Received September 15, 2020, accepted September 23, 2020, date of publication September 30, 2020,
date of current version October 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3027977

Hybrid Bidirectional LSTM Model for Short-Term Wind Speed Interval Prediction

ADNAN SAEED[1], CHAOSHUN LI[1], (Member, IEEE), MOHD DANISH[2], SAEED RUBAIEE[2,3], GENG TANG[1], ZHENHAO GAN[1], AND ANAS AHMED[3]

[1]School of Hydropower and Information Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
[2]Department of Mechanical and Materials Engineering, University of Jeddah, Jeddah 21589, Saudi Arabia
[3]Department of Industrial and Systems Engineering, University of Jeddah, Jeddah 21589, Saudi Arabia

Corresponding author: Chaoshun Li (csli@hust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 51879111 and Grant 51679095, in part by the Frontier Special Project of Applied Foundation of Wuhan City under Grant 2018010401011269, in part by the Hubei Provincial Natural Science Foundation of China under Grant 2019CFA068, and in part by the Fundamental Research Funds for the Central Universities under Grant 2019kfyRCPY072.

ABSTRACT Wind speed interval prediction is gaining importance in optimal planning and operation of power systems. However, the unpredictable characteristics of wind energy makes quality forecasting an arduous task. In this paper, we propose a novel hybrid model for wind speed interval prediction using an autoencoder and a bidirectional long short term memory neural network. The autoencoder initially extracts important unseen features from the wind speed data. The artificially generated features are utilized as input to the bidirectional long short term memory neural network to generate the prediction intervals. We also demonstrate that for time series prediction tasks, feature extraction through autoencoder is more effective than making deep residual networks. In our experiments which involve eight cases distributed among two wind fields, the proposed method is able to generate narrow prediction intervals with high prediction interval coverage and achieve an improvement of 39% in coverage width criterion over the traditional models.

INDEX TERMS Wind speed prediction, bidirectional LSTM, autoencoder, residual LSTM, interval prediction.

NOMENCLATURE

PI	prediction interval
WSIP	wind speed interval prediction
LSTM	long short term memory
BLSTM	bidirectional LSTM
LUBE	lower upper bound estimation
PICP	PI coverage probability
PINRW	PI normalized root-mean-square width
PINAW	PI normalized average width
PINC	PI normal confidence
CWC	coverage width criterion
ReLU	rectified linear unit
NREL	the National Renewable Energy Laboratory

I. INTRODUCTION

Wind energy is a clean and abundantly available alternate sources of energy. However, integration of wind energy in power systems require prior knowledge about the behavior and nature of wind speed in the area. Over the last decade, numerous researchers have contributed a number of different techniques such as physical [1], [2], statistical [3], [4], as well as hybrid frameworks [5]–[7] to improve the accuracy of deterministic wind forecasts. However, deterministic forecasts or point forecasting can be of limited use if there is a large downside to an incorrect prediction like in scheduling and operation of energy systems. Indicating the uncertainty associated with each forecast is therefore more imperative over point forecasting.

One common way of communicating the uncertainty is to indicate the expected upper and lower bounds of the forecasts or in other words generating the prediction intervals (PIs). There exist several interval prediction methods in the literature like Delta method [8], Bayesian technique [9], Bootstrap [10], [11] and Fuzzy models [12]. Fuzzy models suffer from implementation difficulties whereas the traditional models mostly obtain PIs based on certain error distribution assumptions over point predictions.

The associate editor coordinating the review of this manuscript and approving it for publication was Long Wang.

Eliminating the need for any distributional assumption for the original data, lower upper bound estimation (LUBE) is proposed in [13]. The core idea is to generate PIs directly using a certain predictive model, which is trained by minimizing a PI-based cost function which assumes high-quality PIs to capture as many data points while keeping the interval width as narrow as possible. Because of its inherent lucidity, this method finds its implementation in a plethora of application focused works [14]–[18]. Majority of these works implement shallow machine learning models such as ANN, ELM, and SVM as predictive models. The insufficiency of these models has restricted the performance of LUBE method. Like SVMs are incapable in handling large amount of data, and ELM suffers from weak universal approximation property. Also, integration of these models require too many parameters to be tuned for the metaheuristic optimization mechanism adopted in the framework, making the computational cost too expensive.

Recent years have seen a rise of deep learning methods as a new research hotspot in machine learning [19]. In particular, RNNs with the capability of maintaining states between different inputs show advantage in handling time sequences, but suffer from problems like vanishing and exploding gradients. LSTM, an extension of RNN, solves the vanishing gradient problem by introducing memory cells with controlling gates and have been used extensively to simulate the time-series correlation [20], [21]. Further extensions to RNNs have recently become popular where the output is obtained by exploring not only the past but also the future context [22]. These networks connect the outputs from two separate hidden layers (which scan the input sequences in opposite directions) to the same output layer making them bidirectional RNNs. LSTMs can also benefit from this bidirectional modeling resulting in bidirectional LSTMs (BLSTMs).

Due to the intermittent and very volatile nature of wind power the data distribution assumptions seem dubitable, thus in view of the advantage of no distributional assumption of LUBE and the merits of bidirectional LSTM, how to establish an interval prediction model based on BLSTM as the predictive model in the LUBE framework would be interesting and challenging and has seldom been researched.

Traditionally, LSTMs are shallow across layers, instead obtaining their depths across time steps. Recent advancements in residual networks have unlocked the ability to use deep networks. With residual connections the network can directly learn an Identity function resulting in more effective learning [23]. Fascinated by the enticing popularity of these networks, researchers applied residual modeling to make deep residual LSTMs and report improvements in performance specially in the field of natural language processing [24], [25]. This inspires to examine the performance of residual LSTMs in time series prediction which is an area not yet documented. Will it work as a predictive model for complex time series prediction? A soft answer here is, maybe not. The reason based on our findings are discussed later in the paper. An interesting question then arises that why this remarkable idea of identity connection is not successful in time series prediction or is there any other way to utilize this idea? Another very popular network called autoencoder (AE) exists which basically learns an identity function. The fundamental idea in an autoencoder is to learn and reproduce almost the same input. This usually involves two stages: i) contracting the input data into a latent-space representation, and ii) reconstructing the output from this representation. In this process the latent representation learns the most salient features of the training data. This is very interesting as an autoencoder can be trained to preserve as much information as possible which can be utilized to make hybrid networks.

In the present paper, a hybrid model is proposed and integrated as a predictor network in the LUBE framework for improved interval predictions. The method involves a stacked BLSTM prediction model which generate PIs based on the artificial features extracted by a BLSTM autoencoder initially from the wind series data.

The main contributions of this paper are as follows: (1) a novel methodology to generate prediction intervals by integrating a deep neural network into the LUBE framework is proposed; (2) the concept of identity function, in learning the important features across the layers of a deep model is examined, and is shown that utilizing it as a preprocessor (autoencoder) is more effective over its use as a residual, for time series prediction; (3) specifically, a hybrid model combining an autoencoder and BLSTM is presented to perform high-quality PI for wind speed interval prediction (WSIP).

The rest of the paper is organized as follows: Section II gives a brief introduction about the PI quality and evaluation indices as well as the structures of BLSTM, residual networks and autoencoders. Section III proposes the hybrid BLSTM model. Section V lists the conclusions obtained from the discussions and comparisons of several numerical examples presented in section IV.

II. THEORITICAL BACKGROUND

A. BIDIRECTIONAL LSTM

A typical LSTM cell is shown in Fig. 1 (a) which use the following transition equations:

$$f_n = \sigma\left(W_f x_n + U_f h_{n-1} + b_f\right) \tag{1}$$
$$i_n = \sigma\left(W_i x_n + U_i h_{n-1} + b_i\right) \tag{2}$$
$$o_n = \sigma\left(W_o x_n + U_o h_{n-1} + b_f\right) \tag{3}$$
$$\tilde{c}_n = tanh\left(W_c x_n + U_c h_{n-1} + b_c\right) \tag{4}$$

which leads to

$$c_n = \sigma\left(\tilde{c}_n \otimes i_n + c_{n-1} \otimes f_n\right) \tag{5}$$
$$h_n = o_n \otimes tanh\left(c_n\right) \tag{6}$$

wherein x_n, c_n, and h_n represents the input, cell state and the output of the current LSTM respectively whereas h_{n-1} is the mean output of the last LSTM. f_n, i_n, o_n, and $\tilde{c}_n$ represents the forget gate, input gate, output gate, and the cell candidate value. W and U are weight matrices while b is the bias. $\otimes$ denotes the Hadamard product.

The concept of memory cells with controlling gates helps LSTMs to achieve considerable success in removing the problems of long term dependency and vanishing gradients associated with RNNs [26]. The learning process through backpropagation estimates the weights based on which the cells either store or delete the data.

Another variation to RNNs is proposed in [22] wherein a single RNN layer consists of two RNN blocks processing temporal information simultaneously in two opposite directions. The final output at each single time instance is a combination of the outputs of each RNN block. The bidirectional structure can be applied to LSTM to make a bidirectional LSTM (BLSTM) as shown in Fig.1 (b).

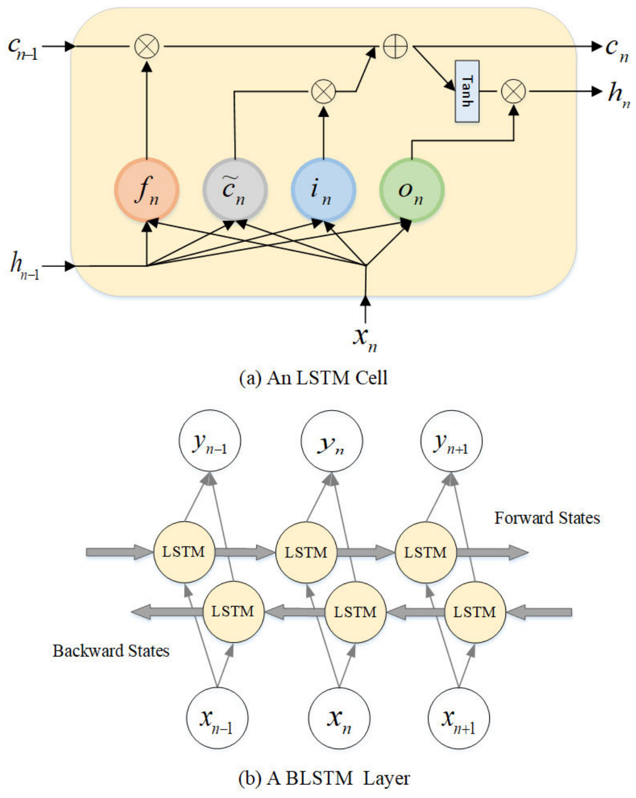


FIGURE 1. General structure of LSTM cell and BLSTM layer.

B. RESIDUAL NETWORKS

During back-propagation in a deep network, repeated multiplication may make the gradient infinitively small due to which deep neural networks are hard to train. Residual networks [23] introduce a so-called ''identity shortcut connection'' between the input and output as shown in Fig. 2 which ensures unimpeded flow of original information layer to layer throughout the network. Consequently, gradients are computed with the original input taking into consideration and would never vanish no matter how deep we go. But why this is called as residual network? Mathematically, for a neural network layer whose input is x and output is $F(x)$. The difference or the residual between this is

$$R(x) = F(x) - x \tag{7}$$

Rearranging it we get,

$$F(x) = R(x) + x \tag{8}$$

The residual block is overall trying to learn the true output, $F(x)$ but as can be seen from Fig. 2 (a), that since there is an identity connection coming due to x, the layer is actually trying to learn the residual, $R(x)$. Hence, the added layer just need to learn the features on top of already available input.

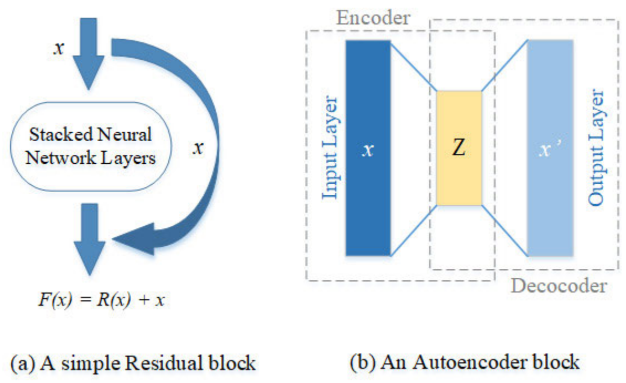


FIGURE 2. General structures of residual block and autoencoders.

C. AUTOENCODERS

Autoencoders (AE) are a category of neural networks for which the output resembles the input as closely as possible. An AE consists of two parts: an encoder ϕ that maps the input data x to latent-space representation Z by combining the data's most important features, and a decoder ψ that aims to reconstruct the input x' from the latent space representation as shown in Fig. 2 (b). These transitions can be represented mathematically by the standard neural network function passed through an activation function.

$$\phi : X \rightarrow Z : x \mapsto \phi(x) = \sigma(Wx + b) := z \tag{9}$$

$$\psi : Z \rightarrow X : z \mapsto \psi(z) = \sigma\left(\tilde{W}z + \tilde{b}\right) := x' \tag{10}$$

There are a few reasons why AEs may be useful: 1) Using latent feature representations may enhance model performance, 2) Dimensionality reduction can reduce training time.

D. EVALUATION INDICES AND QUALITY OF PI

An effective forecasting model must construct sharp and reliable PIs. Number of target values covered or coverage probability is the commonly used measure for reliability whereas width of PI represents its sharpness. Mathematically, these two measures defined through indices like PI coverage Probability (*PICP*) and PI normalized average width (*PINAW*) or PI normalized root-mean-square width (*PINRW*) [18]:

$$PICP = \frac{1}{n}\sum\nolimits_{i=1}^{n} C_i, \quad C_i = \begin{cases} 1, & y_i \in [L_i, U_i] \\ 0, & y_i \notin [L_i, U_i] \end{cases} \tag{11}$$

$$PINAW = \frac{1}{nR}\sum\nolimits_{i=1}^{n} (U_i - L_i) \tag{12}$$

$$PINRW = \frac{1}{R}\sqrt{\frac{1}{n}\sum\nolimits_{i=1}^{n}(U_i - L_i)^2} \tag{13}$$

where U_i and L_i are the upper and lower bounds of the i^{th} PI respectively and R is the range of the minimum and maximum of the time-series, n is the number of data pints. The format for *PINRW* is more close to mean square error which enhance the training performance by magnifying the big error terms [18]. Therefore, we chose *PINRW* as our width assessment index.

The wider the intervals the higher will be the coverage. So, higher coverage may lead to a higher *PINAW* or reduced sharpness and vice-versa. Thus the two indices conflict with each other and are insufficient to independently reflect the quality of the PI. A popular index called coverage width criterion (*CWC*) which makes a comprehensive balance between *PICP* and *PINAW* is used as a measure to examine the quality of PI. Because of its importance the definition of *CWC* keeps evolving in literature. We used the definition of *CWC* as

$$CWC = \begin{cases} \beta \cdot PINAW, & PICP > \mu \\ (\alpha + \beta * PINAW) & \\ \cdot (1 + exp(-\eta(PICP - \mu))), & PICP < \mu \end{cases} \tag{14}$$

where α, β, η, and μ are hyperparameters that govern the measurement of *CWC* index. β is used to linearly increase the influence of *PINAW*, α avoids the *CWC* from vanishing once *PINAW* becomes zero, η is the penalty factor for unqualified *PICP* and μ is a PI normal confidence (PINC) that needs to meet the *PICP*. This definition of *CWC* is very effective because the multiplication operation between *PINAW* and *PICP* solves the problem of *PINAW* losing control of *CWC* [27]. Thus, *PICP* is given more weight when the *PICP* does not meet the required PINC. Once PINC is satisfied, the width of PI will have more influence on *CWC*.

III. NOVEL BLSTM BASED HYBRID AUTOENCODER-BLSTM MODEL FOR INTERVAL PREDICTION

A. FRAMEWORK OF H-BLSTM MODEL

Using the deep learning fundamentals, a novel multi-layer neural network for WSIP is proposed. For simplicity we call it hybrid BLSTM (H-BLSTM) model.

The forecasting framework is easy to elucidate as illustrated in Fig.3. We start with training a BLSTM AE on the initial wind speed series; next we extract the encoder and utilize it as a features creator. The prediction model then acts on these artificially generated features to generate the prediction intervals. The specific training procedure of the proposed model is shown in Fig. 4.

1) BLSTM AUTOENCODER

Usually, the input data may contain some redundant or correlated features which may result in wasted processing time and overfitting in the model. It is thus ideal to only include the features we need. AE comes in handy in extracting those important unknown features. An AE consists of two parts: an encoder that maps the input data to a latent-space representation by combining the data's most important features, and a decoder that aims to reconstruct the input from the latent

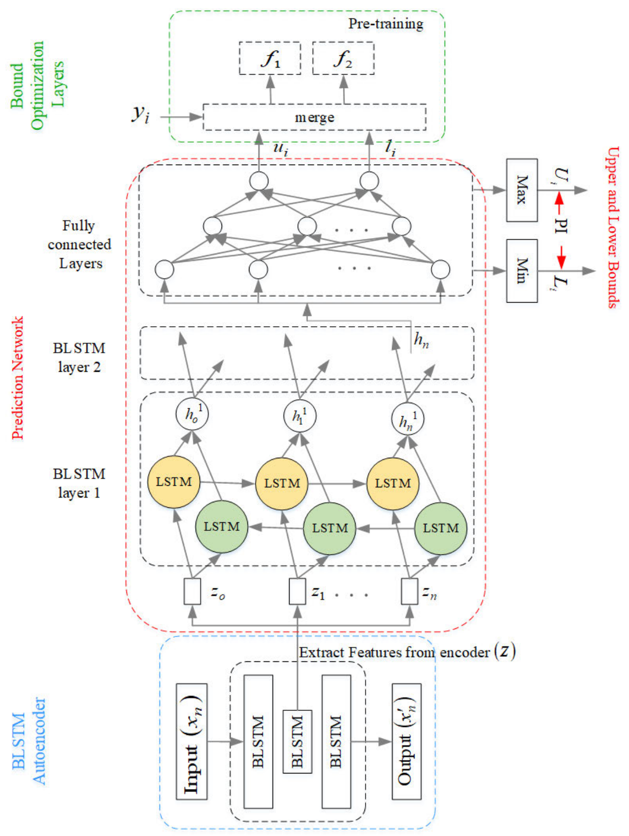


FIGURE 3. Framework of the interval prediction model.

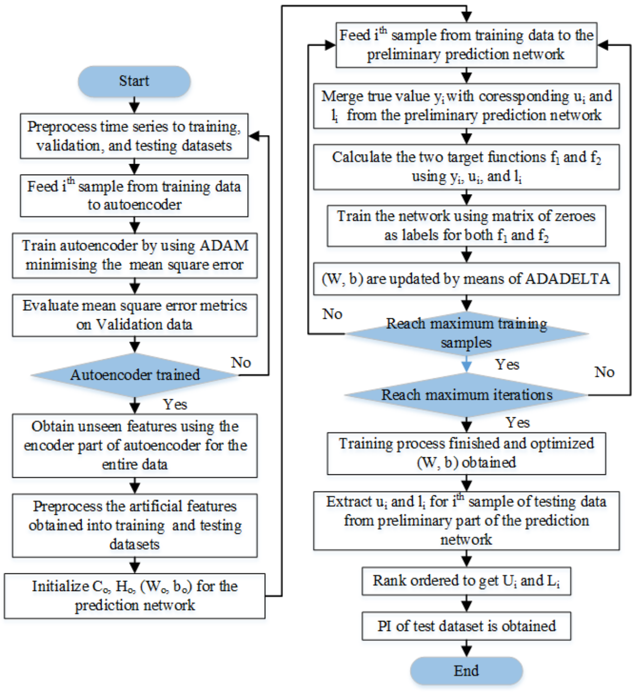

Start
Preprocess time series to training, validation, and testing datasets
Feed i^th sample from training data to autoencoder
Train autoencoder by using ADAM minimising the mean square error
Evaluate mean square error metrics on Validation data
Autoencoder trained
No
Yes
Obtain unseen features using the encoder part of autoencoder for the entire data
Preprocess the artificial features obtained into training and testing datasets
Initialize C_o, H_o, (W_o, b_o) for the prediction network
Feed i^th sample from training data to the preliminary prediction network
Merge true value y_i with coressponding u_i and l_i from the preliminary prediction network
Calculate the two target functions f_1 and f_2 using y_i, u_i, and l_i
Train the network using matrix of zeroes as labels for both f_1 and f_2
(W, b) are updated by means of ADADELTA
No
Reach maximum training samples
Yes
No
Reach maximum iterations
Yes
Training process finished and optimized (W, b) obtained
Extract u_i and l_i for i^th sample of testing data from preliminary part of the prediction network
Rank ordered to get U_i and L_i
PI of test dataset is obtained
End

FIGURE 4. Flow chart of the H-BLSTM model.

space representation. If the "reconstruction" of data is very accurate, the latent space representation can be used as input into another model. Our AE consists of a BLSTM encoder as well as a BLSTM decoder with a BLSTM bottleneck (hidden layer). The decoder is followed by a Time-Distributed output layer. Taking in the train data x at the encoder input the AE reconstructs the input $x^{'}$ at the output of decoder by minimizing the mean square error $1/n\left(x-x^{'}\right)^2$ as the reconstruction loss utilizing Adam [28], as the optimizer algorithm. Once trained the encoder part is removed and is fed in full wind speed data (train + test) to generate the features dataset z. The generated dataset is then divided into train and test data (z_{train}, z_{test}) as earlier and the train data is fed to the prediction model.

2) PREDICTION MODEL

In time series prediction, the temporal characteristics are really important. LSTMs learn by back propagating through time allowing the gradient of weight updates to be estimated from all observations in the sequence. Based on the weights, the memory cells with controlling gates either store or delete the data. Thus LSTMs can disregard the error contribution from prior time-steps in gradient estimates, and can give more weight to the input of most recent moments, while simultaneously keeping important information in the memory cells thus removing problems of long term dependency and vanishing gradients. BLSTM which combines another LSTM that move backward through the sequence is therefore capable to learn a representation that depends on both the past and the future but is most sensitive to the input values around that particular time. Thus the structural characteristics of BLSTM layer suits well at this point. So our prediction model starts with a BLSTM which is designed as multiple input on time scale and outputs a sequence of vectors. These vectors are passed on to a subsequent BLSTM layer which gives single output from last cell. This hierarchy of layers enables more complex representation of the time-series data, capturing information at different scales. Thus, the stacked BLSTM layers transfer the input $z = \{z_1, z_2, \ldots, z_n\}$ into a high dimensional vector h_n, which is full of temporal feature information. Feeding on this feature vector the subsequent fully connected layers, two in number, further apply nonlinear computations (ReLU activation) layer by layer transforming it to a dense representation which is then finally converted into two bound output by the output layer. ReLU [29] as defined in Eq. (15) gives several advantages over sigmoid and Tanh activation functions.

$$ReLU = \begin{cases} x, & x \geq 0 \\ 0, & x \leq 0 \end{cases} \tag{15}$$

In the regime $x > 0$ the gradient has a constant value which helps reducing the likelihood of the vanishing gradient problem as well as results in faster learning whereas in $x < 0$ regime it results in sparse representations. Sparse representations seem to be more beneficial than dense representations. Initializing the weights of these layers with He initialization [30], further ensures a faster and more efficient gradient descent. For the output layer, just a linear activation function can receive a good result in fitting the prediction label. We term this part of network as preliminary prediction network. The bounds obtained from this network are optimized by using the true value and two target functions. In the end a rank-ordered terminal sorts the output to correspond the upper and lower bounds.

B. OPTIMIZATION STRATEGY

The prediction network gives two outputs. We call these as the preliminary upper (u_i) and lower (l_i) bounds of the PI.

$$\begin{cases} u_i = \psi_u\left((W, b)\right), & z_i \\ l_i = \psi_l\left((W, b)\right), & z_i \end{cases} \tag{16}$$

where ψ_u and ψ_l are functions of relationship among the layers of the prediction network, (W, b) contains the weights and biases of the various layers in the network. For accurate predictions these bounds must be optimized. A neural network optimizes the prediction by minimizing a loss function which is generally the error between the predictions and the training labels. However, unlike point prediction, interval prediction lacks in the available labels and require a way of constructing the training labels artificially. Optimal PIs should possess high coverage probability while keeping the PI width as narrow as possible. Based on these two objectives, two target functions f_1, and f_2 are specifically designed as described in (17) and (18) respectively. If the observed value is in PI, it is expected to be closer to the mid-point of PI while demanding a penalty on the value of the target function once it escapes from the PI. The farther it flees the greater the penalty. At the same time, however, a narrow PI width is expected.

$$f_1(W, b) = k_1 \cdot \left(\left|y_i - (u_i + l_i)/2\right| + \lambda_1 \cdot \gamma \cdot d\right) \tag{17}$$

$$f_2(W, b) = k_2 \cdot \left((u_i - l_i) + \lambda_2 \cdot \gamma \cdot d\right) \tag{18}$$

where k_1, and k_2 are weights of f_1, and f_2. These two determine the importance between hit rate and width of PI. y_i is the observed value of the i^{th} sample, λ_1 and λ_2 are the penalty coefficients, γ is the step function given by Eq.(19) and finally d as defined in Eq. (20) measures the distance between y_i and boundary of u_i and l_i, when y_i escapes PI.

$$\gamma = \begin{cases} 0, y_i \in [l_i, u_i] \\ 1, y_i \notin [l_i, u_i] \end{cases} \tag{19}$$

$$d = \left|y_i - \frac{u_i + l_i}{2}\right| - \left|\frac{u_i - l_i}{2}\right| \tag{20}$$

We incorporated these two target functions as succeeding layers to the preliminary prediction network, shown as bound optimization layers in Fig. 3. So now our network makes two predictions namely the distance of the observed value from the center of the bound, and the bound. To train the network we now need labels for these predictions. In ideal scenario the observed value should lie at the midpoint of the preliminary PI. Also, in limiting case of fully accurate prediction the bound should be so narrow that it approaches zero. Thus, the neural network is trained by using a matrix of zeros as the training labels for both f_1, and f_2. Adadelta [28] a gradient based optimization algorithm is employed to train the prediction model. With its update rule adadelta eliminates the need to set a default learning rate thus accelerate convergence. Once trained the bound prediction network is removed and used to predict the optimal bounds U_i, and L_i of the PIs.

The hyperparameters k_1, k_2, λ_1, and λ_2 hugely influence the construction of the optimal bounds and thus finding the best values for these hyperparameters is very significant.

Hyperparameter optimization is generally represented as

$$\theta^* = \underset{\theta \in \chi}{\arg\min}\, g(\theta) \tag{21}$$

where $g(\theta)$ represents an objective score to minimize such as mean square error evaluated on the validation set; θ^* is the set of hyperparameters that yields the lowest score, and θ varies the domain χ. Evaluating the objective function of a deep model iteratively is computationally expensive. So we took a Bayesian approach to find these optimal hyperparameters. In this approach a probabilistic model (“surrogate” for the objective function) is formed mapping the hyperparameters to a probability of a score on the objective function, by tracking the results from previous evaluations. This is represented as $p(y|x)$ where y is the score. The surrogate is much easier to optimize than the objective function. We employ a python library Hyperopt [31] for using the Bayesian model based optimization. Quality forecast involves narrow intervals; thus we design our objective function to minimize the mean average error of the upper and lower bounds from the true value as described in Eq. (22).

$$Loss = \frac{1}{2}\left(\frac{\sum_{i=1}^{n}(u_i - y_i)}{n} + \frac{\sum_{i=1}^{n}(y_i - l_i)}{n}\right) \tag{22}$$

The hyperparameters obtained are listed in Table 1.

TABLE 1. Parameter settings of different models.

Model	Parameters	Quantity
QR	model confidence	0.95
ARIMA	model confidence	0.9
S-ANN	number of layers	3
	number of neurons in each layer	9, 8, 2
	population size of PSO	100
	maximum number of iterations of PSO	100
D-ANN	number of layers	5
	number of neurons in each layer	9, 8-6-4, 2
	Population size of PSO	100
	maximum number of iterations of PSO	100
H-BLSTM	BLSTM autoencoder	64, 32, 64
	number of BLSTM layers	2
	number of time steps in BLSTM layers	9
	dimension of z_n, h_n^1, h_n	64, 128, 32
	number of neurons in fully connected layers	64,16,2
	k_1, k_2, λ_1, and λ_2	5, 5,1, 4.5

An advantage of the optimization methodology adopted, is that by incorporating Eq.17 and Eq. 18, we are able to use mean square error as the loss function, which is differentiable and thus the method can be trained by classical neural network optimization technique called gradient descent. In contrast, the previous LUBE versions [13], [18] employ the commonly used measures of PI evaluation to train their model. To do so, they combine the two conflicting objectives of coverage probability and interval width into a single cost function called *CWC*, which is complex, highly nonlinear, and non-differentiable and thus difficult to optimize via classical local optimization problems.

IV. EXPERIMENTS ON REAL DATASETS

A. DATASET AND METHODOLOGIES

We choose wind speed datasets from two different wind fields collected in 2011 and 2012 from the National Renewable Energy Laboratory (NREL) website. The first dataset is from an offshore wind field (numbered 110197) located in Lake Huron, of which the longitude and latitude are −83.111816 and 45.07209 respectively. The second is an onshore wind field (numbered 71764) located in Pennsylvania with −78.3109 and 40.7147 as the longitude and latitude respectively.

For experiments we divide the wind series data into four subseries and evaluate the model performance in each of them. This is similar to performing a cross-validation to assure better generalization capabilities of the model [32]. Each subseries is extracted using a sliding window of 15 months, of which the first twelve months' data is considered for training the model while the remaining three months' data is kept for testing purpose, e.g., first subseries consists of data from January 2011 to December 2011 as train data and from January 2012 to March 2012 as test data, second subseries from April 2011 to March 2012 as train data and from April 2012 to June 2012 as test data and so on. The three months testing period is taken considering a particular season of a year generally consists of three months. Thus we term the four subseries as “spring”, “summer”, “autumn”, and “winter” based on the season names. Each subseries contains data points with a resolution of 30 min. So, in total we have eight cases, four for each location. In each case we perform multi-step predictions 3 hours ahead every 30 minutes.

We present the experimental results in four steps. The first two steps involve the evaluation of the structural components of the model, like the first step compares the effectiveness of BLSTM over LSTM whereas in the second step we check the advantage of AEs over residual RNN approach. The third step compares the proposed model with a variety of traditional approaches like LUBE based Shallow ANN (S-ANN) as well as Deep ANN (D-ANN) [33] methods and statistical approaches like Bootstrap ARIMA [11] and Quantile Regression (QR) [34]. Finally, in the fourth step the performance of the proposed model is compared for multi-step horizons. The selection of parameters is trial and error based for the proposed model, whereas the parameters of the other models were set as per the instructions of the references. The parameters of different models used in the comparative studies were presented in Table 1.

B. COMPARISON AND ANALYSIS

The indices of *PICP*, *PINRW*, and *CWC*, are used as the evaluation criterion to compare the performance of the different models. To counter the effect of the randomness associated with the initialization of the neural networks, each experiment is repeated 10 times and the mean values of indices are kept for comparison. For a WSIP model, a higher *PICP* and lower *PINRW* which lead to a lower *CWC* represent a high quality PI.

1) COMPARISON BETWEEN LSTM AND BLSTM

This subsection is to compare the performance of BLSTM with LSTM. For pure comparison, experiments are run considering solely the LSTM and BLSTM models without using the AE as in the proposed model. The parameters for both the models were also kept the same. The experiments contain eight cases, each of which contains a week's data with a resolution of 10 min, from the four seasons, for each of the two locations. In each of the eight cases the first five days were devoted to train data whereas the last two days of the week were considered as test data. The selection of small datasets is just to save the computational time, since the sole purpose here is to check the performance of BLSTM over LSTM. This dataset is selected only for the experiments of this subsection, while in the rest of the paper the data described in the previous section is used for all the experiments.

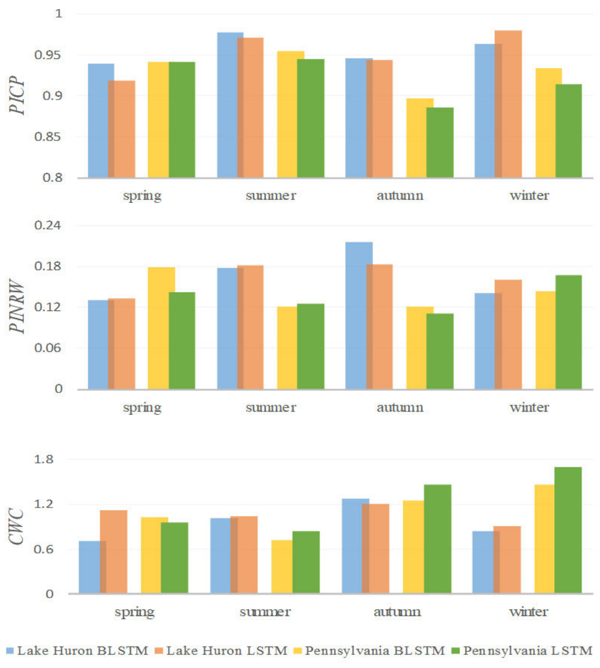


FIGURE 5. Performance indices comparison for LSTM and BLSTM models.

From Fig. 5 one can easily visualize that BLSTM is able to capture a higher *PICP* with lower *PINRW* thus a lower *CWC* in most of the cases. The BLSTM model is able to achieve an average improvement of 11% in *CWC* over LSTM. The model training time for BLSTM is a little higher in experiments than LSTM (not shown in the Fig. 5) which is justified for the two opposite LSTM layers in the BLSTM structure. The improvements in the performance indices at the expense of a little computational cost justify the selection of BLSTM for the proposed model.

2) THE IDENTITY FUNCTION: AUTOENCODER OR RESIDUAL CONNECTIONS

With effectiveness of BLSTM already established in previous subsection, we now attempt to enhance the quality of prediction by making BLSTM based deep neural network. As described in the introduction section that a residual or identity connection is highly successful in deep neural networks. Let us now attempt to answer the interesting question raised in the introduction section, that whether the remarkable success of the identity connections as enjoyed by the deep residual networks in computer vision tasks, can also be duplicated in time series predictions by making deep residual RNNs.

A satisfactory answer requires comparative experiments. For intuitive comparison we run the experiments on three different models. The difference among the three models is in the way they create the high dimensional vector h_n from the input time series. Once h_n is extracted the remaining structure of all the three models is the same i.e., fully connected layers followed by rank ordered terminal to generate PIs. The first model we call BLSTM model as it contains a single BLSTM layer followed by the rest of the structure. The second model which we call residual BLSTM (R-BLSTM) model is deeper across the layers and consists of four BLSTM layers with residual connection between them. Also, it is known that the last layer of BLSTM only outputs the sequence at the last time step, so to make a residual connection at the last layer we sliced the last element of the output sequence from the previous layer. All the four BLSTM layers have equivalent dimensionality of 64. This parameter setting gives two benefits; firstly, the input size matches the BLSTM output size therefore a residual connection is kept at the input also and secondly, it ensures that the number of parameters are almost the same as the total number of parameters including the AE and stacked BLSTM layers in the H-BLSTM model. Since making a bidirectional layer is also a kind of stacking up layers, we think the model possess sufficient depth to effectively utilize the advantages of residual connections. The datasets used are the same fifteen months' datasets as described earlier in the dataset and methodology section.

Table 2 summarize the performance indices obtained from the three models. The results clearly indicate that the residual connection model failed miserably in improving over the forecasts of the BLSTM model, instead a degraded performance is observed in some cases. In four cases it failed to maintain even the PINC of 0.9 and thus a higher *CWC* which implies lower quality forecasts. In contrast, the proposed

TABLE 2. Performance indices comparison of BLSTM, R-BLSTM and H-BLSTM models.

Mean Value of 10 Times		Off-shore wind field in Lake Huron				On-shore wind field in Pennsylvania			
		Spring	Summer	Autumn	Winter	Spring	Summer	Autumn	Winter
BLSTM	PICP	0.9539	0.9190	0.9186	0.9311	0.9521	0.9285	0.9157	0.9440
	PINRW	0.1389	0.1561	0.1700	0.1176	0.1420	0.1912	0.1888	0.1218
	CWC	0.7710	1.1520	1.1785	0.7603	0.7852	1.2164	1.2913	0.6814
R-BLSTM	PICP	0.9124	0.8683	0.8488	0.9171	0.9086	0.8869	0.8806	0.9241
	PINRW	0.1086	0.1375	0.1319	0.1118	0.1256	0.1713	0.1798	0.1124
	CWC	0.9468	2.0938	2.7278	0.8265	0.9884	2.0595	2.3576	0.8992
H-BLSTM	PICP	0.9440	0.9265	0.9251	0.9426	0.9211	0.9363	0.9388	0.9329
	PINRW	0.1192	0.1551	0.1667	0.1163	0.1215	0.1852	0.2088	0.1137
	CWC	0.6634	0.8473	0.8943	0.6377	0.6682	1.0244	1.1438	0.6320

TABLE 3. Performance indices comparison of five models by wind fields.

Mean Value of 10 Times		Off-shore wind field in Lake Huron				On-shore wind field in Pennsylvania			
		Spring	Summer	Autumn	Winter	Spring	Summer	Autumn	Winter
S-ANN	PICP	0.9870	0.9582	0.9517	0.9832	0.9766	0.9638	0.9539	0.9834
	PINRW	0.2321	0.2302	0.2101	0.2165	0.1988	0.2632	0.2487	0.1979
	CWC	1.3926	1.3814	1.2606	1.2991	1.1925	1.5790	1.4923	1.1877
D-ANN	PICP	0.9828	0.9206	0.9374	0.9807	0.9555	0.9406	0.9381	0.9710
	PINRW	0.2791	0.3755	0.2717	0.2299	0.4042	0.5328	0.4107	0.3236
	CWC	1.6744	2.2529	1.6303	1.3794	2.4250	3.1969	2.4644	1.9418
QR	PICP	0.9407	0.8727	0.8618	0.9289	0.9028	0.8983	0.8813	0.9379
	PINRW	0.1087	0.1083	0.1104	0.1012	0.1053	0.1425	0.1368	0.1076
	CWC	0.6472	1.8690	2.1105	0.5992	0.6215	1.9123	2.1197	0.6354
ARIMA	PICP	0.9122	0.9225	0.9185	0.9132	0.9088	0.9232	0.9109	0.9128
	PINRW	0.6014	0.6601	0.5897	0.4935	0.6159	0.5697	0.5803	0.4976
	CWC	3.6032	3.9552	3.5334	2.9569	3.6896	3.4135	3.4765	2.9815
H-BLSTM	PICP	0.9440	0.9265	0.9251	0.9426	0.9211	0.9363	0.9388	0.9329
	PINRW	0.1192	0.1551	0.1667	0.1163	0.1215	0.1852	0.2088	0.1137
	CWC	0.6634	0.8473	0.8943	0.6377	0.6682	1.0244	1.1438	0.6320

TABLE 4. Average and standard deviation of indices of 8 cases.

	PICP		PINRW		CWC	
	MEAN	STDEV	MEAN	STDEV	MEAN	STDEV
S-ANN	0.9691	0.0127	0.2247	0.3534	1.3482	0.1235
D-ANN	0.9533	0.0202	0.0206	0.0862	2.1206	0.5172
QR	0.9031	0.0283	0.1151	0.0145	1.3144	0.6933
ARIMA	0.9153	0.0048	0.5760	0.0050	3.4512	0.2996
H-BLSTM	0.9334	0.0075	0.1483	0.0320	0.8139	0.1782

H-BLSTM model improves the forecast performance with better *CWC* index in all the cases and a maximum improvement of 16.9%. Thus we can assert with ease that our BLSTM AE is a worthy weapon to extract important unseen features from time series. Talking about R-BLSTM model, while expecting a better performance, we were bewildered in the manner this model performed. The reason for this adverse performance may be attributed to the fact that while maintaining the attractive property of unimpeded gradient flow across layers, the identity connection can interact unpredictably with the BLSTM architecture, as the "fast" state of the BLSTM no longer reflects the network's full representation of the data at that point. Extracting the features using autoencoder and then passing those features as input to H-BLSTM does not suffer from this unpredictable interaction thus improved predictions compared to R-BLSTM. The ruinous unpredictable interaction in R-BLSTM might be mitigated by passing the same value both to the next layer as input and to the next time-step as the "fast" state [35]. However, given the remarkable performance of the hybrid BLSTM model we were least interested in improving the R-BLSTM model as more the number of layers higher will be training time and thus less efficient the model will be.

3) COMPARISON BETWEEN PROPOSED MODEL AND TRADITIONAL MODELS

After experimental justification of the structural components of the proposed model in the previous subsections, let us now compare its performance with the traditional models. The average indices for *PICP*, *PINRW* and *CWC* for the different models are listed in Table 3. To evaluate the overall performance, the mean and standard deviation values of the indices of the eight cases are listed in Table 4.

(1) The results from Table 3 dictate that the proposed model performs better in terms of coverage probability than the statistical models but achieves comparatively lower *PICP*

TABLE 5. Performance indices comparison in multi-step forecasting on Lake Huron datasets.

Mean Values of 10 Times		ONE-STEP			TWO-STEP			THREE-STEP		
		PICP	PINRW	CWC	PICP	PINRW	CWC	PICP	PINRW	CWC
S-ANN	spring	0.9870	0.2321	1.3926	0.9838	0.2681	1.6085	0.9786	0.3103	1.8617
	summer	0.9582	0.2302	1.3814	0.9325	0.2725	1.6348	0.9291	0.3110	1.8660
	autumn	0.9517	0.2101	1.2606	0.9394	0.2778	1.6670	0.9355	0.3188	1.9131
	winter	0.9832	0.2165	1.2991	0.9812	0.2540	1.5239	0.9787	0.2919	1.7515
QR	spring	0.9407	0.1087	0.6472	0.9240	0.1624	0.9714	0.9182	0.2060	1.2338
	summer	0.8727	0.1083	1.8690	0.8653	0.1648	2.9120	0.8534	0.2091	4.0754
	autumn	0.8618	0.1104	2.1105	0.8725	0.1736	2.8657	0.8828	0.2266	3.3458
	winter	0.9289	0.1012	0.5992	0.9377	0.1525	0.9129	0.9404	0.1969	1.1786
H-BLSTM	spring	0.9440	0.1192	0.6634	0.9378	0.1782	1.0128	0.9432	0.2393	1.3653
	summer	0.9265	0.1551	0.8473	0.9195	0.2187	1.2128	0.9211	0.2753	1.5613
	autumn	0.9251	0.1667	0.8943	0.9105	0.2298	1.2703	0.9155	0.2773	1.8489
	winter	0.9426	0.1163	0.6377	0.9335	0.1694	0.9513	0.9391	0.2083	1.1917
Mean Values of 10 Times		FOUR-STEP			FIVE-STEP			SIX-STEP		
		PICP	PINRW	CWC	PICP	PINRW	CWC	PICP	PINRW	CWC
S-ANN	spring	0.9701	0.3474	2.0844	0.9640	0.3669	2.2017	0.9626	0.3972	2.3831
	summer	0.9183	0.3505	2.1029	0.9219	0.3712	2.2274	0.9164	0.3947	2.3680
	autumn	0.9339	0.3616	2.1695	0.9275	0.3927	2.3564	0.9244	0.4154	2.4924
	winter	0.9734	0.3263	1.9580	0.9766	0.3586	2.1513	0.9750	0.3775	2.2651
QR	spring	0.9082	0.2418	1.4491	0.9015	0.2750	1.6492	0.8982	0.3031	3.8864
	summer	0.8538	0.2467	4.7354	0.8570	0.2803	5.1761	0.8544	0.3078	5.8006
	autumn	0.8794	0.2678	4.0253	0.8832	0.3037	4.3914	0.8850	0.3348	4.7442
	winter	0.9383	0.2376	1.4187	0.9367	0.2706	1.6127	0.9342	0.2986	1.7834
H-BLSTM	spring	0.9385	0.2726	1.5806	0.9464	0.3168	1.8493	0.9236	0.3209	1.8872
	summer	0.9239	0.3285	1.8868	0.9186	0.3601	2.0867	0.9119	0.3885	2.2605
	autumn	0.9181	0.3219	1.8663	0.9124	0.3501	2.0246	0.9227	0.3918	2.2976
	winter	0.9369	0.2477	1.4260	0.9582	0.2987	1.7445	0.9459	0.3241	1.9251

values than the traditional LUBE models. However, the lower *PICP* is not a downside as the average *PICP* for the proposed model is 0.9334 as depicted from Table 4 which means that the proposed model can guarantee the desirable PINC of 0.9. Moreover, while maintaining the *PICP* higher than PINC, the model shows the least deviation which manifests its stability over those higher performing ANN models. The lower *PICP* can be explained by Eq. 17 and Eq. 18 which maintain a balance between *PICP* and width of interval. The target is to obtain narrow intervals which can attain the PINC.

(2) The higher *PICP* index achieved by the ANN models can be easily explained by the wider PIs produced by these models as suggested by Table 3 and Table 4. Thus, collectively it easy to manifest that H-BLSTM model wins over the ANN models as it achieves significantly lower *PINRW* and still contain the data points over the threshold value. Talking about statistical models, the ARIMA model failed miserably in terms of *PINRW*. However, QR wins over H-BLSTM in generating narrow intervals but since it failed to meet the PINC of 0.9 for those cases the lower *PINRW* gives no advantage. Figure 6, gives a visual of the PIs of wind field in Lake Huron in spring case clearly showing the quality PIs produced by H-BLSTM.

(3) From Table 3 the H-BLSTM model appears to achieve significantly better *CWC* values than other models with exceptions in spring and winter cases where QR shows marginal improvements over H-BLSTM. However, QR lose to H-BLSTM with a notable difference in terms of average *CWC* as shown in Table 4. Further the deviation in *CWC* for

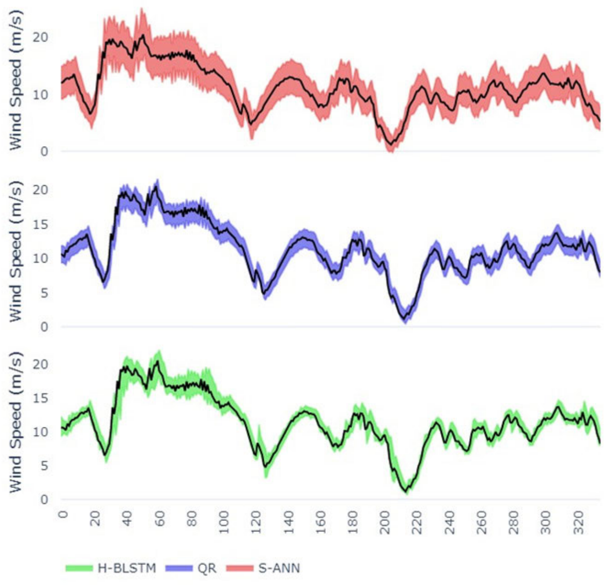


FIGURE 6. Comparison of PI in wind field in Lake Huron in spring.

QR is significant which is a measure of its high instability in terms of *CWC*. Thus, H-BLSTM emerges to be victorious over the other models in terms of *CWC* index with an improvement of 39.6% over its best competitor S-ANN, whereas an improvement of 38.1% and 76.4% over QR and ARIMA methods respectively.

(4) Apart from PI quality, the training efficiency of the model is also of practical importance. To compare the efficiency, we observe the average time taken for an experiment repeated 10 times for one case for S-ANN and H-BLSTM models on the same computing machine. The values obtained are 139.8 s, and 789.6 s for S-ANN and H-BLSTM models. A decline of 460% over S-ANN seems a little frustrating, however, the absolute time of 789 s or 13 minutes is lesser than the 30-min resolution of wind data. Thus the method is practically feasible and competitive considering the massive amount of year round data. Further, a certain expense of efficiency paid off with high quality PIs which is more desirable in real world application.

4) MULTI-STEP VALIDATION

This subsection evaluates the performance comparison of the different models for multi-step prediction. The heretofore results reveal S-ANN and QR to be better performing models in their respective categories of LUBE models and statistical models and therefore we adopt these two models as the representative for multi-step comparisons. Experiments are run for S-ANN, QR, H-BLSTM models adopting off shore Lake Huron wind field data with a varying horizon from one step to six steps. The average *PICP*, *PINRW*, and *CWC* values are summarized in Table 5.

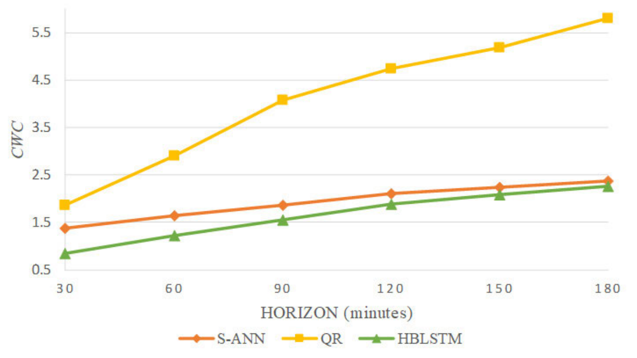


FIGURE 7. Multi-step wind speed interval prediction in "summer" case.

Achieving significantly better *CWC* index H-BLSTM clearly outperforms S-ANN in all the seasons. The reason lies in the narrow interval widths captured by H-BLSTM as compared to S-ANN while keeping the *PICP* well above PINC. While both QR and H-BLSTM models have almost identical performance for spring and winter cases, however, for summer and autumn cases QR appears to be a mediocre performer with *PICP* continuously under the PINC of 0.9 and thus lose the competition significantly to H-BLSTM. Figure 7 shows the variation of the *CWC* index with the increasing horizon from one step to six steps, for the summer case. The green line for H-BLSTM is at the bottom among the three lines indicating H-BLTSM to be the best performer for all the horizons.

V. CONCLUSION

Enhancing the quality of wind forecasts may hugely influence the scheduling and operation of wind integrated power systems. Aiming to obtain high quality prediction intervals a hybrid model is proposed in this paper. The model utilizing the recently introduced RNN called BLSTM, constructs quality intervals based on the high-quality principle of the forecasts to have high coverage probability and narrow interval widths. The model is initially benefitted by a BLSTM autoencoder which extracts the important unseen features from time series and pass it to the BLSTM model.

We examine the power of a BLSTM autoencoder over residual BLSTM model in comparative experiments which proved the BLSTM autoencoder to be a worthy feature extractor from the time series data. Thus utilizing the extracted features, the H-BLSTM model presents excellent prediction performance in comparative experiments with traditional models like ANN based LUBE models as well as statistical models like QR and bootstrap ARIMA models.

As a future work, different types of autoencoders such as sparse autoencoder, denoising autoencoder etc will be tried to preprocess wind speed data, and to improve the prediction neural network, ensemble of BLSTM, GRU, or 1D-CNN will be tried to model the wind speed, to check for the possibility of improvements in the wind speed forecasts.

REFERENCES

[1] J. Jung and R. P. Broadwater, "Current status and future advances for wind speed and power forecasting," *Renew. Sustain. Energy Rev.*, vol. 31, pp. 762–777, Mar. 2014.
[2] Z. Song, Y. Jiang, and Z. Zhang, "Short-term wind speed forecasting with Markov-switching model," *Appl. Energy*, vol. 130, pp. 103–112, Oct. 2014.
[3] G. Sideratos and N. D. Hatziargyriou, "An advanced statistical method for wind power forecasting," *IEEE Trans. Power Syst.*, vol. 22, no. 1, pp. 258–265, Feb. 2007.
[4] E. Erdem and J. Shi, "ARMA based approaches for forecasting the tuple of wind speed and direction," *Appl. Energy*, vol. 88, no. 4, pp. 1405–1414, Apr. 2011.
[5] C. Li, Z. Xiao, X. Xia, W. Zou, and C. Zhang, "A hybrid model based on synchronous optimisation for multi-step short-term wind speed forecasting," *Appl. Energy*, vol. 215, pp. 131–144, Apr. 2018.
[6] H. Liu, C. Chen, H.-Q. Tian, and Y.-F. Li, "A hybrid model for wind speed prediction using empirical mode decomposition and artificial neural networks," *Renew. Energy*, vol. 48, pp. 545–556, Dec. 2012.
[7] A. U. Haque, M. H. Nehrir, and P. Mandal, "A hybrid intelligent model for deterministic and quantile regression approach for probabilistic wind power forecasting," *IEEE Trans. Power Syst.*, vol. 29, no. 4, pp. 1663–1672, Jul. 2014.
[8] G. Chryssolouris, M. Lee, and A. Ramsey, "Confidence interval prediction for neural network models," *IEEE Trans. Neural Netw.*, vol. 7, no. 1, pp. 229–232, Jan. 1996.
[9] D. J. C. MacKay, "The evidence framework applied to classification networks," *Neural Comput.*, vol. 4, no. 5, pp. 720–736, Sep. 1992.
[10] T. Heskes, "Practical confidence and interval predictions," in *Proc. Adv. Neural Inf. Process. Syst.*, 1997, pp. 176–182.
[11] X. Chen, Z. Y. Dong, K. Meng, Y. Xu, K. P. Wong, and H. W. Ngan, "Electricity price forecasting with extreme learning machine and bootstrapping," *IEEE Trans. Power Syst.*, vol. 27, no. 4, pp. 2055–2062, Nov. 2012.
[12] W. Zou, C. Li, and P. Chen, "An inter type-2 FCR algorithm based T–S fuzzy model for short-term wind power interval prediction," *IEEE Trans. Ind. Informat.*, vol. 15, no. 9, pp. 4934–4943, Sep. 2019.

[13] 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 Netw.*, vol. 22, no. 3, pp. 337–346, Mar. 2011.

[14] C. Wan, Z. Xu, P. Pinson, Z. Y. Dong, and K. P. Wong, "Optimal prediction intervals of wind power generation," *IEEE Trans. Power Syst.*, vol. 29, no. 3, pp. 1166–1174, May 2014.

[15] C. Lian, Z. Zeng, W. Yao, H. Tang, and C. L. P. Chen, "Landslide displacement prediction with uncertainty based on neural networks with random hidden weights," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2683–2695, Dec. 2016.

[16] A. Kavousi-Fard, "Modeling uncertainty in tidal current forecast using prediction interval-based SVR," *IEEE Trans. Sustain. Energy*, vol. 8, no. 2, pp. 708–715, Apr. 2017.

[17] 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.*, vols. 418–419, pp. 363–382, Dec. 2017.

[18] H. Quan, D. Srinivasan, and A. Khosravi, "Short-term load and wind power forecasting using neural network-based prediction intervals," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 303–315, Feb. 2014.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[20] Y. Wu, M. Yuan, S. Dong, L. Lin, and Y. Liu, "Remaining useful life estimation of engineered systems using vanilla LSTM neural networks," *Neurocomputing*, vol. 275, pp. 167–179, Jan. 2018.

[21] X. Qing and Y. Niu, "Hourly day-ahead solar irradiance prediction using weather forecasts by LSTM," *Energy*, vol. 148, pp. 461–468, Apr. 2018.

[22] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Dec. 2015, *arXiv:1512.03385*. [Online]. Available: http://arxiv.org/abs/1512.03385

[24] L. Huang, J. Sun, J. Xu, and Y. Yang, "An improved residual LSTM architecture for acoustic modeling," in *Proc. Comput. Commun. Syst.*, 2017, pp. 101–105.

[25] J. Kim, M. El-Khamy, and J. Lee, "Residual LSTM: Design of a deep recurrent architecture for distant speech recognition," Jun. 2017, *arXiv:1701.03360*. [Online]. Available: http://arxiv.org/abs/1701.03360

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] C. Li, G. Tang, X. Xue, X. Chen, R. Wang, and C. Zhang, "The short-term interval prediction of wind power using the deep learning model with gradient descend optimization," *Renew. Energy*, vol. 155, pp. 197–211, Aug. 2020.

[28] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*. [Online]. Available: http://arxiv.org/abs/1609.04747

[29] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 1–8.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.

[31] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: A Python library for model selection and hyperparameter optimization," *Comput. Sci. Discovery*, vol. 8, no. 1, Jul. 2015, Art. no. 014008.

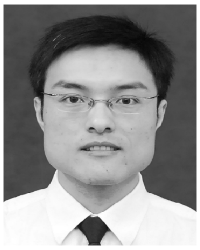
[32] C. Li, G. Tang, X. Xue, A. Saeed, and X. Hu, "Short-term wind speed interval prediction based on ensemble GRU model," *IEEE Trans. Sustain. Energy*, vol. 11, no. 3, pp. 1370–1380, Jul. 2020, doi: 10.1109/TSTE.2019.2926147.

[33] H. Quan, D. Srinivasan, and A. Khosravi, "Particle swarm optimization for construction of neural network-based prediction intervals," *Neurocomputing*, vol. 127, pp. 172–180, Mar. 2014.

[34] R. R. Pullanagari, G. Kereszturi, I. J. Yule, and M. Irwin, "Determining uncertainty prediction map of copper concentration in pasture from hyperspectral data using qunatile regression forest," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2018, pp. 3809–3811.

[35] S. Longpre, S. Pradhan, C. Xiong, and R. Socher, "A way out of the odyssey: Analyzing and combining recent insights for LSTMs," Dec. 2016, *arXiv:1611.05104*. [Online]. Available: http://arxiv.org/abs/1611.05104

ADNAN SAEED received the B.Tech. degree in mechanical engineering and the M.Tech. degree in thermal sciences (mechanical engineering) from Aligarh Muslim University, Aligarh, India, in 2010 and 2012, respectively. He is currently pursuing the Ph.D. degree in hydropower and information engineering with the Huazhong University of Science and Technology, Wuhan, China. He spent several years as an Assistant Professor in Dr. A.P.J. Abdul Kalam Technical University, Lucknow, India. His research interests include deep learning, artificial neural networks, time series analysis, and computational fluid dynamics.

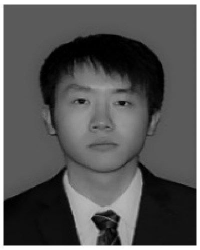


CHAOSHUN LI (Member, IEEE) received the B.S. degree in thermal energy and power engineering from Wuhan University, Wuhan, China, in 2005, and the Ph.D. degree in water conservancy and hydropower engineering from HUST, in 2010. He is currently a Full Professor with the School of Hydropower and Information Engineering, HUST. His research interests include machine learning, intelligent optimization, and control theories and applications.

MOHD DANISH received the bachelor's and master's degrees from the Department of Mechanical Engineering, Aligarh Muslim University, Aligarh, India, in 2010 and 2012, respectively, and the Ph.D. degree from the Department of Mechanical Engineering, Universiti Teknologi PETRONAS, Malaysia. He is currently an Assistant Professor with the Department of Mechanical and Materials Engineering, University of Jeddah, Jeddah, Saudi Arabia. His research interests include sustainable machining processes, cryogenic machining, powder mixed-electro discharge machining (PM-EDM), 3D printing, selective laser melting (SLM), and multi objective optimization.

SAEED RUBAIEE received the B.S. degree in chemical engineering from Tennessee Tech University, Cookeville, TN, USA, in 2009, the M.Sc. degree in industrial/management engineering from the University of South Florida, Tampa, FL, USA, in 2010, and the Ph.D. degree in industrial engineering from the Department of Industrial Systems and Manufacturing, Wichita State University, Wichita, KS, USA, in 2015.

He is currently an Associate Professor with the Department of the Industrial and Systems Engineering, University of Jeddah, Jeddah, Saudi Arabia. His research interests include engineering systems, sustainability and green manufacturing, production equipment operation, renewable energy, energy-efficient production planning, manufacturing engineering, materials engineering, advanced materials, mathematical/statistical optimization, and applied optimization.



GENG TANG received the B.S. degree in water conservancy and hydropower engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2018, and the M.S. degree in water conservancy and hydropower engineering from the Huazhong University of Science and Technology. His research interests include deep learning, artificial neural networks, and forecasting.

ZHENHAO GAN received the B.S. degree in water conservancy and hydropower engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2019, where he is currently pursuing the M.S. degree in water conservancy and hydropower engineering. His research interests include applied deep learning, artificial neural networks, time series forecasting, and computer science and applications.

ANAS AHMED received the B.S. degree in electrical engineering, the M.Sc. degree in electrical and computer engineering, and the M.Sc. degree in industrial engineering from the University of Miami, Coral Gables, FL, USA, in 1999, 2001, and 2008, respectively, and the Ph.D. degree in industrial engineering from the Department of Industrial Engineering, University of Miami, in 2010.

He is currently an Associate Professor with the Department of the Industrial and Systems Engineering, University of Jeddah, Jeddah, Saudi Arabia. His research interests include game theory, optimization, logistics, materials engineering, energy, disaster management, and statistical quality control.