Weather Data Forecasting Using Long Short-Term Memory Model

Ibukunoluwa Adetutu Olajide
Department of Electrical and Electronics Engineering
The Federal University of Technology,
Akure, Nigeria
iaadebanjo@futa.edu.ng

***Abstract*— Weather forecasts have a broad range of applications, making them indispensable in numerous fields. Accurate weather forecasting enables informed decision-making, enhances safety, and supports efficient resource management. With the random nature of the atmosphere and the large computational power required when using traditional methods for forecasting, there arose a need to develop other methods using deep learning. In this paper, a two-layer Long Short-Term Memory model was developed and used to predict a fourteen (14) hours period of weather data. A stationarity check using Augmented Dickey-Fuller (ADF) Test was performed for the historical data obtained for Akure metropolis in Nigeria. With the data passing the test, the LSTM forecast model was built and prediction was made. The Root Mean Squared Error (RMSE) values of 0.078, 0.034, 1.531, 0.363, and 0.460 were obtained after prediciton for visibility, precipitation, humidity, atmospheric pressure, and temperature respectively. LSTM gave a promising performance in forecasting weather entities.**

Keywords*— *forecasting; weather prediction; long short-term memory; stationarity check.

I. Introduction

Weather forecasting plays a crucial role across various sectors, including science and agriculture, by predicting the state of the atmosphere at specific times and locations. This information is essential for planning daily activities, designing structures, and formulating policies. The process involves collecting quantitative data about the current atmospheric conditions and applying scientific principles to project future changes. However, the chaotic nature of the atmosphere, the substantial computational power required to solve atmospheric equations, measurement errors in initial conditions, and an incomplete understanding of atmospheric processes contribute to a decline in forecast accuracy over time [1]. Weather warnings protect life and property, while temperature and precipitation forecasts are crucial for agriculture and commodity markets. Utility companies use temperature forecasts to estimate demand. Daily weather forecasts help individuals decide what to wear and plan activities, particularly those affected by heavy rain, snow, and wind chill [1][2]. Weather forecasts have a broad range of applications, making them indispensable in numerous fields. Accurate weather forecasting enables informed decision-making, enhances safety, and supports efficient resource management. Consequently, addressing this complex task has always been a priority in research [3]. Most weather stations globally utilize numerical weather prediction models, such as the Weather Research and Forecasting (WRF) model, since its public release in 2000. These models require substantial computational power to solve large systems of non-linear equations [4]. In addressing this problem, an alternate method is to use machine learning algorithms adaptation. Machine learning is a promising data-driven method that uses historical data to make predictions. It can be divided into two (2) forms, namely, the shallow and deep learning [3]. The choice of either learning algorithm is determined by application of use, programming constraints and advantages.

The advantages of deep learning models for use in weather forecasting are extensive over shallow learning techniques. Deep learning models can process and analyze large sets of complex and high-dimensional data, making them well-suited for handling the intricate patterns in meteorological data [5]. Unlike traditional models, deep learning algorithms can automatically extract relevant features from raw data, reducing the need for manual feature engineering [6]. Also, deep learning models have been found to be able to capture non-linear relationships and interactions, thereby improving accuracy [7]. In adaptation to new and diverse datasets, deep learning models does better than its shallow counterpart, leading to flexibility and scalability for various weather predictions [8]. Further advantage was highlighted in [9], as being able to learn the entire mapping from input data to prediction in a single framework, and integration with big data technologies [10].

Furthermore, models like Long Short-Term Memory (LSTM) networks are particularly effective at capturing temporal dependencies and trends over time, which are crucial for accurate weather forecasting [11], thereby informing its choice in this paper. This paper utilizes the Long Short-Term Memory (LSTM) networks for weather data forecasting using two-layered network architecture. The design methodology involved pre-processing of weather data, normalization and transformation of data, data stationarity test, definition of Long Short-Term Memory (LSTM) model architecture, training and validation, forecasting and evaluation.

II. LITERATURE REVIEW

A. *Related Work*

Several weather forecasting research are available in literature. A few are selected in this section. Weather is considered to be dynamic, multi-dimensional, and non-linear in nature [12] . Extensive research has led to the development of various methods for weather prediction. Traditionally, the most accurate forecasting has relied on mathematical simulations, which use generative techniques to model atmospheric dynamics through physical simulations [13]. In contrast, statistical and data-centric approaches have facilitated the creation of numerous machine learning models. [14] developed a deep neural network-based feature representation model that primarily employs stacked autoencoders to construct the Deep Neural network (DNN) and utilizes support vector regression for prediction. A similar approach is documented in [15] where stacked denoising autoencoders were used to construct a deep network. In [16], weather forecasting accuracy was aimed at by applying machine learning techniques, specifically to predict the maximum and minimum temperatures over a seven-day period using weather data from the past two days. A linear regression model and variation of a functional regression model were employed. The linear regression model, which is straightforward and effective for short-term data, outperformed the functional regression model in this context. The functional regression model, designed to capture weather trends, was less effective possibly due to the short two-day data window.

In [17], a simple time-series approach to model and forecast daily average temperatures in U.S. cities, particularly for use in the weather derivatives market was employed . The findings indicated that time-series modeling effectively captures significant conditional mean and variance dynamics in daily average temperature, as well as distinct differences between temperature distributions and temperature surprises. Furthermore, in [18], a novel, lightweight, data-driven weather forecasting model utilizing Long Short-Term Memory (LSTM) and Temporal Convolutional Networks (TCN) was proposed. It was concluded that the proposed lightweight model outperforms the complex Weather Research and Forecasting WRF model, showing potential for efficient and accurate weather forecasting up to 12 hours.

For forecasting severe convective weather (SCW), the authors in [19] used a six-layer convolutional neural network (CNN) trained with predictors. Assessing the model's performance using Global Forecast System (GFS), the deep learning model outperformed subjective forecasts by forecasters, significantly improving threat scores for thunderstorm, heavy rain, hail, and convective gusts by 16.1%, 33.2%, 178%, and 55.7% respectively. This model is now operational at the National Meteorological Center of China, aiding SCW forecasting across the country. In [20], for the prediction of air temperature using historical data, two methods were compared.The standard neural network against a deep learning network, specifically one using Stacked Denoising Auto-Encoders (SDAE) were evaluated. The empirical results demonstrate that the deep neural network with SDAE outperformed the standard multilayer feedforward network for this noisy time series prediction task.

Adaptive learning with bidirectional LSTM model was employed in the work done in [21], the result showed that adaptive learning with a bidirectional LSTM model reduced prediction error by 45% compared to baseline models. Another method utilized by [22] was a stacked Auto-Encoder. It was used to simulate 30 years of hourly weather data. Experimental results show that incorporating these newly learned features into classical models improves accuracy in time series forecasting.

B. *Long Short-Term Memory (LSTM) Model*

LSTM was introduced by Hochreiter and Schmidhuber in 1997 [11]. The model is a robust recurrent neural network specifically designed to address the exploding and vanishing gradient issues that commonly occur when learning long-term dependencies, even with substantial time lags [23]. Overall, this issue can be mitigated by employing a constant error carousel (CEC), which preserves the error signal within each unit's cell. Notably, these cells function as recurrent networks themselves, enhanced by the addition of an input gate and an output gate, collectively forming the memory cell. The self-recurrent connections provide feedback with a lag of one time step [24].

A standard LSTM unit consists of a cell, an input gate, an output gate, and a forget gate. The forget gate was not part of the initial design of LSTM, but was added by [25] in order that the network will be able to reset its state. The cell retains values over arbitrary time intervals, while the three gates control the flow of information related to the cell. Figure 1 shows a simple LSTM model architecture with the gates, input, and output.

Assuming a network that has *N* processing blocks and *M* number of inputs, and activation functions for the gates and memory cells are σ and g respectively, In the block input, the block component denoted by z_t are updated by combining the present input x_t and the output of the LSTM unit h_{t-1}, thereby we obtain [26],

$$z_t = g(U_z x_t + R_z h_{t-1} + b_z) \qquad (1)$$

where U_z and R_z are the weights associated with x_t and h_{t-1} , respectively, while b_z is the bias weight factor.

At the input gate, with activation function represented by i_t, we have,

$$i_t = \sigma(U_i x_t + R_i h_{t-1} + \rho_i \otimes c_{t-1} + b_i) \quad (2)$$

where $\otimes$ denotes point-wise multiplication, U_i, R_i, and ρ_i are the weights of x_t, h_{t-1}, and c_{t-1} respectively, while b_i is the bias weight factor for the input block.

At the forget gate with activation fucntion of f_t, the LSTM unit makes a choice of the information to be removed from its previous cell state c_{t-1}, therefore we have,

$$f_t = \sigma(U_f x_t + R_f h_{t-1} + \rho_f \otimes c_{t-1} + b_f) \quad (3)$$

where U_f, R_f and ρ_f are the weights associated with x_t, h_{t-1} and c_{t-1} respectively, while b_f is the bias weight vector.

Now, computing the cell value, the block input z_t, and input gate, i_t, and the forget gate, f_t values are combined with the previous cell value, therefore, we have,

$$c_t = z_t \otimes i_t + c_{t-1} \otimes f_t \quad (4)$$

In the output gate terminal with activation function of o_t, the output value is calculated as:

$$o_t = \sigma(U_o x_t + R_o h_{t-1} + \rho_o \otimes c_{t-1} + b_o) \quad (5)$$

where U_o, R_o and ρ_o are the weights associated with x_t, h_{t-1} and c_{t-1} respectively, while b_o is the bias weight vector.

Finally, the block output is combined to give,

$$h_t = g(c_t) \otimes o_t \quad (6)$$

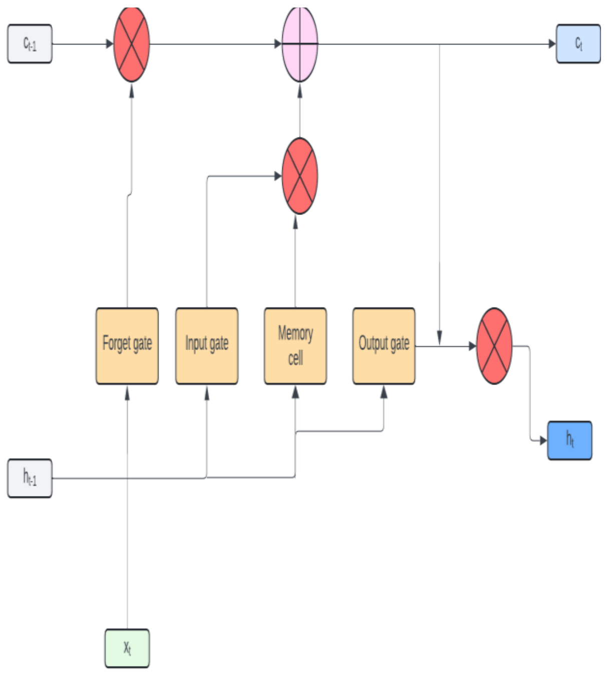

Fig. 1. Basic LSTM Model Structure.

III. METHODOLOGY

A. Dataset

Atmospheric data were obtained from the World Weather Online in Hong Kong for Akure, Ondo State, Nigeria, from July 2009 to October 2022. The weather data comprised of Atmospheric pressure, Visibility, Precipitation, Temperature and Humidity, and it is a hourly time-series data.

B. Software

Data preparation, formatting and programming was conducted using Python 3.9 [27], using the packages imported are the numpy, panda, matplot and the seaborn libraries. The LSTM model was built with keras on Google TensorFlow, and was formulated using the seaborn and sklearn libraries. For the error calculation, the math and sklearn libraries were employed.

C. Data Preparation

The data preparation algorithm imports essential libraries, configures plotting settings, loads data into a pandas dataframe, and performs data cleaning and preprocessing. The steps include handling duplicates, normalizing data, dealing with missing values, and transforming data for analysis. This process prepared the dataset for further machine learning tasks.The pesudocode is presented as :

```
IMPORT TensorFlow AS tf
IMPORT pandas AS pd
IMPORT numpy AS np
IMPORT matplotlib.pyplot AS plt
IMPORT seaborn AS sns
IMPORT datetime
CONFIGURE PLOTTING
   ENABLE inline plotting for IPython
   SET seaborn and matplotlib DPI settings for figures
   SET seaborn context and style
   CONFIGURE matplotlib to use SVG format for outputs
LOAD DATA into DataFrame
IDENTIFY and REMOVE duplicated or irrelevant features in
DataFrame
PERFORM further data cleaning and preprocessing:
   - Normalize data if necessary
   - Handle missing values
   - Transform data for analysis
END
```

D. Stationarity Check

After the normalization, the stationarity check was performed. Stationarity, for time-series analysis refers to a statistical property where the statistical characteristics of any data, such as mean, variance, and auto-correlation structure, do not change over time [28]. When a time-series is stationary, it makes the series easier to model and predict. The Augmented Dickey-Fuller (ADF) Test was used to carry out the stationarity test check, and the alpha value, which is the boundary that must not be exceeded was set to 0.05. Figs. 2, 3, 4, 5, and 6 presents the snippets of stationarity check for visibility, precipitation, humidity, temperature and pressure respectively. The x-axis is in time, while the y-axis is the time series data (pressure, humidity, temperature, visibility and rainfall). The stationarity check result for visibility was obtained to be 2.978E-24, while for precipitation, 0.0 was obtained. For pressure, humidity and temperature, the stationarity check results were 5.16E-17, 3.64E-13 and 6.74E-19 respectively. These values showed that the data is stationary over time.

The pseudocode used is presented as:

```
IMPORT matplotlib.pyplot, pandas, pmdarima.arima,
statsmodels.tsa.stattools, tensorflow.keras, sklearn.preprocessing
LOAD data
SET 'datetime' as index of DataFrame
PLOT 'humidity' time series
PERFORM stationarity tests:
    ADFTest with alpha=0.05
    Augmented Dickey-Fuller test, PRINT p-value
IF data non-stationary, consider differencing or transformations
PREPROCESS data for neural network:
    SCALE features using MinMaxScaler
    PREPARE training data set from selected columns, CONVERT to
float
BUILD LSTM model:
    DEFINE Sequential model
    ADD LSTM layers
    ADD Dropout layers (if needed)
    ADD Dense output layer
    COMPILE model with optimizer and loss function
TRAIN model on training data, SET epochs and batch size
FORECAST using the trained model
END
```

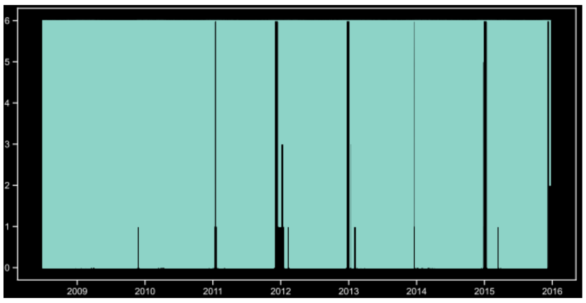

Fig. 2. Stationarity check graph for Visibility

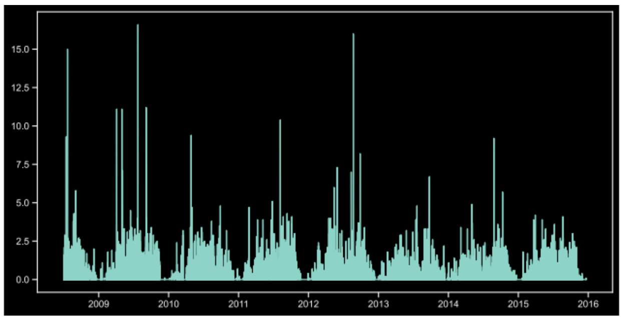

Fig. 3. Stationarity check graph for Precipitation

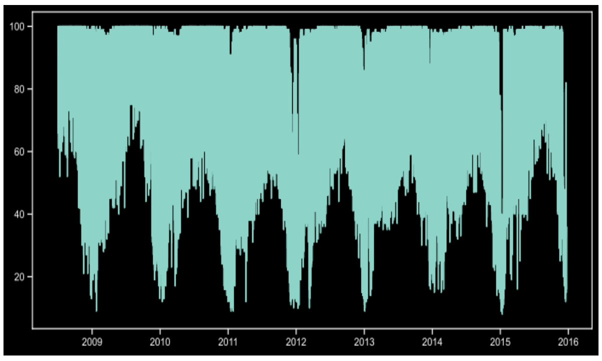

Fig. 4. Stationarity Check Graph for Humidity

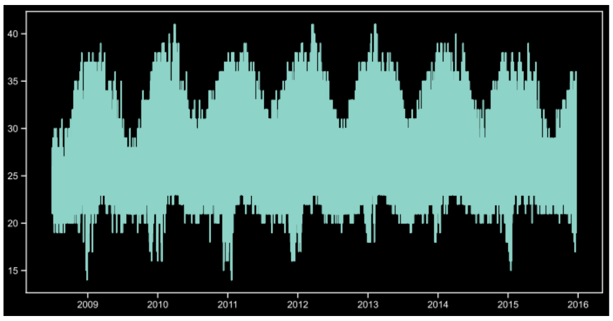

Fig. 5. Stationarity check graph for Temperature

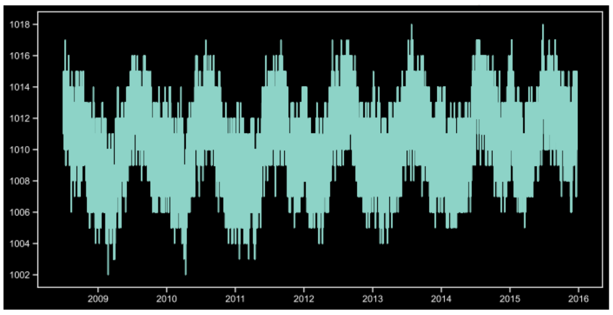

Fig. 6. Stationarity check graph for Pressure

E. Forecasting Model Formulation and Training

After the stationarity check was done, the data was preprocessed for neural network usage by using the MinMaxScaler function to normalize the features to a specific range. The data was prepared for training by choosing the relevant features for training and conversion of data type to float.

Thereafter, the sequential model was defined by using a linear stack of layers, and the LSTM layers were added in order to capture temporal depedencies. To prevent overfitting, a dropout layer was added. To fully produce the output, a dense output layer was connected. For compilation, the 'adam' optimizer and mean squared error was specified.

The model was fitted to the training data by specifying the number of epochs and batch size. The number of epoch used was 5, while the batch size was 16. Fig. 7 presents the LSTM model architecture employed for the forecasting of weather data.

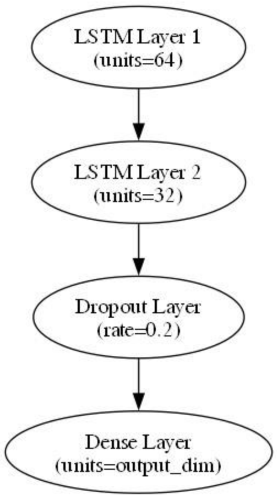

Fig. 7. The LSTM network layout

F. *Prediction*

The number of hours of prediction was defined and specified for fourteen (14) hours. Taking the last number of hours for prediciton samples from the training data, prediction was made using the model. Next, a range of date starting from the end of the training dates and spanning the number of prediciton hours was created. Therafter, the predicitons were repeated across the number of features in the original training data frame for the purpose of inverse scaling. After the inverse scaling, the timestamps were converted back to dates and a construction of a data frame containing the forecast dates and the predicted weather entity were done. Finally, using the matplotlib library, the actual weather entity was plotted.

IV. RESULTS

A. *Training and Validation losses*

The training and validation loss values for visibility, precipitation, humidity, temperature, and pressure is stated in Tables I to V. The training and validation plot for the number of epochs for the five weather entities are presented in Figs. 8 to 12. Significant reduction in validation loss values as the training loss reduces indicates that the model is not experiencing overfitting- that is the model has learned to fit the training data too closely, including noise or irrelevant patterns, and does not generalize well to new data.

TABLE I. TRAINING AND VALIDATION LOSS FOR VISIBILITY

Number of Epoch	Training Loss	Validation Loss
1	0.0252	0.0077
2	0.0114	0.0075
3	0.0097	0.0053
4	0.0090	0.0060
5	0.0087	0.0056

TABLE II. TRAINING AND VALIDATION LOSS FOR PRECIPITATION

Number of Epoch	Training Loss	Validation Loss
1	0.000450	0.000252
2	0.000347	0.000258
3	0.000305	0.000194
4	0.000282	0.000164
5	0.000261	0.000218

TABLE III. TRAINING AND VALIDATION LOSS FOR HUMIDITY

Number of Epoch	Training Loss	Validation Loss
1	0.0103	0.00170
2	0.0028	0.00088
3	0.0022	0.00087
4	0.0020	0.00060
5	0.0019	0.00063

TABLE IV. TRAINING AND VALIDATION LOSS FOR TEMPERATURE

Number of Epoch	Training Loss	Validation Loss
1	0.00390	0.00069
2	0.00130	0.00047
3	0.0011	0.00045
4	0.00099	0.00068
5	0.00091	0.00035

TABLE V. TRAINING AND VALIDATION LOSS FOR PRESSURE

Number of Epoch	Training Loss	Validation Loss
1	0.0065	0.0019
2	0.0019	0.0011
3	0.0016	0.0010
4	0.0015	0.0013
5	0.0014	0.00089

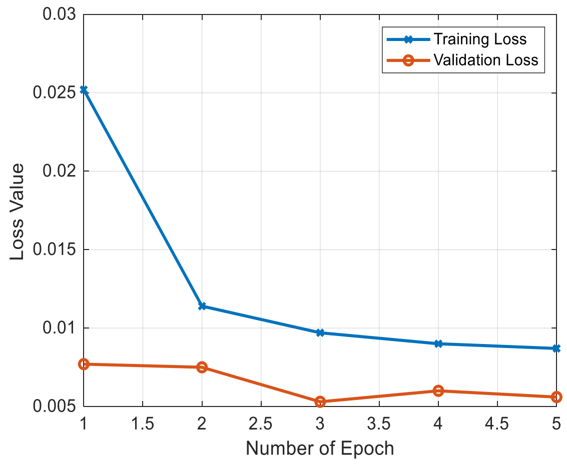

Fig. 8. Training and Validation loss plot for Visibility

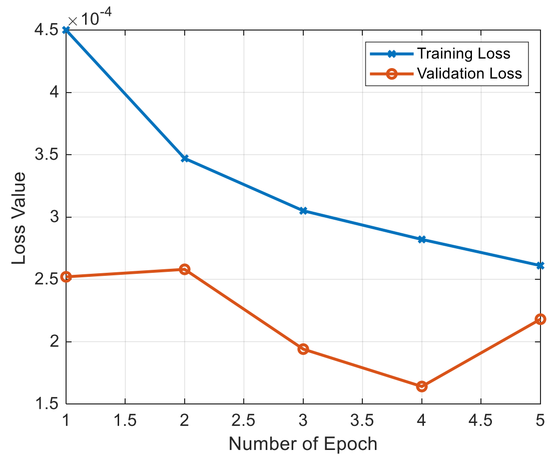

Fig. 9. Training and Validation loss plot for Precipitation

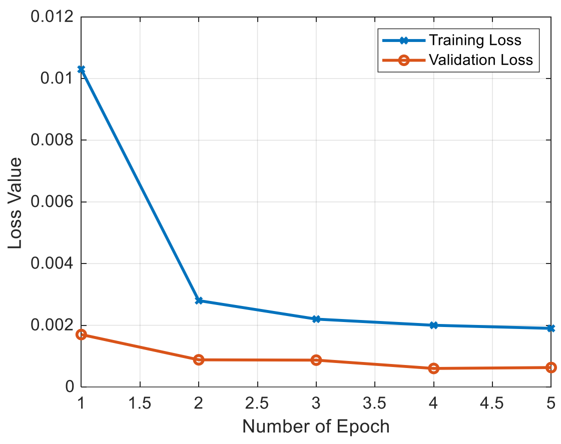

Fig. 10. Training and Validation loss plot for Humidity

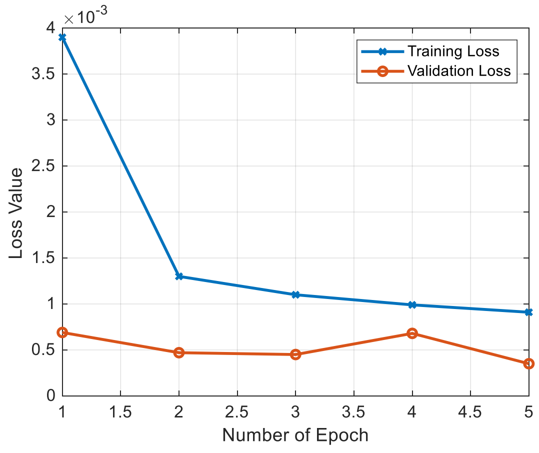

Fig. 11. Training and Validation loss plot for Temperature

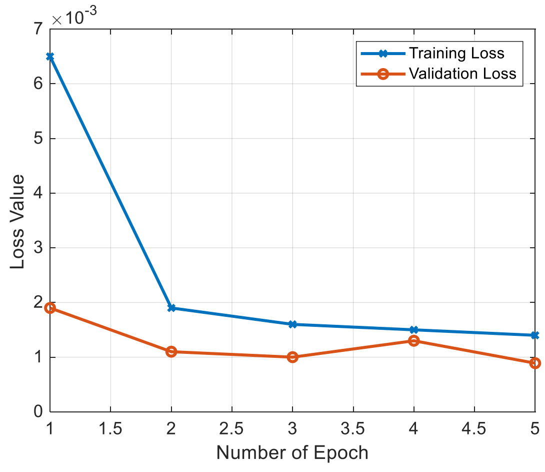

Fig. 12. Training and Validation loss plot for Pressure

B. Prediction Outcome

The predicted and expected values for the five weather entities were plotted and are shown in Fig. 13 to 17. The figures depict the closeness of the original values and the predicted values. The use of LSTM gave a close margin of error, thereby showing the performance of this deep learning method for weather forecasting. The Root Mean Squared Error (RMSE) value for visibility and precipitation was obtained to be 0.078 and 0.034 respectively. For humidity, presure and temperature, the RMSE values obtained were 1.531, 0.363, and 0.460 respectively. A lower RMSE value indicates a better fit between the predicted and actual values. For visibility, the RMSE of 0.078 suggests that, on average, the forecasted visibility values deviates by approximately 0.078 units from the actual visibility values. Similarly, for precipitation, the RMSE of 0.034 indicates an average deviation of approximately 0.034 units between the predicted and actual rainfall values. This deviation is applcable to humidity, pressure and temperature.

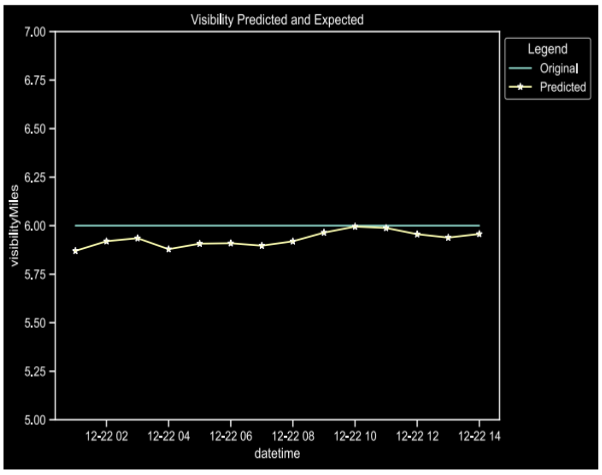

Fig. 13. Original and Predicted values for Visibility

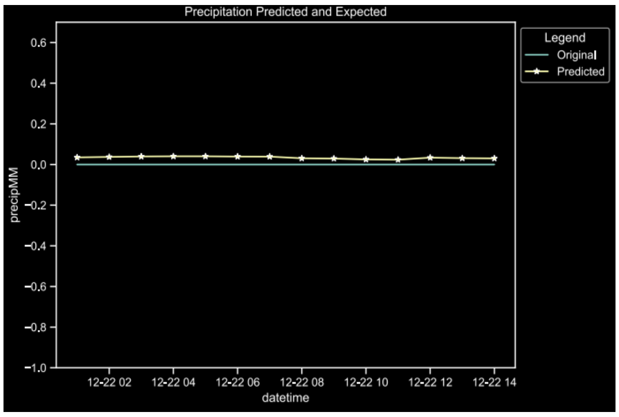

Fig. 14. Original and Predicted values for Precipitation

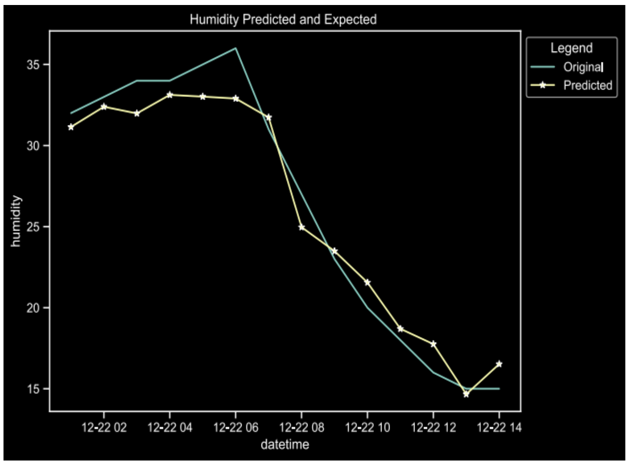

Fig. 15. Original and Predicted values for Humidity

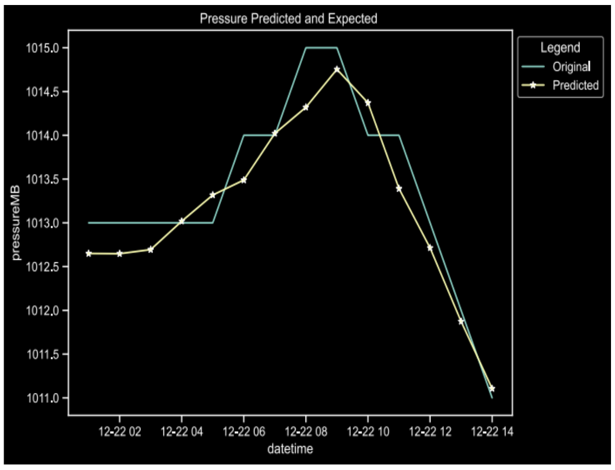

Fig. 16. Original and Predicted values for Pressure

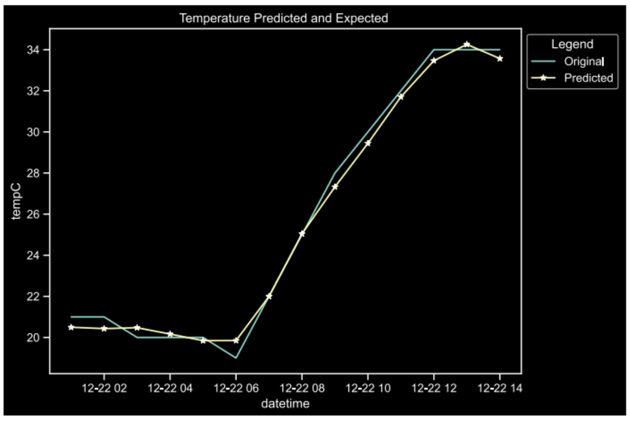

Fig. 17. Original and Predicted values for Temperature

V. CONCLUSION

In weather forecasting, different statistical and traditional methods have been employed. Machine learning tools have recently found usage in predicting the state of the atmosphere , therefore the LSTM- which is a deep learning model was used to predict the next fourteen hours of weather data. A two-layer LSTM network architecture was modelled in this paper. It was clearly observed that the validation losses obtained were lower than the training loss, thereby indicating no overfitting. The measure of error used was the Root Mean Squared Error (RMSE), with values of 0.078, 0.034, 1.531, 0.363, 0.460 for visibility, precipitation, humidity, pressure and temperature respectively. These results shows that LSTM is a deep learning model that can accurately forecast and make predictions with less margin of error. Future work will be to test different variants of LSTM and make a comparison with other deep learning method.

ACKNOWLEDGMENT

Olawuni Bayo is duly acknoweledged for the help in the research.

References

[1] Abhishek, K., Singh, M. P., Ghosh, S., and Anand, A, "Weather forecasting model using artificial neural network". *Procedia Technology*, *4*, 2012, pp. 311-318.

[2] Böcker, L., Dijst, M., and Prillwitz, J, "Impact of everyday weather on individual daily travel behaviours in perspective: a literature review". *Transport reviews*, *33*(1), 2013, pp. 71-91.

[3] Al Sadeque, Z., and Bui, F. M, "A deep learning approach to predict weather data using cascaded LSTM network". In *2020 IEEE Canadian conference on electrical and computer engineering (CCECE)* , 2020, pp. 1-5

[4] Powers, J. G., Klemp, J. B., Skamarock, W. C., Davis, C. A., Dudhia, J., Gill, D. O., and Duda, M. G, "The weather research and forecasting model: Overview, system efforts, and future directions". *Bulletin of the American Meteorological Society*, *98*(8), 2017, pp. 1717-1737.

[5] Goodfellow, I., Bengio, Y., and Courville, A, " Deep learning". MIT press, 2016.

[6] LeCun, Y., Bengio, Y., and Hinton, G, "Deep learning". *nature*, *521*(7553), 2015, pp. 436-444.

[7] Krizhevsky, A., Sutskever, I., and Hinton, G. E. "Imagenet classification with deep convolutional neural networks". *Advances in neural information processing systems*, *25, 2012*.

[8] Schmidhuber, J, "Deep learning in neural networks: An overview". *Neural networks*, *61*, 2015, pp. 85-117.

[9] Bengio, Y., Courville, A., and Vincent, P, "Representation learning: A review and new perspectives". *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 2013, pp. 1798-1828.

[10] Chen, X. W., and Lin, X, "Big data deep learning: challenges and perspectives", *IEEE access*, *2*, 2014, pp. 514-525.

[11] Hochreiter, S., and Schmidhuber, J, "Long short-term memory". *Neural computation*, *9*(8), 1997, pp. 1735-1780.

[12] Maqsood, I., Khan, M. R., and Abraham, A, "An ensemble of neural networks for weather forecasting", *Neural Computing and Applications*, *13*, 2004, pp. 112-122.

[13] Grover, A., Kapoor, A., and Horvitz, E, "A deep hybrid model for weather forecasting" , In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* , 2015, pp. 379-386.

[14] Liu, J. N., Hu, Y., You, J. J., and Chan, P. W, "Deep neural network based feature representation for weather forecasting", In *Proceedings on the International Conference on Artificial Intelligence (ICAI)* (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2014.

[15] Hossain, M., Rekabdar, B., Louis, S. J., and Dascalu, S, "Forecasting the weather of Nevada: A deep learning approach". In *2015 international joint conference on neural networks (IJCNN), 2015,* pp. 1-6, IEEE.

[16] Holmstrom, M., Liu, D., and Vo, C, "Machine learning applied to weather forecasting". *Meteorol. Appl*, *10*, 2016, pp. 1-5.

[17] Campbell, S. D., and Diebold, F. X, "Weather forecasting for weather derivatives". *Journal of the American Statistical Association*, *100*(469), 2005, pp. 6-16.

[18] Hewage, P., Trovati, M., Pereira, E., and Behera, A, "Deep learning-based effective fine-grained weather forecasting model". *Pattern Analysis and Applications*, *24*(1), 2021, pp. 343-366.

[19] Zhou, K., Zheng, Y., Li, B., Dong, W., and Zhang, X, "Forecasting different types of convective weather: A deep learning approach", *Journal of Meteorological Research*, *33*, 2019, pp. 797-809.

[20] M. Hossain, B. Rekabdar, S. J. Louis and S. Dascalu, "Forecasting the weather of Nevada: A deep learning approach," *2015 International Joint Conference on Neural Networks (IJCNN)*, Killarney, Ireland, 2015, pp. 1-6, doi: 10.1109/IJCNN.2015.7280812.

[21] Abdulla, N., Demirci, M., and Ozdemir, S, "Design and evaluation of adaptive deep learning models for weather forecasting", *Engineering Applications of Artificial Intelligence*, *116*, 2022, pp. 105440.

[22] Liu, J.N.K., Hu, Y., He, Y., Chan, P.W., and Lai, L, "Deep Neural Network Modeling for Big Data Weather Forecasting", In: Pedrycz, W., Chen, SM. (eds) Information Granularity, Big Data, and Computational Intelligence. Studies in Big Data, vol 8. Springer, 2015. https://doi.org/10.1007/978-3-319-08254-7_19

[23] Hochreiter, S., Schmidhuber, J, "Lstm can solve hard long time lag problems". In: M.C. Mozer, M.I. Jordan (eds.) Advances in Neural Information Processing Systems 9, pp. 473–479. MIT Press (1997)

[24] Yu, Y., Si, X., Hu, C., and Zhang, J, "A review of recurrent neural networks: LSTM cells and network architectures". *Neural computation*, *31*(7), 2019, pp. 1235-1270.

[25] Gers, F. A., Schmidhuber, J., and Cummins, F, "Learning to forget: Continual prediction with LSTM", *Neural computation*, *12*(10), 2000, pp. 2451-2471.

[26] Van Houdt, G., Mosquera, C., and Nápoles, G, "A review on the long short-term memory model", *Artificial Intelligence Review*, *53*(8), 2020, pp. 5929-5955..

[27] Python Software Foundation, 2021, Python 3.9.13 documentation, Avaliable at Download — Python 3.9.13 documentation

[28] Kočenda, E., and Černý, A, "Elements of time series econometrics: An applied approach". Charles University in Prague, Karolinum Press, 2015.