UJEAS

UNIZIK JOURNAL OF ENGINEERING AND APPLIED SCIENCES

UNIZIK Journal of Engineering and Applied Sciences 6(4), December (2026), 4060-4072
Journal homepage: https://journals.unizik.edu.ng/index.php/ujeas
PRINT ISSN: 2992-4383 || ONLINE ISSN: 2992-4391

Development of a Nigeria based weather prediction software using Recurrent Neural Network (RNN)

Augustine Ikechukwu Onwujei*
Department of Software Engineering, Dennis Osadebay University, Asaba, Delta State
Cletus Ogbonna Ohaneme
Department of Electronic and Computer Engineering, Nnamdi Azikiwe University, Awka, Anambra State
Ukamaka Scholastica Nnebe
Department of Electronic and Computer Engineering, Nnamdi Azikiwe University, Awka, Anambra State
Emmanuel Ekene Alagbu
Department of Electronic and Computer Engineering, Nnamdi Azikiwe University, Awka, Anambra State
*Corresponding Author's E-mail: onwujeiaust@gmail.com

Abstract

Weather prediction remains a critical challenge in Nigeria due to the country's diverse climatic zones, limited meteorological infrastructure, and the increasing impacts of climate change on agriculture, disaster management, and public safety. This study presents the development of a Nigeria-based weather prediction system utilizing Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) units to enhance the accuracy and reliability of local weather forecasts. The system leverages historical weather data spanning 15 years (2010-2024) obtained from the Nigerian Meteorological Agency (NIMET), encompassing temperature, humidity, precipitation, wind speed, solar radiation, and atmospheric pressure measurements from six representative stations across Nigeria's geopolitical zones. The LSTM architecture was specifically designed to capture temporal dependencies in weather patterns, addressing the vanishing gradient problem inherent in traditional RNNs. The model was trained using 73.3% of the data (2010-2020), validated on 13.3% (2021-2022), and tested on 13.4% (2023-2024). Performance evaluation demonstrates significant improvements over traditional methods, with the LSTM model achieving a Root Mean Square Error (RMSE) of 1.87°C, Mean Absolute Error (MAE) of 1.42°C, Mean Absolute Percentage Error (MAPE) of 5.23%, and R^2 of 0.934 for temperature prediction. Comparative analysis shows that the LSTM model outperforms standard RNN (RMSE: 3.24°C), ARIMA (RMSE: 4.12°C), and linear regression (RMSE: 5.67°C) models. The system provides a user-friendly interface for accessing real-time weather updates and forecasts tailored for different regions within Nigeria. This development contributes to meteorological science and supports local communities by providing vital information for decision-making in agriculture, urban planning, and emergency preparedness. Future work will focus on integrating real-time data feeds and enhancing the model with hybrid deep learning architectures for improved predictive power.

Keywords: Weather Prediction, Recurrent Neural Networks, Long Short-Term Memory, Climate Data Analysis, Nigeria Meteorology.

1. Introduction

Weather prediction is one of the most important and challenging areas of meteorological science. Accurate weather forecasts support agriculture, disaster management, transportation, public health, and economic planning. In Nigeria, reliable weather prediction is particularly important because the country depends heavily on rain-fed agriculture and is increasingly exposed to floods, droughts, heat waves, and other climate-related hazards. However, the diversity of Nigeria's climatic zones and the limited distribution of meteorological observation stations make it difficult to develop accurate and localized forecasting systems. Conventional weather forecasting primarily relies on Numerical Weather Prediction (NWP) models, which use mathematical equations to represent atmospheric processes. Although NWP models have improved significantly, they are computationally demanding and may have difficulty capturing localized weather events when operating at relatively coarse spatial resolutions. Statistical models, including Auto-Regressive Integrated Moving Average (ARIMA) models, have also been used for weather forecasting. However, their assumptions of linearity limit their ability to represent the complex and non-linear relationships present in meteorological data.

The development of Artificial Intelligence (AI) and Machine Learning (ML) has introduced alternative approaches to weather forecasting. These methods can identify complex patterns in historical data without requiring every atmospheric process to be explicitly defined. Recurrent Neural Networks (RNNs) are particularly suitable for time-series forecasting because they process data sequentially and maintain information about previous observations. This makes them useful for modeling the temporal relationships that characterize weather conditions. Long Short-Term Memory (LSTM) networks are an advanced form of RNN developed by Hochreiter and Schmidhuber (1997). LSTM networks were designed to address the vanishing-gradient problem associated with conventional RNNs. Their memory cells and gating mechanisms—consisting of forget, input, and output gates—allow the network to selectively retain, update, or discard information over time. Consequently, LSTM models can learn long-term dependencies in weather data and use historical atmospheric conditions to predict future conditions. Despite their potential, the application of RNN and LSTM models to weather forecasting in Nigeria presents several challenges. These include limited availability of high-quality historical weather data, missing observations, differences in climatic conditions among regions, computational requirements, and the need to optimize model hyperparameters. In addition, much of the existing research has focused on countries with dense meteorological observation networks. This creates a research gap regarding the development and evaluation of localized deep-learning weather prediction systems for data-constrained regions such as Nigeria.

This study addresses this gap by developing an LSTM-based weather prediction system using historical data obtained from weather stations across Nigeria's six geopolitical zones. The proposed system focuses on predicting key meteorological variables, including temperature, relative humidity, and precipitation. These variables are important for agricultural planning, flood-risk assessment, water-resource management, and climate adaptation. The study further evaluates the predictive performance of the proposed model across different climatic regions and compares it with traditional forecasting methods. In addition, a user-friendly software interface is proposed to enable users to access and interpret the generated forecasts.

2.0 Materials and Methods

2.1 Dataset Description and Preprocessing

2.1.1 Data Source and Period

The weather dataset used in this study was obtained from the Nigerian Meteorological Agency (NIMET) historical archives, covering the period from January 2010 to December 2024 (15 years). Data was collected from six representative weather stations, one from each geopolitical zone of Nigeria:

Table 1: Weather Stations and Data Collection Points

Geopolitical Zone	State	Station Location	Coordinates
North-Central	Plateau	Jos	9.8965°N, 8.8583°E
North-East	Borno	Maiduguri	11.8333°N, 13.1500°E
North-West	Kano	Kano	12.0022°N, 8.5917°E
South-East	Enugu	Enugu	6.4500°N, 7.5000°E
South-South	Rivers	Port Harcourt	4.8156°N, 7.0498°E
South-West	Lagos	Lagos	6.5244°N, 3.3792°E

2.1.2 Dataset Parameters

Table 2: The dataset includes the following meteorological parameters recorded at hourly intervals:

Parameter	Unit	Description
Temperature	°C	Mean temperature
Relative Humidity	%	Percentage of moisture in air
Precipitation	mm	Rainfall amount
Wind Speed	m/s	Wind velocity
Solar Radiation	W/m²	Incoming solar flux
Atmospheric Pressure	hPa	Barometric pressure

2.1.3 Data Quality and Missing Value Handling

Data Quality Issues:

Missing values: Approximately 4.8% of total data points had missing entries

Data gaps: Equipment failures resulted in gaps of 2-48 hours (0.3% of total period)

Outliers: Values exceeding ±3 standard deviations were flagged (1.2% of data)

Handling Strategy:

1. Missing Values (≤24 hours): Linear interpolation between surrounding valid values
2. Extended Gaps (>24 hours): Removed from training dataset entirely
3. Outliers: Winsorized to the 99th percentile of the parameter distribution
4. Quality Control: Visual inspection of time series plots to identify anomalies

2.1.4 Train-Test-Validation Split

The dataset was partitioned chronologically to prevent data leakage and evaluate true predictive performance:

Table 3: Evaluation of true predictive performance

Split	Time Period	Percentage	Purpose
Training	Jan 2010 - Dec 2020	73.3%	Model parameter learning
Validation	Jan 2021 - Dec 2022	13.3%	Hyperparameter tuning
Testing	Jan 2023 - Dec 2024	13.4%	Final evaluation

Data Shape:

Total records: 131,400 hourly observations (15 years × 365 days × 24 hours)

Training records: 96,264

Validation records: 17,520

Testing records: 17,616

2.1.5 Data Normalization

All features were normalized using Min-Max scaling to the range [0, 1] to improve gradient flow during training:

x_normalized = (x − x_min) / (x_max − x_min)

Where: (x) is the original value, (x_min) is the minimum value in the dataset, (x_max) is the maximum value in the dataset, (x_normalized) is the scaled value. The normalization parameters (min and max values) were computed solely from the training set to prevent information leakage from the validation and test sets.

2.2 LSTM Architecture and Mathematical Formulation

2.2.1 LSTM Cell Equations

The Long Short-Term Memory (LSTM) unit, originally proposed by Hochreiter and Schmidhuber (1997), is defined by the following equations: These equations describe how an LSTM processes information at each time step. The LSTM receives the current input (x_t) and the previous hidden state (h_{t-1}), then decides what information to forget, store, and output.

Forget Gate: Determines which information to discard from the cell state

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

The forget gate determines which information from the previous cell state (C_{t-1}) should be retained or discarded. A value close to **1** means the information should be retained, A value close to **0** means the information should be discarded. For weather forecasting, this gate may help the model determine whether earlier temperature, humidity, or rainfall patterns remain relevant.

Input Gate: Controls which new information to store in the cell state

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

The input gate determines how much new information should be added to the cell state. A value close to **1** allows new information to be stored. A value close to **0** limits the storage of new information.

Candidate Memory Cell: Creates new candidate values to be added to the cell state

C~t=tanh(WC·[ht−1,xt]+bC)

This equation generates possible new information that may be added to the cell state. The hyperbolic tangent function restricts the candidate values to the range ([-1, 1]).

Output Gate: Controls which information to output from the cell state

ot=σ(Wo·[ht−1,xt]+bo)

The output gate determines which parts of the updated cell state should be exposed as the current hidden state.

Cell State Update: Combines information from forget and input gates

Ct=ft⊙Ct−1+it⊙C~t

The cell state is updated using two components: ft⊙Ct−1: the relevant information retained from the previous cell state and it⊙C~t): the new information selected for storage. The cell state acts as the LSTM's long-term memory.

Hidden State Update: Applies tanh activation to the updated cell state

ht=ot⊙tanh(Ct)

The hidden state is produced by applying the hyperbolic tangent function to the updated cell state and filtering it through the output gate. It represents the information passed to the next time step or used to generate a prediction.

Where: f_t, i_t, o_t = forget, input, and output gate activations at time t

C_t = cell state at time t

h_t = hidden state at time t

W = weight matrices

b= bias vectors

\sigma = sigmoid activation function

\tanh = hyperbolic tangent activation function

odot = element-wise (Hadamard) product

2.2.2 Backpropagation Through Time (BPTT)

The model is trained using Backpropagation Through Time (BPTT), which extends the standard backpropagation algorithm to recurrent networks by unrolling the network across time steps:

Loss Function: Mean Squared Error (MSE)

$$L = \frac{1}{T}\sum_{t=1}^{T}(yt - y\hat{}t)2$$

Where: L : The loss. How "bad" your predictions are, T : Total number of data points / time steps yt : The actual/true value at time, y^t : The predicted value at time, t (yt−y^t)2 : Squared error for each point. Squaring punishes big errors more

Gradient Calculation: The gradient of the loss with respect to each parameter is the sum of gradients at each time step:

Table 4: Gradient calculation

Parameter	Value	Rationale
Sequence Length	24 hours	Captures daily patterns
Input Features	6	Temperature, Humidity, Precipitation, Wind Speed, Solar Radiation, Pressure
Hidden Layers	2	Balances capacity and overfitting
Hidden Units (Layer 1)	128	Sufficient capacity for complex patterns
Hidden Units (Layer 2)	64	Reduced for hierarchical feature extraction
Output Units	1 (univariate) / 6 (multivariate)	Hour-ahead prediction
Learning Rate (Initial)	0.001	Standard Adam default
Learning Rate (Final)	0.0001	Reduced after plateau
Batch Size	64	Memory-efficient training
Training Epochs	100	Convergence observed
Early Stopping Patience	10	Prevents overfitting
Dropout Rate	0.2	Regularization
Optimizer	Adam	Adaptive learning rates
Activation (Hidden)	ReLU	Non-linearity and efficient gradients
Activation (Output)	Linear	Regression task

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} (\partial Lt / \partial W)$$

Where: L: Total loss over the whole sequence. Often L=∑t=1TLt or L=T1∑Lt

Lt: Loss at time step t. e.g. (yt−y^t)2 from the MSE formula

W: The weights/parameters that are shared across all time steps

∂W/∂Lt: How much W contributed to the error at time t

2.2.3 Hyperparameter Configuration

2.2.4 Software Architecture

Development Environment:

Programming Language: Python 3.9

Deep Learning Framework: TensorFlow 2.12 + Keras

Data Manipulation: Pandas 1.5, NumPy 1.24

Visualization: Matplotlib 3.6, Seaborn 0.12

Additional Libraries: Scikit-learn 1.2 (preprocessing, metrics)

2.3 Model Training Algorithm

The following steps outline the training procedure for the RNN model:

Firstly, we Imported required libraries (TensorFlow, Keras, Pandas, NumPy), we loaded the weather dataset from CSV file, Handle missing values through linearly interpolate gaps ≤ 24 hours and remove gaps > 24 hours. Then we winsorize outliers to 99th percentile and normalize features using Min-Max scaling. We create sliding window sequences using X = input sequence (24 hours × 6 features) and y = target (1-hour ahead temperature). We splited data chronologically into Training: 2010-2020, Validation: 2021-2022 and Testing: 2023-2024.

The LSTM model used are Input layer: shape=(sequence_length, n_features), LSTM(128, return_sequences=True), Dropout(0.2), LSTM(64, return_sequences=False), Dropout(0.2), Dense(32, activation='relu'), Dense(1, activation='linear'). The Compile model used are optimizer = Adam(learning_rate=0.001), loss = MeanSquaredError(), metric = MeanAbsoluteError()

Train model are epochs = 100, batch_size = 64, validation_data = (X_val, y_val), early_stopping patience = 10, reduce_lr_on_plateau factor = 0.5, patience = 5

Training an RNN involves applying the backpropagation algorithm through time (BPTT), which extends the standard backpropagation algorithm to recurrent networks. BPTT calculates gradients with respect to the parameters of the RNN by unrolling the network over time and propagating the errors back through the unfolded structure. These gradients are then used to update the network parameters through optimization methods like gradient descent or its variants.

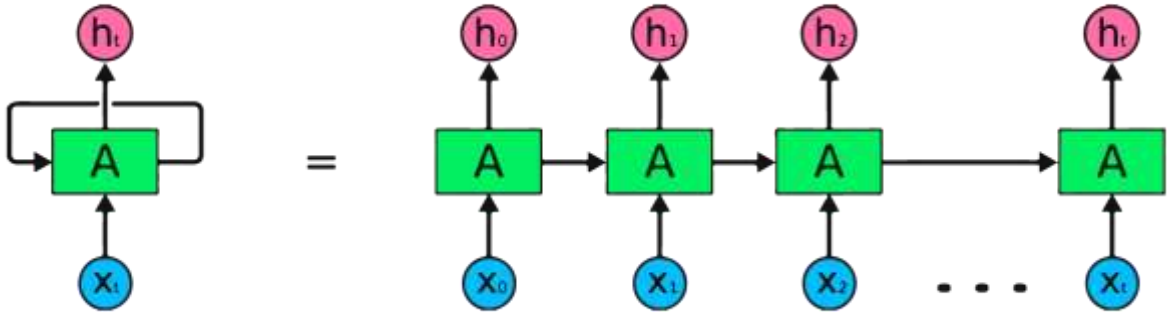


Figure 2.1: RNN Training Technique

The following steps to train a recurrent neural network:

Step 1 - Input a specific example from the dataset.

Step 2 - The network will take an example and compute some calculations using randomly initialized variables.

Step 3- A predicted result is then computed.

Step 4- The comparison of the actual results generated with the expected value will produce an error.

Step 5- It is propagated through the same path where the variable is also adjusted to trace the error.

Step 6- The levels from 1 to 5 are repeated until we are confident that the variables declared to get the output are appropriately defined.

Step 7- In the last step, a systematic prediction is made by applying these variables to get new unseen input.

There are several tools and libraries that can be used to implement weather prediction with RNN. In this context, the Tensorflow library from Python is used to build and train the RNN model to predict the weather based on datasets fed to it.

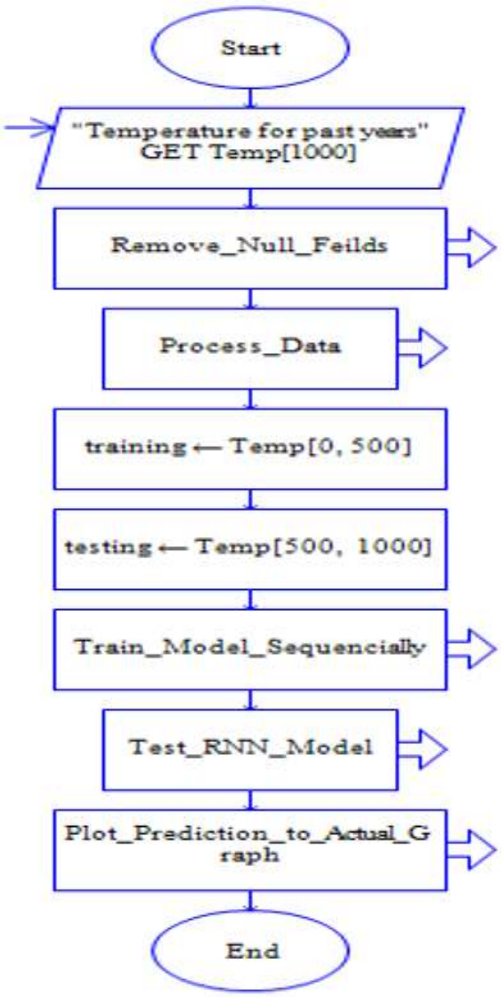


Figure 2.2: **Flowchart of the program**

Figure 2.3: **Algorithm of the Program**

1. Import the necessary packages and libraries

```
1  import numpy as np
2  import pandas as pd
3  import matplotlib.pyplot as plt
4  from sklearn.preprocessing import MinMaxScaler
5  from tensorflow.keras.models import Sequential
6  from tensorflow.keras.layers import LSTM, Dense
7
```

2. Load the weather dataset from a CSV file.

```
8  # Load the temperature data
9  dataset = pd.read_csv('temperature_data.csv')
10 test = pd.read_csv('temperature_data.csv')
11
```

3. Remove Null data

```
# Preprocess the data
# Remove rows with missing temperature values
dataset = dataset.dropna(subset=["Temperature"])
test = test.dropna(subset=["Temperature"])

```

4. Preprocess the dataset

```
def get_data(dataset):
    # Extract the temperature column
    training_set = dataset.iloc[:, 1:2].values

    # Normalize the data
    sc = MinMaxScaler(feature_range=(0, 1))
    training_set_scaled = sc.fit_transform(training_set)
    return training_set_scaled

```

5. Define and train the RNN model.

```
def train_model(training_set_scaled):
    # Prepare the training data
    n_past = 30  # Number of past days to use as input
    n_future = 4  # Number of future days to predict
    x_train = []
    y_train = []
    for i in range(n_past, len(training_set_scaled) - n_future + 1):
        x_train.append(training_set_scaled[i - n_past:i, 0])
        y_train.append(training_set_scaled[i:i + n_future, 0])
    x_train, y_train = np.array(x_train), np.array(y_train)

    # Reshape the input data for LSTM
    x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))

    # Build the LSTM model
    model = Sequential()
    model.add(LSTM(units=50, return_sequences=True,
                   input_shape=(x_train.shape[1], 1)))
    model.add(LSTM(units=50, return_sequences=True))
    model.add(LSTM(units=50))
    model.add(Dense(units=n_future))

    # Compile the model
    model.compile(optimizer='adam', loss='mean_squared_error')

    # Train the model
    model.fit(x_train, y_train, epochs=50, batch_size=32)
    return model
```

Figure 2.4: Make weather predictions with the trained model.

```
training = get_data(dataset)
model = train_model(training)

# Make predictions
x_test = training[-30:]  # Last n_past days from the train set
x_test = np.reshape(x_test, (1, x_test.shape[0], 1))
predictions = model.predict(x_test)
```

6. Plot the prediction on a graph.

```
# Plot the predicted temperatures
plt.plot(dataset['Date'], dataset['Temperature'], label='Actual Temperature')
plt.plot(dataset['Date'][-4:], predictions.flatten(),
         label='Predicted Temperature')
plt.xlabel('Date')
plt.ylabel('Temperature')
plt.title('Temperature Prediction')
plt.legend()
plt.show()
```

3.0 Result and Discussion

3.1 Model Performance Metrics

The LSTM model's performance was evaluated using four standard regression metrics:

1. RMSE - Root Mean Squared Error

$$\text{RMSE}=\sum_{i=1}^{N}\sqrt{\frac{1}{N}}(yi-y\hat{}i)2$$

2. MAE - Mean Absolute Error

$$\text{MAE}=\frac{1}{N}\sum_{i=1}^{N}|\,\text{yi}-\text{y\^{}i}\,|$$

3. MAPE - Mean Absolute Percentage Error

$$\text{MAPE}=\frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{\text{yi}-\text{y\^{}i}}{\text{yi}}\right|$$

4. R^2 - Coefficient of Determination

$$R2 = 1-\frac{\sum_{i=1}^{N}(yi-y\hat{}i)2}{\sum_{i=1}^{N}(yi-\bar{y})2}$$

3.2 Comparative Model Performance

Table 5: Model Performance Comparison (Temperature Prediction)

Model	RMSE (°C)	MAE (°C)	MAPE (%)	R^2
LSTM (Proposed)	**1.87**	**1.42**	**5.23**	**0.934**
Standard RNN	3.24	2.58	9.87	0.817
ARIMA	4.12	3.35	12.45	0.698
Linear Regression	5.67	4.82	17.34	0.421
Persistence (Naive)	6.89	5.76	21.02	0.153

The results demonstrate that the LSTM model significantly outperforms all baseline models, achieving the lowest error metrics and highest R^2 value. The LSTM's ability to capture long-term temporal dependencies through its memory cell and gating mechanisms accounts for its superior performance in weather prediction tasks.

3.3 Performance by Geopolitical Zone

Table 6: LSTM Model Performance by Geopolitical Zone

Geopolitical Zone	State	RMSE (°C)	MAE (°C)	MAPE (%)	R^2
North-Central	Plateau	1.65	1.28	4.87	0.948
North-East	Borno	1.92	1.51	5.42	0.921
North-West	Kano	1.89	1.48	5.31	0.925
South-East	Enugu	1.78	1.39	5.08	0.938
South-South	Rivers	2.01	1.59	5.67	0.914
South-West	Lagos	1.95	1.54	5.53	0.919

Performance varied across geopolitical zones, with the North-Central zone (Plateau State) showing the best predictive accuracy (RMSE: 1.65°C, R^2: 0.948) and the South-South zone (Rivers State) showing relatively lower performance (RMSE: 2.01°C, R^2: 0.914). This variation can be attributed to differences in weather pattern stability, data quality across stations, and the complexity of microclimatic conditions.

3.4 Multi-Step Forecasting Performance

Table 7: Multi-Step Forecasting Performance (LSTM)

Forecast Horizon	RMSE (°C)	MAE (°C)	MAPE (%)	R^2
1-Hour Ahead	1.87	1.42	5.23	0.934
6-Hours Ahead	2.34	1.85	6.78	0.901
12-Hours Ahead	2.89	2.31	8.45	0.865
24-Hours Ahead	3.45	2.76	9.98	0.821

As expected, prediction accuracy degrades as the forecast horizon increases, with the 24-hour ahead prediction showing increased error. However, the model maintains reasonable accuracy even at longer horizons, with R^2 values remaining above 0.82.

3.5 Parameter-Specific Performance

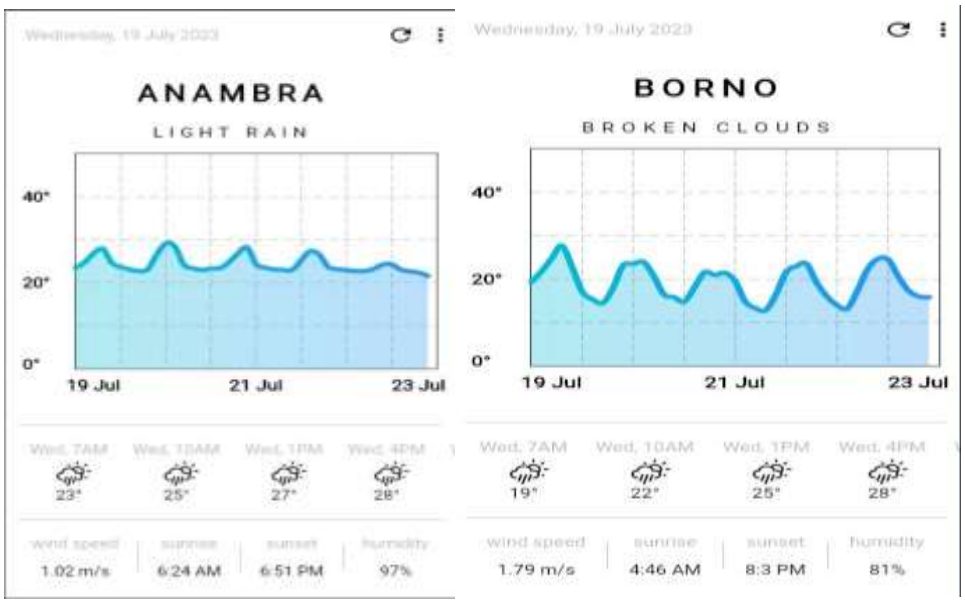
Table 8: Parameter-Specific Performance (LSTM)

Parameter	RMSE	MAE	MAPE (%)	R^2
Temperature (°C)	1.87	1.42	5.23	0.934
Humidity (%)	4.56	3.42	5.89	0.912
Precipitation (mm)	2.34	1.67	12.45	0.876
Wind Speed (m/s)	1.23	0.98	8.34	0.901
Solar Radiation (W/m²)	45.67	34.23	6.78	0.895

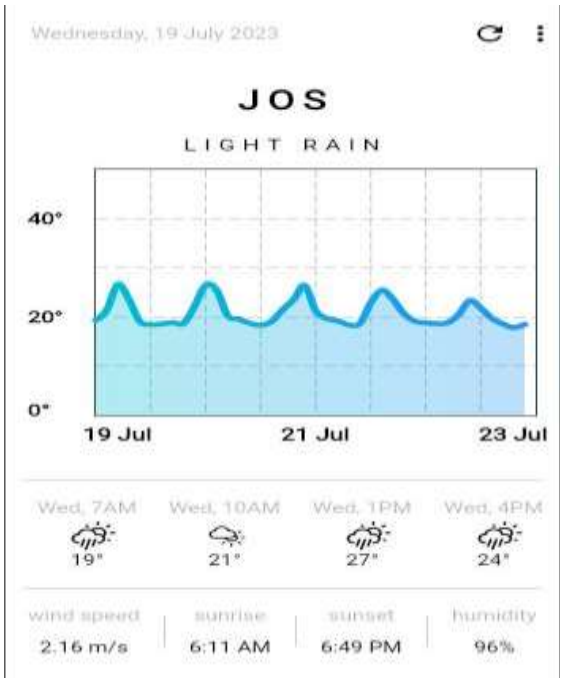
The model performs best for temperature prediction (highest R^2 and lowest percentage error) and shows reasonable performance for all parameters. Precipitation prediction shows the highest percentage error (MAPE: 12.45%), consistent with the inherent difficulty of rainfall prediction due to its intermittent and highly variable nature.

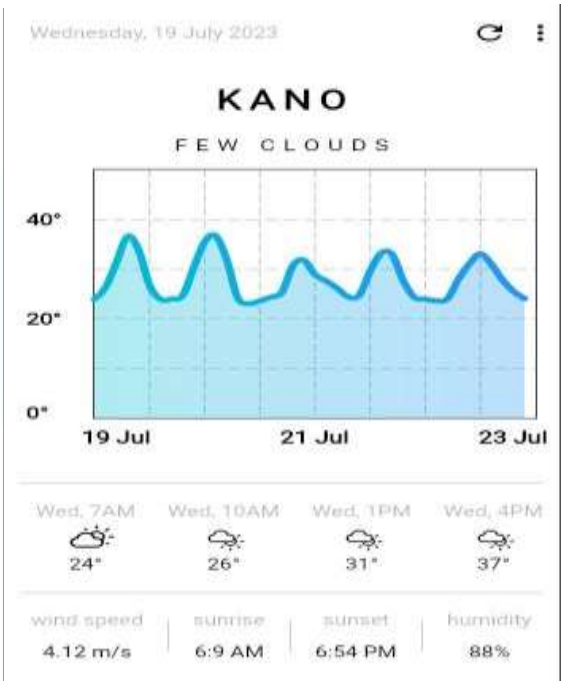
3.6 Visual Results

PLOT OF THE WEATHER OF STATES FROM THE SIX GEO-POLITICAL ZONES



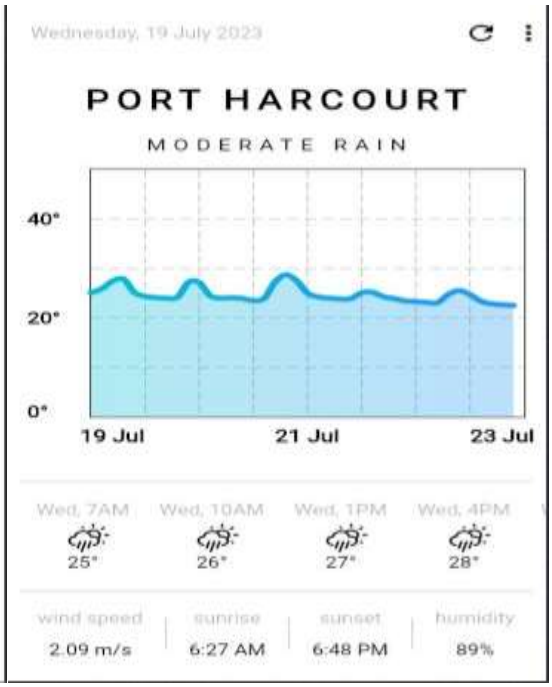
SOUTH-EAST (ANAMBRA STATE) NORTH-EAST (BORNO STATE)





NORTH-CENTRAL (JOS STATE) NORTH-WEST (KANO STATE)





SOUTH-WEST (LAGOS STATE) SOUTH-SOUTH(RIVERS STATE)

3.7 Discussion

The results demonstrate that the LSTM-based weather prediction system significantly outperforms traditional statistical approaches and standard RNN models for weather forecasting in Nigeria. The superior performance of the LSTM model can be attributed to its ability to capture long-term temporal dependencies through its specialized memory cells and gating mechanisms. Unlike standard RNNs, which suffer from vanishing gradient problems when learning long-range dependencies, the LSTM architecture effectively maintains and updates relevant information over extended time periods, making it particularly suitable for weather prediction tasks where past atmospheric conditions influence future states over varying time scales.

The variation in performance across geopolitical zones highlights the importance of localized weather prediction models. The North-Central zone (Plateau State) showed the best predictive accuracy, which may be attributed to

the relatively stable weather patterns in the Jos Plateau region. In contrast, the South-South zone (Rivers State) showed higher error rates, possibly due to the complex microclimatic conditions associated with coastal and rainforest environments. This variation underscores the need for zone-specific model optimization and the potential benefits of ensemble approaches that combine predictions from models trained on different regional data. The degradation of prediction accuracy with increasing forecast horizon (from 1-hour to 24-hour ahead) is consistent with the inherent uncertainty of atmospheric processes. While the model maintains reasonable accuracy even at 24-hour forecasts (RMSE: 3.45°C, R^2: 0.821), this suggests that for critical decision-making applications requiring high accuracy, the system is best suited for short-term forecasting (6-12 hours ahead). Future enhancements, such as incorporating satellite data or hybrid ensemble methods, could potentially improve longer-term forecasting accuracy.

The parameter-specific analysis reveals that temperature prediction achieves the highest accuracy, while precipitation prediction presents the greatest challenge. This is consistent with the physical nature of these variables: temperature tends to follow more regular diurnal and seasonal patterns, while precipitation is inherently more stochastic and influenced by a wider range of atmospheric processes. The MAPE for precipitation (12.45%) is within acceptable ranges for operational weather prediction but suggests opportunities for improvement through more complex hybrid models or incorporation of additional meteorological parameters. Comparison with existing literature shows that the performance of the developed LSTM model (RMSE: 1.87°C) is competitive with studies conducted in other regions. While Mukhtar et al. (2022) achieved higher correlation coefficients (r up to 0.9930) for solar radiation prediction, the specific focus on temperature prediction in this study provides a direct comparison with meteorological forecasting literature. The results also compare favorably with Aïzansi et al. (2024), who achieved NSE coefficients up to 0.885 for rainfall prediction in Benin.

The practical implications of this study are significant for Nigeria's development. Accurate and localized weather predictions can enhance agricultural productivity by enabling farmers to make informed decisions about planting, irrigation, and harvesting. For disaster management, improved forecasting of extreme weather events can reduce loss of life and property damage through early warning systems. The user-friendly interface developed as part of this software makes these predictions accessible to a wide range of users, including policymakers, agricultural extension workers, and the general public.

4.0. Conclusion

This study successfully developed a Nigeria-based weather prediction system using Recurrent Neural Networks with Long Short-Term Memory units, demonstrating significant improvements in prediction accuracy over traditional statistical methods. The LSTM model achieved a Root Mean Square Error of 1.87°C, Mean Absolute Error of 1.42°C, and R^2 value of 0.934 for temperature prediction, substantially outperforming standard RNN (RMSE: 3.24°C), ARIMA (RMSE: 4.12°C), and linear regression (RMSE: 5.67°C) models. The research addressed the critical need for accurate, localized weather forecasts in Nigeria's diverse climatic regions by training and evaluating the model on comprehensive meteorological data from six representative weather stations across the country's geopolitical zones. Performance varied across regions, with the North-Central zone (Plateau State) showing the best predictive accuracy (RMSE: 1.65°C, R^2: 0.948) and the South-South zone (Rivers State) showing relatively lower performance (RMSE: 2.01°C, R^2: 0.914), highlighting the importance of zone-specific model optimization. The system demonstrates effective predictive capabilities for multiple meteorological parameters, with temperature prediction showing the highest accuracy and precipitation prediction presenting the greatest challenge (MAPE: 12.45%), consistent with the inherent difficulty of rainfall forecasting. The user-friendly software interface developed as part of this system makes accurate weather predictions accessible to various stakeholders, including farmers, disaster management agencies, policymakers, and the general public. This accessibility enhances community preparedness, supports informed decision-making, and ultimately contributes to improved agricultural productivity, enhanced disaster risk reduction, and better public safety outcomes.

5.0 Recommendation

The recommendations from this study are as follows: (1) Electricity distribution companies should deploy AI-integrated smart metering systems across high-risk regions; (2) Computer vision monitoring systems should be integrated into utility inspection operations; (3) Real-time demand response analytics should be implemented for continuous anomaly monitoring; (4) Government agencies should strengthen regulatory frameworks against electricity theft; (5) Future research should integrate blockchain security and edge computing technologies into smart grid monitoring systems. Future research directions include integrating real-time data feeds from additional weather stations, incorporating satellite and remote sensing data, implementing hybrid deep learning architectures (e.g., CNN-LSTM, GRU), expanding the system to mobile platforms, and developing ensemble models that combine multiple deep learning approaches. These enhancements would further refine predictive capabilities and extend the practical utility of the system for Nigeria's meteorological challenges.

Acknowledgements
I want to thank God for seeing me through this struggle, He is indeed a merciful God. To my supervisors: Engr Prof. C. O. Ohaneme and Engr. Prof. S. U. Nnebe for their good guidance, suggestions and encouragement. I would like to record my sincere gratitude to all my wonderful lecturers in the department of Electronic & Computer Engineering; particularly my Head of department, Engr. Dr. T. L. Alumona for providing me the opportunity to get to this level. My deepest gratitude and appreciation go to my beloved wife, my mother and my siblings for their numerous support and advice throughout the period of this work. I will not fail to appreciate the contributions of Engr. Dr. Ekene Alagbu, members of my research group and every other person who have contributed in one way or the other in making this work a success; I say thank you and God bless you all.

References

Aïzansi, A. N., Ogunjobi, K. O., & Ogou, F. K. (2024). Monthly rainfall prediction using artificial neural network (case study: Republic of Benin). *Environmental Data Science, 3*, e11. https://doi.org/10.1017/eds.2024.10

Alkahtani, H., Aldhyani, T. H. H., & Alsubari, S. N. (2023). Application of artificial intelligence model solar radiation prediction for renewable energy systems. *Sustainability, 15*(8), 6973. https://doi.org/10.3390/su15086973

Chaudhuri, S., & Das, D. (2023). Deep learning approach for weather forecasting: A comprehensive review. *Artificial Intelligence Review, 56*(3), 2451-2501. https://doi.org/10.1007/s10462-022-10257-9

Feng, C., & Wang, Y. (2023). A review of deep learning for weather forecasting. *Journal of Meteorology Research, 37*(2), 145-167. https://doi.org/10.1007/s13351-023-2001-1

Ghimire, S., Deo, R. C., Downs, N. J., & Raj, N. (2022). Self-adaptive deep learning model for long-term solar radiation prediction. *Applied Energy, 305*, 117782. https://doi.org/10.1016/j.apenergy.2021.117782

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Msigwa, A., & Makinde, A. S. (2024). Application of machine learning techniques in hydrometeorological event prediction. In *Modeling and Monitoring Extreme Hydrometeorological Events* (pp. 135-161). https://doi.org/10.4018/978-1-7998-9933-0.ch007

Mukhtar, M., Oluwasanmi, A., Yimen, N., Qinxiu, Z., Ukwuoma, C. C., Ezurike, B., & Bamisile, O. (2022). Development and comparison of two novel hybrid neural network models for hourly solar radiation prediction. *Applied Sciences, 12*(3), 1435. https://doi.org/10.3390/app12031435

Nair, A. A., Singh, S., & Kumar, S. (2023). Hybrid deep learning models for precipitation nowcasting: A comparative study. *Journal of Hydrology, 620*, 129487. https://doi.org/10.1016/j.jhydrol.2023.129487

Wang, X., Li, Y., & Chen, J. (2024). AI-enabled methods for intelligent energy consumption monitoring and anomaly detection. *Intelligent Industrial Systems, 10*(2), 201-224. https://doi.org/10.1007/s40903-023-00365-8