BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **22**, No 1
Sofia • 2022 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2022-0011

Long Short Term Memory Neural Network-Based Model Construction and Fne-Tuning for Air Quality Parameters Prediction

Virendra Barot[1], *Viral Kapadia*[2]

[1]*I. T department, Government Engineering College, Bhavnagar, Gujarat, India*
[2]*Department of CSE, Faculty of Technology and Engineering, The M S University of Baroda, Vadodara, India*
E-mails: viren.rao82@gmail.com viral.kapadia-cse@msubaroda.ac.in

***Abstract*:** *Air pollution has increased worries regarding health and ecosystems. Precise prediction of air quality parameters can assist in the effective action of air pollution control and prevention. In this work, a deep learning framework is proposed to predict parameters such as fine particulate matter and carbon monoxide. Long Short Term Memory* (*LSTM*) *neural network-based model that processes sequences in forward and backward direction to consider the influence of timesteps in both directions is employed. For further learning, unidirectional layers' stacking is implemented. The performance of the model is optimized by fine-tuning hyperparameters, regularization techniques for overfitting resolution, and various merging options for the bidirectional input layer. The proposed model achieves good optimization and performs better than the simple LSTM and a Recurrent Neural Network* (*RNN*) *based model. Moreover, an attention-based mechanism is adopted to focus on more significant timesteps for prediction. The self-attention approach improves performance further and works well especially for longer sequences and extended time horizons. Experiments are conducted using real-world data collected, and results are evaluated using the mean square error loss function.*

***Keywords*:** *Air quality forecasting, Air pollution forecasting, Deep learning, Long short term memory, attention.*

1. Introduction

Due to fast economic development, urbanization, construction of industrial parks, and production procedures, air pollution has become an alarming issue for society. Also, rapid urban development has given rise to global warming, climate changes, and disturbance of the ecosystem. Fuel-burning generates carbon dioxide as well carbon monoxide, both influence rising of earth temperature. In recent years, numerous cities

across the globe have been attacked by smog. The smog affects citizens' daily life and causes many health hazards.

According to [1], a substantial fear for air pollution is due to the existence of a high concentration of particulate matter (PM 2.5 and PM10) and carbon monoxide in the surrounding atmosphere. Incomplete coal burning and automobile emissions can cause damaging effects on health because they discharge fine particulate matter (PM2.5) straight into the atmosphere. PM 2.5 can cause harmful effects to various human organs such as the lungs, nervous system [2], and the outermost layer of skin [3]. Carbon monoxide (CO) present in the atmosphere is responsible for an enormous proportion of the poisonings and life losses, reported all around the globe [4]. In various cases, Carbon monoxide level increases to a level that causes coma and even deaths sometimes [5]. The development of accurate air quality (air pollution) parameter forecasting model or system in urban areas can help people in avoiding health hazards by the decision making of canceling to be outdoor or to visit certain critical places. Also, the forecasting results inform government agencies to execute traffic control or any such policy implementation, looking at the critical levels of air pollution in the future.

2. Related work

Numerous air quality prediction methodologies have been presented by researchers, which can be classified as statistical methods, machine learning-based approaches, and recently, deep learning-based approaches. Statistics based approach comprises Principal Component Analysis (PCA), Coefficient analysis, linear as a well non-linear regression-based model [6, 11, 8], and interpolation-based [7] implementations. These methods suffer due to a lack of ability to model non-linear as well as multivariate types of data. Machine learning-based approach comprises fuzzy methods [12, 13], genetic algorithm [9], and support vector [10, 13] based implementations.

Recent development in deep learning-based forecasting approaches has shown very good prediction accuracy, outperforming the statistics and machine learning-based methods in various domains. Deep learning methods include Recurrent Neural Network (RNN) and Long Short Term Memory (LSTM) based neural network model [14, 15]. LSTM network has performed better compared to RNN due to the gated cell-based mechanisms in the LSTM unit [15, 16, 26]. LSTM based fully connected network has been utilized to predict particulate matter concentration at targeted stations [17, 18] by the researchers. Authors in [19] have predicted air pollution parameters with a combination of Convolutional Neural Network (CNN) and LSTM based networks to further improve the efficiency of a simple LSTM based network model. Thanongsak Xayasouk and team [20] have predicted air quality using the deep auto encoder and LSTM based model. The authors have fine-tuned the performance by hyperparameter setting and compared the performance using two approaches – PM2.5 predictions by combining convolution networks and bidirectional GRU (Gated Recurrent Unit) network over particulate matter data of Beijing city have been carried out to enhance the performance of simple GRU [21]. Saba Gul and his team

[43] have implemented and fine-tuned the performance of LSTM based model for the classification of the Air Quality Index in six different categories. Authors [22] have used a bi-directional LSTM network for prediction the severity of air pollution category. The bi-directional approach learns features in both forward and backward directions to improve further the LSTM performance. Weitian Tong and his team [23] have used bidirectional LSTM for spatiotemporal interpolation of particulate matter PM2.5 concentrations. The model has focused on both spatial and temporal factors. Yue-Shan Chang and his team [41] have used aggregated LSTM model. The model has used three sets of data and aggregated predictive features from those three sets in the prediction of PM2.5. The first set has utilized the data from the local station which contained 17 attributes along with PM2.5. The second data set of PM2.5 and PM10 has been derived from nearby stations to realize the effect of the nearby pollutant on local pollutants. The third dataset has been derived from the industrial zone to consider the effect of external sources. The model utilizes the effect of other (other than PM2.5) pollutants from local stations and the effect of nearby as well external sources of PM2.5 and PM10 on the prediction of local stations' PM2.5. Jun Ma and the team [44] have utilized the data from the existing station during training, which assists to improve the predictive performance of the new station to overcome the shortage of data at such new stations. The model named transfer learning (from existing to new stations) utilizes a stacked bi-direction LSTM network. Authors [42] have used a flexible dropout layer to adjust the dropout rate automatically along with widow size for specific intervals and have found good results by the addition of such layer. The model performs better compared to other alternatives (without flexible dropout) such as GRU, LSTM, and bi-LSTM. So LSTM based approach is one of the popular choices in air quality parameter prediction. Some attempts have been also made to further improve the performance of LSTM based model as discussed. In the proposed work LSTM network-based model is implemented for the prediction of air quality parameters like Carbon monoxide (CO), Particulate matter 2.5, and Particulate matter 10. The major contribution of the proposed work can be given as follows.

- LSTM based deep learning network model that takes the advantage of both forward and backward direction time step observation in learning during training is applied in the proposed work. The data being observed are modeled into sequences and a sliding window-based approach for the transformation of training data into supervised data is used. The model can predict the next time step value for air quality parameters from the given test sequence. In the field of air quality prediction, one such approach of using bidirectional LSTM [22] is available but the work is done for label classification where the label is various severity categories rather than actual sequence to sequence (moving window based) value prediction. In another approach, authors [44] have used bidirectional stacked LSTM during transfer learning from existing station data to the new station for such time series prediction. In our approach, the input layer is only kept bidirectional while the stacking is done with unidirectional LSTM layers. The performance of such bi-directional LSTM is heavily influenced by the way forward and backward layer outputs are merged. In the proposed work, the model is critically evaluated with various merging functions

(options) to optimize the performance. Moreover, the stacking model is tested with an incremental approach to decide the best stacking approach for optimized performance.

- The performance of the deep learning model suffers from an overfitting issue. In the model being proposed, the issue is resolved by analyzing and implementing two regularization techniques. The model is also fine-tuned by a proper hyperparameter setting to improve the performance.
- The proposed work also implements a self-attention mechanism which shows better performance by minimizing the loss function further. The applied self-attention mechanism is also one of the first attempts to the best of our knowledge, in the field of sequence to sequence air quality prediction.

The proposed article is organized as follows: Section 3 presents the necessary concepts and LSTM based neural network methodology (model) employed in the proposed work. Section 4 discusses data preparation or data modeling, preprocessing, overall prediction framework, and results of experiments in detail. The concluding remarks and future enhancements discussions have been carried out in Section 5.

3. Necessary concepts and methodology employed

3.1. LSTM neural network

A Recurrent Neural Network (RNN) is mostly utilized to process sequential data. RNN handles dependencies among data by the information/knowledge gained from the succeeding timesteps by the use of network loops and by the concept of timing in learning. The activation of the previous time step is provided as input to the current time step for forecasting purposes in the loop. RNN is very promising in processing short-term dependencies but does not perform well with long-term dependencies or long sequences due to the gradient disappearance or explosion problem in training [24, 25].

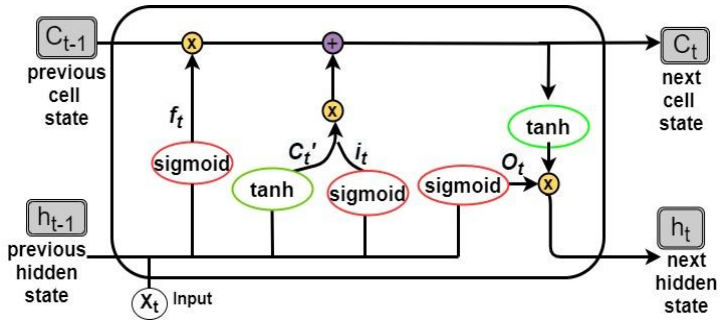


Fig. 1. Internal gated structure of basic LSTM unit/cell

Long Short Term Memory (LSTM) network (a special type of RNN) has been proposed to solve the vanishing gradient problem and handle the prediction of long sequences in an effective manner [26]. The traditional perceptron architecture has

been replaced with a memory function to maintain cell state and gated functions for regulating input information. The architecture includes an input gate, a forget gate, and an output gate. Fig. 1 shows the architectural design of a single LSTM unit. The internal cell state at time step t is denoted as c_t and at time $t-1$ is denoted as c_{t-1}. The inputs to LSTM at time t are an input vector x_t, a hidden vector h_{t-1}, and a cell state c_{t-1}. LSTM produces the output c_t and h_t from the given input and with the use of the three gates. The function of forget gate, input gate, and output gate f_t, i_t, and o_t, respectively, can be stated as per the equations (1)-(3). The previously hidden state h_{t-1} and the input at current unit x_t are given to the sigmoid function. The output f_t (between 0 to 1) is multiplied with c_{t-1} to decide the extent of information to be forgotten at forget gate

(1) $$f_t = \sigma(W_f\,[h_{t-1}, x_t] + b_f),$$

(2) $$i_t = \sigma(W_i\,[h_{t-1}, x_t] + b_i),$$

(3) $$o_t = \sigma(W_O\,[h_{t-1}, x_t] + b_O).$$

The sigmoid at the input gate generates a value between 0 and 1 using Equation (2), which decides values to be updated or added in the neuron state. The intermediate cell state c_t' is calculated as per Equation (4) using previous time point output h_{t-1} and current unit input x_t. The intermediate cell state c_t' and i_t are multiplied element-wise and the result is added with the output of forget gate to calculate the cell state c_t at time t as per Equation (5). The output gate decides information to be outputted in the state of the neuron. The output state of the input gate(c_t) is given to tanh layer and this output from tanh (between 1 and –1) is multiplied by o_t to generate output hidden state h_t at timestep t as shown in Equation (6). Parameter W is weight matrices and b is biased, which are applied respectively at each stage; σ is the logistic sigmoid activation function and tanh is the hyperbolic tangent function,

(4) $$c_t' = \tanh(W_C\,[h_{t-1}, x_t] + b_C),$$

(5) $$c_t = f_t * c_{t-1} + i_t * c_t',$$

(6) $$h_t = o_t * \tanh(c_t).$$

3.2. Proposed LSTM based Neural Network (bidirectional and stacking) model

Authors have proposed in [27] for the first time a bidirectional recurrent neural network, which works on sequences in two directions – forward as well backward. Forward pass and backward pass are handled by separate RNN layers. A bidirectional Long Short Term Memory network was proposed in the year 2005 [28] in the field of signal processing. It connects two separate layers (forward and backward processing) to the same output (merge) layer. The approach proved to perform brtter than unidirectional [28, 29].

In our model being proposed for training, two sequences are processed as shown in Fig. 2. The forward pass layer outputs sequence $\boldsymbol{h}$ iteratively which is calculated using inputs in forwarding direction $T-1$ to T+1. The backward pass layer outputs vector $\boldsymbol{h'}$ iteratively which is calculated using inputs in reverse direction T+1 to $T-1$. Both the forward and backward layer outputs are calculated by equations used for the standard LSTM unit. The input vector to the next layer is generated using the merge function. The Merge function uses the output of the respective time step from the LSTM cell in the forward and backward layer as shown in Fig. 2.

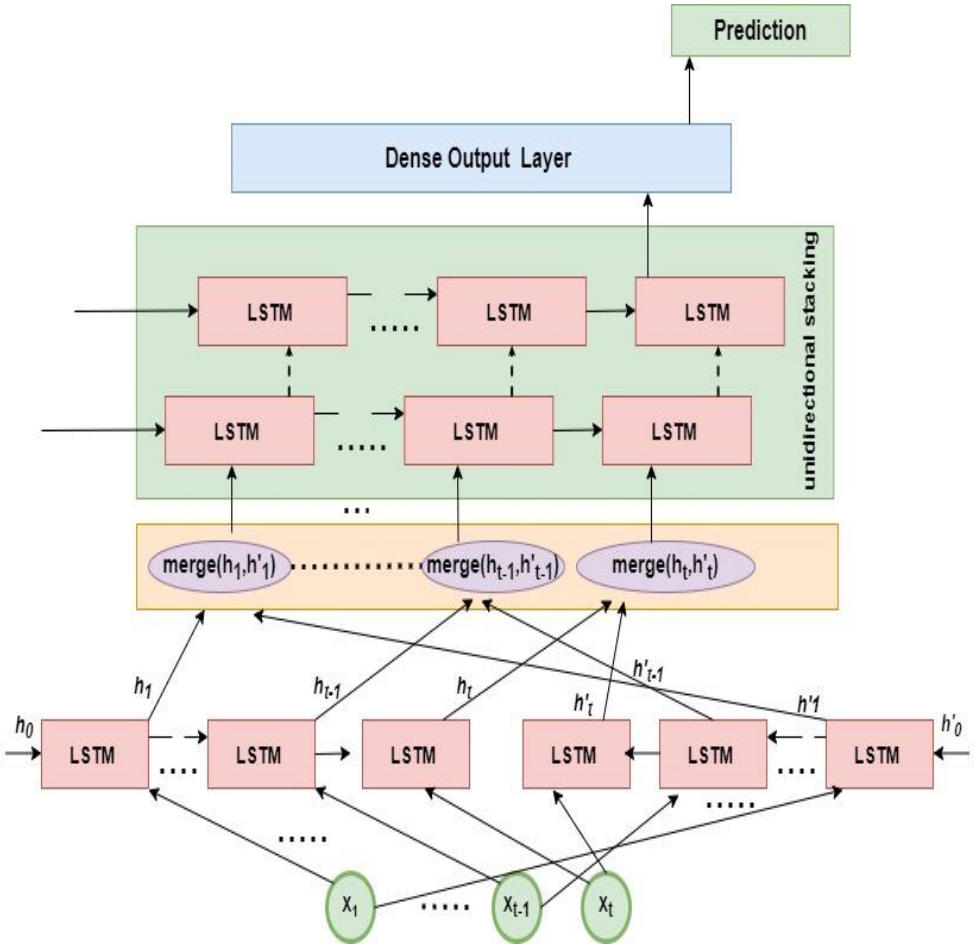


Fig. 2. Proposed LSTM based model

Various researches [30-32] have presented that deep neural network building by progressively stacking recurrent layers on top of each other works more effectively. The approach lets the hidden state at every level to work at different timescale and every layer provides its abstraction level. The model being proposed implements deep LSTM architectures where several stacked unidirectional LSTM layers are added on top of each other. The output of one such layer is given as input to the next layer. The bottom layer in the training model being proposed utilizes forward and backward pass for learning the features in given sequences. The top layers of the model utilize the learned features from the bottom layer for further learning. The number of layers to be stacked, which gives the optimum result, is one of the hyperparameters, and it is decided through experiments. The proposed training model with forward and backward layer and the first layer with only forward unidirectional stacking as shown in Fig. 2 will be referred to as the FBLSTM model further on in the paper.

3.3. Attention mechanism

Attention mechanism in a neural network lets the training model give more importance to the features that are having more influence on the output. Self-attention is applied to the input timestep vector of the RNN layer to focus more on important timestep values [33, 34]. The self-attention mechanism assigns some weight to each input sample according to its importance or influence on the output. Self-attention is proven to improve the performance of the neural network [33-35]. Self-attention is applied in the model shown in Fig. 2 and experiments are being carried out to check the effect on learning with different dimensions, discussed later in the result section. Output vector $V = \{V_1, V_2, V_3, \ldots, V_t\}$ of the last unidirectional hidden layer in LSTM stacking is given as input to the attention mechanism or attention layer. The self-attention mechanism assigns the weight α_i to each of the V_i based on the importance to output as per Equations (7) and (8):

$$\alpha_i = \frac{\exp(e_i)}{\sum_{i=1}^{T} \exp(e_i)}, \tag{7}$$

$$e_i = \text{fun}(W_i, V_i), \tag{8}$$

$$V'_i = \alpha_i * V_i, \tag{9}$$

where W_i is the weight applied during training for each timestep V_i in LSTM based learning through backpropagation with time and fun is the function applied for calculation of e_i which is tanh function in experiments. Softmax in Keras is used for self-attention which calculates the normalized weight α_i as per Equation (7). The softmax function takes care that the addition of all the weights(α_i) is one. The final Context vector output from the self-attention layer is $V' = \{V'_1, V'_2, V'_3, \ldots, V'_t\}$, where V'_i can be obtained as per Equation (9).

4. Experimentation and results

4.1. Data preparation

Real-time data of air quality parameters for the forecasting purpose are collected by an IoT (internet of technology) based air quality monitoring system [36]. The monitoring system contains a smart node with an ESP8266 12E/Node MCU controller and a variety of air quality parameter sensors interfaced with it. The smart node is connected to cloud broker HiveMQ via MQTT protocol. The data observed of the remote site is logged in at the server every 90 seconds, which works as the timestep of the input sequences in the proposed prediction model. The IoT-based system utilizes ZE07-CO (electrochemical sensor module) for carbon monoxide and SDS 021(laser scattering sensor module) for PM 10 and PM 2.5 parameter collection.

To develop a model for prediction of the time series data of pollutants collected, we need to transform the time series data into a suitable data structure for supervised learning. The approach having been used in [37] relies on a prediction methodology that models many to many mapping of inputs to outputs by keeping the stochastic interdependencies of time-series events. It consists of predicting the next *k* values as per the next equation:

$$P(x_t, \ldots, x_{t-l+1}) = (x_{t+k}, \ldots, x_{t+1}), \tag{10}$$

where l indicates the number of past observations utilized for prediction of k next events and $t \in \{l, \ldots, n-k\}$. The right-hand side of the equation indicates observation included in the target or output window and k represents the size of the output window. The left-hand side of the equation represents observations included in the input window with size l. The moving window or sliding window method is employed for partitioning the time-series observations of pollutants of length n into sample sequences of length [input window (l) + output window (k)]. So overall there are total [n-(input window size +output window size) +1] such samples in sample space. The sample space X after segmentation over the time-series with n timesteps can be given as:

$$X = \begin{bmatrix} x_1 & x_2 & \cdots & x_l & x_{l+1} & \cdots & x_{l+k} \\ x_2 & x_3 & \cdots & x_{l+1} & x_{l+2} & \cdots & x_{l+k+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{n-(l+k)+1} & x_{n-(l+k)+2} & \cdots & x_{n-k} & x_{n-k+1} & \cdots & x_n \end{bmatrix}. \tag{11}$$

The output window is kept of size one, so only the next time step is predicted in experiments and input window size l is kept to be 60. During the training of the LSTM network, training samples are obtained using the moving input window as shown in Fig. 3. The l sized input window slides over the pollutant time-series observations until the last window is encountered. The output window observation is used as a target value that is used for further learning using backpropagation through time and for error gradient calculation over the employed batch size.

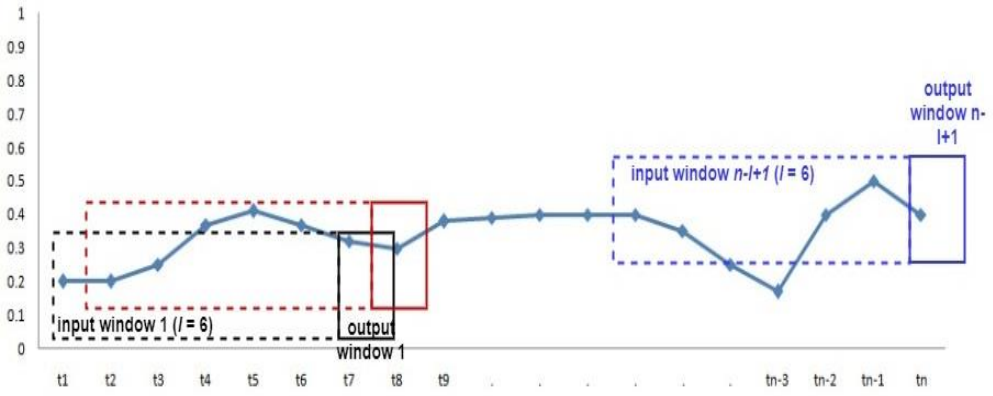


Fig. 3. Moving window over timesteps in time-series

4.2. Pre-processing and Metrics

Fig. 4 shows the complete framework of the proposed research work. During data pre-processing normalization of time series data is performed using linear scaling given with Equation (12) below. The linear scaling method transforms the observations into a new interval defined by lower bound (lb) and upper bound (ub). The new transformed pollutant time series data lies in the range [0, 1]:

$$X_{i\_\text{new}} = \text{lb} + \frac{X_i - \text{MIN}(X)}{\text{MAX}(X) - \text{MIN}(X)} * (\text{ub} - \text{lb}). \tag{12}$$

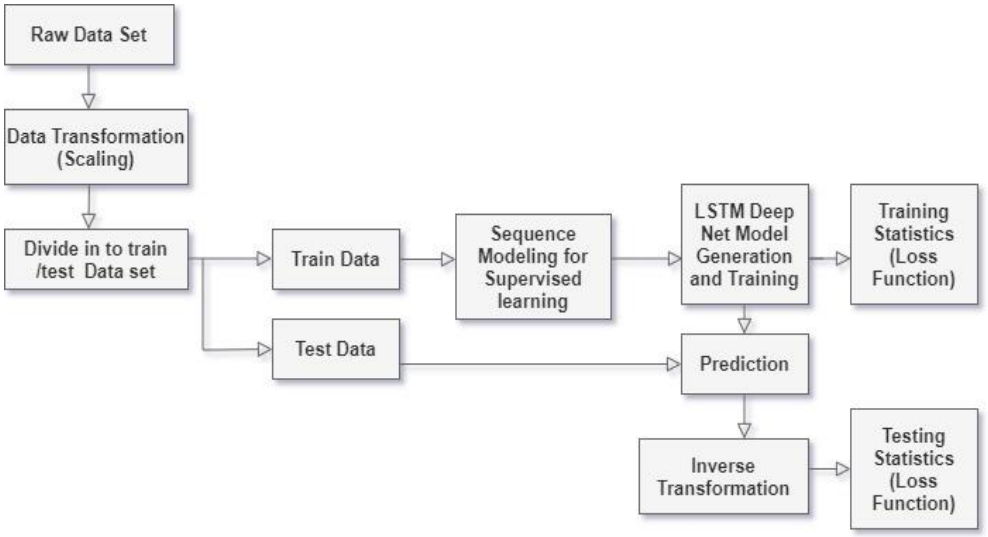


Fig. 4. Detailed training and testing framework of proposed forecasting task

Transformed training data are modelled into supervised learning data as discussed in the data preparation subsection. There are various evaluation metrics available, e.g., MAPE, MSE, and RMSE that can be used in the performance evaluation of the prediction model. Mean Square Error (MSE) is used as an evaluation metric (MSE loss function in Keras) in the experiments conducted that can be defined as per Equation (13), where $X_{\text{pred\_}i}$ is the predicted value and $X_{\text{actual\_}i}$ is the actual value of the *i*-th air quality parameter observation:

$$\text{MSE} = \frac{1}{n}\sum_{n=1}^{n}(\text{X}_{\text{pred\_i}} - \text{X}_{\text{actual\_i}})^2. \tag{13}$$

4.3. Results

Implementation of the proposed FBLSTM model and experimentations are carried out using Keras 2.1.6 which uses Tensorflow in the back end. The model uses 60 units in each layer and tanh function is utilized as an activation function. The input window or sequence size in each sample is set to 60. In Keras, the LSTM layer is shaped with a three-dimensional vector with fields (total sample space size, number of timestep observations in individual sample, feature). Training in neural networks is done using a stochastic gradient descent optimization algorithm. This algorithm compares the prediction to actual observation and uses the difference as an estimation for the error gradient. The error gradient in turn is used to update the weights and biases. SGD [38] algorithms are suffering from the problem of deciding optimal step size. The problem is solved by the availability of a new optimization algorithm, i.e, ADAM [38]. ADAptive Moment estimation (ADAM) is the best stochastic optimization algorithm for deep learning purposes and comprehends the benefits of two widely used algorithms AdaGrad and RMSProp. The ADAM algorithm is utilized in the model for optimization which adapts the rate of learning based on the average of first as well second moments of the gradients. The algorithm provides fast

convergence with less memory necessity comparing to the two other stochastic algorithms [38].

Table 1. Comparison of MSE of the proposed model(FBLSTM) with LSTM and RNN for training and validation

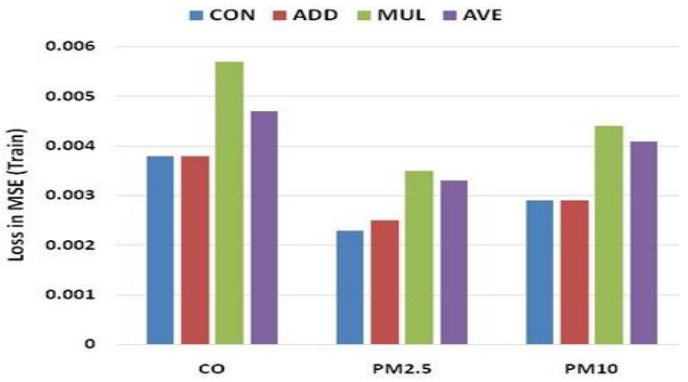
Model		Epochs 100		Epochs 200		Epochs 300	
		Train	**Value**	**Train**	**Value**	**Train**	**Value**
RNN (sequence size =10)	**CO**	0.0761	0.0903	0.0579	0.1079	0.0443	0.1127
	PM2.5	0.0643	0.0834	0.0408	0.0986	0.0392	0.1023
	PM10	0.0617	0.0782	0.0510	0.1087	0.0398	0.1324
RNN (sequence size =60)	**CO**	1.8967	1.9942	1.6993	2.0934	1.6341	2.2832
	PM2.5	2.1032	2.4926	1.8976	2.5503	1.7689	2.6845
	PM10	1.9873	2.3031	1.6973	2.3314	1.5961	2.7941
LSTM	**CO**	0.0118	0.0140	0.0107	0.0129	0.0104	0.0144
	PM2.5	0.0104	0.0119	0.0100	0.0130	0.0089	0.0147
	PM10	0.0111	0.0135	0.0105	0.0137	0.0102	0.0153
FBLSTM (1 hidden layer, CONCAT merge function, sequence size =60)	**CO**	0.0059	0.0082	0.0049	0.0069	0.0042	**0.0093**
	PM2.5	0.0041	**0.0057**	0.0039	0.0072	0.0026	0.0083
	PM10	0.0047	0.0077	0.0040	0.0073	0.0036	0.0088
FBLSTM (2 hidden layer, CONCAT merge function, sequence size =60)	**CO**	**0.0052**	**0.0078**	**0.0041**	**0.0061**	**0.0038**	**0.0093**
	PM2.5	**0.0030**	0.0058	**0.0025**	**0.0059**	**0.0023**	**0.0077**
	PM10	**0.0041**	**0.0070**	**0.0032**	**0.0067**	**0.0029**	**0.0080**
FBLSTM (3 hidden layer, CONCAT merge function, sequence size =60)	**CO**	0.0060	0.0080	0.0048	0.0074	0.0048	0.0098
	PM2.5	0.0043	0.0074	0.0045	0.0079	0.0033	0.0088
	PM10	0.0052	0.0084	0.0048	0.0081	0.0043	0.0094

The total number of individual training samples utilized (after which error gradient calculated) for estimation of error gradient is a hyperparameter for the ADAM optimization algorithm which is known as batch size. A batch size of 32 is used during the experiments. The number of unidirectional layers in stacking to gain minimum loss for prediction of air pollutant time series data is decided through experiments. The return sequence attribute is kept to be true in Keras, while the output of one LSTM layer is given as input to the subsequent layer. So, instead of giving one output, the LSTM layer gives output for each timestep. The backward pass layer is implemented by setting go_backwards to be true in Keras. We use the functional API of Keras for building the proposed training model. Table 1 shows the loss in MSE (mean squared error) with various stacking options for the proposed FBLSTM model. The MSE values listed in the tables are averaged values over six repeated

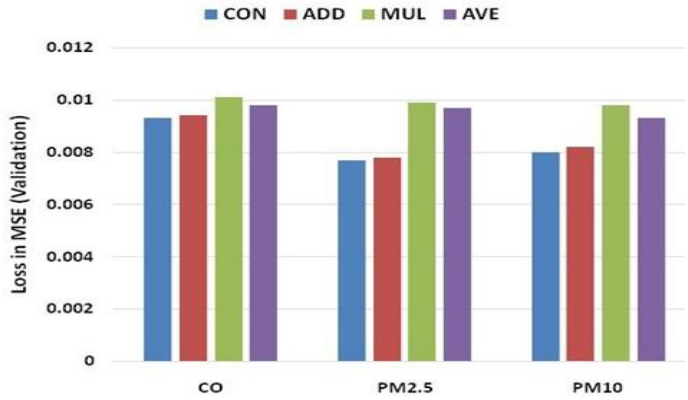
runs. Experiments are conducted till 300 epochs and loss reported at the end of 100, 200, and 300 epochs are listed for both train and test (validation) data in the table. Table 1 shows the output of the experiments with the "CONCAT"(Con) function for merging the two layers (forward and backward), which is the default one in Keras. The results are compared with RNN and simple LSTM. RNN suffers from a vanishing gradient problem as the sequence sample size increases, which also can be observed in the performance evaluation. As the sequence size increases from 10 to 60 the loss value increases. Mean Squared Error-values highlighted in bold in the table show the minimum observed loss during experiments. The minimum value of loss in turn indicates the best accuracy for time series prediction. It can be observed that in the proposed FBLSTM model with the discussed parameter setting, the minimum loss is observed with the stacking of two hidden layers. Moving further by adding one more layer to the existing unidirectional stacking, i.e, three hidden layers, the performance starts degrading. The proposed model outperforms both the RNN and the simple LSTM layer.

There are four merging functions available in Keras. Concat (*Con*) is the default merging option where the output of the respective cell state from forward and backward layers are simply concatenated together. *Mul* and *Add* are the merge modes where such outputs are multiplied or added respectively. *Ave* is the merge function where the average of the corresponding outputs is taken. We utilized the optimum FBLSTM model setup which has two hidden layers and experimented with all possible four merge modes. As shown in Fig. 5 the *Con* function performs best over train as well as test data and achieves minimum loss function. *Add* function also perform near equal to the *Con* function while *Mul* function has the highest loss amongst all four.

Table 1 shows the MSE value at 100, 200, and 300 epochs, from the result it can be seen that the MSE value decreases as the number of epochs increased for the train data. The epoch represents the number of scans throughout the total sample space. It is very obvious and expected that with the increase of epochs the loss gets decreased and at a certain point it becomes stable. Such behavior has been also exhibited for the train data but for validation data (test data) the same effect is not observed. The detailed behavior of the model is represented in Fig. 6 by collecting and plotting loss value after every epoch for the training data and validation data for a specific sample space of PM 2.5 time-series data. It can be seen that initially with fast convergence and after obtaining the minimum value of loss function, the model performance starts decreasing with the increase of epochs for validation. The behavior exhibited is due to an overfitting issue.



(a)



(b)

Fig. 5. MSE for merge function alternatives over: training data (a); validation data (b)

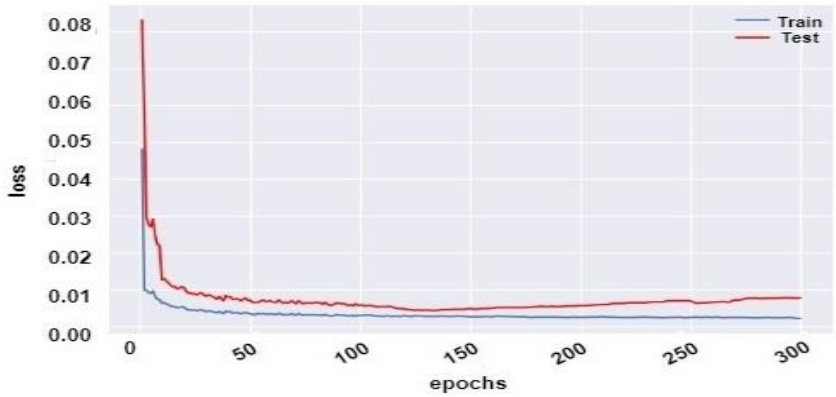
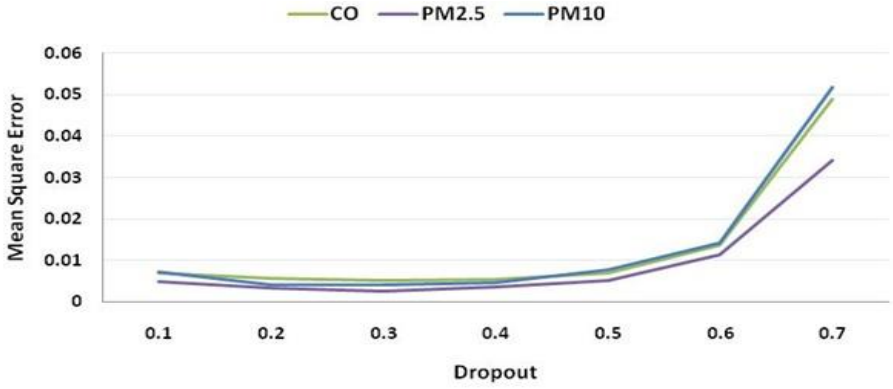


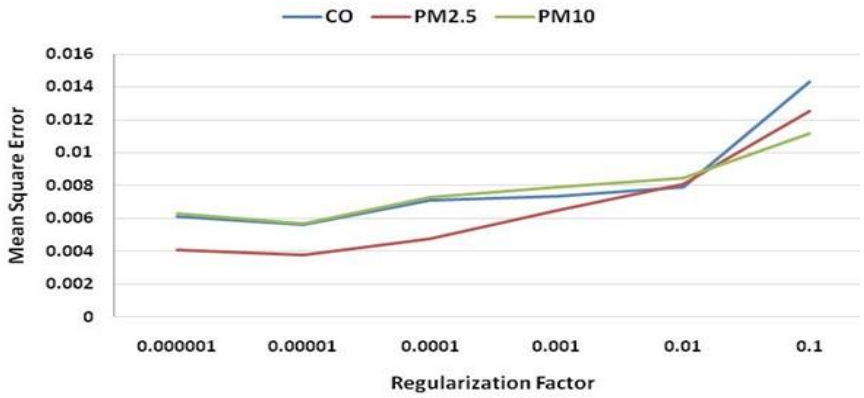
Fig. 6. Plotting of MSE per every epoch for training and validation

To reduce overfitting, various regularization techniques are proposed by researchers in the field of neural networks. Dropout is one such regularization technique used to avert the model from overfitting. Dropout regularization is

executed by randomly setting the leaving or outgoing edges of hidden cells in the hidden layer to zero at each update of the training phase [39]. Keras supports the implementation of the dropout by using dropout layers. Dropout layers are added in between hidden layers. Input and recurrent edges or connections to LSTM units are excluded by setting probability for activation and updates of weights during training of a network. The addition of dropout layers results in simulating a large number of networks with a very dynamic network structure in parallel. Also due to dropout, a neural network can never rely on any input node because every node has the probability to be removed. So, the network cannot assign any high weight to a specific feature. The probability of such dropout is again a hyperparameter which needs to be chosen with experiments. Fig. 7a shows all experimented values of dropout, plotted against the MSE loss recorded for a particular dropout value. Applied dropout values can vary in the range of 0 to 1. It can be seen from Fig. 7a that minimum loss is recorded for 0.3 value of dropout and after 0.5 dropout value, there is a sudden increase in the loss function. Dropout value 0.3 in Keras is representing a 30 percent probability of node removal during training.



(a)



(b)

Fig. 7. Performance of the model for: various values of dropout parameter under dropout technique (a); various values of lambda or regularization factor under L2 regularization (b)

Table 2. MSE comparison of the proposed model (FBLSTM) under regularization techniques for validation data

Parameter	Epochs 100			Epochs 200			Epochs 300		
	No Regularization	**Dropout (0.3)**	**L2 regularization ($\times 10^{-5}$)**	**No Regularization**	**Dropout (0.3)**	**L2 regularization ($\times 10^{-5}$)**	**No Regularization**	**Dropout (0.3)**	**L2 regularization ($\times 10^{-5}$)**
CO	0.0078	0.0075	0.0081	0.0061	0.0057	0.0059	0.0093	0.0052	0.0056
PM 2.5	0.0058	0.0047	0.0054	0.0059	0.0031	0.0043	0.0077	0.0025	0.0038
PM 10	0.0070	0.0061	0.0063	0.0067	0.0047	0.0059	0.0080	0.0041	0.0057

Another technique of regularization is using weight decay known as L2 regularization. The neural network always tries to minimize the cost function by adjustment of weights and biases. For L2 regularization a component is added which penalizes the large weights. The component is added to the cost function. The addition of the component drives the overall weight matrix values down, which in turn reduces the activation function influence. Due to that comparatively less complex activation function may be fit to observations, which helps in reducing overfitting. The component is given in the next equation:

$$\text{New Minimization Goal} = L(W, B) + \lambda \|W\|^2. \quad (14)$$

Here λ in the component is a regularization or tuning parameter that balances the tradeoff between a low value of weights and a low loss of training. λ is the hyperparameter which is needed to be optimized. Authors in [40] report the value of λ (regularization factor) to be 10^{-6} for getting optimum results in their LSTM based network. We tried the initial value of λ which is given as an argument to the L2 regularization in Keras starting from 10^{-1} to 10^{-6}. Fig. 7b shows all experimented values of λ plotted against the MSE loss recorded for a particular λ value (regularization factor). It can be seen from the given figure that minimum loss has been recorded for 10^{-5} values of λ.

Table 2 shows loss function MSE recorded for three time-series data of pollution parameters at the end 100, 200, and 300 epochs for validation data. The table compares the MSE values for 0.3 dropouts and 10^{-5} for λ value which is found to be providing optimum results during respective regularization experiments. Results show that dropout-based regularization outperforms weight decay (L2) regularization and is more suitable to our model. Dropout regularization can achieve stable conversation along with loss in MSE up to 0.0052, 0.0025, and 0.0041 for CO, PM 2.5, and PM 10, respectively.

As discussed in Subsection 3.3, the self-attention mechanism is also applied and tested during the experiments. The Self-attention layer is kept as the last layer in the model (FBLSTM) shown in Fig. 2. The output of the self-attention layer is given as input to the final dense layer for prediction. To understand the effect on loss function and improvement to the existing model, we analyze the self-attention mechanism with two dimensions, time horizon as well input window size or input lag. While increasing the time horizon, the sequence size is kept of the same length. By keeping the same sequence size and increasing in time horizon, employed recorded parameter

samples in training realize more fluctuations compared to small-time horizon. Table 3 compares the loss function value (mean squared error) obtained for the FBLSTM (with two hidden layers), without attention, and with attention mechanism for the three air quality parameters. The table shows the effect on MSE value with the increase in the time horizon. The first two rows in the table depict the MSE value for T_x (the basic time step in the input sequence) which is 90 seconds. The time horizon increment further is obtained by aggregating the recorded value for the basic timestep, i.e., $4T_x$ horizon is the aggregated value over 360 seconds and so on. To keep the total sample size and samples in each sequence the same with the extension of the time horizon (for each horizon), more training data(samples) are required. So training data (recorded observations of air quality parameters) covers data of one complete week. The table shows the results obtain over data of two such weeks. The rate column in the table shows the percentage of increase in MSE value with the increase of time horizon from the previous one. It can be seen from the table that with the extension of the time horizon, the rate of increase in MSE (compared to the previous horizon) remains small for the model with an attention mechanism. The high rate of increase in MSE represents the rapid reduction in prediction performance with the extension in the horizon. It can be seen that initially there is not much difference between the performance of the two models but with a higher time horizon, the model with attention mechanism performs substantially better compared to the model without attention. Table 4 shows the performance of the two models with the increase in input window size over recorded air quality parameters observations of a single day. The MSE value and rate of increase in MSE are listed for an input window size of 60, 80, and 120. The table indicates that the model with attention mechanism realizes lower MSE and a slow rate of increase in MSE for higher input window size. Thus the table depicts self-attention mechanism provides better performance for longer sequences.

Table 3. MSE comparison of FBLSTM with attention and without attention after 300 epochs for various time horizons

Week No	**Para-meter**	**Simple FB-LSTM**	**With self-atten-tion**	**Simple FB-LSTM**	**With self-atten-tion**	**Simple FB-LSTM**	**With self-atten-tion**	**Simple FB-LSTM**	**With self-atten-tion**
		Tx		**4*Tx***		**8*Tx***		**12*Tx***	
Week 1	**CO**	0.0051	0.0047	0.0055	0.0049	0.0073	0.0053	0.011	0.006
	Rate	-	-	7.84	4.26	32.73	8.16	50.68	13.21
	PM2.5	0.0025	0.0023	0.0028	0.0025	0.0044	0.0029	0.0083	0.0034
	Rate	-	-	12.00	8.70	57.14	16.00	88.64	17.24
	PM10	0.0041	0.0036	0.0045	0.0039	0.0066	0.0044	0.0118	0.0052
	Rate	-	-	9.76	8.33	46.67	12.82	78.79	18.18
Week 2	**CO**	0.0039	0.0038	0.0044	0.004	0.0061	0.0043	0.0097	0.0051
	Rate	-	-	12.82	5.26	38.64	7.50	59.02	18.60
	PM2.5	0.0032	0.0031	0.0035	0.0033	0.0058	0.0037	0.0103	0.0044
	Rate	-	-	9.38	6.45	65.71	12.12	77.59	18.92
	PM10	0.0045	0.0038	0.0051	0.0041	0.0076	0.0047	0.0125	0.0052
	Rate	-	-	13.33	7.89	49.02	14.63	64.47	10.64

Table 4. MSE comparison of FBLSTM with attention and without attention after 300 epochs for various input windows

Model	Input Window size	CO	Rate	PM2.5	Rate	PM10	Rate
FBLSTM	60	0.0055	–	0.0035	–	0.0046	–
FBLSTM+ Attention		0.0036	–	0.0021	–	0.0034	–
FBLSTM	80	0.0084	52.73	0.0057	62.86	0.0074	60.87
FBLSTM+ Attention		0.0045	25.00	0.0029	38.10	0.0045	32.35
FBLSTM	120	0.0223	165.48	0.0191	235.09	0.0208	181.08
FBLSTM+ Attention		0.0086	91.11	0.0061	110.34	0.0091	102.22

5. Conclusion and future enhancements

Reliable and precise prediction of air pollution or air quality parameters is of great importance. In the presented work LSTM based deep learning framework is proposed for the prediction of air quality parameters. The framework includes a forward and backward LSTM based model for learning the influence of other observations on current prediction in two directions. The proposed model is using bidirectional learning with unidirectional further stacking to improve the performance. Optimum performance is achieved with two hidden layers and CONCAT merging function (out of four merging function alternatives available in Keras) in the proposed model and the model outperforms the simple RNN and LSTM based model. The proposed work is also solving the overfitting issue by applying L2 regularization and dropout methods. The best fit for validation is found with a 0.3 dropout value and $10^{-5}\,\lambda$ value under dropout and L2 regularization methods respectively. The dropout method performs better and achieves a lower value of MSE with good convergence compared to L2 regularization. Finally, self-attention is applied as the last layer in the model and the effect is assessed on two dimensions, for various extended time horizons (T_x, $4T_x$, $8T_x$, and $12T_x$) and with varying input windows (60, 80, and 120). The results show significant improvement in the performance with both dimensions. Future work of the current study includes an extension of attention application and model capability in learning the influence of temperature and humidity (recorded at same timesteps) in the prediction of air quality parameters.

References

1. World Health Assembly, 69. Health and the Environment: Draft Road Map for an Enhanced Global Response to the Adverse Health Effects of Air Pollution – Report by the Secretariat. – World Health Organization, 2016 (Online).
https://apps.who.int/iris/handle/10665/252673
2. Hu, R., X. Xy, S. K. Xu, M. Wang, L. Jiang, R. Wen, W. Lai, L. Guan. PM2.5 Exposure Elicits Oxidative Stress Responses and Mitochondrial Apoptosis Pathway Activation in HaCaT Keratinocytes. – Chin. Med. J., Vol. **130**, 2017, pp. 2205-2214.

3. P i a o, M., H. Y J, K. S h i l n i k o v a, J. J W. Particular Matter 2.5 Damages Skin Cells by Inducing Oxidative Stress Subcellular Organelle Dysfunction and Apoptosis. – Arch. Toxicol, Vol. **92**, 2018, pp. 2077-2.
4. C h i n g C h a n g, Y u, H. Y u a n L e e, J. L o n g H u a n g, C. H s u n C h i u, C. C h e n, C. T e n g W u. Risk Factors and Outcome Analysis in Children with Carbon Monoxide Poisonin – Pediatrics & Neonatology, Vol. **58**, 2017, No 2, pp. 171-177.
5. K u m a r, S., S. D i x i t, O. P. M u r t y. Fatal Carbon Monoxide Poisoning: A Lesson from a Retrospective Study at All India Institute of Medical Sciences, New Delhi. – Journal of Family Medicine and Primary Care, Vol. **6**, 2017, No 4, pp. 791-794.
6. H e, H.-D., M. L i, W. L. W a n g, Z.- Y. W a n g, Y. X u e. Prediction of PM2.5 Concentration Based on the Similarity in Air Quality Monitoring Network – Building and Environment, Vol. **137**, 2018, pp. 11-17.
7. Z h a n g, Y., Y. H e, J. Z h u. Research on Forecasting Problem Based on Multiple Linear Regression Model PM2.5. – J. Anhui Sci. Technol. Univ., Vol. **30**, 2016, No 3, pp. 92-97.
8. B a k e r, K. R., K. M. F o l e y. A Nonlinear Regression Model Estimating Single Source Concentrations of Primary and Secondarily Formed PM2:5. – Atmos. Environ., Vol. **45**, 2011, No 22, pp. 3758-3767.
9. W a n g, Z., Z. L o n g. PM2.5 Prediction Based on Neural Network. – In: Proc. of 11th Int. Conf. Intell. Comput. Technol. Automat. (ICICTA'18), Changsha, China, 2018, pp. 44-47.
10. C h u e n t a w a t, R., Y. K a n-N g a n. The Comparison of PM2:5 Forecasting Methods in the Form of Multivariate and Univariate Time Series Based on Support Vector Machine and Genetic Algorithm. – In: Proc. of 15th Int. Conf. Electr. Eng./Electron., Comput., Telecommun. Inf. Technol. (ECTI-CON'18), Chiang Rai, Thailand, 2018, pp. 572-575.
11. H o e k, G., R. B e e l e n, K. H o o g h, D. V i e n n e a u, J. G u l l i v e r, P. F i s c h e r, D. B r i g g s. A Review of Land-Use Regression Models to Assess Spatial Variation of Outdoor Air Pollution. – Atmos. Environ., Vol. **42**, 2008, pp. 7561-7578.
12. K a b u r l a s o s, V. G., I. N. A t h a n a s i a d i s, P. A. M i t k a s. Fuzzy Lattice Reasoning (FLR) Classifier and Its Application for Ambient Ozone Estimation. – International Journal of Approximate Reasoning, Vol. **45**, 2007, No 1, pp. 152-188.
13. Z h a n g, S., X. L i, Y. L i, J. M e i. Prediction of Urban PM2.5 Concentration Based on Wavelet Neural Network. – In: Proc. of 2018 Chinese Control and Decision Conference (CCDC'18), Shenyang, 2018, pp. 5514-5519.
14. S a l m a n, G., Y. H e r y a d i, E. A b d u r a h m a n, W. S u p a r t a. Single Layer & Multi-Layer Long Short-Term Memory (LSTM) Model with Intermediate Variables for Weather Forecasting. – Procedia Comput. Sci., Vol. **135**, 2018, pp. 89-98.
15. T s a i, Y., Y. Z e n g, Y. C h a n g. Air Pollution Forecasting Using RNN with LSTM. – In: Proc. of IEEE 16th Int. Conf. Dependable, Autonomic Secure Comput., 16th Int. Conf. Pervasive Intell. Comput., 4th Int. Conf. Big Data Intell. Comput. Cyber. Sci. Technol. Congr., Athens, Greece, 2018, pp.1074-1079.
16. K o n g, W., Z. Y. D o n g, Y. J i a, D. J. H i l l, Y. X u, Y. Z h a n g. Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network. – IEEE Trans. Smart Grid, Vol. **10**, 2017, No 1, pp. 841-851.
17. Z h a o, J., F. D e n g, Y. C a i, J. C h e n. Long Short-Term Memory – Fully Connected (LSTM-FC) Neural Network for PM2.5 Concentration Prediction. – Chemosphere, Vol. **220**, 2019, pp. 486-492.
18. K i m, S., J. M. L e e, J. L e e, J. S e o. Deep-Dust: Predicting Concentrations of Fine Dust in Seoul Using LSTM. – ArXiv, Vol. abs/1901.10106, 2019.
19. Q i n, D., J. Y u, G. Z o u, R. Y o n g, Q. Z h a o, B. Z h a n g. A Novel Combined Prediction Scheme Based on CNN and LSTM for Urban PM2.5 Concentration. – IEEE Access, Vol. **7**, 2019, pp. 20050-20059.
20. X a y a s o u k, T., H. L e e, G. L e e. Air Pollution Prediction Using Long Short-Term Memory (LSTM) and Deep AutoEncoder (DAE) Models. – Sustainability, Vol. **12**, 2020, No 6.
21. T a o, Q., F. L i u, Y. L i, D. S i d o r o v. Air Pollution Forecasting Using a Deep Learning Model Based on 1D Convnets and Bidirectional GRU. – IEEE Access, Vol. **7**, 2019, pp. 76690-76698.

22. V e r m a, I., R. A h u j a, H. M e i s h e r i, L. D e y. Air Pollutant Severity Prediction Using Bi-Directional LSTM Network. – In: Proc. of IEEE/WIC/ACM International Conference on Web Intelligence (WI), Santiago, 2018, pp. 651-654.
23. T o n g, W., L. L i, X. Z h o u, et al. Deep Learning PM2.5 Concentrations with Bidirectional LSTM RNN. – Air Qual Atmos Health, Vol. **12**, 2019, pp. 411-423.
24. B e n g i o, Y., P. S i m a r d, P. F r a s c o n i. Learning Long-Term Dependencies with Gradient Descent is Difficult. – IEEE Trans. Neural Netw., Vol. **5**, 1994, No 2, pp. 157-166.
25. P a s c a n u, R., T. M i k o l v, Y. B e n g i o. On the Difficulty of Training Recurrent Neural Networks. – In: Proc. of 30th International Conference on Machine Learning, Atlanta, Georgia, USA, Vol. **28**, 2013, pp. III-1310-III-1318.
26. H o c h r e i t e r, S., J. S c h m i d h u b e r. Long Short-Term Memory. – Neural Comput., Vol. **9**, 1997, No 8, pp. 1735-1780.
27. S c h u s t e r, M., K. K. P a l i w a l. Bidirectional Recurrent Neural Network – IEEE Transactions on Signal Processing, Vol. **45**, 1997, No 11, pp. 2673-2681.
28. G r a v e s, A., J. S c h m i d h u b e r. Frame Wise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. – Neural Networks, Vol. **18**, 2005, No 5, pp. 602-610.
29. G r a v e s, A., N. J a i t l y, A. M o h a m e d. Hybrid Speech Recognition with Deep Bidirectional Lstm. – In: Proc. of 2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU'13), Olomouc, Czech Republic, IEEE, 2013, pp. 273-278.
30. L e C u n, Y., Y. B e n g i o, G. H i n t o n. Deep Learning. – Nature, Vol. **521**, 2015, No 7553, pp. 436-444.
31. H e r m a n s, M., S. B e n j a m i n. Training and Analysing Deep Recurrent Neural Networks. – In: Advances in Neural Information Processing Systems, Vol. **26**, Nevada, USA, 2013, pp. 190-198.
32. P a s c a n u, R., C. G u l c e h r e, K. C h o, Y. B e n g i o. How to Construct Deep Recurrent Neural Networks. – In: Proc. of Second International Conference on Learning Representations, ICLR, Banff, Canada, 2014.
33. C h o i, H., K. C h o, Y. B e n g i o. Fine-Grained Attention Mechanism for Neural Machine Translation. – Neurocomputing, Vol. **284**, 2018, pp. 171-176.
34. B a h d a n a u, D., K. C h o, Y. B e n g i o. Neural Machine Translation by Jointly Learning to Align and Translate. – In: Proc. of 3rd International Conference on Learning Representations (ICLR'15), San Diego, CA, USA, 2015, pp. 7-9.
35. L i, W., D. G u o, X. F a n g. Multimodal Architecture for Video Captioning with Memory Networks and an Attention Mechanism. – Pattern Recognit. Lett., Vol. **105**, 2018, pp. 23-29.
36. B a r o t., V., V. K a p a d i a, S. P a n d y a. QoS Enabled IoT Based Low-Cost Air Quality Monitoring System with Power Consumption Optimization. – Cybernetics and Information Technologies, Vol. **20**, 2020, No 2, pp. 122-140.
37. A u n g i e r s., J. Time Series Prediction Using LSTM Deep Neural Network. – Altum Intelligence, 2018 (Online).
https://www.altumintelligence.com/articles/a/Time-Series-Prediction-Using-LSTM-Deep-Neural-Networks
38. K i n g m a, D. P., J. B a. Adam: A Method for Stochastic Optimization. – CoRR, Vol. **abs/1412.6980**, 2015 (Online).
https://arxiv.org/abs/1412.6980
39. S r i v a s t a v a, N., G. H i n t o n, A. K r i z h e v s k y, S. I l y a, R. S a l a k h u t d i n o v. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. – Journal of Machine Learning Research, Vol. **15**, 2014, pp. 1929-1958.
40. P e r e y r a, G., G. T u c k e r, J. C h o r o w s k i, L. K a i s e r, G. E. H i n t o n. Regularizing Neural Networks by Penalizing Confident Output Distributions. – CoRR, Vol. **abs/1701.06548**, 2017.
41. C h a n g, Y.-S., H.-T. C h i a o, S. A b i m a n n a n, Y.-P. H u a n g, Y.-T. T s a i, K.-M. L i n. An LSTM-Based Aggregated Model for Air Pollution Forecasting. – Atmospheric Pollution Research, Vol. **11**, 2020, No 8, pp. 1451-1463.
42. K a y a, K., Ş. G ü n d ü z Ö ğ ü d ü c ü. Deep Flexible Sequential (DFS) Model for Air Pollution Forecasting. – Sci. Rep., 10: 3346, 2020.

43. G u l, S., G. M. K h a n. Forecasting Hazard Level of Air Pollutants Using LSTM's. – In: Proc. of IFIP International Conference on Artificial Intelligence Applications and Innovations (AIAI'20), IFIP Advances in Information and Communication Technology, Vol. **584**, 2020, pp. 143-153.
44. M a, J., Z. L i, C. P. J. C h e n g, Y. D i n g, C. L i n, Z. X u. Air Quality Prediction at New Stations Using Spatially Transferred Bi-Directional Long Short-Term Memory Network. – Science of the Total Environment, Vol. **705**, 2020, 135771.

Received: 28.07.2021; Accepted: 17.12.2021