DECLARATION

We, the undersigned students, hereby submit our project report entitled **“Weather Forecasts”** to the **Department of Computer Engineering, Universal Engineering and Science College, Chakupat, Lalitpur**. This report has been prepared as a partial fulfillment for the degree of Bachelor of Engineering in **Computer Engineering**.

We declare that the work presented in this report is the result of our own efforts and has not been submitted to any other university or institution for the award of any degree or diploma. We take full responsibility for the originality and authenticity of the content and have duly acknowledged all sources and references used in the course of our research.

We express our sincere gratitude for the support and guidance received throughout the completion of this project.

Submitted By:

Mamta Dangaura [22070517]

Sadikshya Dhakal [22070522]

Date: July,2025

CERTIFICATE OF APPROVAL

This is to certify that we have thoroughly reviewed the minor project report entitled **"Weather Forecasts"** submitted by **[Mamta Dangaura]** and **[Sadikshya Dhakal]** to the Department of Computer Science and Engineering, **Universal Engineering and Science College**, Chakupat, Lalitpur. This project has been carried out in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in **Computer Engineering.**

The project work was undertaken under our supervision and successfully completed within the time frame prescribed by the curriculum. Throughout the project, the students demonstrated diligence, technical competency, and a strong commitment to their work. We are confident in recommending this project for acceptance as a part of the academic requirements for the Bachelor's degree in **Computer Engineering**.

Date: July, 2025

Project Supervisor
Mr. Hemant Joshi
Head of Department,
Department of Computer Engineering, UESC
Signature: ______________________

External Examiner
Mr. ____________________________
Signature: ______________________

External Examiner
Mr. ____________________________
Signature: ______________________

Head of the Department
Mr. Hemant Joshi
Department of Computer Engineering, UESC
Signature: ______________________

COPYRIGHT

ACKNOWLEDGEMENT

This project work has been prepared as part of the partial fulfillment of the requirements for the degree of Bachelor of Computer Engineering at the Faculty of Engineering. The successful completion of this project, titled **‘Weather Forecasts’**, would not have been possible without the support and guidance of several individuals and organizations.

We extend our heartfelt gratitude to the administration of **Universal Engineering and Science College** for their continuous support and encouragement throughout this project. We are also deeply thankful to the faculty members of the BE Computer program at UESC for their invaluable guidance and insights.

Special thanks go to our supervisor, Mr. Hemant Joshi, for his expert advice, constructive feedback, and constant supervision during the preparation of this project.

Lastly, we express our sincere appreciation to all those who directly or indirectly contributed to the successful completion of this project. Your support and encouragement have been instrumental in achieving this milestone.

Sincerely,

Mamta Dangaura [22070517]

Sadikshya Dhakal [22070522]

ABSTRACT

Weather forecasting is essential for informed decision-making in sectors such as agriculture, aviation, disaster management, and daily life. Traditional statistical and machine learning models often struggle to capture the nonlinear and dynamic nature of weather patterns. This project proposes a hybrid deep learning model that combines Convolutional Neural Networks for spatial feature extraction with Long Short-Term Memory networks for modeling temporal dependencies in time-series weather data. The system is designed to predict 7-day weather conditions with improved accuracy by learning both short- and long-term patterns from historical and real-time inputs. Real-time weather data is fetched from Weather API and processed using a trained CNN-LSTM model, which is integrated into a user-friendly web-based interface for seamless interaction. The model's performance is evaluated using metrics such as accuracy and loss, demonstrating its potential for reliable and accessible weather forecasting applications.

Keywords: Weather Forecasting, CNN-LSTM Hybrid Model, Deep Learning.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolution Neural Network
CSV	Comma-Separated Values
DL	Deep Learning
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HTML	HyperText Makeup Language
HTTP	HyperText Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
PY	Python
ReLU	Rectified Linear Unit
VS CODE	Visual Studio Code
XML	eXtensible Markup Language

CHAPTER 1

OVERVIEW

1.1 Introduction

Weather forecasting involves predicting future atmospheric conditions by analyzing various meteorological parameters including temperature, humidity, air pressure, wind speed, and precipitation. Accurate weather forecasts are crucial for a wide range of applications such as agriculture, aviation, disaster management, and daily life activities. They enable timely warnings, help optimize resource allocation, and support informed decision-making. Traditional forecasting methods often struggle to capture the highly nonlinear and dynamic nature of weather patterns.

To address these challenges, this project implements both Long Short-Term Memory networks and a hybrid CNN-LSTM deep learning model for 7-day weather prediction. While LSTM networks effectively model temporal dependencies in sequential data, integrating CNN layers helps extract meaningful spatial features from weather datasets. The models are trained and evaluated using historical weather data combined with real-time inputs from Weather API. By leveraging these deep learning approaches, the system aims to provide more accurate and reliable weather forecasts accessible via a user-friendly web interface, enhancing practical decision-making across multiple sectors.

1.2 Background

Weather forecasting plays a crucial role in sectors like agriculture, transportation, energy, and disaster management by enabling better planning and risk reduction. Traditionally, forecasting relied on observational and statistical methods, but with advancements in technology, machine learning models have emerged as powerful tools for predicting complex and nonlinear weather patterns. These models can analyze large volumes of atmospheric data to improve prediction accuracy.

Various classification algorithms such as Support Vector Machines, K-Nearest Neighbors, Naive Bayes, Artificial Neural Networks, and Random Forests have been used to classify weather conditions like rain or sunshine. Among them, Random Forest is valued for its robustness and accuracy. More recently, deep learning approaches such as Long Short-Term Memory networks have demonstrated exceptional performance in capturing temporal patterns in time series weather data. Hybrid models like CNN-LSTM combine spatial and temporal feature learning, offering improved forecasting accuracy by addressing the complex nature of weather dynamics.

1.2 Problem Definition

Weather is highly dynamic and nonlinear, making it difficult for traditional computer models to simulate accurately. These models divide the atmosphere into grids, which leads to approximations and limited prediction accuracy, especially for short-term and local forecasts.

1.3 Motivation

Accurate weather prediction is essential for minimizing weather-related risks and improving public safety, agriculture planning, energy management, and daily life. With rising climate variability, there is a growing need for smarter, data-driven forecasting methods.

1.4 Objectives

- To develop a 7-day real time weather prediction model using API Key for improved accuracy.

1.5 Scope and Application

- **Disaster Management:** Accurate weather forecasting enables early warnings for natural disasters like floods, storms, and wildfires, supporting timely evacuation and emergency response.
- **Agriculture:** Reliable predictions assist farmers in planning crop cycles, irrigation, fertilization, and pest control, thereby improving productivity and food security.
- **Energy Sector:** Weather forecasts help utility companies manage energy demand and optimize production from renewable sources such as solar, wind, and hydroelectric power.
- **Transportation:** Forecasting road and flight conditions enhances transportation safety and efficiency by helping avoid weather-related delays and accidents.

CHAPTER 2

LITERATURE REVIEW

Suleman and Shridevi (2022) proposed an LSTM-based model enhanced with spatial feature attention mechanisms to improve short-term weather forecasting. The model shows high accuracy and robustness, addressing spatial variability in meteorological data, and outperforms conventional LSTM networks in terms of prediction precision [1].

Fente and Singh (2018) applied a basic ANN model for weather prediction and demonstrated its ability to approximate nonlinear trends in meteorological data, laying a foundation for more complex neural forecasting systems [2].

Salman et al. (2018) compared single- and multi-layer LSTM architectures for weather forecasting using auxiliary variables. Their results showed that deeper models yielded better long-term predictions and effectively captured variable dependencies [3].

Gong et al. (2024) introduced a CNN-LSTM model to integrate spatial and temporal aspects of weather data, significantly improving temperature prediction accuracy while managing high-dimensional input and missing values [4].

Fu et al. (2019) built a hybrid model combining 1D CNN for local spatial pattern detection and Bi-LSTM for capturing long-range temporal dependencies. The model was effective in forecasting weather across multiple geographic stations [5].

Yu et al. (2019) utilized LSTM networks for short-term solar irradiance prediction under dynamic weather scenarios. Their method exhibited strong generalization and robustness, improving energy forecast reliability in solar power systems [6].

Karevan and Suykens (2018) introduced a two-layer spatio-temporal stacked LSTM architecture for temperature forecasting. The model captures spatial dependencies by training individual LSTM models for each location in the first layer, then combining hidden states in the second. This architecture showed improved accuracy in temperature prediction by integrating spatial context into temporal learning [7].

Table 1: Literature Review of Deep Learning Models for Weather Forecasting.

N	Study Title	Meth-od Used	Specific purpose	Accuracy/Results	Key Findings
1	Short-Term Weather Forecasting Using Spatial Feature Attention Based LSTM Model (2022)	Spatial Feature Attention LSTM	Enhance short-term weather forecasting by capturing spatial dependencies	Achieved state-of-the-art prediction accuracy; specific metrics not specified	Integrates spatial attention mechanisms into LSTM to improve forecasting accuracy by considering spatial features.
2	Weather Forecasting Using Artificial Neural Network (2018)	Artificial Neural Network (ANN)	Basic weather prediction using historical meteorological data	Demonstrated improvements in forecasting precision and accuracy; specific metrics not specified	Showed the capability of ANN in modeling nonlinear weather data trends, serving as a foundation for more complex models.
3	Single Layer & Multi-layer LSTM Model with Intermediate Variables (2018)	Single- & Multi-layer LSTM	Compare single vs. multi-layer LSTM using additional variables (e.g.,	Validation accuracy up to 80.60% with RMSE of 0.0775 for pressure variable	Incorporating intermediate variables like pressure and dew point significantly enhances the

			pressure, dew point)		accuracy of visibility predictions.
4	Deep Learning for Weather Forecasting: CNN-LSTM Hybrid Model (2024)	CNN + LSTM Hybrid	Predict historical temperature by combining spatial and temporal patterns	R² score of 0.901; MAE reduced to 0.901	The hybrid model effectively captures spatial and temporal dependencies, resulting in stable and accurate temperature predictions.
5	Multi-Stations' Weather Prediction Using 1D CNN and Bi-LSTM (2019)	1D CNN + Bi-LSTM Hybrid	Forecast weather across multiple locations simultaneously	Achieved higher precision compared to FNN, 1D-CNN, and LSTM models; specific metrics not specified	Combining 1D CNN and Bi-LSTM enhances the model's ability to capture both spatial and temporal features, improving multi-station forecasting accuracy.
6	An LSTM Short-Term Solar Irradiance Forecasting Under	LSTM	Forecast solar irradiance under complex weather scenarios	Demonstrated effective handling of variability in solar irradiance	LSTM models effectively handle the

	Complicated Weather Conditions (2019)			data; specific metrics not specified	variability in solar irradiance data, providing reliable short-term forecasts even under complicated weather conditions.
7	Spatio-temporal Stacked LSTM for Temperature Prediction in Weather Forecasting (2018)	Two-layer Spatio-temporal Stacked LSTM	Temperature forecasting with spatial structure awareness	MAE reduced to 1.43°C for minimum temperature and 1.22°C for maximum temperature predictions	Utilizing a two-layer stacked LSTM that combines outputs from multiple locations improves the model's ability to capture spatial dependencies, enhancing temperature prediction accuracy.

From the above table, it is evident that various deep learning approaches have enhanced the accuracy and robustness of weather forecasting models. LSTM models with spatial attention improve spatial variability handling, while basic ANN models effectively capture nonlinear trends in meteorological data. Deeper LSTM architectures yield better long-term forecasts by capturing complex dependencies. Hybrid models like CNN-LSTM and 1D CNN with Bi-LSTM integrate spatial and temporal features, leading to improved accuracy and the ability to manage high-dimensional or incomplete datasets. Overall, incorporating both spatial and temporal learning significantly strengthens the performance of forecasting systems.

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 System Requirements

To run the weather forecasting system efficiently, certain hardware and software prerequisites must be met. These requirements ensure optimal performance, especially when handling real-time data, training deep learning models, and running the web-based user interface.

3.1.1 Hardware Requirements

The hardware requirements for developing and running the model are moderate. A standard modern PC or laptop with the following specifications is sufficient:

- **Processor:** Intel i5 or higher (or AMD equivalent)
- **RAM:** Minimum 4 GB (16 GB recommended for training models)
- **Storage:** At least 10 GB free space
- **GPU:** Optional (NVIDIA GPU recommended for faster training with TensorFlow/Keras)

3.1.2 Software Requirements

The weather forecasting system is developed using Python and incorporates several machine learning, deep learning, and web technologies. The required software components for implementation and deployment are as follows:

1. **Programming Language**
    - Python – the primary language for model development, data preprocessing, and integration.
2. **Development Environment / IDE**
    - VS Code – for backend and frontend integration with Flask.
3. **Model Handling**
    - joblib, pickle – for loading pre-trained LSTM/CNN-LSTM models and scalers during runtime prediction.

4. **Libraries and Frameworks:**

   I. **Data Handling**

   - pandas, numpy – for preprocessing and manipulating datasets (e.g., temperature, precipitation, wind, humidity).

   II. **Machine Learning & Deep Learning**

   - scikit-learn – for data splitting, preprocessing, and evaluation metrics.
   - tensorflow, keras – for building, training, and saving LSTM and CNN-LSTM hybrid models.

   III. **Visualization**

   - matplotlib, seaborn – to generate accuracy and loss plots for model performance analysis.

   IV. **Web Development**

   - Flask – for building the backend server and routing between the frontend UI and model predictions.
   - HTML/CSS/JavaScript – for the frontend weather interface and visualization.

   V. **API Integration**

   - requests – for fetching real-time weather data from Weather API based on user location.

3.1.3 User Requirement Definition

The primary user requirements for the weather prediction system are accuracy, real-time responsiveness, and ease of use. The system is intended for general users, farmers, travelers, or anyone who needs reliable 7-day forecasts through an intuitive web interface. Key user requirements include:

- The system should provide accurate weather forecasts using trained LSTM and CNN-LSTM models.
- The system should fetch and process real-time weather data automatically.
- The interface should be simple, clean, and responsive for any user.
- The system should deliver results quickly and with minimal user input.
- Weather predictions should be displayed with clear icons and easy-to-read visuals.

3.1.4 System Requirement Specification

The system requirements are divided into Functional and Non-Functional requirements.

1. **Functional Requirements**

   Functional requirements define the core functionalities of the weather forecasting system must perform:

   - The system should allow users to enter a location or automatically fetch current location weather.
   - It shall fetch real-time weather data using the Weather API.
   - It must preprocess and scale the input data using saved scalers (scaler. pkl).
   - The system shall load and run predictions using LSTM and CNN-LSTM trained models.

2. **Non-Functional Requirements**

   Non-functional requirements define how the system should perform rather than what it does:

   - **Accuracy**: The system should use deep learning models (LSTM and CNN-LSTM) trained on historical weather data to ensure accurate forecasts.
   - **Efficiency**: The backend must process inputs, load models, and return predictions within a few seconds using optimized computation and libraries like TensorFlow/Keras.
   - **Scalability**: The system should be able to handle additional locations, longer forecast ranges, or increased model complexity with minimal redesign.
   - **Reliability**: The system should maintain consistent performance, handle exceptions gracefully, and recover from API failures or incorrect inputs.
   - **Usability**: The user interface must be visually appealing and intuitive, even for non-technical users, using clear weather icons and labels.
   - **Maintainability**: The codebase should be modular and documented, enabling future upgrades or model replacements easily.

CHAPTER 4

SYSTEM ARCHITECTURE AND METHODOLOGY

This section outlines the methods used to design and implement the weather forecasting system using deep learning. The development process includes dataset collection, preprocessing, model training (LSTM and CNN-LSTM), evaluation, and deployment into a web-based user interface using Flask.

4.1 Block Diagram

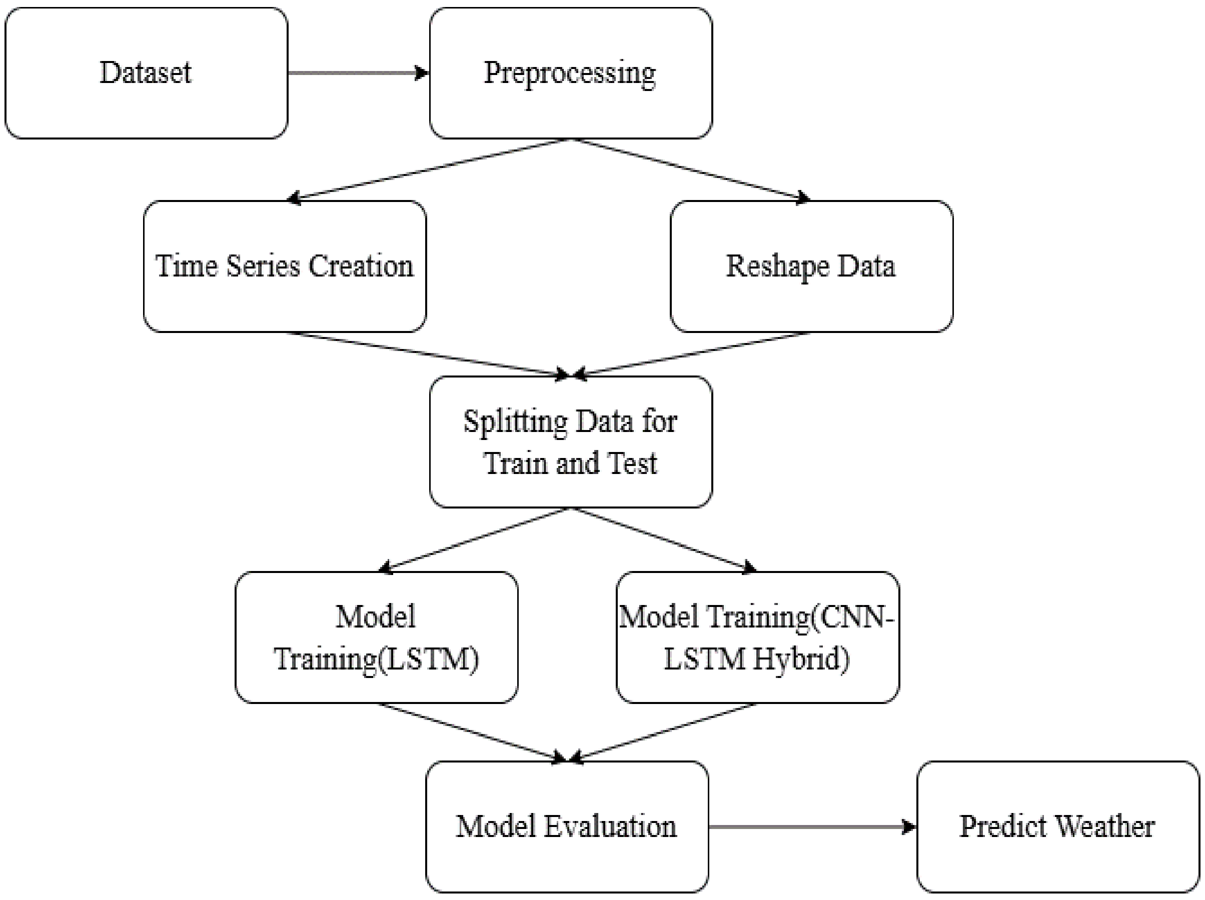


***Figure 1*:** Block Diagram of the System

4.1.1 Data Collection

To build an accurate weather forecasting model, a reliable and comprehensive dataset is essential. For this project, the historical weather dataset was obtained from Kaggle, specifically the Seattle Weather Dataset. It contains weather variables such as:

- Temperature
- Precipitation
- Wind Speed
- Maximum and Minimum Temperatures

4.1.2 Data Visualization

Data visualization is used to uncover patterns and relationships in the weather dataset prior to modeling. By graphically representing key variables over time, it becomes easier to detect trends, seasonal changes, and anomalies. These insights are crucial for guiding feature selection and engineering, ultimately enhancing the performance and reliability of the predictive models.

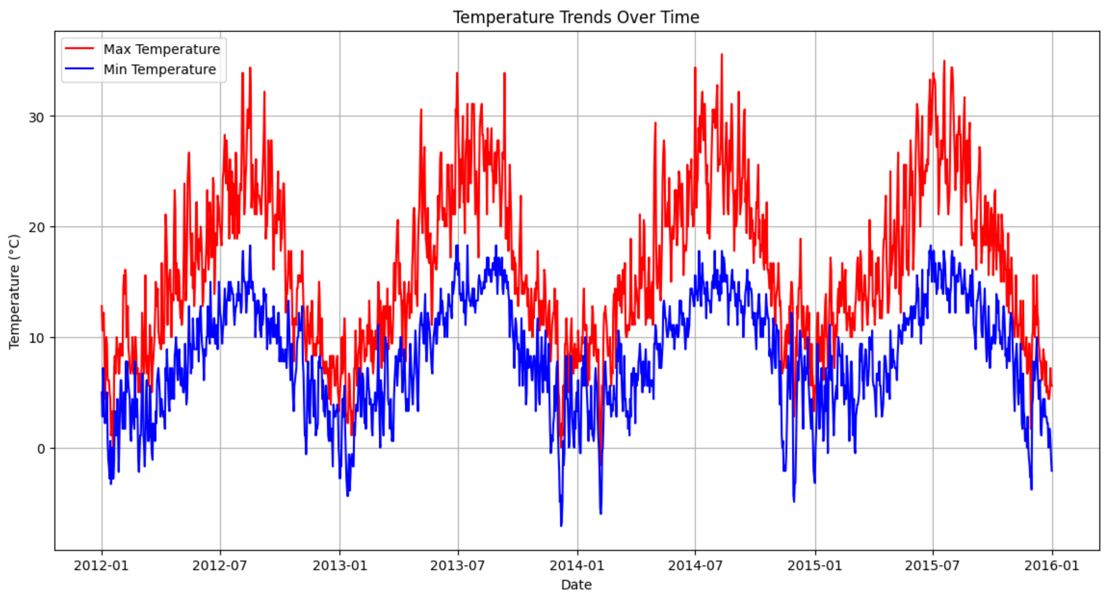


Figure 2: Temperature Trends Over Time

4.1.3 Data Pre-processing

The input features were scaled using MinMax Scaler to normalize values between 0 and 1. The scaler was saved using pickle for consistent use during deployment. Target weather conditions were one-hot encoded using pd.get_dummies() to convert categorical labels into a machine-readable format.

Table 2: Feature and Target Selection with Missing Value Handling

Date	Precipitation	Temp-max	Temp-min	Wind	Weather
2012-01-01	0.0	12.8	5.0	4.7	Drizzle
2012-01-02	10.9	10.6	2.8	4.5	Rain
2012-01-03	0.8	11.7	7.2	2.3	Rain
2012-01-04	20.3	12.2	5.6	4.7	Rain
2012-01-05	1.3	8.9	2.8	6.1	Rain

4.1.4 Data Preparation and Train-Test Split

The preprocessed dataset is split into training and testing sets (80%-20%) to evaluate model performance. Input features are scaled and reshaped to fit the 3D input format required by LSTM models: (samples, timesteps, features). The target labels are one-hot encoded to prepare them for multi-class classification. The data preparation steps ensure the input format aligns with deep learning model requirements.

4.1.5 Model Selection and Training

Two deep learning models were developed and trained using the preprocessed weather dataset. Both models were compiled using the categorical crossentropy loss function and Adam optimizer, and trained for up to 50 epochs. An EarlyStopping callback with patience=5 and min_delta=0.001 was applied to prevent overfitting by stopping training early if no significant improvement was seen in validation loss.

1. **LSTM Model**

   In this project, a Sequential Long Short-Term Memory model was implemented to learn temporal patterns in weather data, such as temperature, precipitation, and wind over time. The model consists of two LSTM layers that capture dependencies between past and future weather conditions, followed by dropout layers to reduce overfitting and a final dense layer with softmax activation for multi-class classification. This architecture is effective in understanding how historical weather trends influence the prediction of upcoming conditions, making it suitable for forecasting weather classes like sunny, rainy, or cloudy.

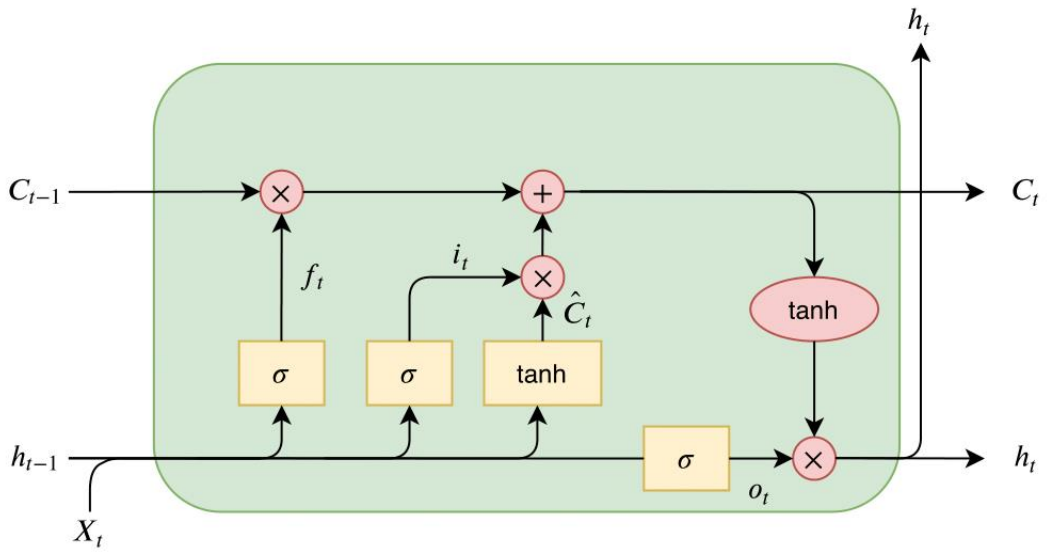


Figure 3: LSTM Architecture [8]

The LSTM architecture is designed to learn temporal patterns and long-term dependencies in sequential data, such as time-series weather records. A typical LSTM network consists of the following key components:

1. **Memory Cell**

   Acts as the core of the LSTM, responsible for storing information over long periods. It helps the network remember important patterns or sequences across time steps. The cell state carries this memory forward, updated at each step. Cell state update:

   C_t = f_t * C_{t-1} + i_t * Ĉ_t

2. **Forget Gate**

   Decides what part of the previous memory cell state should be erased. A sigmoid function generates values between 0 and 1 to scale the previous cell state where 0 means "completely forget" and 1 means "completely keep." Forget gate:

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

3. **Input Gate**

   Controls how much of the current input should be added to the memory cell. It uses a sigmoid function to filter the input and a tanh function to generate candidate values to be stored. Input gate:

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

$$\hat{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

4. **Hidden State**

   Acts as the short-term memory of the LSTM. It is updated at every time step and passed to the next LSTM unit. It reflects both the current input and the long-term memory (cell state), enabling sequence learning.

5. **Output Gate**

   Determines which information from the memory cell should be output as the hidden state. It uses the current cell state and a sigmoid filter to produce the final output that is passed to the next time step or layer. Output gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

2. CNN Model

In the weather forecasting model, the Convolutional Neural Network is used to automatically extract important spatial patterns and features from multivariate weather data such as precipitation, temperature, and wind. By applying convolutional filters over the input sequences, the CNN captures local trends and dependencies between weather variables. These extracted features are then passed to subsequent layers (e.g., LSTM in a hybrid model) for temporal analysis. CNN helps improve the accuracy of the forecast by learning relevant patterns in the data, making it well-suited for predicting complex weather conditions.

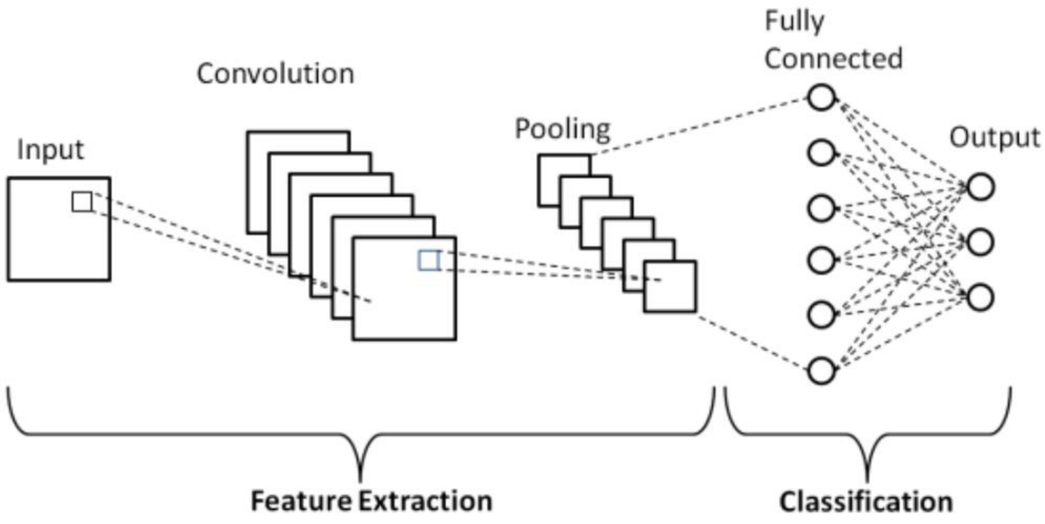


Figure 4: CNN Architecture [9]

The CNN architecture is designed to automatically learn spatial features from input data, especially images or structured grids like weather data. A typical CNN consists of the following main layers:

1. **Input Layer**
   This layer takes in raw input data, such as multivariate weather features (e.g., precipitation, temperature, wind). Each input can be treated like a matrix or image with multiple channels.

2. **Convolutional Layers**

   These are the core building blocks of a CNN. They apply a set of filters (kernels) that scan over the input data to extract local patterns. Each filter produces a feature map, highlighting important spatial relationships. Activation functions like ReLU (Rectified Linear Unit) are applied to introduce non-linearity.

3. **Pooling Layers**

   Pooling (typically max pooling) reduces the spatial dimensions of the feature maps. This helps in lowering computational load and controlling overfitting. It also makes the extracted features more robust by summarizing the presence of features in patches.

4. **Flatten Layer**

   After multiple convolution and pooling operations, the multidimensional feature maps are flattened into a one-dimensional vector to be fed into fully connected layers.

5. **Fully Connected (Dense) Layers**

   These layers act like a traditional neural network. They take the learned features and perform high-level reasoning. In weather forecasting, this might include predicting weather categories or values like temperature.

6. **Output Layer**

   The final layer provides the output—such as a class label (e.g., rain, sunny) or a numerical weather prediction. The activation function depends on the task (e.g., softmax for classification, linear for regression).

4.1.6 Model Performance and Evaluation

Deep Learning Model Performance and Evaluation focuses on how effectively the neural network learns patterns from data and generalizes to new inputs. It is assessed using metrics like accuracy, loss, precision, recall, and F1-score to ensure the model's predictive quality and robustness.

4.1.6.

A. Predictions on Test Data

After training, the saved models were loaded and used to predict classes on the unseen test data. The predicted class labels were obtained by selecting the class with the highest predicted probability (argmax) from the model outputs. Similarly, the true labels were extracted from the one-hot encoded test targets.

B. Classification Report and Confusion Matrix

The predicted classes and true classes were compared using classification reports and confusion matrices generated via sklearn's classification_report and confusion_matrix functions. These reports provide detailed insights into the models' precision, recall, F1-score per class, and overall accuracy.

C. Evaluation Metrics Theory

- **Accuracy** measures the ratio of correctly predicted instances to total instances.

  Accuracy = Total Number of Predictions / Number of Correct Predictions

  Accuracy = TP + TN / TP + TN + FP + FN

- **Precision** indicates the proportion of positive identifications that were actually correct.

  Precision= TP / TP + FP

- **Recall** (or sensitivity) measures how well the model identifies actual positives.

  Recall= TP / TP + FN

- **F1 Score** is the harmonic mean of precision and recall, balancing the two.

  F1= 2 × (Precision × Recall / Precision + Recall)

- **Confusion Matrix** is a table showing correct and incorrect predictions broken down by class:

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

D. Model Comparison and Selection

Both models were evaluated using these metrics. The LSTM model consistently showed higher accuracy and better precision, recall, and F1 scores across most classes compared to the CNN-LSTM model. Hence, the LSTM model was selected as the final model for deployment and detailed analysis.

Table 3: Model Evaluation and Comparison

Model	Batch-size	Epochs	Accuracy	Val-accuracy	Loss	Val-loss
LSTM	32	50	0.8405	0.8020	0.5325	0.6027
CNN-LSTM	32	50	0.8110	0.7816	0.5674	0.6250

From the above table, we can observe the performance metrics of two models: LSTM and CNN-LSTM. The LSTM model achieved a high accuracy on the training set and a strong validation accuracy, with a relatively low training loss and validation loss. In contrast, the CNN-LSTM model recorded a slightly lower training accuracy and validation accuracy, along with a higher training loss and validation loss. These metrics provide insights into the models' performance in terms of accuracy and loss on both training and validation datasets.

E. Visualizing Performance

Training and validation accuracy and loss curves for the LSTM model are shown in Figures respectively. These plots confirm stable learning and good generalization during training.

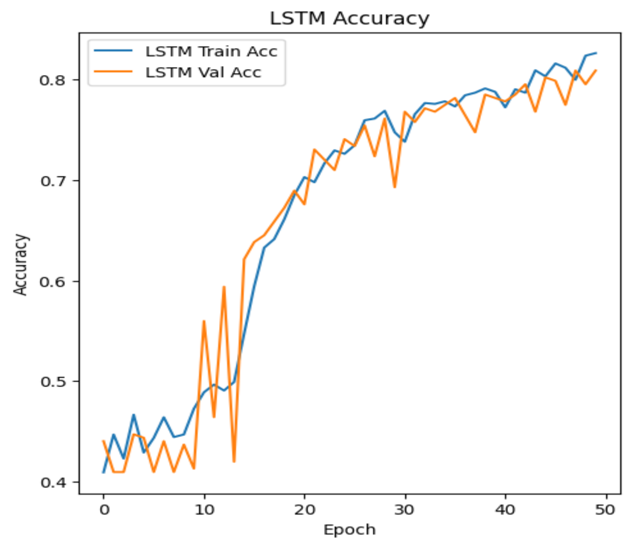


Figure 5: Training and Validation Accuracy of LSTM Model

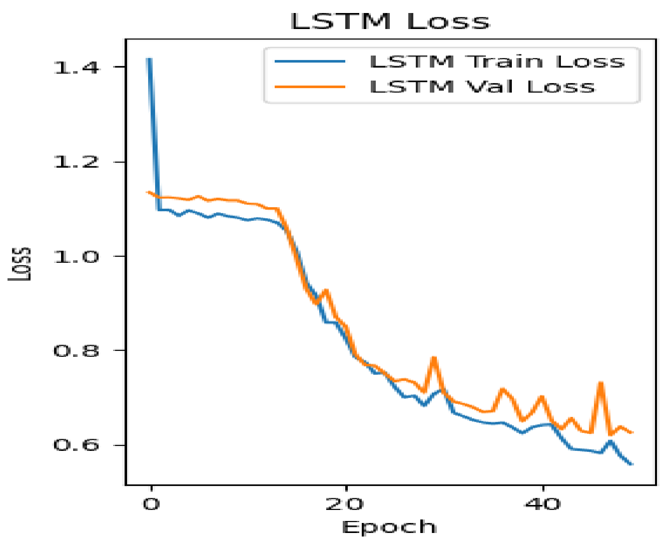


Figure 6: Training and Validation Loss of LSTM Model

4.1.7 Real-Time Weather Forecasting and Interpretation

Real-time weather forecasting and interpretation were implemented to enhance the practical usability of the system. Using Weather API, live weather data for the next seven days is retrieved for a specified location. The system interprets the forecast by analyzing weather conditions, average temperature, and precipitation levels. It highlights days with favorable weather such as sunny, clear, or cloudy to assist users in identifying ideal days for outdoor activities. This feature adds dynamic, real-world relevance to the model, making the application more user-centric and informative.

4.1.8 Model Deployment

After training and evaluating both LSTM and CNN-LSTM models, the selected model (LSTM) was deployed into a functional weather forecasting web application. This deployment stage integrates the trained predictive model with a user interface, allowing users to input their location and receive 7-day weather classification forecasts in real time. The backend handles data fetching from Weather API, preprocessing, model prediction, and result delivery. The frontend displays these results with weather icons and descriptions, enabling interactive and accessible weather forecasting through a browser-based system.

4.2 Use Case Diagram

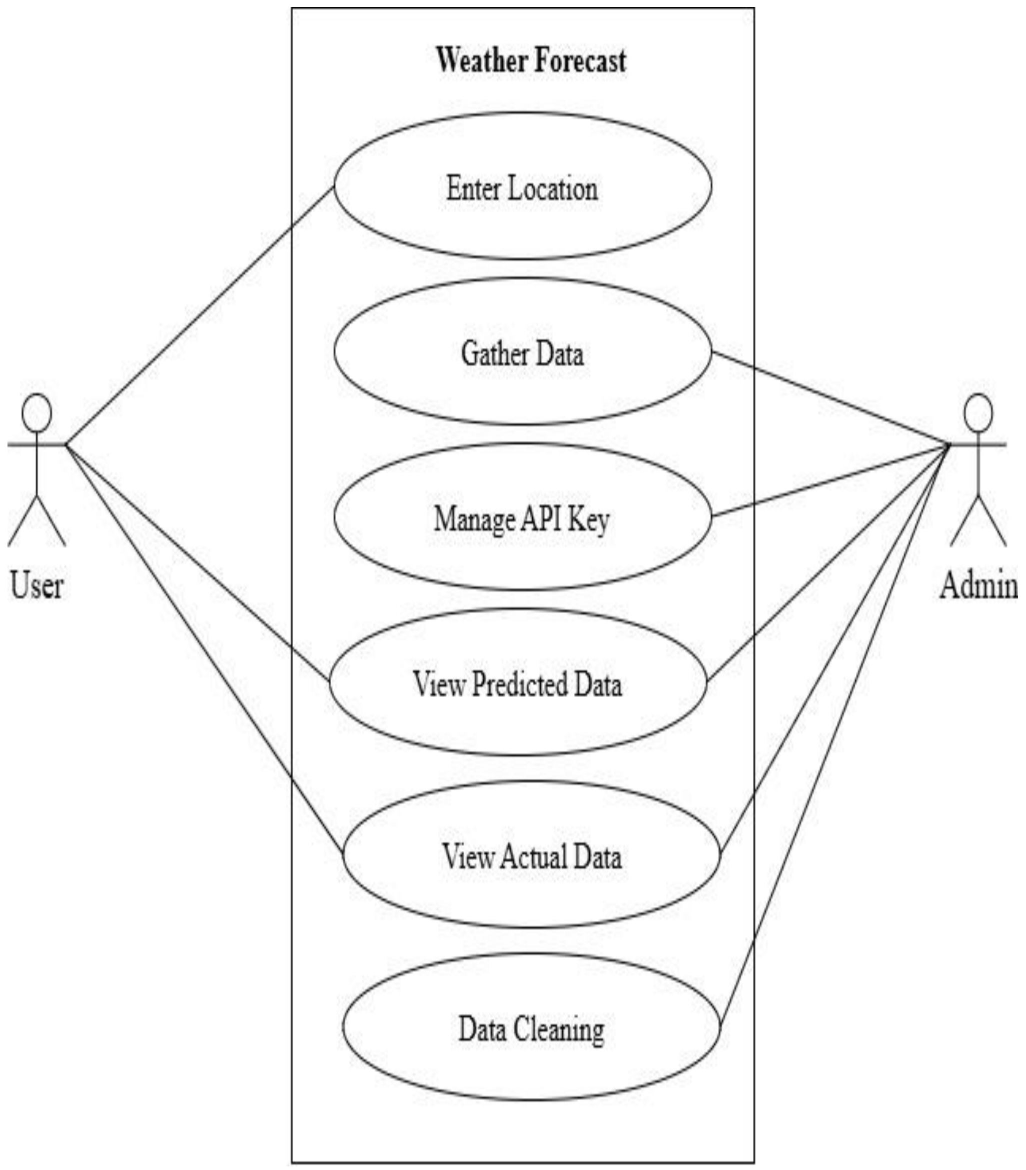


Figure 7: Use Case Diagram of the System

CHAPTER 5

IMPLEMENTATION

5.1 Implementation

System implementation focuses on building a fully functional and usable weather forecasting system. After the conceptual design and model training phases, coding was initiated to achieve the desired forecasting functionality. The core idea was to develop a real-time weather classification and prediction system that is easy to use, accessible from any browser, and supports intuitive interaction. A web-based application architecture was chosen to align with these objectives.

This phase also included detailed system and user documentation to provide clear guidance on using and understanding the system. The final implementation ensures that the system meets all the specifications and objectives defined in earlier stages, such as accuracy, usability, and real-time forecasting capability.

5.1.1 Design and Development Tools

- **Python**: Primary programming language used for data preprocessing, model training (LSTM, CNN-LSTM), evaluation, and backend API integration.
- **Flask**: For creating the backend API to integrate model predictions with real-time data and UI.
- **HTML/CSS/JavaScript**: For building a user-friendly front-end interface.
- **Matplotlib / Seaborn**: For generating accuracy and loss graphs and visualizing model performance.

5.1.2 Applications Used for Report Writing

- **MS Word**: For preparing the final report with well-formatted sections and equations.
- **Draw.io**: For drawing architecture diagrams, flowcharts, and block diagrams.
- **Excel**: For basic data exploration and visualization during preprocessing.

5.2 Tools and Techniques

The project uses LSTM and CNN-LSTM models with Keras/TensorFlow for weather forecasting. Techniques include data preprocessing, time-series input creation, and real-time WeatherAPI integration. Some of the tools and techniques used are Python, Jupyter Notebook, Pandas, NumPy, Scikit-learn, Keras, TensorFlow, and WeatherAPI.

1. **Python**

   Python served as the core programming language. Its extensive support for data science libraries, clean syntax, and integration with APIs made it ideal for building and deploying machine learning models.

2. **TensorFlow & Keras**

   Used to build and train LSTM and CNN-LSTM models for weather classification. These frameworks simplify neural network creation and provide tools for model evaluation and saving.

3. **NumPy**

   Used for efficient numerical computations, reshaping data, and handling arrays during preprocessing and model input shaping.

4. **Pandas**

   Used for reading, cleaning, and manipulating weather datasets. Pandas DataFrames enabled efficient handling of tabular data and label encoding.

5. **Scikit-learn**

   Used for data splitting (train_test_split), label encoding (LabelEncoder), and generating classification reports and confusion matrices for model evaluation.

6. **Matplotlib & Seaborn**

   Used to plot training and validation accuracy/loss graphs to visualize model performance over epochs.

7. **Weather API**

   Real-time weather data for various locations was fetched using Weather API. This API allowed dynamic integration of live weather forecasts, enabling predictions based on current and upcoming data.

8. **Flask**

   A lightweight web framework used to create the backend API which connects the frontend to the trained LSTM model and provides real-time predictions.

CHAPTER 6

RESULTS AND ANALYSIS

As this project is about weather prediction using deep learning, it focuses on developing and evaluating models that can effectively learn temporal patterns from historical weather data to classify future weather conditions. Two deep learning architectures: LSTM and CNN-LSTM hybrid were implemented, with the LSTM model achieving higher accuracy and thus selected for final deployment. The system's performance was assessed using metrics such as accuracy, precision, recall, F1-score, and confusion matrix to ensure robust evaluation. Furthermore, real-time weather data was integrated through Weather API, enabling the system to provide timely and accurate weather forecasts. The results were presented in both numerical and graphical forms, demonstrating that the deep learning-based system meets its objectives of delivering reliable and practical weather prediction.

6.1 Web Application Output and Visualization

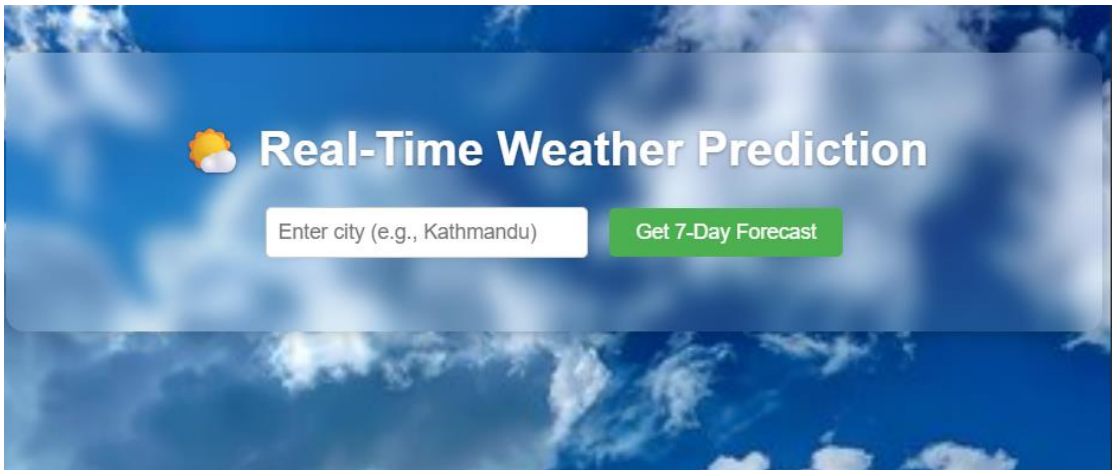
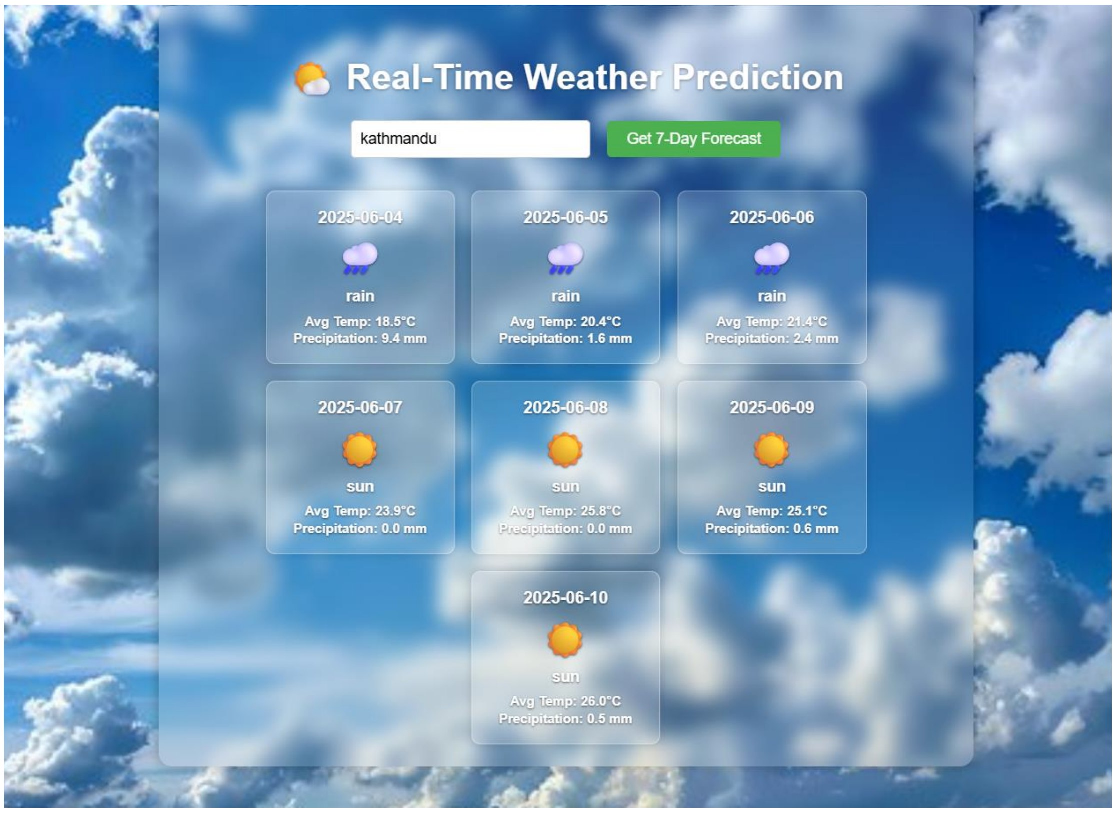


Figure 8: Web App Input Interface for Location Entry



2025-06-04
rain
Avg Temp: 18.5°C
Precipitation: 9.4 mm

2025-06-05
rain
Avg Temp: 20.4°C
Precipitation: 1.6 mm

2025-06-06
rain
Avg Temp: 21.4°C
Precipitation: 2.4 mm

2025-06-07
sun
Avg Temp: 23.9°C
Precipitation: 0.0 mm

2025-06-08
sun
Avg Temp: 25.8°C
Precipitation: 0.0 mm

2025-06-09
sun
Avg Temp: 25.1°C
Precipitation: 0.6 mm

2025-06-10
sun
Avg Temp: 26.0°C
Precipitation: 0.5 mm

Figure 9: Real-Time 7-Day Weather Forecast Display

6.2 Confusion Matrix Output

A confusion matrix is a table used to evaluate the performance of a classification model. It shows the counts of true positives, true negatives, false positives, and false negatives. This helps in measuring accuracy, precision, recall, and other metrics.

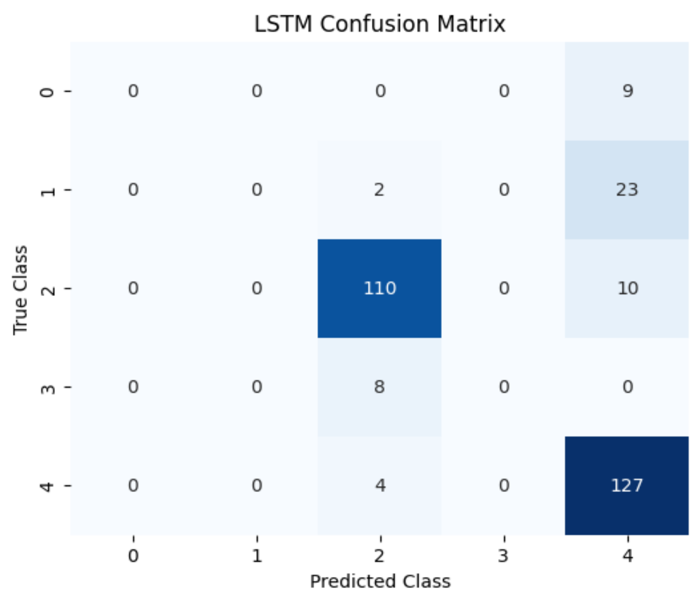


Figure 10: LSTM Confusion Matrix

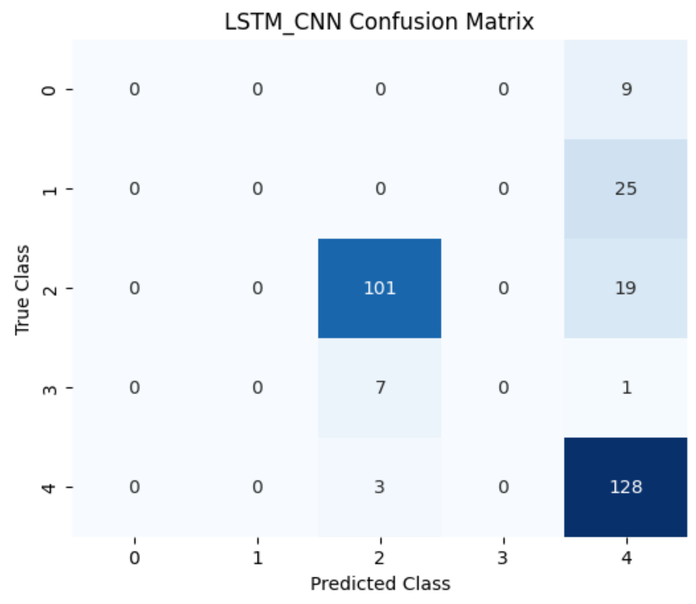


Figure 11*:* LSTM_CNN Confusion Matrix

CHAPTER 7

CONCLUSION

7.1 Conclusion

In this project, we developed a weather forecasting system using two deep learning models: Long Short-Term Memory and CNN-LSTM. Both models were trained on historical weather data to predict future weather conditions. The LSTM model effectively captured temporal patterns in the data, while the CNN-LSTM model combined convolutional layers to learn spatial features along with temporal sequences. Based on the evaluation results, the LSTM model achieved higher accuracy compared to the CNN-LSTM model. This indicates that for our dataset and prediction task, the LSTM architecture was more effective. Overall, the project demonstrates the usefulness of deep learning techniques in weather forecasting and highlights how model selection depends on the nature of the data and problem.

7.2 Future Work

For future enhancements, the project can be expanded by incorporating more sophisticated deep learning architectures such as Transformer models or hybrid CNN-LSTM with attention mechanisms to potentially improve prediction accuracy. Additionally, integrating larger and more diverse datasets, including real-time sensor data, could further refine forecasting performance. Enhancing the web application with more interactive visualizations, location-based services, and mobile app compatibility will also improve user engagement and accessibility.

REFERENCES

[1] M. A. R. S. a. S. Shridevi, "“Short-Term Weather Forecasting Using Spatial Feature Attention Based LSTM Model"," *IEEE Access,* vol. 10, no. 1, p. 23393–23403, 2022.

[2] D. N. F. a. D. K. Singh, " “Weather Forecasting Using Artificial Neural Network”," *Proc. 2nd Int. Conf. Intelligent Computing and Control Systems (ICICCS), Madurai, India,* p. 862–867, 2018.

[3] Y. H. E. A. a. W. S. A. G. Salman, " “Single Layer & Multi-layer Long Short-Term Memory (LSTM) Model with Intermediate Variables for Weather Forecasting,”," *Procedia Computer Science,* vol. 135, p. 89–98, 2018.

[4] Y. Z. F. W. a. C.-H. L. Y. Gong, "“Deep Learning for Weather Forecasting: A CNN-LSTM Hybrid Model for Predicting Historical Temperature Data,”," *arXiv (Cornell University),* 2024.

[5] D. N. Z. Z. J. H. a. L. D. Q. Fu, "“Multi-Stations’ Weather Prediction Based on Hybrid Model Using 1D CNN and Bi-LSTM,”," *2019 Chinese Control Conference (CCC),* p. 8890–8895, 2019.

[6] J. C. a. J. Z. Y. Yu, " “An LSTM Short-Term Solar Irradiance Forecasting Under Complicated Weather Conditions,”," *IEEE Access,* vol. 7, no. 1, p. 145651–145666, 2019.

[7] Z. K. a. J. A. K. Suykens, "“Spatio-temporal Stacked LSTM for Temperature Prediction in Weather Forecasting,”," *arXiv (Cornell University),* 2018.

[8] S. Hesaraki, "Long Short-Term Memory (LSTM)," *Online,* 2023.

[9] S. Cota, "Deep Learning Basics — Part 8 — Convolutional Neural Network (CNN)," *Online,* 12 Dec 2023.

APPENDICS

Appendix A: Project Timeline

Table 4: Project Timeline

	Jan-Feb	Mar-Apr	May-Jun	July
Requirement Gathering	■			
Analysis	■			
Model Creation	■	■		
Web Design		■	■	
Testing and Debugging		■	■	■
Documentation	■	■	■	■