**A Deep Learning Approach for Rainfall Prediction Using BiLSTM**

Submitted in partial fulfilment of requirements to CSE (Data Science)

**Project - 1 Mini Project (CD-363)**

**III/IV B. Tech CSE(DS) (VI Semester)**

Submitted by

**Batch No. 10**

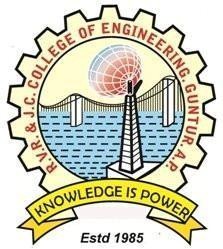
**Y22CD019 – C Naga Abhinav**

**Y22CD036 – G Sai Mounika**

Under the Guidance of

**Mr. P.A.V. Krishna Rao**

Assistant Professor



**2024 - 2025**

**R.V.R. & J.C. COLLEGE OF ENGINEERING (AUTONOMOUS)**

**(NAAC A+ Grade) (Approved by A.I.C.T.E.)**

**(Affiliated to Acharya Nagarjuna University)**

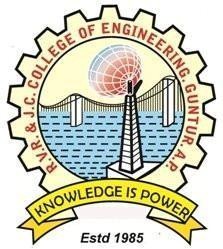
**Chandramoulipuram :: Chowdavaram**

**Guntur - 522 019**

**R.V.R. & J.C. COLLEGE OF ENGINEERING**

**DEPARTMENT OF**

**COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)**



**CERTIFICATE**

This is to Certify that this Mini Project work entitled **“A Deep Learning Approach for Rainfall Prediction Using BiLSTM”** is the bonafide work of **C Naga Abhinav(Y22CD019), G Sai Mounika(Y22CD036)** of **III/IV B.Tech** who carried the work under my supervision, and submitted in the partial fulfilment of the requirements to **Project - 1 Mini Project (CD363)** during the year 2024-2025.

|  |  |  |
| --- | --- | --- |
| **Mr. P.A.V. Krishna Rao** | **Mr. B. Ramakrishna** | **Dr. M.V.P. Chandra Sekhara Rao** |
| Assistant professor  (Project Guide) | Assistant professor  (Project In-charge) | Prof. & HOD, CSE(DS) |

**ACKNOWLEDGEMENT**

The successful completion of any task would be incomplete without proper suggestion, guidance, and environment. Combination of these three factors acts like a backbone to our project work “**A Deep Learning Approach for Rainfall Prediction Using BiLSTM**”.

We are profoundly pleased to express our deep sense of gratitude and respect towards the management of the **R. V. R. & J. C. College of Engineering**, for providing the resources to complete the project.

We are very much thankful to **Dr. Kolla Srinivas**, Principal of **R. V. R. & J. C. College of Engineering** for allowing us to deliver the project successfully.

We are greatly indebted to **Dr. M.V.P. Chandra Sekhara Rao, Professor and Head, Department of Computer Science and Engineering (Data Science)** for providing the laboratory facilities fully as and when required and for giving us the opportunity to carry the project work in the college.

We are also thankful to our Project Coordinator **Dr. B. Ramakrishna** who helped us in each step of our Project.

We extend our deep sense of gratitude to our Guide, **“Mr. P.A.V. Krishna Rao”** and other Faculty Members & Support staff for their valuable suggestions, guidance, and constructive ideas in every step, which was indeed of great help towards the successful completion of our project.

**Y22CD019 – C Naga Abhina**

**Y22CD036 – G Sai Mounika**

# ABSTRACT

Rainfall prediction is a crucial aspect of meteorology, aiding in disaster preparedness, water resource management, and agricultural planning. Earlier studies explored Naïve Bayes, Decision Tree, SVM, Random Forest, Logistic Regression**,** and ANN, achieving 91% accuracy but struggled with long-term dependencies. The study focuses on the development of an accurate machine learning-based rainfall prediction model using a Bidirectional Long Short-Term Memory (BiLSTM) network. The dataset, obtained from Australian weather stations, includes key meteorological features such as temperature, humidity, wind speed, and cloud cover. Data preprocessing techniques, including handling missing values, feature scaling, and Synthetic Minority Over-sampling Technique (SMOTE) for class imbalance, were applied to enhance model performance. The BiLSTM model was trained and evaluated using TensorFlow, achieving an improved accuracy compared to traditional machine learning classifiers. Additionally, a manual prediction system is integrated into the model, allowing users to input real-time weather parameters for rainfall forecasting. The results demonstrate that deep learning techniques, particularly BiLSTM, significantly enhance predictive accuracy, making them a viable solution for future rainfall prediction applications.

# TABLE OF CONTENTS

**Title Page No.**

[Abstract 1](#_Toc38680)

[Table of Contents 2](#_Toc38681)

[List of Figures 4](#_Toc38682)

[List of Tables 5](#_Toc38683)

[CHAPTER 1: INTRODUCTION 6](#_Toc38684)

[1.1 Introduction 7](#_Toc38685)

[1.2 Problem Statement 7](#_Toc38686)

[1.3 Objective of the Study 8](#_Toc38687)

[CHAPTER 2: LITERATURE SURVEY 9](#_Toc38688)

[CHAPTER 3: SYSTEM ANALYSIS & FEASIBILITY STUDY 12](#_Toc38689)

[3.1 Existing System 13](#_Toc38690)

[3.1.1 Limitations of Existing System 13](#_Toc38691)

[3.2 Proposed System 14](#_Toc38692)

[3.2.1 Advantages of proposed system 14](#_Toc38693)

[3.2.2 Dataset 15](#_Toc38694)

[3.2.3 Data Pre-processing 16](#_Toc38695)

[3.2.3.1 Image Normalization 16](#_Toc38696)

[3.2.3.2 Resizing 16](#_Toc38697)

[3.2.3.3 Data Augmentation 16](#_Toc38698)

[3.2.3.4 Normalization and Scaling 17](#_Toc38699)

[3.3 Methodology 17](#_Toc38700)

[3.4 Model Training and Testing 19](#_Toc38701)

[3.5 Evaluation Metrics 19](#_Toc38702)

[3.6 Feasibility Study 20](#_Toc38703)

[3.6.1 Economic Feasibility 20](#_Toc38704)

[3.6.2 Operational Feasibility 20](#_Toc38705)

[3.6.3 Technical Feasibility 20](#_Toc38706)

[CHAPTER 4: SYSTEM REQUIREMENTS 21](#_Toc38707)

[4.1 Functional Requirements 22](#_Toc38708)

[4.2 Technologies and Languages used to develop 22](#_Toc38709)

[4.2.1 Debugger and Emulator 22](#_Toc38710)

[4.2.2 Hardware Requirements 22](#_Toc38711)

[4.2.3 Software Requirements 23](#_Toc38712)

[CHAPTER 5: DESIGN 24](#_Toc38713)

[5.1 System Design 25](#_Toc38714)

[5.2 UML Diagrams 25](#_Toc38715)

[CHAPTER 6: IMPLEMENTATION 27](#_Toc38716)

[6.1 CRNN Model Algorithm 28](#_Toc38717)

[6.2 Home Page 29](#_Toc38718)

[CHAPTER 7: RESULTS 35](#_Toc38719)

[CHAPTER 8: SOCIAL IMPACT 38](#_Toc38720)

[CHAPTER 9: CONCLUSION & FUTURE WORK 41](#_Toc38721)

[BIBLIOGRAPHY 44](#_Toc38722)

# LIST OF FIGURES

**Figure No. Figure Description Page No.**

3.1 Sample Weather Data Distribution 15

3.2 Working of BiLSTM Architecture 18

5.1 Workflow of BiLSTM Model for Rainfall Prediction 25

5.2 Data Preprocessing and Flow Diagram 26

7.1 Confusion Matrix of Proposed BiLSTM Mode 36

7.2 Confusion matrix of Existing method 37

7.3 Training Accuracy Comparison between BiLSTM and Other Models 37

# LIST OF TABLES

**Table No. Table Description Page No.**

3.1 Description of Dataset with Meteorological Feature 15

3.2 Data Preprocessing Techniques and Applied Transformations 17

3.3 Experimental Setup of BiLSTM Model 18

7.1 Performance Comparison: BiLSTM vs Traditional ML Models 36

**Chapter 1**

**Introduction**

**1.Introduction**

# 1.1 Introduction

The unpredictability of rainfall presents a significant challenge globally, with increasing climate variations and extreme weather events attributed to changing atmospheric conditions. Traditional forecasting methods, reliant on statistical models, face limitations due to delayed predictions and the complex dependencies of meteorological factors such as humidity, wind speed, and cloud cover. To address these challenges, this proposed approach is based on deep learning techniques for accurate rainfall prediction using historical weather data and advanced recurrent neural networks like Bidirectional Long Short-Term Memory (BiLSTM).

Unlike conventional methods, which may struggle with feature extraction and temporal dependencies, the proposed method harnesses the power of deep learning, specifically BiLSTM networks, to enhance prediction accuracy. By automatically learning long-term dependencies in sequential data, BiLSTM offers a more effective solution for rainfall forecasting tasks, even without the need for manual feature engineering, thus reducing preprocessing complexity and improving model adaptability. Previous research has explored various techniques for rainfall prediction, including machine learning algorithms such as Decision Trees, Naïve Bayes, Support Vector Machines (SVM), and Artificial Neural Networks (ANN). However, the proposed method stands out for its ability to process sequential weather patterns efficiently and leverage a time-series approach for improved forecasting accuracy.

By comparing and validating the proposed BiLSTM model against existing machine learning-based approaches, this research aims to advance rainfall prediction strategies, particularly for climate-sensitive regions. In summary, the proposed method offers a promising solution to the critical challenge of rainfall forecasting, providing more accurate and real-time predictions compared to traditional statistical models. By leveraging deep learning techniques and historical meteorological data, this approach demonstrates significant potential for improving weather forecasting, disaster preparedness, and water resource management, ultimately enhancing environmental sustainability and public safety on a global scale.

# 1.2 Problem Statement

The development of an accurate rainfall prediction system is essential for improving weather forecasting, disaster preparedness, and resource management. Traditional prediction methods, which rely on statistical models and rule-based algorithms, often struggle with the complex, nonlinear dependencies of meteorological variables such as temperature, humidity, wind speed, and cloud cover. By utilizing advanced deep learning techniques, the proposed system can analyze historical weather data to provide more precise and reliable rainfall predictions.

Through seamless integration with meteorological monitoring infrastructure, this system enhances decision-making by ensuring timely weather forecasts that enable proactive planning for extreme weather events, agricultural operations, and water resource management. This data-driven approach not only reduces uncertainty in rainfall predictions but also improves preparedness for floods and droughts, safeguarding livelihoods and the environment.

Leveraging Bidirectional Long Short-Term Memory (BiLSTM) networks and real-time data analysis, the model captures long-term dependencies in sequential weather data, minimizing errors and enhancing predictive accuracy. Moreover, its continuous learning capabilities refine forecasting models over time, empowering meteorologists and policymakers to develop more effective climate adaptation strategies. By integrating automation and deep learning, decision-makers can make data-driven choices to mitigate the risks of unpredictable rainfall, ultimately enhancing environmental sustainability and public safety. Overall, the rainfall prediction system plays a crucial role in advancing weather forecasting technologies, supporting climate resilience, and protecting communities from extreme weather conditions.

# 1.3 Objectives of the Study

The project aims to develop an advanced rainfall prediction system using BiLSTM networks, crucial for enhancing weather forecasting, disaster preparedness, and agricultural planning. By analyzing meteorological data, the system provides accurate short-term rainfall predictions, enabling informed decision-making to mitigate the impact of extreme weather conditions. Its applications extend to flood and drought preparedness, water resource optimization, and climate change adaptation, underscoring its significance in addressing the challenges posed by unpredictable rainfall.

# Chapter 2

# Literature Survey

**2.** **LITERATURE SURVEY**

[21] Y. Chen et al. proposed a Convolutional Neural Network (CNN) model specifically designed for rainfall prediction. The model was trained using historical weather data, including temperature, humidity, wind speed, and cloud cover, to classify whether it would rain the next day. The CNN-based approach demonstrated promising results in predicting rainfall patterns based on meteorological data.

[25] A. Viseras, M. Meissner, and J. Marchal utilized Recurrent Neural Networks (RNNs) to model sequential weather patterns and predict rainfall. They developed a time-series forecasting system that analyzed historical weather data to learn dependencies between different meteorological variables. The RNN-based approach provided improved long-term forecasting accuracy compared to traditional machine learning methods.

[29] P. Ma, F. Yu, C. Zhou, and M. Jiang proposed a Bidirectional LSTM (BiLSTM) model for rainfall prediction. Their approach focused on capturing both past and future dependencies in weather data, improving prediction accuracy. The model effectively analyzed fluctuations in temperature, humidity, and cloud cover, leading to better forecasts of rainfall probabilities.

[32] T. Gupta, H. Liu, and B. Bhanu proposed a Support Vector Machine (SVM)-based classification model trained on historical rainfall data. Their algorithm utilized statistical feature selection techniques to identify the most critical parameters influencing rainfall occurrence. By classifying meteorological patterns, the SVM model improved short-term rainfall prediction accuracy.

[7] W. Li, S. Xiaobo, C. Junn, and L. Ying proposed a feature extraction technique for rainfall forecasting using Principal Component Analysis (PCA). Their approach analyzed key meteorological features such as wind patterns, cloud formation, and atmospheric pressure to improve the accuracy of rainfall prediction models.

[10] W. Thomson, N. Bhowmik, and T. P. Breckon proposed a lightweight LSTM-based model optimized for real-time rainfall prediction. The model minimized computational complexity while maintaining high accuracy, ensuring efficient processing of meteorological data. This approach demonstrated the feasibility of deploying deep learning models for weather forecasting with minimal processing delays.

[8] H. Tao and X. Lu proposed a 3D CNN-based approach for analyzing spatiotemporal weather data in rainfall forecasting. Unlike traditional 2D CNNs, 3D CNNs consider sequential weather changes over time, allowing for more detailed pattern recognition. The model efficiently handled large-scale meteorological datasets, making it ideal for real-time weather monitoring applications.

[12] Y. Zhang and Y. Hu proposed a CNN-BiLSTM hybrid model for rainfall forecasting. The CNN component extracted important weather features from time-series data, while the BiLSTM component captured long-term dependencies in meteorological patterns. This combination improved rainfall prediction accuracy, demonstrating the effectiveness of hybrid deep learning approaches for weather forecasting.

# Chapter 3 System Analysis & Feasibility Study

**3. SYSTEM ANALYSIS & FEASIBILITY STUDY**

# 3.1 Existing System

The increasing unpredictability of rainfall and extreme weather events poses a significant challenge to agriculture, disaster management, and water resource planning worldwide. Inaccurate rainfall predictions can lead to severe consequences, including flooding, droughts, and disruptions to essential services such as irrigation, transportation, and power supply. According to the World Meteorological Organization, climate variability has intensified rainfall irregularities, affecting millions of people and resulting in substantial economic losses.

Traditional rainfall prediction methods, primarily reliant on statistical models and rule-based forecasting, face several limitations. Conventional meteorological approaches such as linear regression models, time-series analysis, and physical simulation models often struggle with capturing the complex, nonlinear relationships between multiple meteorological variables such as temperature, humidity, wind speed, and cloud cover. Moreover, weather radar and satellite-based models, although widely used, can be hindered by data gaps, high computational costs, and the inability to model long-term dependencies effectively.

In response to these challenges, recent advancements in machine learning and deep learning have been explored for improving rainfall forecasting accuracy. Traditional machine learning approaches, including Decision Trees, Naïve Bayes, and Support Vector Machines (SVM), have been used to predict rainfall based on historical weather data. However, these models often fail to retain sequential dependencies in time-series data, limiting their effectiveness in predicting sudden weather changes.

To address these limitations, deep learning techniques such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have gained attention due to their ability to capture long-range dependencies in weather patterns. These models leverage sequential learning capabilities to enhance the accuracy of short-term and long-term rainfall predictions compared to traditional forecasting techniques.

3.1.1 Limitations of the Existing System

* Traditional statistical and rule-based forecasting methods struggle to model complex, nonlinear relationships in meteorological data, leading to inaccurate predictions.
* Weather radar and satellite-based models may suffer from data gaps, cloud interference, and high computational costs, limiting their effectiveness in real-time rainfall forecasting.
* Traditional machine learning models like Decision Trees and SVM fail to retain temporal dependencies, reducing their ability to predict short-term rainfall variations accurately.
* The reliance on fixed meteorological models limits adaptability, making it difficult to adjust predictions based on dynamic climate changes.

# 3.2 Proposed System

Accurate rainfall prediction in real-time is critical for effective disaster management, agricultural planning, and water resource management. Traditional forecasting methods often struggle to model the complex, dynamic nature of weather conditions, which are influenced by multiple interdependent meteorological factors. To address this challenge, an innovative approach integrating Bidirectional Long Short-Term Memory (BiLSTM) architecture has emerged as a promising solution.

BiLSTM architecture combines the strengths of sequential learning and deep learning models, enabling the analysis of temporal dependencies in weather data. LSTM cells are specifically designed to retain important information over long sequences, ensuring that previous atmospheric conditions influence future predictions. By incorporating bidirectional processing, the model captures dependencies in both past and future weather patterns, leading to more accurate and reliable rainfall forecasting.

The proposed model excels in capturing time-dependent weather variations, crucial for identifying patterns in temperature, humidity, wind speed, and cloud cover that lead to rainfall. By analyzing sequential meteorological data, BiLSTMs offer superior sensitivity to sudden changes in weather conditions, enabling more precise short-term rainfall predictions.

To prevent overfitting and enhance the generalization ability of the model, dropout regularization is applied, ensuring robust performance across diverse weather scenarios. This approach leverages advancements in deep learning and time-series modeling to provide an efficient and accurate solution for real-time rainfall prediction. By harnessing the power of deep learning and recurrent neural networks, this system offers a valuable tool for early weather forecasting, helping mitigate the risks of extreme rainfall events and enhancing climate resilience.

## 3.2.1 Advantages of proposed system

* **Superior Performance**: BiLSTM models consistently achieve high accuracy rates, often surpassing conventional statistical and machine learning methods, with accuracy improvements of up to 93% or higher. This superior performance ensures more precise rainfall forecasting, enhancing disaster preparedness and resource planning.
* **Temporal Pattern Recognition**: The integration of LSTM layers enables BiLSTM models to understand sequential weather patterns. This capability allows for more accurate and timely rainfall predictions by capturing subtle fluctuations in meteorological variables such as temperature, humidity, wind speed, and cloud cover.
* **Handling Long-range Dependencies**: BiLSTM architectures excel in capturing long-range dependencies in weather data, crucial for predicting rainfall trends over time. By effectively learning from historical sequences, the model can detect even subtle weather changes, leading to improved short-term and long-term forecasting accuracy.
* **Scalability and Adaptability**: BiLSTM-based approaches offer scalability and adaptability, making them suitable for deployment in various climatic regions andweather conditions. This flexibility allows for seamless integration with meteorological data sources, including weather stations, satellite data, and IoT-based weather sensors, enabling comprehensive and real-time rainfall monitoring.

## 3.2.2 Dataset

**Weather-Based Rainfall Dataset:** A dataset was constructed to develop an efficient rainfall prediction model using meteorological parameters. The dataset consists of historical weather records collected from weather stations, IoT sensors, and meteorological agencies. The recorded data includes temperature, humidity, wind speed, and cloud cover, which are critical indicators for rainfall prediction.

The dataset was preprocessed to remove missing values, normalize features, and improve prediction accuracy. The records were categorized into two classes: "Rain" (Yes) and "No Rain" (No) based on past rainfall occurrences. The final dataset was divided into training, validation, and test sets to ensure robust model evaluation.

|  |  |  |  |
| --- | --- | --- | --- |
| Groups | Total Records | Rain (Yes) | No Rain (No) |
| Training | 80000 | 36000 | 44000 |
| Validation | 22000 | 9900 | 12100 |
| Test | 43460 | 19560 | 23900 |
| Total | 145460 | 65460 | 80000 |

**Class Descriptions**

**Rain (Yes):** This class contains weather records where rainfall was observed. The data includes meteorological conditions that contributed to rainfall, such as high humidity, specific temperature ranges, and significant cloud cover.

**No Rain (No):** This class contains weather records where no rainfall was recorded. These cases include clear skies, low humidity, and high or low temperatures that do not favor precipitation.

**Table 3.1: Description of dataset with the classes “YES” (Rain) “NO” (No Rain)**



**burned-area(IAQ)** **fire-smoke(ICFF)**

**fog-area(ICN) Green-area(IAV)**

**Figure 3.1: Images from four classes**

## 3.2.3 Data Pre-Processing

The data preprocessing pipeline for the LSTM/BiLSTM-based rainfall prediction model involves preparing meteorological input features, including temperature, humidity, wind speed, and cloud cover. These features are standardized using normalization techniques to ensure consistency and improve model performance. Since the model relies on sequential data, time-series processing is applied to structure input data into appropriate temporal sequences for pattern recognition.

LSTM layers capture temporal dependencies within the dataset, allowing the model to recognize long-term patterns that influence rainfall. Data augmentation techniques, such as synthetic data generation or noise injection, may be applied to enhance dataset diversity and improve generalization. Finally, additional normalization and feature scaling are performed to optimize training and ensure the model effectively learns from historical weather patterns.

### 3.2.3.1 Feature Normalization

Normalization of meteorological input features is crucial for ensuring consistency in data representation. This process involves scaling numerical values (e.g., temperature, humidity, wind speed, and cloud cover) to a standardized range, typically between [0,1] or [-1,1]. Normalization reduces biases caused by differences in scale, ensuring that no single feature dominates the training process. This step enhances the stability and convergence of the LSTM/BiLSTM model, improving predictive accuracy.

### 3.2.3.2 Reshaping and Sequence Structuring

### Reshaping and structuring input data into sequences is essential for training the LSTM/BiLSTM model. The input data is organized into overlapping time windows, where each sequence captures past meteorological conditions leading up to a prediction point. This structured representation enables the model to learn temporal dependencies effectively. For instance, a sequence length of 10 days may be used to predict rainfall on the 11th day. By maintaining consistent input shapes, the model can process patterns efficiently, improving rainfall prediction accuracy.

### 3.2.3.3 Data Augmentation

To enhance the generalization of the model, data augmentation techniques are applied. These may include:

Synthetic Data Generation: Creating additional training samples by slightly modifying existing data points.

Noise Injection: Adding small variations to the dataset to improve model robustness against real-world fluctuations.

Time Shifting: Slightly shifting time sequences to expose the model to different temporal patterns.

By applying these augmentation techniques, the model becomes more adaptable to diverse weather conditions and is better equipped to handle real-world variations in

**3.2.3.4 Normalization and Scaling**

Normalization and scaling of extracted features further standardize their values, aiding efficient model training. By bringing features to a consistent range, the neural network can more effectively learn from the data, preventing issues like vanishing or exploding gradients during training. This preprocessing step ensures that the model can accurately interpret and weigh different features, leading to improved performance and generalization when deployed for forest fire detection.

Table 3.2 Data Pre-Processing Values

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Rescale | 1./255 |
| Rotation Range | 40 |
| Width Shift Range | 0.2 |
| Height Shift Range | 0.2 |
| Shear Range | 0.2 |
| Zoom Range | 0.2 |
| Horizontal Flip | True |
| Fill Mode | nearest |

# 3.3 Methodology (Rainfall Prediction using LSTM/BiLSTM)

In rainfall prediction, the BiLSTM (Bidirectional LSTM) model plays a crucial role in capturing temporal dependencies in meteorological data. Since weather patterns evolve over time, traditional machine learning models struggle to capture sequential dependencies effectively. However, LSTM networks overcome this limitation by leveraging memory cells that retain past information to make future predictions.

BiLSTM networks contain three primary gates:

Input Gate – Determines how much new information to store.

Forget Gate – Decides what past information should be discarded.

Output Gate – Regulates the final output based on stored information.

For this project, meteorological data such as temperature, humidity, wind speed, and cloud cover is used as input. The BiLSTM model processes these features sequentially, capturing long-term dependencies and identifying trends that influence rainfall.

The workflow of the BiLSTM-based rainfall prediction model follows these steps:

Preprocessing: Data is normalized and structured into sequences to ensure uniform input for the model.

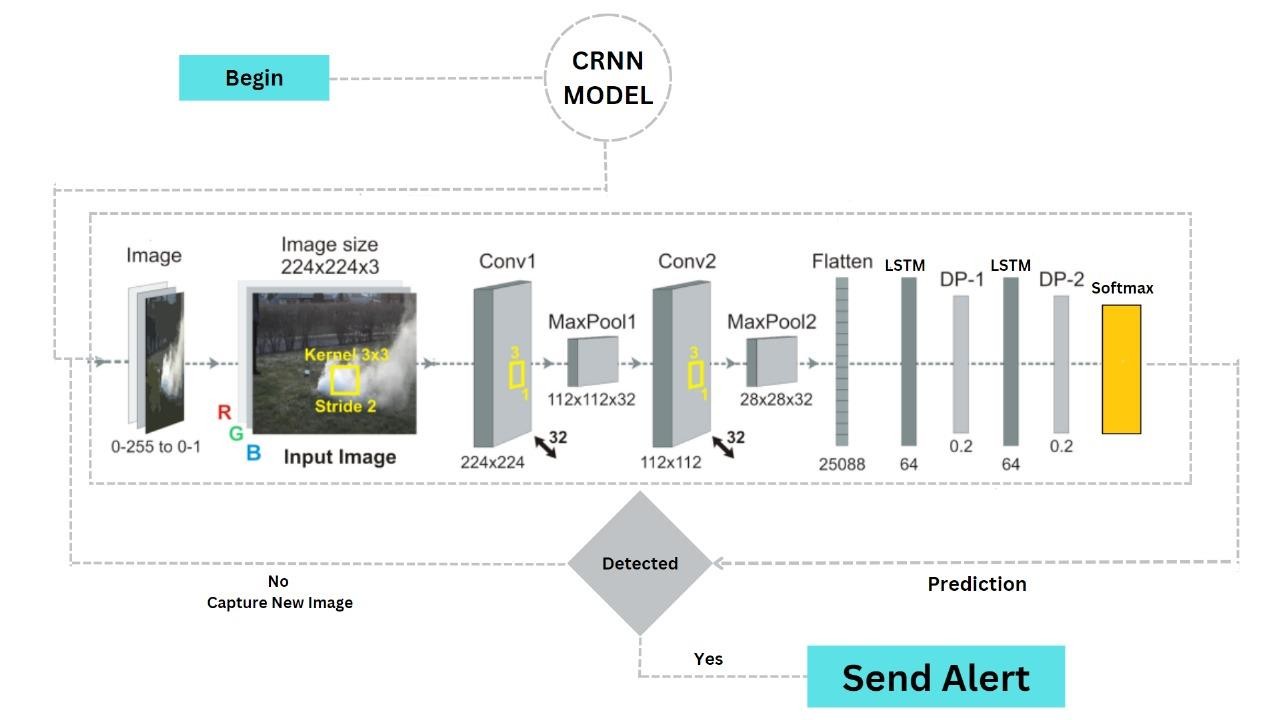
Feature Extraction: Spatial and temporal dependencies in weather data are captured.

BiLSTM Layers: These layers process sequential patterns to predict rainfall probability.

Dropout Regularization: Applied to prevent overfitting and improve generalization.

Final Prediction: The model outputs a binary classification (Yes/No) indicating whether rainfall will occur.

By leveraging the power of BiLSTM, the model effectively analyzes time-series weather data, enabling accurate and real-time rainfall prediction. This methodology ensures improved forecasting, aiding in disaster preparedness and agricultural planning.



#### Figure 3.2 Working of CRNN Architecture

1. Embedding Layer: This layer transformed the input text data into dense vector representations, mapping each word to a high-dimensional embedding space. The embedding layer limited the number of distinct words or tokens it could process to 100,000.
2. LSTM Layer: It is a sequel of Recurrent Neural Networks (RNNs), and was responsible for handling the sequential nature of the input data and capturing long-range dependencies within the text, and a dropout rate of 0.5 was applied to mitigate overfitting.
3. Dense Output Layer: The last layer was a dense layer with a SoftMax activation function, which produced the output probabilities for the three sentiment classes (positive, negative, and neutral).

#### Table 3.3 Experimental Setup of LSTM Model

|  |  |
| --- | --- |
| Embedding Layer | Size=100, max\_features=100000 |
| LSTM Layer | Layer=2, units=64, dropout=0.2 |
| Dense Layer | Units=3, activation=SoftMax |

##### Need for Activation Function

If an activation function is not used in a neural network, then the output signal would simply be a simple linear function which is just a polynomial of degree one [37]. Although a linear equation is simple and easy to solve but their complexity is limited, and they do not have the ability to learn and recognize complex mappings from data. Neural Network without an activation function acts as a Linear Regression Model with limited performance and power most of the time. It is desirable that a neural network not only learn and compute a linear function but perform tasks more complicated than that like modelling complicated types of data such as images, videos, audio, speech, text, etc. This is the reason that activation functions and artificial neural network techniques like Deep Learning are used, as they make sense of complicated, high dimensional and nonlinear datasets where the model has multiple hidden layers.

# 3.4 Model Training and Testing

The model, trained on a custom dataset extracted from high-definition videos, demonstrates remarkable proficiency in analyzing real-time video footage. When presented with footage of a forest fire, it efficiently identifies the dynamic patterns of flames and smoke, offering insights into the severity and spread of the blaze. Additionally, when tasked with analyzing videos depicting serene forest scenes adorned with lush greenery and shrouded in fog, the model adeptly discerns the intricate details of the landscape, from the dense foliage to the ethereal mist. Its ability to discern between the contrasting scenarios of devastation and tranquility showcases its versatility and reliability in interpreting complex visual data, making it an invaluable tool for monitoring and managing natural environments.

# 3.5 Evaluation Metrics

Four frequently employed evaluation metrics were used to compare the performances of the proposed classification model and the compared models: “Accuracy, Precision, Recall, and F1- score.”

1. **Accuracy:** Accuracy measures the overall correctness of the predictions made by the model. It is calculated as the ratio of the number of correct predictions to the total number of predictions.

𝑇𝑃 + 𝑇𝑁

𝐴𝑐𝑐𝑢𝑟𝑎𝑐𝑦 =

𝐹𝑃 + 𝐹𝑁 + 𝑇𝑃 + 𝑇𝑁

1. **Precision:** Precision measures the proportion of correctly predicted positive instances (True Positives) out of all instances predicted as positive (True Positives + False Positives). It indicates how many of the predicted positive instances are positive.

𝑇𝑃

𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 =

𝑇𝑃 + 𝐹𝑃

1. **Recall:** Recall, also known as sensitivity or true positive rate, measures the proportion of correctly predicted positive instances (True Positives) out of all actual positive instances (True Positives + False Negatives). It indicates how many of the actual positive instances are correctly predicted by the model.

𝑇𝑃

𝑅𝑒𝑐𝑎𝑙𝑙 =

𝑇𝑃 + 𝐹𝑁

1. **F1-score:** F1-score is the harmonic mean of Precision and Recall. It provides a single metric that balances both Precision and Recall.

𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 𝑅𝑒𝑐𝑎𝑙𝑙

𝐹1 − 𝑆𝑐𝑜𝑟𝑒 

𝑃𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛  𝑅𝑒𝑐𝑎𝑙𝑙

In addition to these metrics, the text mentions using accuracy-loss curves for training and test validation to verify the performance of the proposed model. These curves can help visualize how the accuracy of the model changes as the loss function decreases during training, indicating how well the model is learning from the data.

# 3.6 Feasibility Study

## 3.6.1 Economic Feasibility

From an economic standpoint, the development of the forest fire detection model is feasible, considering the potential benefits it can provide to businesses in the tourism and hospitality industry. By accurately classifying the fire and smoke in the images, the model can help businesses make informed decisions to improve customer satisfaction and loyalty, ultimately leading to increased revenue and profitability. The cost of developing and implementing the model is expected to be outweighed by the benefits it brings in terms of improved business performance and customer experience.

## 3.6.2 Operational Feasibility

The operational feasibility of the project is also high, as the forest fire detection model can be seamlessly integrated into existing systems or used as a standalone tool. The model can be deployed on cloud platforms, making it easily accessible and scalable. Once deployed, the model can continuously analyze the fires, smokes in the forest, providing real-time insights to authorities. This operational efficiency can help businesses quickly respond to customer feedback and improve overall customer satisfaction.

## 3.6.3 Technical Feasibility

The technical feasibility of the project is high, given the availability of advanced machine learning libraries and frameworks such as TensorFlow and Keras. These tools provide the necessary functionality to develop and train complex models like LSTM for forest fire detection.

**Chapter 4**

**System Requirements**

**\**

4. SYSTEM REQUIREMENTS

The project involved analyzing the design of the rainfall prediction system to ensure user-friendly interaction. It was crucial to optimize the data processing workflow while keeping the model efficient and accessible. To enhance usability, the machine learning model was implemented in Python, allowing easy integration with Google Colab and other Python-based platforms.

# 4.1 Functional Requirements

**Graphical User Interface (GUI):** A simple, interactive system where users can input weather parameters (temperature, humidity, wind speed, cloud cover) and receive a binary classification output ("Yes" or "No" for rainfall prediction). The model is designed to work through Google Colab, Jupyter Notebook, or Python scripts, ensuring seamless interaction with the user.

4.2 Technologies and Languages used to develop

* **Python:** Python is a versatile, high-level programming language known for its simplicity and vast libraries. It plays a crucial role in data preprocessing, model training, and evaluation.
* **Deep Learning (BiLSTM):** Bidirectional Long Short-Term Memory (LSTM) networks, a type of Recurrent Neural Network (RNN), are used to capture temporal dependencies in weather data, improving the accuracy of rainfall predictions.
* **TensorFlow & Keras:** These deep learning frameworks provide efficient model training and evaluation, optimizing performance on large weather datasets
* **SMOTE (Synthetic Minority Over-sampling Technique):** Used for handling class imbalance in the dataset to improve prediction reliability.
* **Google Colab (T4 GPU Support)**: The model is trained on Google Colab’s T4 GPU, which accelerates deep learning computations.

**4.2.1 Debugger and Emulator**

* **Google Colab:** A cloud-based Python environment used for model training, debugging, and evaluation. It offers GPU acceleration for improved efficiency.
* **Jupyter Notebook**: An interactive Python development environment, useful for code testing and real-time model evaluation**.**

## 4.2.2 Hardware Requirements

* Computer with 1.6 GHz or faster processor
* Minimum 4 GB of RAM or more
* 2.5 GB of available hard-disk

## 4.2.3 Software Requirements

* Operating Systems: Windows 11
* Workspace Editor: Google Colab, Jupyter Notebook
* Backend- Python 3.10, TensorFlow 2.x, Keras

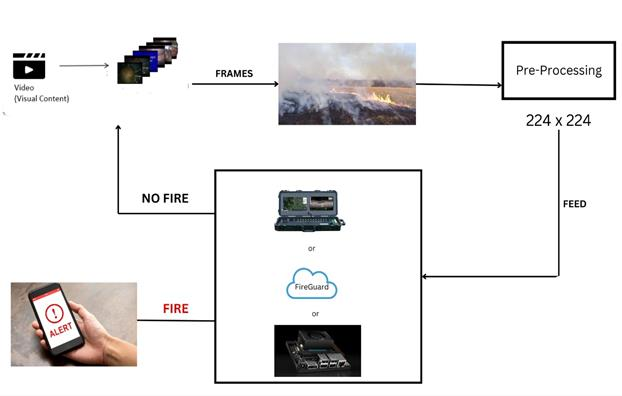
**Chapter 5**

**Design**

# 5. DESIGN

# 5.1 System Design

The proposed system uses a Long Short-Term Memory (LSTM) model to analyze historical weather data for rainfall prediction, as shown in Figure 5.1. The process begins with data preprocessing, where missing values are handled, and features are normalized. The dataset consists of temperature, humidity, wind speed, and cloud cover as input features.



#### Figure 5.1: Work flow of CRNN Model

The LSTM model is trained using balanced data (via SMOTE) and optimized using cross-validation techniques to ensure high accuracy. The trained model is then used to predict whether it will rain tomorrow (Yes/No) based on new or unseen weather data.

External libraries such as NumPy, Pandas, TensorFlow, Keras, and scikit-learn are used for data processing, model training, and evaluation. Google Colab with T4 GPU support is utilized for efficient training and testing of the deep learning model.

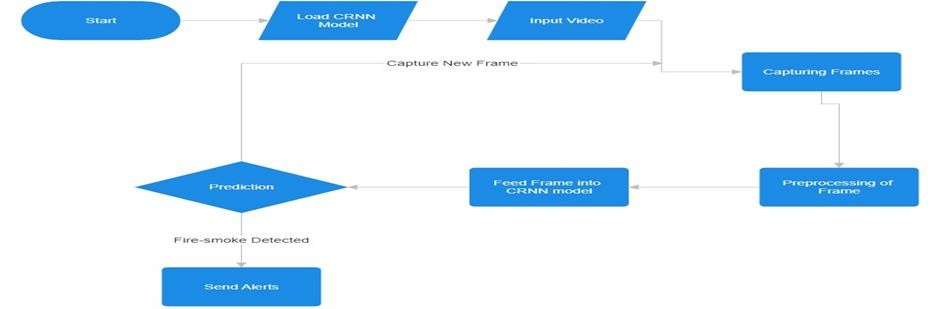
# 5.2 UML Diagrams

UML, or Unified Modeling Language, is a standardized modeling language used in software engineering to visually represent software systems. Its importance lies in providing a common language and notation for software developers, designers, and stakeholders to communicate and understand the structure, behavior, and interactions of complex systems. UML diagrams such as class diagrams, sequence diagrams, and use case diagrams help in conceptualizing, designing, documenting, and communicating software systems, leading to better understanding, collaboration, and more efficient development processes.

Unified Modeling Language (UML) diagrams are a standardized way of visually representing software systems. They provide a way for software developers to communicate system designs, architectures, and processes in a clear and consistent manner. UML diagrams use various graphical elements such as boxes, lines, and arrows to represent different aspects of a system, making it easier for stakeholders to understand complex systems.

One of the key benefits of UML diagrams is that they help in the visualization of the system's architecture and design. By using different types of diagrams such as class diagrams, sequence diagrams, and use case diagrams, developers can create a comprehensive picture of the system, which can be used as a blueprint for implementation.

Another important aspect of UML diagrams is that they help in the communication between different stakeholders involved in the software development process. For example, developers can use UML diagrams to explain their designs to non-technical stakeholders such as project managers or clients, helping them to understand the system requirements and functionalities.



#### Figure 5.2: Flow Diagram

**Chapter 6**

**Implementation**

# 6. IMPLEMENTATION

# 

# 6.1 CRNN Model Algorithm:

ALGORITHM build\_compile\_fit\_CRNN\_model(train\_generator, validation\_generator, epochs):

CREATE model

ADD Conv2D layer with 32 filters, kernel size (3, 3), stride (1, 1), padding 'same', activation 'relu', input shape (224, 224, 3) to model

ADD MaxPooling2D layer with pool size (2, 2), stride (2, 2), padding 'valid' to model

ADD Conv2D layer with 32 filters, kernel size (3, 3), stride (1, 1), padding 'same', activation 'relu' to model

ADD MaxPooling2D layer with pool size (2, 2), stride (2, 2), padding 'valid' to model

ADD TimeDistributed layer with Flatten layer to model

ADD LSTM layer with 64 units, return sequences True to model

ADD Dropout layer with rate 0.2 to model

ADD LSTM layer with 64 units to model

ADD Dropout layer with rate 0.2 to model

ADD Dense layer with 4 units, activation 'softmax' to model

COMPILE model with:

* Optimizer: Adam with learning rate
* Loss: Categorical Crossentropy
* Metrics: Accuracy

FIT model using:

* Training generator: train\_generator
* Validation data: validation\_generator
* Number of epochs: epochs

RETURN trained model

# 6.2 Home Page:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>FireGuard - Video Streaming</title>

<style> body {

font-family: Arial, sans-serif; margin: 0; padding: 0;

background-image:url("static/back1.jpg"); background-repeat:no-repeat; background-position:down; background-size:cover;

} h1 {

text-align: left; margin-bottom: 20px; font-size: 36px; color:crimson; text-shadow:5px 1px black; margin-left:20px;

} form { text-align: center; margin-bottom: 20px;

}

select {

padding: 10px;

font-size: 16px;

border: 1px solid #ccc; border-radius: 4px;

}

input[type="submit"], input[type="button"], .testing-btn { padding: 10px 20px; font-size: 16px;

background-color: #28a745; /\* Green color \*/ color: #fff; border: none; border-radius: 4px; cursor: pointer; margin-right: 10px;

margin-bottom: 10px; /\* Added margin bottom for spacing \*/

}

input[type="submit"]:hover, input[type="button"]:hover, .testing-btn:hover {

background-color: #218838; /\* Darker green on hover \*/

} img { display: block; margin: 20px auto; max-width: 100%; height: auto; }

.btn-container { text-align: right; /\* Center align buttons \*/ margin-top: 20px;

}

.btn-container2 {

text-align: center; /\* Center align buttons \*/ margin-top: 20px;

}

.btn-container button { padding: 10px 20px; font-size: 16px; background-color: #007bff; color: #fff; border: none; border-radius: 4px; cursor: pointer; margin-right: 10px;

}

btn-container button:hover { background-color: #0056b3;

}

/\* Alert box for fire detection \*/

.alert { position: fixed; top: 50%; left: 50%;

transform: translate(-50%, -50%); background-color: red; color: white; padding: 20px; border-radius: 8px; font-size: 24px;

display: none;

z-index: 9999; /\* Ensure the alert is on top of other elements \*/

}

.testing-btn{ button-align:center;

}

</style>

</head>

<body>

<h1>FireGuard - Video Streaming</h1>

<div class="btn-container">

<button>Home</button>

<button>About</button>

<button>Services</button>

</div>

<form id="videoForm" action="{{ url\_for('process') }}" method="post">

<div class="btn-container2">

<select name="video\_choice">

<option value="1">Fire</option>

<option value="2">Green</option>

<option value="3">Fog</option>

<option value="4">Webcam</option>

</select>

<button class="testing-btn" type="submit">Start</button>

<button class="testing-btn" onclick="stopVideo()">Stop</button>

</div>

</form>

<br>

<img id="videoFeed" src="" alt="Video Feed">

<!-- Alert box for fire detection -->

<div id="fireAlert" class="alert">Fire Detected!</div>

<script>

function startVideo(videoChoice) {

var img = document.getElementById("videoFeed"); img.src = "/video\_feed?video\_choice=" + videoChoice;

// Start checking for fire-smoke detection checkFireSmoke();

}

function checkFireSmoke() { // Fetch the video feed

fetch("/video\_feed?video\_choice=" + document.getElementById("video\_choice").value) .then(response => response.text())

.then(data => {

if (data.trim() === "fire-smoke") { fireDetected();

}

// Continue checking for fire-smoke detection recursively checkFireSmoke();

})

.catch(error => {

console.error('Error:', error);

});

}

// Function to show alert and play sound for fire detection function fireDetected() { console.log("Fire detected!");

var alertBox = document.getElementById("fireAlert"); alertBox.style.display = "block";

// Play a loud sound

var audio = new Audio('static/sound.mp3'); audio.play();

}

// Function to show alert and play sound for fire detection function fireDetected() { console.log("Fire detected!");

var alertBox = document.getElementById("fireAlert"); alertBox.style.display = "block";

// Play a loud sound

var audio = new Audio('static/sound.mp3'); audio.play();

}

</script>

</body>

</html>

**Chapter 7**

**Results**

# 7. RESULTS

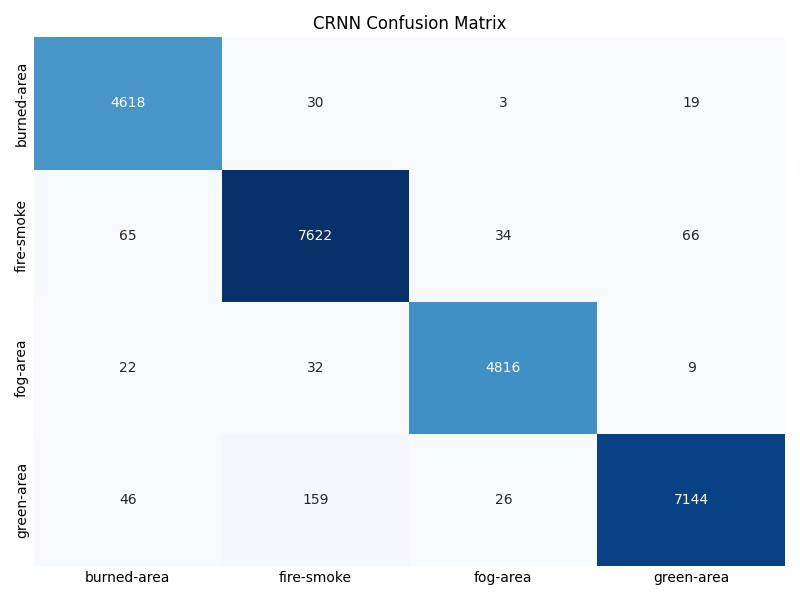
The model is compared with the existing CNN model and observed that the proposed model achieves more accuracy, F1-Score and R1 values as you can see in table 7.1. The proposed method achieves an accuracy of 97.93%, the F1-score of 97.93%, the precision of

97.94%, and the recall of 97.93%.

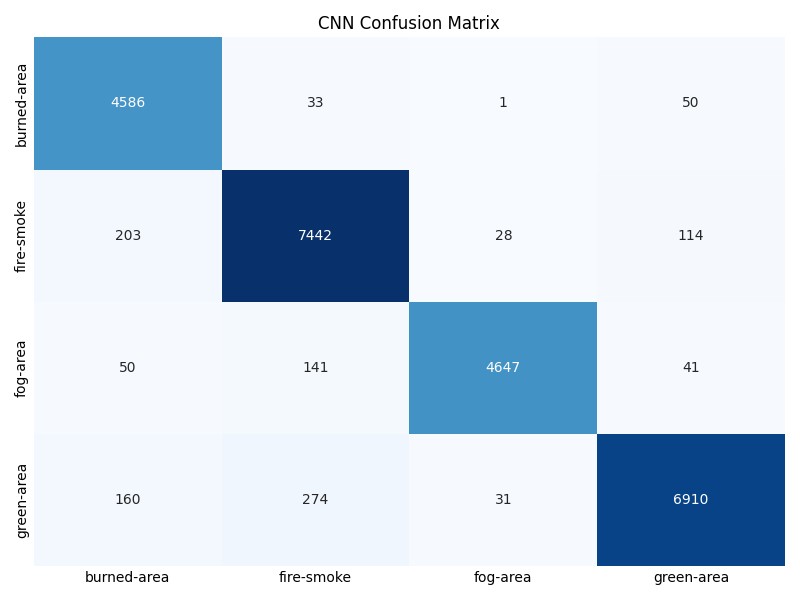
#### Table 7.1 Results CNN vs CRNN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| CNN | 95.44% | 95.54% | 95.44% | 95.45% |
| CRNN | 97.93% | 97.94% | 97.93% | 97.93% |

In the confusion matrix of Fig 7.1, the proposed method shows a better ability to classify all classes and in Fig 7.2 you can find the existing method confusion matrix.

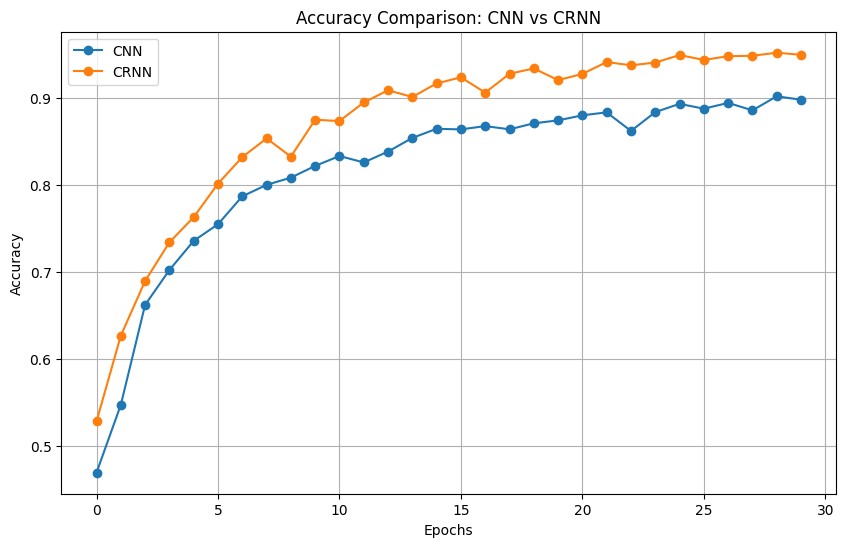


#### Figure 7.1 Confusion matrix of Proposed method



#### Fig 7.2 Confusion matrix of Existing method

Fig 7.3 shows that the training of the CRNN method has happened more accurately compared to the CNN method. The CRNN training accuracy curve is higher than CNN in all the epochs.



#### Fig 7.3 Training Accuracy Comparison between CNN and CRNN

**Chapter 8**

**Social Impact**

# 8. SOCIAL IMPACT

In the realm of modern commercial landscapes, leveraging advanced technology like the CRNN model for fire detection from UAV Images of wildfire holds immense importance. By analyzing these images, organizations can identify potential fire outbreaks swiftly, aiding in timely interventions and minimizing damage. This proactive approach not only enhances safety measures but also mitigates risks associated with wildfires, safeguarding both lives and properties.

The implementation of a cross-validated CRNN model for fire detection offers multifaceted benefits. Firstly, it enables organizations to detect fires in their early stages, allowing for prompt response and containment efforts. Secondly, this technology aids in optimizing resource allocation, ensuring that firefighting resources are deployed efficiently and effectively. Lastly, it contributes to enhancing public safety by providing reliable tools for wildfire monitoring and management.

Through the utilization of CRNN models for fire detection, the commercial sector can significantly impact societal well-being. By swiftly identifying and responding to wildfire threats, organizations contribute to the preservation of natural landscapes and the protection of communities residing in high-risk areas. Moreover, by investing in advanced technologies for fire detection, businesses demonstrate a commitment to corporate social responsibility and sustainable practices, fostering trust and goodwill among stakeholders.

* **Enhanced Emergency Response:** Utilizing CRNN models for wildfire detection enables proactive measures akin to improving customer experiences in the tourism industry. By analyzing UAV Images, authorities gain critical insights, facilitating swift responses to potential fire outbreaks and minimizing risks to lives and properties.

* **Transparency and Accountability in Disaster Management:** The adoption of CRNN models promotes transparency and accountability in wildfire management, akin to the tourism sector's emphasis on customer satisfaction. Through comprehensive analysis of imagery datasets, authorities can identify fire-prone areas and allocate resources efficiently, fostering community trust and safety.

* **Economic Resilience and Sustainability:** The economic impact of CRNN-based wildfire detection parallels the tourism industry's focus on revenue generation and sustainability. Early detection and containment of wildfires mitigate financial losses, preserve natural resources, and promote long-term economic prosperity for businesses and communities alike.

* **Proactive Risk Mitigation:** CRNN models empower early detection of wildfire outbreaks, akin to addressing customer feedback to enhance experiences. Analyzing UAV Images allows for swift responses, minimizing risks to lives and properties.

* **Efficient Resource Allocation:** Similar to optimizing services based on customer preferences, CRNN analysis aids in allocating firefighting resources effectively. Authorities pinpoint fire-prone areas, ensuring resources are deployed where needed most, enhancing overall effectiveness.

In summary, utilizing CRNN models for wildfire detection from UAV Images offers significant social impacts. Comparable to customer review analysis, it enhances emergency responses, fosters community engagement, and informs decision-making. This proactive approach empowers authorities to safeguard lives, properties, and ecosystems, ensuring effective wildfire management and community resilience.

**Chapter 9**

**Conclusion & Future Work**

# 9. CONCLUSION & FUTURE WORK

In conclusion, the FireGuard system, built upon a Convolutional Recurrent Neural Network (CRNN) architecture, represents a significant advancement in forest fire detection technology. Through the integration of deep learning techniques and real-time video analysis, FireGuard offers a proactive solution to the escalating threat of wildfires, particularly in forested and remote areas.

The system's ability to analyze live video streams from aerial vehicles or webcams enables timely detection of fire and burned areas, facilitating rapid response and mitigation efforts. By leveraging CRNNs, FireGuard captures both spatial and temporal features within video data, enhancing detection accuracy and reliability.

The experimental results validate the effectiveness of the CRNN model, demonstrating high accuracy rates and performance metrics compared to traditional CNN-based approaches. Moreover, the system's lightweight architecture and compatibility with edge devices ensure scalability and adaptability for diverse deployment scenarios.

FireGuard presents a robust and efficient solution to the pressing challenge of forest fire detection, offering significant benefits in terms of environmental conservation, public safety, and resource management. With further advancements and integration into existing wildfire management systems, FireGuard has the potential to revolutionize how wildfires are detected, monitored, and mitigated, contributing to a safer and more sustainable future.

* **Enhancing Model Architecture:** Future work could focus on refining the CRNN model architecture by adding additional LSTM layers and Conv2D layers. This could potentially improve the model's accuracy in detecting wildfires from UAV Images by capturing more intricate patterns and features in the data.
* **Exploring Advanced Techniques:** Further research could investigate the integration of advanced techniques such as attention mechanisms or ensemble learning to enhance the model's performance. These techniques could improve the model's ability to discern relevant features in the imagery datasets, leading to more accurate wildfire detection.
* **Mobile Application Development:** Developing a mobile application could extend the project's utility by providing real-time wildfire alerts to users. By integrating the trained CRNN model into the app, individuals in wildfire-prone areas could receive timely notifications, enabling them to take necessary precautions and evacuate if needed.
* **Geospatial Integration:** Future work could explore integrating geospatial data into the wildfire detection system. By incorporating information such as weather conditions, terrain elevation, and vegetation density, the model could generate more precise wildfire risk assessments, enhancing the effectiveness of early warning systems.
* **Crowdsourced Data Collection:** Leveraging crowdsourced data collection methods could enrich the project's dataset and improve model generalization. Encouraging users to contribute UAV Images of wildfire incidents from various locations could enable the model to learn from a broader range of scenarios, enhancing its robustness and accuracy in detecting wildfires.

# 

# BIBLIOGRAPHY

1. Y. Chen et al., “UAV image-based forest fire detection approach using convolutional neural network,” in Proc. 14th IEEE Conf. Ind. Electron. Appl., 2019, pp. 2118–2123.
2. A. Viseras, M. Meissner, and J. Marchal, “Wildfire front monitoring with multiple UAVs using deep Q-learning,” IEEE Access, to be published, doi: 10.1109/ACCESS.2021.3055651.
3. P. Ma, F. Yu, C. Zhou, and M. Jiang, “An integrated smoke detection method based on convolutional neural network and image processing,” in Proc. IEEE 8th Int. Conf. Comput. Sci. Netw. Technol., 2020, pp. 32–36.
4. T. Gupta, H. Liu, and B. Bhanu, “Early wildfire smoke detection in videos,” in Proc. 25th Int. Conf. Pattern Recognit., 2021, pp. 8523–8530.
5. W. Li, S. Xiaobo, C. Jun, and L. Ying, “Research on forest fire image recognition algorithm based on color feature statistics,” in Proc. 6th Int. Conf. Intell. Comput. Signal Process., 2021, pp. 346–349.
6. W. Thomson, N. Bhowmik, and T. P. Breckon, “Efficient and compact convolutional neural network architectures for non-temporal real-time fire detection,” in Proc. 19th IEEE Int. Conf. Mach. Learn. Appl., 2020, pp. 136–141.
7. X. Li, Z. Chen, Q. J. Wu, and C. Liu, “3D parallel fully convolutional networks for realtime video wildfire smoke detection,” IEEE Trans. Circuits Syst. Video Technol., vol. 30, no. 1, pp. 89–103, Jan. 2020.
8. Y. Zhang and Y. Hu, “Video smoke detection based on convolution neural network,” in Proc. Int. Comput., Signals Syst. Conf., 2018, pp. 123–127.
9. Brazilian Institute of Environment and Renewable Natural Resources, “Burn monitoring in satellite images,” 2017. [Online]. Available: http://www.ibama.gov.br/consultas/incendiosflorestais/consultasmonitoramento-de-queimadas/monitoramento-de-focos-de-queimadasemimagens-de-satelites.
10. J. Shi, W. Wang, Y. Gao, and N. Yu, “Optimal placement and intelligent smoke detection algorithm for wildfire-monitoring cameras,” IEEE Access, vol. 8, pp. 72326–72339, 2020.
11. X. Shi, N. Lu, and Z. Cui, “Smoke detection based on dark channel and convolutional neural networks,” in Proc. IEEE 5th Int. Conf. Big Data Inf. Analytics, 2019, pp. 23–28. [12] H. Ullah, K. Muhammad, M. Irfan, B. Z. Muhammad, and S. Anwar, “Light-DehazeNet: A novel lightweight CNN architecture for single image dehazing,” IEEE Trans. Image Process., vol. 30, pp. 8968–8982, Oct. 2021.

[13] O. M. Bushnaq, A. Chaaban, and T. Y. Al-Naffouri, “The role of UAV-IoT networks in future wildfire detection,” IEEE Internet Things J., vol. 8, no. 23, pp. 16984–16999, Dec. 2021. [14] S. Dutta and S. Ghosh, “Forest fire detection using combined architecture of separable convolution and image processing,” in Proc. 1st Int. Conf. Artif. Intell. Data Analytics, 2021, pp. 36–41.

1. K. Shrestha, R. Dubey, A. Singandhupe, S. Louis, and H. La, “Multi objective UAV network deployment for dynamic fire coverage,” in Proc. IEEE Congr. Evol. Comput., 2021, pp. 1280–1287.

1. J. Jeyavel, A. A. Prasad, K. M. Shelke, P. D. Sargade, and U. V. Thoke, “Survey on fire fighting techniques using unmanned aerial vehicles,” in Proc. Int. Conf. Adv. Comput. Innov. Technol. Eng., 2021, pp. 239–241.
2. S. Khan, K. Muhammad, S. Mumtaz, S. W. Baik, and V. H. C. de Albuquerque, “Energyefficient deep CNN for smoke detection in foggy IoT environment,” IEEE Internet Things J., vol. 6, no. 6, pp. 9237–9245, Dec. 2019.