampling time of 15 minutes.

Regarding the model structure of the neural network, various neural networks were evaluated modifying the numbers of hidden units. Then, for each proposed structure, the relevant input variables were selected by sensitivity analysis [19]. The best structure was defined when the validation error was either increased or stable compared with the training error when the structure of the model increased in complexity. Finally, for the best model, all the parameters were calculated using the training data set. The optimal structure, consisting of 11 regressors and 15 neurons for the proposed method, and 12 neurons for the covariance method in the hidden layer, was obtained:

$$\begin{aligned}\hat{y}(k) = f^{NN}(&y(k-1), y(k-2), y(k-3),\\ &y(k-4), y(k-5), y(k-91),\\ &y(k-92), y(k-95), y(k-96),\\ &y(k-97), y(k-100))\end{aligned} \quad (26)$$

where $\hat{y}(k)$ is the electric load output prediction, and $y(k-100)$ is the input corresponding to one previous day and one previous hour. Specifically, in this study, 15- minute, 1-hour, and 1-day forward prediction horizons were considered. For each model (covariance and joint supervision) and prediction horizon, the performance indices are computed as is shown in Table II.

As can be seen in Table II, the Joint Supervision Method achieves better *RMSE* and *MAE* indexes for all of the prediction horizons, meaning that the crisp prediction made by the neural network trained under joint supervision is more accurate than the prediction made by a network using only the classical loss function.

Fig. 6 and Fig. 7 show one-day ahead prediction intervals of both models, tuned with 90% coverage probability, using a rolling horizon strategy, i.e., the forecast for one-day ahead was performed every 15 minutes.

The results (see Table II) suggest that the prediction interval was tuned to appropriately 90% of the desired coverage probability, and the interval width corresponded to the minimum width for characterizing the uncertainty of the load. The width values are due to the high variability of the data in this case study. Since the information level given by an interval has a direct relationship to its width, the proposed method of interval generation gives better information than the covariance method.

In particular, the crisp prediction for the states with low noise is much better for the joint supervision network than for the classical network, while the crisp predictions for states with high noise are very similar for both networks. For this case study, 157 parameters were found for the covariance method and 213 for the proposed method. Therefore, the highest training time was with the Joint Supervision Method, as shown in Table II.

TABLE II. PERFORMANCE METRICS

Prediction Horizon	**Performance indexes**	**Neural Network Models**	
		Covariance	**Joint Supervision**
One step ahead	RMSE (kW)	2.4503	2.1739
	MAE (kW)	1.8522	1.6040
	PINAW (%)	43.94	39.56
	PICP (%)	90.29	91.09
	Training time (sec)	9	21
One hour ahead	RMSE (kW)	2.4656	2.3918
	MAE (kW)	1.8474	1.7590
	PINAW (%)	45.67	43.89
	PICP (%)	91.38	92.92
	Training time (sec)	10	18
One day ahead	RMSE (kW)	2.5558	2.3836
	MAE (kW)	1.9766	1.7965
	PINAW (%)	48.78	47.90
	PICP (%)	92.36	92.91
	Training time (sec)	9	23

Authorized licensed use limited to: National Library of Technology. Downloaded on June 11,2021 at 22:24:13 UTC from IEEE Xplore. Restrictions apply.

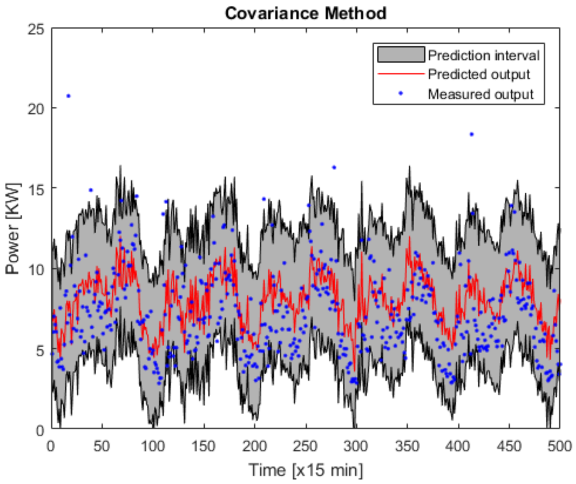

Fig. 6. One-day-ahead Prediction Interval based on Covariance.

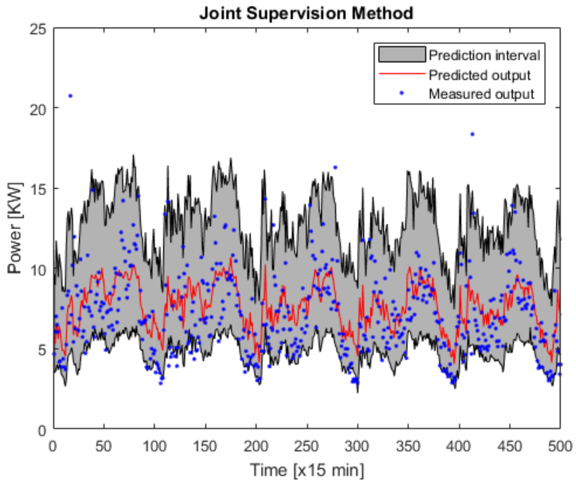

Fig. 7. One-day-ahead Prediction Interval based on Joint Supervision.

V. Conclusions

In this paper, a Prediction Interval Model based on a Joint Supervision Method is proposed. The model is used to construct a prediction interval that includes a representation of the uncertainty. A total loss function that is composed of both a classical loss function and a new interval loss function is used for training the neural network prediction interval. Thus, the parameters of the models are determined with the objective of minimizing the prediction error and the width of the prediction interval, while maintaining the desired coverage probability.

The proposed model was compared with a covariance prediction interval method. Based on the results, it was found that in all of the cases, the proposed method generated the largest amount of information in terms of the relationship between the width of the interval and the coverage probability. Additionally, the proposed method is capable of predicting state dependent noise accurately as is shown with the benchmark used for evaluation. It was also found that joint supervision by proposed total loss function can improve the crisp results of the neural network. Much of this success is due to the derivation of the total loss function that takes the coverage probability, as well as the width of the interval, into account, and that is well suited for backpropagation algorithm. The final algorithm unifies the generation of the prediction interval and the crisp prediction in a solution for modeling uncertain data. Furthermore, the proposed method benefits from all the optimization found for neural networks, making the training very efficient. Future work can include integrating the interval models in the actual operation of a microgrid, for instance, in the design of robust energy management strategies.

Acknowledgment

This study was partially supported by the Complex Engineering Systems Institute (CONICYT – PIA – FB0816), the Solar Energy Research Center SERC-Chile (CONICYT/FONDAP/ Project under Grant 15110019) and FONDECYT Chile Grant Nr.1170683 "Robust Distributed Predictive Control Strategies for the Coordination of Hybrid AC and DC Microgrids". The authors would like to thank Prof. Mark Sumner, University of Nottingham UK, for kindly sharing with us information related to the consumption from a town in UK. Luis G. Marín has also been supported by a Ph.D. scholarship from CONICYT-PCHA/Doctorado Nacional para extranjeros/2014-63140093.

References

[1] A. Ghanbari, E. Hadavandi, and S. Abbasian-Naghneh, "Comparison of artificial intelligence based techniques for short term load forecasting," in *2010 Third International Conference on Business Intelligence and Financial Engineering*, 2010, pp. 6–10.

[2] J. L Patel and R. Goyal, "Applications of Artificial Neural Networks in Medical Science," *Curr. Clin. Pharmacol.*, vol. 2, pp. 217–226, 2007.

[3] M. Norgaard, O. Ravn, N. K. Poulsen, and L. K. Hansen, *Neuronal Networks for Modelling and Control of Dynamic Systems*. London: Springer-Verlag, 2000.

[4] J. Vilar, G. Aneiros, and P. Raña, "Prediction Intervals for Electricity Demand and Price Using Functional Data," *Int. J. Electr. Power Energy Syst.*, vol. 96, pp. 457–472, 2018.

[5] T. Heskes, "Practical confidence and prediction intervals," *Adv. Neural Inf. Process. Syst.*, vol. 9, no., pp. 176–182, 1997.

[6] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Lower upper bound estimation method for construction of neural network-based prediction intervals," *IEEE Trans. Neural Networks*, vol. 22, no. 3, pp. 337–346, 2011.

[7] K. Li, R. Wang, H. Lei, T. Zhang, Y. Liu, and X. Zheng, "Interval Prediction of Solar Power Using an Improved Bootstrap Method," *Sol. Energy*, vol. 159, pp. 97–112, 2018.

[8] I. M. Galván, J. M. Valls, A. Cervantes, and R. Aler, "Multi-Objective Evolutionary Optimization of Prediction Intervals for Solar Energy Forecasting with Neural Networks," *Inf. Sci. (Ny).*, vol. 418–419, pp. 363–382, 2017.

[9] Y. A. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, "Efficient BackProp," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7700, pp. 9–48, 2012.

[10] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica*, vol. 31, pp. 249–268, 2007.

[11] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A Discriminative Feature Learning Approach for Deep Face Recognition," *Springer Int.*, pp. 499–515, 2016.

[12] C. Xu *et al.*, "Multi-Loss Regularized Deep Neural Network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 12, pp. 2273–2283, 2016.

[13] 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 *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2016, pp. 328–335.

Authorized licensed use limited to: National Library of Technology. Downloaded on June 11,2021 at 22:24:13 UTC from IEEE Xplore. Restrictions apply.

[14] N. A. Shrivastava, K. Lohia, and B. K. Panigrahi, “A multiobjective framework for wind speed prediction interval forecasts,” *Renew. Energy*, vol. 87, no. Part 2, pp. 903–910, 2016.

[15] A. C. Rencher and G. B. Schaalje, *Linear models in statistics*, 2 edition. USA: John Wiley & Sons, Inc, 2008.

[16] D. Sáez, F. Ávila, D. Olivares, C. Cañizares, and L. G. Marín, “Fuzzy prediction interval models for forecasting renewable resources and loads in microgrids,” *IEEE Trans. Smart Grid*, vol. 6, no. 2, pp. 548–556, 2015.

[17] S. Chen, S. A. Billings, and P. M. Grant, “Non-linear system identification using neural networks,” *Int. J. Control*, vol. 51, no. 6, pp. 1191–1214, 1990.

[18] F. Veltman, L. G. Marín, D. Sáez, L. Gutierrez, and A. Nuñez, “Prediction interval modelling tuned by an improved teaching learning algorithm applied to load forecasting in microgrids,” in *2015 IEEE Symposium Series on Computational Intelligence*, 2015, pp. 651–658.

[19] D. Sáez and R. Zuñiga, “Cluster optimization for takagi & sugeno fuzzy models and its application to a combined cycle power plant boiler,” in *Proceedings of the 2004 American Control Conference*, 2004, vol. 2, pp. 1776–1781.

Authorized licensed use limited to: National Library of Technology. Downloaded on June 11,2021 at 22:24:13 UTC from IEEE Xplore. Restrictions apply.