Indonesian Journal of Electrical Engineering and Computer Science
Vol. 22, No. 3, June 2021, pp. 1611~1618
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v22.i3.pp1611-1618

A hybrid deep learning model for air quality time series prediction

Samit Bhanja[1], Abhishek Das[2]
[1]Department of Computer Science, Government General Degree College, Hooghly, India
[2]Department of Computer Science and Engineering, Aliah University, Kolkata, India

Article Info

Article history:

Received Dec 4, 2020
Revised May 27, 2021
Accepted Jun 13, 2021

Keywords:

Air quality
CNN
Deep learning
LSTN
Time-series forecasting

ABSTRACT

Air quality (mainly PM2.5) forecasting plays an important role in the early detection and control of air pollution. In recent times, numerous deep learning-based models have been proposed to forecast air quality more accurately. The success of these deep learning models heavily depends on the two key factors viz. proper representation of the input data and preservation of temporal order of the input data during the feature's extraction phase. Here we propose a hybrid deep neural network (HDNN) framework to forecast the PM2.5 by integrating two popular deep learning architectures, viz. Convolutional neural network (CNN) and bidirectional long short-term memory (BDLSTM) network. Here we build a 3D input tensor so that CNN can extract the trends and spatial features more accurately within the input window. Here we also introduce a linking layer between CNN and BDLSTM to maintain the temporal ordering of feature vectors. In the end, our proposed HDNN framework is compared with the state-of-the-art models, and we show that HDNN outruns other models in terms of prediction accuracy.

Corresponding Author:

Abhishek Das
Department of Computer Science and Engineering
Aliah University
IIA/27, NewTown, Kolkata 700160, India
Email: adas@aliah.ac.in

1. INTRODUCTION

Air pollution is one of the serious issues during this urbanization era, and PM2.5 is one of the most important pollutants that influence air pollution. Long-time exposer in the environment with a high concentration of PM2.5 causes serious public health issues, which include asthma, chronic bronchitis, and heart disease. So, the forecasting of PM2.5 has a severe impact on our living environment and also the physical health of human beings. Air quality depends on various factors, these factors are very much dynamic and complex in nature. These factors vary over time. Thus, air quality data is time-series data. Time-series data is a progression of data points recorded in time requests. Time series information may be univariate or multivariate. In multivariate time-series (MTS) information, a bunch of variables moves in time order. Since the air quality data are collected from various sensors over fixed time intervals, so air quality forecasting is a multivariate time-series forecasting (MTSF) problem.

In the last decade, machine learning achieves great success in time-series prediction, only because of the availability of massive data samples and significant improvement of computing power. Some of the popular machine learning algorithms are support vector machines (SVM), Naive Bayes, k-means, and random forest. All of these machine learning algorithms suffer from the following challenges viz. 1. long design time 2. domain expert's knowledge is required for feature designing. Recently, deep learning methods

have become a popular tool for multivariate time series data processing [1]-[4]. It is a subset of machine learning methods, and its objective is to learn the model parameters from the representation of the data. Deep neural networks (DNNs) are the backbone for the implementation of deep learning architecture. A neural network with more than one hidden layer is called a DNN, where the output of a layer turns into the input of the subsequent layer. Due to this layered architecture, the DNNs become a popular feature learning tool for extracting features from the historical dataset. In recent times, the two most popular deep learning techniques are convolutional neural network (CNN) [5] and recurrent neural network (RNN) [6], [7]. These networks have received interest in different application areas, like computer vision, natural language processing, and time series forecasting. Long short-term memory (LSTM) [8], [9] recurrent neural networks are the modified version of the RNNs, that overcomes some of the shortfalls of the RNNs. CNN [10], [11] has produced a great deal of success in image recognition, and image classification. Whereas, LSTM achieves enormous success in machine translation, natural language processing, and speech recognition. In contrast with the MTS data, CNNs are popular for their feature learning capabilities within an input window but are not useful for exploring the temporal features. On the other hand, LSTMs are outstanding to model the long-term dependencies and temporal features extraction from univariate time series data, but not suitable for extracting the complex and spatial features from multivariate time-series data. In the last few years, different models [12]-[18] is proposed by combining two or more deep learning architectures for MTSF problems, and these models have shown excellent results. Du *et al*. [15] proposed a hybrid deep learning model for air health monitoring. This model is constructed by combining 1D-CNNs and bidirectional LSTMs. The authors show that the combined model produced better results compared to the shallow deep learning and machine learning models. The transferred bi-directional long short-term memory (TL-BLSTM) model is proposed in [19] predict the air quality. Here the author used the bidirectional LSTM to extract the temporal dependencies of PM2.5 and use transfer learning to transfer the knowledge from the smaller temporal window to the larger temporal window. But these hybrid deep learning models do not concentrate on the proper representation of the input data and do not maintain the proper temporal ordering of the extracted features by the first phase of the models.

So deviating from the recent research work, here we propose a hybrid deep neural network (HDNN) framework by combining two most popular deep learning architectures such as convolutional neural network (CNN) and bidirectional long short-term memory (BDLSTM) recurrent neural network for air quality forecasting. In this framework, we propose a 3D tensor formation scheme to convert the multivariate time-series data to image like 3D data as input of the CNN module. We also introduce a linking layer between the CNN module and the LSTM module to maintain the temporal order of the features extracted by the CNN module. The rest of the paper is organized as follows: In section 2, we describe the research methodology. The experimental process is presented in section 3. Section 4 includes discussion about the detailed results achieved, and the conclusion is drawn in section 5.

2. RESEARCH METHODOLOGY

2.1. Dataset description and correlation analysis

In this work, we use the air pollutant and the meteorology datasets of Sydney, Australia and Delhi, India. The detail description of the datasets is represented in the Table 1. In this study, we consider the PM2.5 as our target output to predict the air quality, and other air pollutants and meteorological data are treated as the input of the model. To reduce the number of input parameters without degrading the overall impact on the target output, here the correlation analysis is conducted on the input dataset using the Pearson's correlation coefficient . We remove one of the parameters from a pair of the parameter, which are highly correlated.

Table 1. Dataset description

Location	Pollutants	Meteorological Factors	Periods	No. of rows	Frequency
Sydney	CO, NO, NO2, SO2 OZONE, PM10. PM2.5	Temperature, Wind Direction Humidity, Wind Speed	01.01.2018 24:00 Hours - 31.12.2019 23:00 Hours	17403	Hour
Delhi	CO, NO, NO2, SO2 OZONE, PM10. PM2.5	Temperature, Wind Direction Wind Speed, Humidity, Pressure	01.01.2015 03:00 Hours - 24.04.2017 23:00 Hours	20277	Hour

2.2. Data preprocessing

Since the air pollution data is collected from different air pollution sensors, so there is a possibility of missing and noisy data [20]. The performance of any deep learning models heavily depends on the quality of the input data. So, to produce high-quality data from it, data preprocessing is an important part of any deep

learning algorithm. In this experiment, we replace every missing value by the mean value of just before and after the missing value. In the next step, we normalize the reduced dataset by min-max normalization technique. The data normalization [21] is an important step of data preprocessing because the time-series data can vary over a large range and also produce a good quality of data from it, the data must be scale-down to a certain range.

2.3. Input tensor formation

In this section, we represent the input tensor formation scheme. CNN achieves a great deal of success in image classification, and image recognition. Image data is different from the time-series data. Hence, to feed the time-series data into the CNN model we apply the image-like tensor formation scheme to encode the multidimensional time-series data. Each red-green-blue (RGB) image consists of three 2D arrays of pixels, which means each RGB image can be represented as a 3D tensor of the dimension $h \times w \times 3$, here 3 represent the depth of the tensor. By applying this encoding method, we transform our multivariate time-series data into 3D tensor, where the number of sensors represents the depth of the tensor.

The input tensor formation scheme consists of two-phase as depicted in Figure 1. It represents the tensor formation technique for four sensors (input) signals (a, b, c and d) where the length of the sliding window $SW_1 = 3$. In the first phase, we construct a tensor of the dimension $3 \times 1 \times 1$ from every single sensor signal within the sliding window and then such four 3D tensors of that sliding window are concatenated in the direction of z axis to form a 3D tensor of the dimension $3 \times 1 \times 4$, as depicted in the Figure 1 (a). In this way, we construct a series of 3D tensors by striding the sliding window with the striding length SL_1 $(1 > SL_1 \leq SW_1)$ from the multivariate time-series dataset. In the next phase, all three consecutive 3D tensors of the sliding window ($SW_2 = 3$) are concatenated to the direction of x axis to form an input tensor of the size $3 \times 3 \times 4$, as depicted in the Figure 1 (b). In such a way, we construct a series of 3D input tensors by striding the sliding window with the striding length SL_2, where $1 < SL_2 \leq SW_2$.

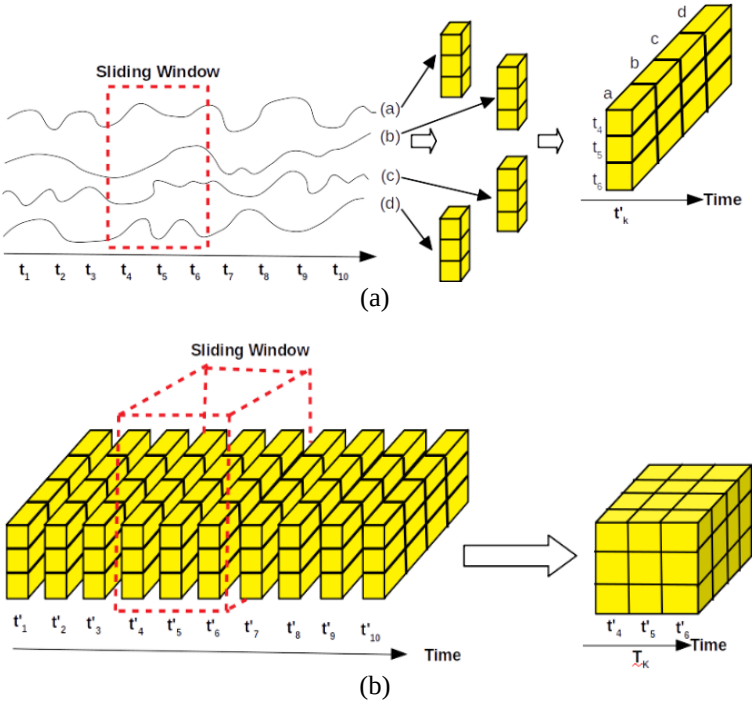


Figure 1. Input tensor formation: (a) phase one and (b) phase two

2.4. Forecasting model construction

Here we describe our proposed forecasting model (HDNN). Figure 2 represents the overall architecture of the HDNN forecasting model. This forecasting model is constructed by two popular deep learning architectures-CNN and LSTM. Here our HDNN forecasting model is divided into four modules-CNN Module, linking module, LSTM Module, and fully connected module.

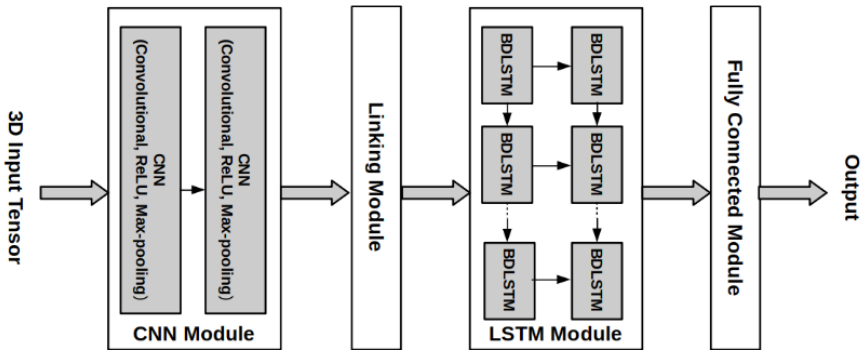


Figure 2. Network architecture of proposed HDNN model

2.4.1. CNN module

CNN is a deep learning architecture that is specially designed to process image data. CNNs are widely used in the area of image recognition, image classifications, and object detection. As recent research work [22], [23] shows that the deep learning model with the combination of CNN and LSTM architectures produces more accurate results for MTSF problems compared to the shallow deep learning models. Thus the extracted higher-order features by CNN construct helps the outer level LSTM construct in our proposed work for more accurate prediction.

In the proposed model, the CNN module is consists of stacking of two CNN layers where the output of one CNN layer is considered as the input of the next CNN layer as shown in Figure 2. Each CNN layer performs three different operations on the input data-convolutional, ReLU [24], and pooling. The inputs of the CNN module are the 3D input tensors of the normalized multivariate time-series data as illustrated in section 2.3. The operations of the CNN layer are presented in Figure 3. Here, $X$is the 3D input tensor of size 6×6 with depth 3and W is the feature detector of size 3×3 with $m_k = 4$ channels (filters). The convolution operation with a stride length of 1 will generation 4 feature maps of the dimension 4×4. Then the non-linear activation functions ReLU is applied on the feature map. The formula of the ReLU is as follows:

$$f(z) = \begin{cases} z, if\ z > 0 \\ 0, if\ z \leq 0 \end{cases} \quad (1)$$

After that, we carry out the max-pooling operation to pool the maximum over each interval as its output. If the max-pooling size is 2×2 then it, reduced to each feature matrix of sized 2×2 as shown in Figure 3. In this way, the first CNN layer compressed the input feature matrix$X$. This compressed output of the first CNN layer is the local feature vectors of X. These local feature vectors are then fed as the input of the second CNN layer, and in this way, the second CNN layer generates the higher-order feature vectors.

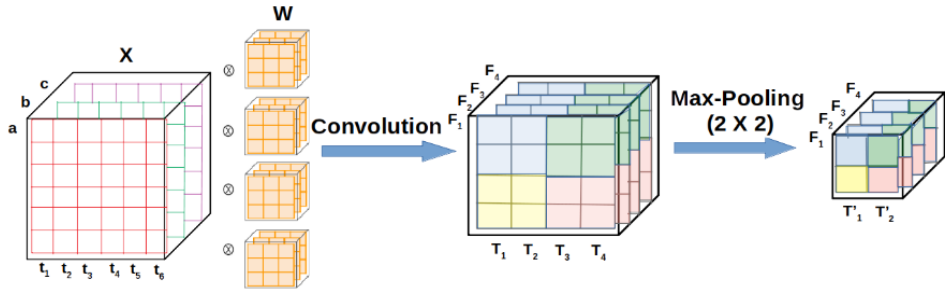
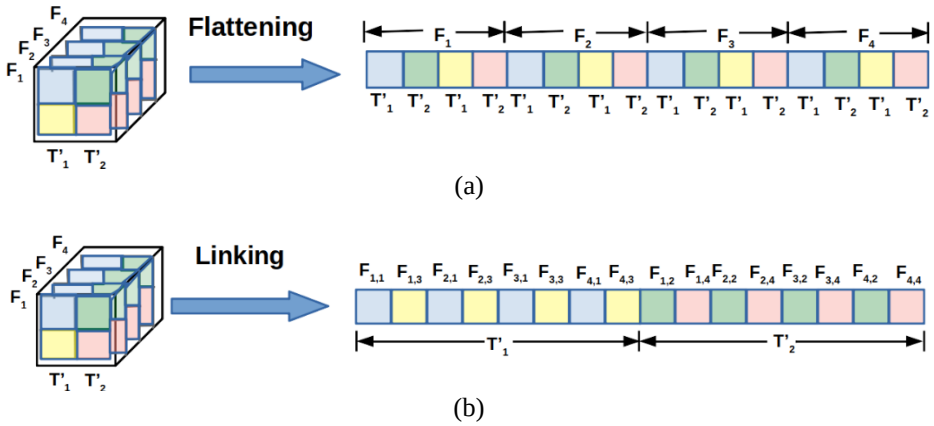


Figure 3. Operation of CNN layer

2.4.2. Linking of CNN module with LSTM module

The output of the CNN module is the feature vectors, expressed as F_i where $i = 1,2,\therefore, m_k$ and each feature vector has the length L, so we have $F_i = \{F_{i,1}, F_{i,2}, \therefore, F_{i,L}\}$. Here each feature vector is the higher-order feature of the original multivariate time-series dataset along the time axis. Analogous to the benchmark CNN, where the flattening layer is used to flatten the feature vectors as shown in Figure 4 (a),

here we introduced a linking module. In this module, we redistributed the components of the feature vectors along the time axis as shown in Figure 4 (b). These regenerated feature vectors are represented in a more relevant time order compared to the flattening layer of benchmark CNN. These newly generated feature vectors are used as the input of the LSTM module.



(a)

(b)

Figure 4. These figures are; (a) Flattening layer and (b) linking module

2.4.3. LSTM module

In this LSTM module, we create two layers of bidirectional LSTMs (BDLSTMs) and each BDLSTM layer can be unrolled to *L* number of BDLSTM units along the time axis as presented in Figure 2. Conventional LSTM network can only process in the forward direction and as a result, it can miss some of the useful information during the extraction of the temporal features. But the BDLSTM network can process in both the forward and backward direction, so it is efficient to extract the temporal features compared to the conventional LSTM network. This is the reason why we choose the BDLSTM [25], [26] over LSTM. Due to this bidirectional property, each BDLSTM unit accumulates the feature information from both sides of the multivariate time-series data within its time frame. In this way, each unit extracts temporal features.

2.4.4. Fully connected module

It is the last module of our proposed model. This module consists of two layers–a dropout layer and a dense layer. The dropout layer randomly selects some of the nodes and dropout them. This is a regularization technique that is used to dela with the overfitting problem. The next layer of this module is the dense layer. This layer connects the output vectors from all the BDLSTM units of the second BDLSTM layer to a single node. Finally, the output of this layer is the forecasted result of our proposed work.

3. EXPERIMENT

3.1. Experimental setup

In this work, all the experiments are conducted in Python programming language with Keras as the deep learning library. Here we forecast the next hour PM2.5 based on the previous 8 hours pollutants concentration (CO, NO2, and SO2) and the meteorological factors (temperature, wind direction, humidity, and wind speed). We have used 70% of the data for training purposes, 15% of data for the validation purpose, and we test the performance of the model by the remaining 15% of data. In the input tensor formation scheme, for the first phase, we set the length of the sliding window $SW_1 = 5$, and striding-length of the sliding window $SL_1 = 1$. In the second phase, we set the length of the sliding window $SW_2 = 4$, and striding-length $SL_2 = 2$. Therefore, the dimension of the constructed 3D input tensors becomes $5 \times 4 \times 7$.

The selection of the hyperparameter plays a crucial role in the performance of any deep learning model. After the fine-tuning of our forecasting model, we set the hyperparameters are as follows:
For the first CNN layer, the number of filters $m_1 = 16$, the filter size is 4, striding length $s_1 = 1$, and the window size of the max-pooling is 2. For the second CNN layer, the number of filters $m_2 = 12$, the filter size is 3, striding length $s_2 = 1$, and the window size of the max-pooling is 2. For the first BDLSTM layer, the number of neurons is 64. and in the second layer, number of neurons is 32. The training batch size is 16, the model's optimizer is stochastic optimization (Adam), and the number of iteration is 100. The dropout regularization

technique is used to reduce the overfitting problem of the model. After the fine-tuning of our model, we set the dropout rate to 0.6, and this dropout layer is added before the final dense layer of the fully connected module.

3.2. Evaluation

To analyze the performance of the model, here we consider three performance matrics, viz. Root mean squared error (RMSE), mean absolute percentage error (MAPE), and symmetric mean absolute percentage error (SMAPE). These errors are calculated using the following formula.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(o_t - \hat{o}_t)^2} \quad (2)$$

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}(|o_t - \hat{o}_t|) \times 100 \quad (3)$$

$$SMAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|o_t - \hat{o}_t|}{(|o_t + \hat{o}_t|)/2} \quad (4)$$

Where o_t is actual value and $\hat{o}_t$ is the predicted value at time-stamp *t*.

In this study, we compare the performance of our proposed HDNN with the four popular shallow models and three states-of-the-art hybrid deep learning models. The shallow models are SVR, Auto Regressive Integrated Moving Average (ARIMA), CNN, and BDLSTM, and the hybrids models are DAQFF[15] and TL-BLSTM[19]. The parameters of each of the models are shown in Table 2.

Table 2. Parameters of different models

Model	Parameters	Input
SVR	kernel=rbf, C=100, gamma=0.1	PM2.5 Time-series
ARIMA	p=8, d=1, q=1	PM2.5 Time-series
CNN	filters=64, kernel=3 × 3, MaxPooling= 2 × 2	Multivariate Time-series
BDLSTM	2 layers of BDLSTM, Layer1: 80 Neurons, Layer2: 32 Neurons	Multivariate Time-series
DAQFF [15]	2 layers of 1D-CNN and two layers of LSTM	Multivariate Time-series
TL-BLSTM [19]	3 layers of BDLSTM	Multivariate Time-series

4. RESULT AND DISCUSSION

In Figures 5 and 6, we graphically represent (zoomed for 500 hours) the predicted PM2.5 concentration for the both the testing datasets. From these figures, it is quite clear that the HDNN successfully predicts the fluctuation of PM2.5 concentration and produces a stable performance. The results of the comparative analysis are presented in Tables 3 and 4. From these tables, we can observe that the shallow machine learning models SVR and ARIMA produces almost the same result, whereas the shallow deep learning models BDLSTM and CNN produces better performance compared to the shallow machine learning models in terms of RMSE and MAPE. On the other hand, testing errors of all the hybrid deep learning models are significantly low compared to the shallow deep learning models. From these tables, we can also see that our proposed framework HDNN yields the lowest testing errors compared to all other models. The results confirm that HDNN not only captures the features of local trends and also the long term temporal dependencies from the historical dataset pollutant concentrations and meteorological factors.

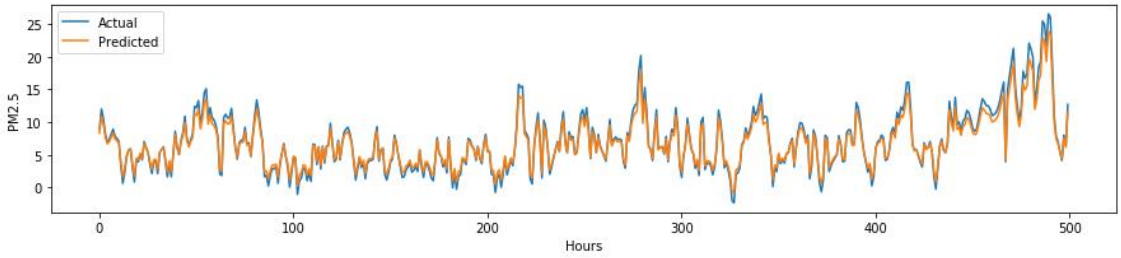


Figure 5. Forecasting results of PM2.5 concentration of Sydney

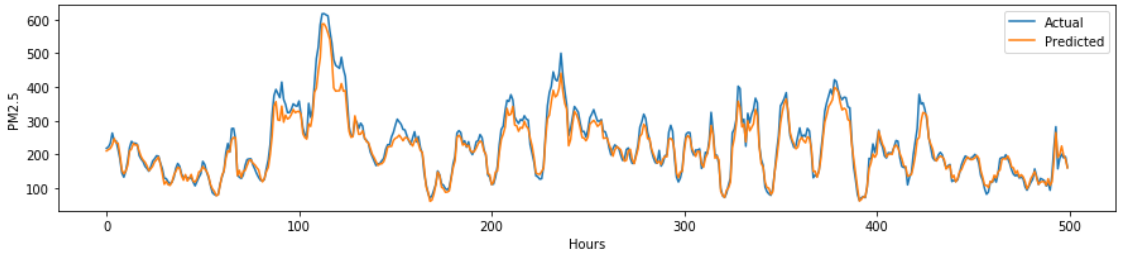


Figure 6. Forecasting results of PM2.5 concentration of Delhi

Table 3. The performance comparison of different models for Sydney dataset

Model	SVR	ARIMA	BDLSTM	CNN	DAQFF	TL-BLSTM	HDNN
RMSE	21.7531	20.1904	14.2091	15.3711	11.0753	11.3574	9.3042
MAPE	35.7816	34.0873	28.2431	29.6108	19.3281	19.4384	17.8551
SMAPE	0.30125	0.2653	0.1810	0.1875	0.1426	0.1397	0.1173

Table 4. The performance comparison of different models for Delhi dataset

Model	SVR	ARIMA	BDLSTM	CNN	DAQFF	TL-BLSTM	HDNN
RMSE	32.1920	31.0847	26.9921	28.3008	24.3520	24.1083	23.8171
MAPE	18.5490	15.1008	13.2871	13.9812	8.0931	8.1159	7.0846
SMAPE	0.3829	0.2546	0.2409	0.2571	0.1509	0.1532	0.1481

5. CONCLUSION

Air quality forecasting is of utmost importance for the early detection of air pollution. In this study, to predict the air quality more accurately, we propose a hybrid deep learning framework. In this framework, we introduce an innovative 3D input tensor formation technique to convert the time-series data to a 3D image like data for feeding it to the CNN module. Here we also introduce a linking layer between the CNN module and the LSTM module. The experimental results show that the proposed framework outshined other state-of-the-art models in terms of different testing errors. The proposed framework also provides stable performance, especially in the periods of wave peak and wave valley. So, it can be concluded that the proper input data representation plays a vital role in the performance of any deep learning models. Therefore, this framework will help the administrative authority for controlling the air pollution well in advance and able to warn the citizen. Here the framework has applied for only the single-time-step ahead prediction. The performance of the framework has not been analyzed for the multi-time-step ahead prediction. In the future, we will like to apply our model for multi-time-step ahead prediction.

REFERENCES

[1] J. F. Torres, A. Galicia, A. Troncoso, and F. Martinez-Álvarez, "A scalable approach based on deep learning for big data time series forecasting," *Integr. Comput. Aided. Eng.*, vol. 25, no. 4, pp. 335–348, 2018, doi: 10.3233/ICA-180580.

[2] J. C. B. Gamboa, "Deep learning for time-series analysis," *arXiv Prepr. arXiv1701.01887*, 2017.

[3] B. S. Freeman, G. Taylor, B. Gharabaghi, and J. Thé, "Forecasting air quality time series using deep learning," *J. Air & Waste Manag. Assoc.*, vol. 68, no. 8, pp. 866–886, 2018, doi: 10.1080/10962247.2018.1459956.

[4] S. Bhanja and A. Das, "Deep Neural Network for Multivariate Time-Series Forecasting," in *Proceedings of International Conference on Frontiers in Computing and Systems*, 2020, pp. 267–277.

[5] Y. LeCun, *et al.*, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, 1990, pp. 396-404.

[6] D. E. Rumelhart, G. E. Hinton, R. J. Williams, and others, "Learning representations by back-propagating errors," *Cogn. Model.*, vol. 5, no. 3, p. 1, 1988.

[7] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179-211, 1990, doi: 10.1016/0364-0213(90)90002-E.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735-1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[9] M. Rhanoui, S. Yousfi, M. Mikram, and H. Merizak, "Forecasting financial budget time series: ARIMA random walk vs LSTM neural network," *IAES International Journal of Artificial Intelligence*, vol. 8, no. 4, pp. 317-327, 2019, doi: 10.11591/ijai.v8.i4.pp317-327.

[10] N. A. Rahmad, N. A. J. Sufri, N. H. Muzamil, and M. A. As'ari, "Badminton player detection using faster region convolutional neural network," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 14, no. 3, pp. 1330–1335, 2019, doi: 0.11591/ijeecs.v14.i3.pp1330-1335.
[11] N. Kasim, N. Rahman, Z. Ibrahim, and N. N. A. Mangshor, "Celebrity Face Recognition using Deep Learning," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 12, no. 2, pp. 476-481, 2018, doi: 10.11591/ijeecs.v12.i2.pp476-481.
[12] S. Du, T. Li, Y. Yang, and S.-J. Horng, "Deep air quality forecasting using hybrid deep learning framework," *arXiv Prepr. arXiv1812.04783*, 2018.
[13] G. Yang, H. Lee, and G. Lee, "A Hybrid Deep Learning Model to Forecast Particulate Matter Concentration Levels in Seoul, South Korea," *Atmosphere (Basel).*, vol. 11, no. 4, p. 348, 2020, doi: 10.3390/atmos11040348.
[14] Z. Han, J. Zhao, H. Leung, K. F. Ma, and W. Wang, "A review of deep learning models for time series prediction," *IEEE Sens. J.*, vol. 21, no. 6, pp. 7833-84, 2019, doi: 10.1109/JSEN.2019.2923982.
[15] S. Du, T. Li, Y. Yang, and S.-J. Horng, "Deep air quality forecasting using hybrid deep learning framework," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 6, pp. 2412-2424, 2019, doi: 10.1109/TKDE.2019.2954510.
[16] S. Bhanja and A. Das, "Deep Learning-based Integrated Stacked Model for the Stock Market Prediction," *Int. J. Eng. Adv. Technol.*, vol. 9, no. 1, pp. 5167-5174, Oct. 2019, doi: 10.35940/ijeat.A1823.109119.
[17] Q. Sun, Y. Zhu, X. Chen, A. Xu, and X. Peng, "A hybrid deep learning model with multi-source data for PM 2.5 concentration forecast," *Air Quality, Atmosphere & Health*, vol. 14, no. 4, pp. 1–11, 2020, doi: 10.1007/s11869-020-00954-z.
[18] B. Ghose and Z. Rehena, "A Deep Learning Approach for Predicting Air Pollution in Smart Cities," in *Computational Intelligence and Machine Learning*, Springer, 2020, pp. 29-38, doi: 10.1007/978-981-15-8610-1_4.
[19] J. Ma, J. C. P. Cheng, C. Lin, Y. Tan, and J. Zhang, "Improving air quality prediction accuracy at larger temporal resolutions using deep learning and transfer learning techniques," *Atmos. Environ.*, vol. 214, p. 116885, 2019, doi: 10.1016/j.atmosenv.2019.116885.
[20] N. H. Abd Rahman and M. H. Lee, "Artificial neural network forecasting performance with missing value imputations," *IAES International Journal of Artificial Intelligence*, vol. 9, no. 1, pp. 33-39, 2020, doi: 10.11591/ijai.v9.i1.pp33-39.
[21] S. Bhanja and A. Das, "Impact of Data Normalization on Deep Neural Network for Time Series Forecasting," *arXiv Prepr. arXiv1812.05519*, 2018.
[22] I. Livieris, E. Pintelas, and P. Pintelas, "A CNN--LSTM model for gold price time-series forecasting," *Neural Comput. Appl.*, vol. 32, no. 23, pp. 17351–17360, 2020, doi: 10.1007/s00521-020-04867-x.
[23] T. Li, M. Hua, and X. Wu, "A hybrid CNN-LSTM model for forecasting particulate matter (PM2. 5)," *IEEE Access*, vol. 8, pp. 26933–26940, 2020, doi: 10.1109/ACCESS.2020.2971348.
[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.
[25] X. Ma, J. Zhang, B. Du, C. Ding, and L. Sun, "Parallel architecture of convolutional bi-directional LSTM neural networks for network-wide metro ridership prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 6, pp. 2278-2288, 2018, doi: 10.1109/TITS.2018.2867042.
[26] A. Sagheer and M. Kotb, "Time series forecasting of petroleum production using deep LSTM recurrent networks," *Neurocomputing*, vol. 323, pp. 203-213, 2019, doi: 10.1016/j.neucom.2018.09.082.