Demonstrating Various Pre-Processing Steps for Multi-Modal Flood Data Generated in Several Coastal Regions, India

[1,*]Suresh Kapare, [2]Dr. V. Maria Anu

[1] Research Scholar, Sathyabama Institute of Science and Technology, Chennai.

[2] Associate Professor, VIT, Chennai.

Email-ID: [1,*]kaparesuresh8@gmail.com, [1,*]sureshkapare2012@gmail.com , [2]mariaanu.v@vit.ac.in

Abstract

Floods are among the most distressing natural disasters, adversely affecting coastal regions, especially in countries such as India and China. India has a dense population, and seasonal cloudbursts strengthen their impacts. The government of India and other private organizations project various schemes and strategies to alert the people living in coastal areas and protect them from floods and related disasters. Several earlier research methods were used to forecast flooding; however, their accuracy is low, and their false positive rate is high, which dissatisfies the public and makes them lethargic in critical scenarios. The accuracy of flood predictions is low because data analytics indicates that real-time data from coastal areas is unreliable. Also, they used only one data source, such as satellite images or sensor readings, which is not enough to predict real-world scenarios. Hence, this work has aimed to predict the real-time flood data generated using IoT-sensor networks, satellite images, drone footage, and hydrological time-series data, and improve the data quality by applying various robust preprocessing methods that can ensure high quality and actionable insights. This work systematically presents a set of preprocessing methods tailored to multimodal flood data collected from several hazardous coastal areas in India. The dataset comprises spatial, temporal, and spatio-temporal data, and its quality is improved by applying normalization, segmentation, stationarity verification, time-based feature extraction, outlier detection, resizing, color conversion, image enhancement, denoising, and other techniques. It accurately prepares data for subsequent flood prediction and forecasting, helping people protect themselves by making immediate decisions. It qualitatively and quantitatively evaluates the data to assess the efficiency in increasing data quality and reliability. It also makes computational models more accurate for flood prediction, risk mapping, and emergency response preparation. The experiment is conducted using a comprehensive multimodal dataset, and the results are validated by comparing prediction accuracy before and after preprocessing. After applying preprocessing methods such as feature scaling, KNN imputation, and IQR-based outlier removal, training and validation accuracies increased to approximately 97% and 93%, respectively. It also concludes that preprocessing is a crucial step that must be performed before any data analytics.

Keywords: ***Flood Data Preprocessing, Multimodal Data, Satellite Imagery, Time Series Normalization, Image Augmentation, Noise Reduction, CLAHE, India's Coastal Regions, Disaster Management, Data Quality Enhancement.***

1. Introduction

Floods are among the most frequent and devastating natural disasters worldwide, posing risks to hundreds of lives and threatening infrastructure and ecosystems [1]. The Indian coastal regions are more vulnerable due to the superimposition of several factors, such as high population density, poor drainage infrastructure, rapid urbanization, and exposure to monsoonal and cyclonic weather systems [2]. Coastal areas contain numerous geomorphological systems; therefore, flood management is complex and difficult [3, 4]. Furthermore, about 23 percent of the world's population lives within 100 km of the coastline and at an altitude of 100 m [5]. The sample flood prediction structure is shown in figure-1.

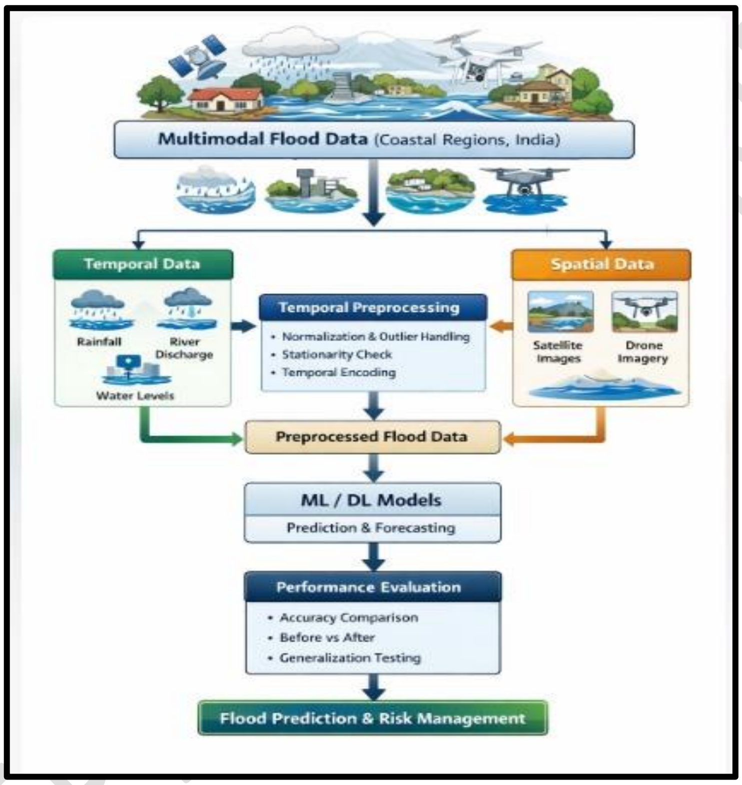


Figure-1 Sample Flood Prediction

Effective disaster risk reduction requires more accurate, timely flood prediction, monitoring, and response, which is critical in these areas. Recent years have seen an acceleration in the availability of data from multiple sources, such as satellite imagery, drone-based imagery, and hydrological sensor readings, for flood and rainfall time series. Nevertheless, preprocessing this heterogeneous dataset is more important, as these data sources are much more heterogeneous (format, scale, resolution, noise characteristics) than other sources.

All downstream flood analytics tasks, such as segmentation, risk mapping, forecasting, and emergency response optimization, are built on the upstream preprocessing tasks. Raw data contain various errors, such as missing values, inconsistencies, noise, and misaligned formats, but with proper preprocessing, high-quality, reliable results can be obtained. Normalization methods help with temporal learning in time series and with image-centric operations such as resizing, contrast enhancement, and noise reduction, which help detect more spatial patterns in ML and DL models.

However, such an important preprocessing framework has not yet been explored in the existing literature, particularly in a unified, comprehensive manner for multimodal flood data in highly diverse and complex settings. Recent studies either focus on a specific data or model type, excluding preprocessing as a routine step. A major limitation for practitioners was the inability to leverage different data models to enhance flood analytics performance.

This research provides a detailed, structured explanation of the preprocessing method for multimodal flood data from various coastal regions in India. It also classified the data into two modalities, such as temporal and spatial, and applied a range of preprocessing methods for each type. It uses windowing, normalization, outlier handling, temporal encoding, transformation, and a stationary approach for temporal data, such as river discharge, rainfall, and water levels. For spatial data such as drone or satellite images, it implements and uses grayscale conversion, Gaussian filtering, resizing, contrast enhancement using CLAHE, data augmentation, pixel normalization, and centering with padding [6]. Each of these methods is presented with a detailed explanation, real-world flood data samples, a mathematical basis, and insights from implementation.

By systematically evaluating and documenting these preprocessing steps, one can help to provide an adaptable and reproducible pipeline for flood data preparation. The proposed framework not only improves the interpretability and data consistency of ML models but also enhances their robustness and overall performance, which are crucial for applications in flood prediction, decision support systems, and mapping. Moreover, this research contributes to intelligent flood management and more resilient strategies for predicting unfortunate events in coastal regions of India.

The goal is to design a comprehensive preprocessing framework for multimodal flood data collected across multiple coastal regions of India to improve data quality and enhance flood prediction accuracy. The work aims to bridge the gap between raw heterogeneous data (time-series sensor readings, satellite images, and drone imagery) and machine learning/deep learning flood-forecasting models. The preliminary objectives include classifying flood data into spatial and temporal modalities. This preprocessing step applies suitable techniques to each data type (such as normalization, outlier removal, segmentation, stationarity checks, image resizing and enhancement, and augmentation) and defines the processing pipeline to enhance consistency, reproducibility, and scalability. This systematic evaluation of the study aims to determine whether robust preprocessing significantly improves model performance, reliability, and generalization.

The motivation for this research stems from the limitations of existing flood prediction systems, which often rely on single-modal data or apply minimal preprocessing, resulting in lower accuracy and higher false alarms. Given the high vulnerability of Indian coastal regions to floods, there is a critical need for structured, domain-specific preprocessing models that can handle noisy, inconsistent, and heterogeneous real-time data. The scope of the proposed work covers the development of a suitable preprocessing pipeline to combine temporal and image-based flood datasets and the analysis of its impact on ML/DL models, with applications to real-world disaster management systems. These are the models that are transferred to other geographic regions and disaster types, supporting scalable, data-driven mitigation of urgency and response risks. The following section discusses prior research, the proposed model's workflow and results, and concludes by highlighting the efficiency of the current work and future directions.

The key objective of this work is to design and demonstrate the preprocessing methods for multimodal flood data obtained from many coastal areas in India. This will improve the data quality, increase prediction and forecast accuracy. It is obtained by providing solutions to the additional objectives:

1. Classifies the multi-modal flood data into spatial and temporal.
2. Apply an appropriate preprocessing method for different data modalities to increase the data quality, consistency, and readiness for further data processing.
3. Standardize the preprocessing practices for flood data obtained from various geographic and environmental constraints in coastal areas, India.
4. Demonstrate the preprocessing steps with mathematical expressions and examples to emphasize the essentiality of robust flood data analytics.

To provide better solutions, this paper makes the following essential contributions.

- ➢ We present a detailed preprocessing framework that integrates spatial and temporal preprocessing methods for multimodal flood data.
- ➢ It demonstrates advanced preprocessing methods for handling time-series data, with a focus on temporal dependencies.
- ➢ It demonstrates advanced preprocessing methods for handling spatial image data, with a focus on spatial dependencies.
- ➢ Each preprocessing method is explained with supporting mathematical expressions and real-world examples using flood data to promote clarity and reproducibility.
- ➢ It highlights the practical relevance of preprocessing methods for flood accident-prone regions, using real flood data from multiple coastal areas in India.
- ➢ It provides a standard, scalable preprocessing framework that integrates multiple ML and DL pipelines to support effective flood forecasting and risk analysis.

2. Literature Survey

This section explains various preprocessing methods for multimodal flood data, particularly those collected in Indian coastal regions. It includes satellite images, time-series data, and other data that provide comprehensive situational awareness. Research has also recognized that preprocessing, such as handling heterogeneous data, is essential for extracting reliable insights and training effective models.

IoT-enabled approaches have been frequently used to analyse and predict floods. Dai et al. (2021) presented an advanced Bayesian model combination (BMC-EL) for predicting flood occurrence by analysing data, which improves the efficiency of short-term depth forecasting using IoT sensor data in Macao and China. Similarly, Bhardwaj et al. (2022) introduced an IoT-driven approach for monitoring water quality and managing devices. This research has compared the performance of the proposed model with several other models to evaluate its effectiveness. The results show that the proposed model has effectively monitored water levels.

Similarly, Tiwari et al. (2021) have presented a LightGBM model for detecting IoT attacks in the marine environment. The experimental results showed that the proposed model outperformed by reducing computational cost and achieving a detection accuracy of 98.52%. While this research has shown many benefits in detecting and monitoring attacks in the marine environment, it still faces challenges in handling missing, noisy, and inconsistent values. To overcome these challenges, the preprocessing steps have been discussed, which make the models robust for real-time deployment.

In flood prediction, ML-based methods enhance the prediction accuracy. Motta et al. (2021) integrated an ML classifier with GIS methods for flood prediction in an urban region, achieving 96% accuracy and an MCC of approximately 0.77. Mangukiya and Sharma (2022) used the Random Forest (RF) algorithm to classify flood risk zones in the lower Narmada basin in India and to detect factors such as rainfall, land, and elevation. Ogbune et al. (2024) evaluated the ML classifier in Nigeria. The results showed that the naïve Bayes algorithm outperformed other ML models, such as NN, LR, SVM, and RF, in terms of timely flood prediction. Kumar et al. (2023) have studied state-of-the-art DL approaches for flood prediction but highlight the complexity of model adaptability and stability. Patel and Yadav (2023) proposed a hybrid model for Dharoi Dam flood prediction by integrating an ensemble hydrological model. As a result, the fundamental pre-processing techniques are applied to the model to enhance its performance. The aforementioned research underscores the strength of ML models in flood prediction, but they often use less-processed or raw data, which affects their accuracy. In a few studies, heterogeneous flood datasets are standardized by integrating preprocessing pipelines with flood prediction models.

The study on water quality monitoring also requires a pre-processing approach for environmental datasets. Gambin et al. (2021) investigated various technologies and approaches, including IoT, Remote Sensing (RS), cloud computing, big data, and AI, for assessing water quality, thereby helping ensure sustainability in the marine environment. In particular, it focused on various DL-based models for estimating water quality. The results demonstrate that the marine environment still has some issues. To address these issues, Reinforcement Learning, TL, edge computing, and decision-making approaches have been implemented and investigated. Tselemponis et al. (2023) investigated various ML models, including DT, Decision Jungle, and Boosting Decision Tree (BDT), to assess their overall effectiveness in predicting water quality in Thrace and Eastern Macedonia. The result demonstrates that the proposed ML-based model outperforms the baseline in forecasting, achieving an accuracy of 99%. Essamlali et al. (2024) presented an investigation into integrating IoT wireless technologies such as Zigbee, LPWAN, Wi-Fi, and RFID with an advanced ML model for water quality monitoring. Zhu et al. (2022) investigated ML models across environments such as sewage, groundwater, and seawater to validate their performance.

Drogkoula et al. (2023) have highlighted issues related to reproducibility and transparency in ML-based water resource management. Chansi et al. have studied the AI, ML, and IoT-based sensor for pollution detection in marine environments. Alprol and Khairy (2025) have evaluated ML-based applications for water pollution control and treatment and addressed complexities, such as model explainability and handling large datasets. Yang et al. (2025)

proposed an ML model that combines corrosion sensor data to predict corrosion risk levels in marine areas. The proposed model integrates K-means clustering and ML classifiers such as RF and SVM, achieving effective performance in accuracy, precision, recall, and F1-score. It also underscores the need for sophisticated preprocessing to handle the marine dataset.

Karthik et al. (2025) proposed an advanced deep learning framework for flood prediction in the Chennai region using long-term climatological data collected between 2000 and 2023. Their approach integrates an Extended Elman Spiking Neural Network (ExESNN) with a Chaotic Artificial Hummingbird Optimizer (Ch-AHO) for robust parameter tuning. A MaxAbsScaler-based preprocessing method is employed to handle missing and inconsistent data. The model demonstrated superior predictive performance across multiple hydrological evaluation metrics, including R^2, NSE, KGE, RMSE, and MAE, highlighting the effectiveness of spiking neural architectures in capturing complex rainfall–flood relationships under severe meteorological conditions.

Babu et al. (2024) introduced an integrated early flood prediction system combining Sentinel-2 satellite imagery, Vehicular Ad Hoc Networks (VANET), Multi-Agent Reinforcement Learning (MARL), and a deep recurrent neural network. Satellite data provided large-scale spatial observations, while VANET-MARL enabled real-time, distributed ground-level information sharing and adaptive decision-making. The deep RNN effectively learned spatiotemporal flood patterns, resulting in improved accuracy, precision, recall, and prediction lead time. The proposed system achieved 94.8% prediction accuracy, demonstrating strong adaptability and resilience under dynamically changing flood conditions.

Sundarapandi et al. (2024) presented a novel flood prediction model that combines a Lightweight Dense Network with a Tree-Structured Simple Recurrent Unit (LDTSRU). The dense network converts meteorological variables such as rainfall, humidity, and wind speed into grayscale representations to capture hidden inter-variable patterns, while the TS-SRU models temporal dependencies efficiently. On a public flood dataset, LDTSRU outperformed state-of-the-art methods in accuracy, precision, and recall while significantly reducing training time. This work emphasizes the importance of lightweight architectures for fast and accurate flood forecasting.

Qiu et al. (2026) investigated flood prediction across the Lower Mekong River Basin using an interpretable Long Short-Term Memory (LSTM) model combined with Shapley Additive exPlanations (SHAP) and Universal Multifractal (UM) analysis. Their findings revealed that soil-related and vegetation-related variables played a dominant role in earlier years, while hydrometeorological variables became more influential after 2017. The UM analysis further demonstrated that hydrometeorological factors exert stronger control over extreme discharge at smaller temporal scales, providing valuable insights into scale-independent flood behavior.

ShravanKumar et al. (2026) presented a comprehensive review of remote sensing and artificial intelligence applications in flood prediction. Their study highlighted the growing importance of integrating SAR-based microwave sensing, LiDAR, and multispectral imaging with advanced AI models such as deep neural networks and ensemble learning techniques.

While these integrated approaches have significantly improved the accuracy of real-time flood mapping and forecasting, challenges remain in data integration, computational costs, and access to high-resolution satellite imagery. The authors emphasized future directions involving IoT-based monitoring, digital twins, and quantum sensing technologies for resilient flood management systems.

limitations and issues of the existing models

Although many flood prediction and forecasting methods offer advantages, limitations persist in the current literature, especially for multi-modal flood data analysis. Most earlier works have focused on either image or time-series data separately, and there is no integration across multimodal data. It limits the deployment of the global flood analytics system. Flood prediction models are not generic and have involved a very few preprocessing methods. Most preprocessing techniques used in earlier works are basic, such as resizing and scaling. Preprocessing methods should be applied correctly based on image resolution and seasonal variations. They frequently used idealized datasets and failed to address complexities in real-world flood data. Given India's high susceptibility to floods, several studies have focused on preprocessing tasks, issues, and challenges specific to the aforementioned geography and its associated data behavior. And they lack reproducibility and transparency. Hence, this work aims to develop a systematic and practical method for preprocessing multi-modal flood data. It aims to bridge the research gap between raw multimodal data generated from coastal regions and subsequent data analytics for flood forecasting, risk mapping, and decision-making.

3. Proposed Methodology

In this section, an overall workflow of the proposed preprocessing framework for processing the input flood data. The main objective of this study is to design a more effective and efficient model for analysing input flood data collected from various sources, including drones, IoT sensors, satellite imagery, and time-series data. To preprocess these images, a KNN model is used for missing value imputation, IQR for outlier removal, and min-max scaling for data normalization. Using the results obtained from these preprocessing techniques, the final output is predicted with high accuracy. The following subsection explains each step of the proposed approaches in detail.

Preprocessing in multimodal data:

Preprocessing multimodal data is essential for preparing raw, heterogeneous data types, such as time-series and image data, for effective machine learning or deep learning models. This technique transforms unstructured, noisy, and incomplete data into a clean, consistent, and analyzable form, ensuring higher model accuracy, generalizability, and performance. The crucial divisions involved in preprocessing time sequence data.

(A) Time-series Data Pre-processing

The major techniques, such as cleaning, transforming, and preparing temporal data, are utilized for preprocessing time-series data for analysis or modeling. These methods also include handling missing values, eliminating outliers, smoothing noisy data, resampling to consistent time intervals, generating lag features or rolling statistics, and normalizing or scaling values. This model primarily focuses on enhancing data quality, ensuring temporal consistency, and producing a dataset relevant for anomaly detection, time-series forecasting, and more.

(i) Missing value handling in air pollution data:

Missing value handling fills in records affected by sensor issues or data logging problems using techniques such as statistical imputation, linear interpolation, and forward and backward filling. In air pollution datasets, missing values frequently occur due to sensor failures, maintenance downtime, or transmission errors. Addressing these missing values is essential to maintain the accuracy and reliability of pollution prediction models, environmental assessments, and policy-making. Many methods are available to address such missing data, depending on whether the missingness is random or systematic. Usual techniques involve mean or median imputation, linear or spline interpolation, forward or backward filling, regression or machine learning-based input methods, and K-nearest neighbors (KNN) imputation, each suitable for different datasets and analytical needs.

1. For mean imputation:

The missing entries m and pollutant X with n values are denoted as equation (1):

$$\bar{X}=\frac{1}{n-m}\sum_{i=1}^{n-m} X_i \quad (1)$$

The individual missing value X_j, is replaced as (2):

$$X_j=\bar{X} \qquad (2)$$

2. Linear interpolation:

The missing value X_t, between X_{t-1}andX_{t+1} is denoted as (3) and (4):

$$X_t = X_{t-1} + \frac{(X_{t+1}-X_{t-1})}{2} \qquad (3)$$

Also

$$X_t = X_n + \frac{(t-t_1)}{(t_2-t_1)}(X_{t_2} - X_n) \quad (4)$$

Where the timestamps before and after t are t1 and t2.

3. K- nearest neighbours(KNN) imputation:

The missing value at time t and find the k denotes similar time windows (based on other variables or past pollution patterns) and take the mean, which is defined as (5):

$$X_t=\frac{1}{k}\sum_{i=1}^{k} X_{t_i} \quad (5)$$

(ii) Noise reduction in air pollution data:

Noise reduction is crucial for reducing fluctuations and extra data, using smoothing methods such as wavelet transforms, exponential smoothing, and moving averages. Noise in air pollution data refers to unwanted fluctuations or irregularities in the recorded values that do not reflect actual environmental changes. It often arises from sensor inaccuracies or calibration issues, ecological disturbances such as electrical interference or weather conditions, and sudden spikes caused by temporary local events, such as a passing vehicle. Such noise can obtain meaningful trends and mislead analysis or predictions, particularly in time-series air quality datasets. Reducing noise is important because it improves data quality and reliability, enhances signal clarity for modeling, reduces false alerts in pollution monitoring systems, and supports better policy-making and health-related decisions.

1. Moving average filter:

This widely used method that smooths short-term fluctuations is denoted as (6):

$$\widehat{X_t} = \frac{1}{2k+1}\sum_{i=-k}^{k} X_{t+i} \quad (6)$$

Where the smoothed value at time t is$\widehat{X_t}$, the raw value at time t is X_t, the window size is denoted as k.

2. Exponential moving average (EMA)

$$\hat{X}_t = \alpha X_t + (1-\alpha)\hat{X}_{t-1} \quad (7)$$

Where: the smoothing factor is denoted as $\alpha \in (0,1)$

3. Gaussian smoothing:

For smoothing,

$$\hat{X}_t = \sum_{i=-k}^{k} w_i \cdot X_{t+i}, \quad where\ \ w_i = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{t^2}{2\sigma^2}} \qquad (8)$$

Where, $\hat{X}_t$ denoted as the smoothed value at time/index t, X_{t+i} denotes the input data value at position t+I, w_i denotes Gaussian weight, I denotes Offset index within the smoothing window, K denotes Half-window size , σ denotes Standard deviation, *e* denotes base of the natural logarithm, and π is a constant (≈ 3.14159).

(iii) <u>Resampling air pollution data:</u>

Resampling converts time intervals (for instance, from irregular to defined intervals such as hourly, daily, or weekly) to ensure temporal consistency, which is often required for temporal models. Resampling is the process of altering the frequency of time-series data by upsampling (e.g., from hourly to minutely) or downsampling (e.g., from minute-wise to hourly or daily). In air pollution data analysis, it is imperative to ensure consistent time intervals when combining datasets from multiple sensors, reduce data granularity for storage and long-term trend analysis, and prepare the data for machine learning models that require fixed time steps. Hence, downsampling smooths out high-frequency noise, making the data more accurate for analysis and visualization.

1. Down-sampling (Aggregation):

The pollutant readings X(t) are taken at a minute interval, and hourly averages are denoted as (9):

$$X_h^{hour} = \frac{1}{n}\sum_{i=1}^{n} X_{t_i}, \qquad where\ t_i \in [h{:}\,00, h{:}\,59] \qquad (9)$$

Where the average pollution value for h hours is X_h^{hour}The number of observations in that hour is n.

2. Up-sampling (interpolation):

If the data is available at hourly intervals, then minute values are used in linear interpolation, expressed as (10):

$$X(t) = X(t_1) + \frac{(t-t_1)}{(t_2-t_1)} \cdot [X(t_2)\text{-}X(t_1)] \qquad (10)$$

Where t1 and t2 are the known times before and after the missing minute t.

(iv) <u>Detrending in air pollution data:</u>

By using polynomial regression or differencing to eradicate long-term trends, degrading aids in identifying the underlying patterns. Detrending removes long-term trends from time-series data to focus on short-term fluctuations or underlying patterns. In air pollution determinations, such trends may result from seasonal changes (e.g., higher PM levels in winter), long-term environmental policies, or urbanization and industrial growth. Removing these trends helps improve the accuracy of short-term forecasting models, reveals cyclic or periodic patterns such as daily traffic-related spikes, and makes the data suitable for stationarity-based models like ARIMA or LSTM. Overall, detrending filters out non-informative trends, allowing for clearer insights and better anomaly detection in air quality data. The observed air pollution is denoted as (11):

$$Y(t) = T(t) + S(t) + R(t) \qquad (11)$$

Where: the original pollutant concentration at time t is denoted as $Y(t)$, the trend component is denoted as $T(t)$, the seasonal component is denoted as $S(t)$, the residual component is denoted as $R(t)$. The detrended signal is obtained by removing the trend $T(t)$is by (12):

$$Y_{detrended}(t) = Y(t) - T(t) \qquad (12)$$

To estimate the trend T(t): Moving average smoothing is defined as (13)

$$T(t) = \frac{1}{2k+1} \sum_{i=-k}^{k} Y(t+i) \qquad (13)$$

Where, $T(t)$ Smoothed value at time/index t, and $(t+i)$ denoted as Observed data value at position.

(v) <u>Seasonal decomposition in air pollution data:</u>

Seasonal decomposition improves model understanding by separating a time series into trend, seasonal, and residual components. Seasonal decomposition is a time-series analysis technique that breaks down data into three components: trend (long-term progression, such as increasing pollution due to urbanization), seasonality (repetitive patterns at regular intervals, like daily traffic peaks or seasonal weather effects), and residual (random, irregular fluctuations). In air pollution studies, this method is important because it reveals hidden periodic patterns, improves forecasting accuracy by isolating noise, aids in designing targeted environmental policies (such as addressing winter pollution spikes), and supports data preprocessing by removing seasonal biases, making the data more suitable for analysis and predictive modeling. There are two main models:

1. Additive model

$$Y(t) = T(t) + S(t) + R(t) \qquad (14)$$

Equation (14) is used when the seasonal variations are roughly constant over time.

2. Multiplicative model

$$Y(t) = T(t) \times S(t) \times R(t) \qquad (15)$$

Where,

In Equation (15), the observed air pollution value at time t is $Y(t)$, and the trend component is denoted as $T(t)$, The seasonal component is denoted as $S(t)$The residual is denoted as $R(t)$.

(vi) Data Normalization in Flood Data

In distance-based algorithms or gradient-based optimization in deep learning, adjusting the value ranges through normalization or scaling, such as Min-Max normalization or Z-score standardization, is crucial. Normalization is one of the most important preprocessing tasks while processing flood data. Different features have different values and units of measure. Without scaling, computer programming can be biased towards features with larger ranges. This paper applies the min-max normalization procedure to rescale the flood data to a specified range, typically 0 to 1 (depending on the data). It is expressed mathematically as (16):

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (16)$$

Where the raw data obtained from the coastal area is **X**, the minimum value obtained from the entire dataset is $\boldsymbol{X_{min}}$, and the maximum value is $\boldsymbol{X_{max}}$. The normalized output data is $\boldsymbol{X'}$. The other method is Z-score normalization, which normalizes the flood data. It transforms the input data to have a mean (μ) of zero and a standard deviation (σ) of 1, which distributes the data normally. It is expressed mathematically as (17):

$$X' = \frac{X - \mu}{\sigma} \quad (17)$$

Table-1 Flood Data Analysis Result

Coastal Area	Water level (m)	Rainfall (mm)	River flow (m^3/s)
A	12.5	200	150
B	15.0	150	180
C	10.0	180	200
D	18.0	250	220

The data in Table-1 are used in the normalization process, and the output for the water level alone is obtained for A=0.3125, B=0.625, C=0, and D=1.0 using min-max normalization.

(vii) Windowing/Segmentation for Flood Data:

By splitting sequences into smaller, more manageable fixed-size windows (overlapping or non-overlapping), windowing or segmentation enables local analysis or feature extraction. Windowing involves splitting continuous data into fixed-size chunks, or windows. This technique helps analyze time-series data, such as flood levels or rainfall, by focusing on shorter periods and simplifying the processing or modeling of temporal patterns. It consists of the primary types of windowing:

1. Non-overlapping windows: Data is split into apparent, non-overlapping segments.
2. Overlapping windows: Each window overlaps with the next, allowing for smoother transitions between windows and capturing more temporal patterns.

For time-series data, windowing is typically done by specifying:

- Window size (W): The length of each window (in terms of time steps or data points).
- Step size(S): The shift between consecutive windows. For overlapping windows, the step size is smaller than the window size, whereas for non-overlapping windows, the step size equals the window size.

1. Non-overlapping Windowing:

Each window contains w data points. The windows do not overlap. Accordingly, for a time-series data sequence X = [x1, x2, …, x_n], the windows can be represented as (18):

$$\text{Window(i)} = [x_{(i-1)W+1,} x_{(i-1)W+2,\ldots,} x_{iW,}] \qquad (18)$$

Where i is the window index.

2. Overlapping Windowing:

Each window contains exactly W data points. The windows overlap by S points, where S < W. For a time-series data sequence X= [x1, x2, …, xn], the windows can be represented as (19):

$$\text{Window i} = [x_{(i-1)S+1,} x_{(i-1)S+2,\ldots,} x_{iW,}] \qquad (19)$$

Where I is the window index, the step size S determines the overlap.

Windowing or segmentation is beneficial for breaking down constant flood data into smaller chunks, making it easier to analyze and model. Non-overlapping windows simplify the data into distinct periods. In contrast, overlapping windows capture more subtle temporal patterns that may be important for prediction or classification tasks, especially when using time-series models.

(viii) <u>Stationarity check and Transformation for flood data:</u>

Stationarity tests and transformations are used to preserve critical aspects of the mean and variance in many statistical models, employing methods such as differencing, logarithmic transformation, or Box-Cox transformation. In time-series analysis, stationarity refers to the statistical property where the mean and autocorrelation structure of the data do not change over time. For many time-series forecasting models, such as the Autoregressive Integrated Moving Average (ARIMA) model, it is essential to work with stationary data, as non-stationary data can lead to misleading conclusions.

Types of Stationarity:

1. Strict Stationarity: All stages (mean, variance, etc.) do not change over time. This is a stronger form of stationarity.
2. Weak Stationarity: The mean, variance, and autocovariance function are time-invariant. This is often sufficient for practical time-series determination.

Stationarity Check:

- Visual Inspection: Plans the time-series data and looks for trends, patterns, or seasonal components.
- Autocorrelation Function (ACF): Examine the autocorrelation at different lags to check if the values decline to zero or not.
- Statistical Tests:

- Augmented Dickey-Fuller (ADF): To test the null hypothesis that the time series is non-stationary.
- Kwiatkowski-Phillips-Schmidt-Shin (KPSS): Tests the null hypothesis that the time series is stationary around a deterministic trend.

Transformation Methods to Achieve Stationarity:

If the data are non-stationary, transformations such as the Box-Cox transformation or other transformations are used to stabilize the mean and variance.

1. Differencing

It involves subtracting the previous observation from the current one, thereby removing trends or seasonality. This method is widely used in ARIMA models. The first difference of a time series X_t at time t is (20):

$$Y_t = X_t - X_{t-1} \quad (20)$$

Where, X_t is the original value at time t, X_{t-1} is the value at time t-1, and Y_t is a different value.

If a first-order difference does not make the series stationary, apply higher-order differencing until stationarity is achieved.

2.Log Transformation

It is mostly used when the data has exponential growth or the variance increases over time. It stabilizes the variance and makes the series more stationary. It is expressed as (21):

$$Y_t = \log(X_t) \quad (21)$$

Where, X_t is the original value at time t and Y_t is the transformed value. This method of calculation is typically used when the data is strictly positive.

3. Box-Cox Transformation

This is a more general method for stabilizing variance and making data more normal (Gaussian). It works for both positive and negative values. It is defined as (22) and (23),

$$Y_t = \frac{x_t^{\lambda} - 1}{\lambda} \quad for\ \lambda = 0 \quad (22)$$

$$Y_t = \log(X_t) \quad for\ \lambda = 0 \quad (23)$$

Where: X_t is the original value at time t and λ is the transformation parameter, estimated from the data. The optimal λ is often estimated using maximum likelihood.

method.

(ix) Outlier detection and removal for flood data

Outlier detection and removal include identifying anomalies or abrupt spikes using statistical tests, Z-scores, or model-based techniques to avoid mispredictions. Finally, encoding time-based information, such as determining the day of the week, month, or hour of the day from timestamps, helps identify cyclical patterns in the data that models can use to learn. In flood data detection, outliers are data points that deviate significantly from the majority of the data. This model represents weather events and sensor errors that don't represent normal conditions. Detecting and handling outliers is crucial to enhancing the accuracy of data analysis and model predictions.

Methods for outlier detection

1. The z-score method detects outliers by looking far from the mean standard deviation; if the z-score exceeds the threshold, it is considered an outlier. It's represented by a mathematical model (24):

$$Z=\frac{X_t-\mu}{\sigma} \quad (24)$$

Where; X_t is a data point at time t, μ is the mean, σ is the standard deviation of the data, Z is the z-score for the data. if the value of $|Z|>3$, then X_t is an outlier.

2. The interquartile range (IQR) method is used for calculating the interquartile range and identifying data points that lie outside the threshold points beyond the IQR. A mathematical model has bounds, they are,
   The lower bound for outliers(25%quartile): $Q_1 - 1.5\times$ IQR
   The upper bound for outliers (75% quartile): $Q_3 + 1.5\times$ IQR

3. A box plot is a visual inspection tool often used to identify outliers by showing the spread of data.
4. The isolation forest algorithm is a machine learning method that profiles normal data points. It's useful for high-dimensional and complex datasets, such as flood data.

Outlier detection is a crucial step in identifying anomalous values in flood data. This analysis used the Z-score and IQR methods, which identified Day 12 (25.0 meters) as an outlier. Addressing outliers—whether through removal or transformation—can enhance the performance of a sufficient model, yielding more similar results in flood prediction analysis.

(x) <u>Encoding time features for flood data:</u>

Capturing time-series data in flood data makes it challenging to understand seasonal patterns, trends, and abnormalities. Key components such as hour of the day, day of the week, and month are valuable and enhance predictive modeling by capturing cyclical patterns in the data.

(1) Types of time features to extract
The time series features for the hour capture daily patterns; the day of the week captures weekly trends; the month captures seasonal trends; the day of the year captures more granular patterns; the week of the year captures weekly trends; the quarter of the year; and weekends.

(2) Encoding methods for time features
One-hot encoding is used to convert categorical time features such as day of the week or month. Another feature is sinusoidal encoding for hour of the day and day of the week, which prevents models from interpreting time as a linear sequence. To capture the cyclical nature, the encoding variables are hours (2-23), day of week (0-6), and month (1-12), using the following expression, which is calculated in steps (25)-(30).

$$Hour_{sin} = \sin(\frac{2\pi\times hour}{24}) \quad (25)$$

$$Hour_{cos} = \cos\left(\frac{2\pi \times hour}{24}\right) \quad (26)$$

$$Day_{sin} = \sin(\frac{2\pi\times day}{7}) \quad (27)$$

$$Day_{cos} = \cos\left(\frac{2\pi \times day}{7}\right) \quad (28)$$

$$month_{sin} = \sin(\frac{2\pi \times month}{12}) \quad (29)$$

$$month_{cos} = \cos\left(\frac{2\pi \times month}{12}\right) \quad (30)$$

(B) Image Data Preprocessing

Image data preprocessing focuses on converting visual inputs into a standardized format suitable for object detection, image classification, or segmentation.

(i) <u>Resizing – Standardize Image Dimensions for Flood Images:</u>

Resizing is the first critical step to ensuring that all images have a similar size (e.g., 224×224 pixels), which is particularly significant for batch processing in neural networks. When color information is irrelevant, RGB images are converted to single-channel grayscale images, reducing computational complexity and memory consumption. Resizing images to a standard dimension is an essential preprocessing step in machine learning and image processing. It's crucial to ensure that all flood images, whether satellite photos, aerial photographs, or other flood-related images, are the same size to enable accurate analysis and model input. By standardising image dimensions, you can ensure the model gets consistent input, improving training effectiveness, model performance, and error prevention.

Importance of resizing

Since deep learning models, especially Convolutional Neural Networks (CNNs), require input images of the same proportions, resizing photographs to a consistent input size is crucial. The model struggles to process photos in batches when they vary in size. Large images consume more memory and compute resources, so reducing their size improves memory efficiency. The model struggles to process photos in batches when they vary in size. Pretrained models such as VGG16 and ResNet frequently employ this standard size to ensure compatibility with the expected input size. Also, flood images can vary widely in size and come from different sources and time periods. Consistent analysis and feature extraction are maintained by resizing them to a uniform dimension, resulting in dependable model performance.

The main goal of scaling an image is to change its proportions, including height and width, without affecting the essential visual information it contains. This process relies on interpolation techniques, which modify pixel values. Consider an image with original dimensions. $H_{orig} \times W_{orig}$. Where H_{orig} and W_{orig} are the original height and width of the image. The original image needs to be resized into a new target size of 244×244, which is denoted as $H_{new} \times W_{new}$.

Interpolation method

The value of the closest corresponding pixel from the original image is assigned to each pixel in the resized image using nearest neighbour interpolation. Based on the ratio of the new and original dimensions, this method is straightforward: it takes the new pixel value directly from the closest pixel in the original grid. For this approach, the mathematical model is (31):

$$I_{new}(x,y) = I_{orig}\left(round\left(\frac{x}{W_{new}}\cdot W_{orig}\right), round\left(\frac{y}{H_{new}}\cdot H_{orig}\right)\right) \quad (31)$$

Image quality can be enhanced during resizing using the technique known as Bilinear Interpolation. Unlike previous methods that used the value of a single neighboring pixel, this technique uses data from a complete 2x2 array of known pixels centered on the target pixel. Even though this pixel is not known at the moment, its neighbors to the left and below can be used. It also calculates the average intensity of surrounding pixels, but not equally. Rather, the value of each pixel is determined by how its coordinates compare with those of the target pixel. Hence, pixels that closely match the target value receive higher weights, ensuring a logical, smooth transition with no abrupt changes.

$$I_{new}(x,y) = (1-f)\cdot(1-g)\cdot I_{orig}(x_1,y_1) + f\cdot(1-g)\cdot I_{orig}(x_2,y_1) + (1-f)\cdot g \cdot I_{orig}(x_1,y_2) + f\cdot g\cdot I_{orig}(x_2,y_2) \quad (32)$$

In equation (32), x_1, y_1 represent the coordinates of the top-left pixel and x_2, y_2 those of the bottom-right pixel in the original image. The variables f and g are the fractional distances from the original grid. Compared to nearest neighbor interpolation, bilinear interpolation produces visually better results, especially in terms of smoothness and quality.

Bicubic interpolation is a step up from basic methods, replacing them with a 4x4 array of nearby pixels. It uses cubic equations to determine new pixel values, producing smoother, higher-quality results than bilinear interpolation. In the math behind bicubic interpolation, the result is a weighted sum of pixel values, obtained by fitting cubic polynomials to the original image. While this uses more computational power, it is chosen when image quality is highest.

$$I_{new}(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}\cdot(x-i)^3\cdot(y-j)^3 \quad (33)$$

Using sinc functions to derive new pixel values is a high-quality interpolation known as Lanczos resampling. It performs exceptionally well at image downsampling, preserving detail while minimizing artifacts. The method performs weighted averaging of several nearby pixel values, with the sinc function determining the weights based on the distance to the target pixel. Although computationally expensive and underperforming in scenes where high visual fidelity is not required, Lanczos resampling delivers unparalleled results for high-detail imaging.

$$I_{new}(x,y) = \sum_{i=0}^{N}\sum_{j=0}^{N} I_{orig}\left(x_i,y_j\right)\cdot sinc(x-x_i)\cdot sinc\left(y-y_j\right) \quad (34)$$

Rescaling flood images to a standard size (for example, 224x224 pixels) is required in machine learning models. This also ensures uniform input sizes, optimizing memory use and improving performance across many datasets. The choice of interpolation method (nearest neighbor, bilinear, bicubic, or Lanczos) balances computation time and output image quality. For high-quality models that require smooth results, bilinear or bicubic interpolation is the go-to, but for quick, low-resource image resizing, nearest-neighbor interpolation is used.

(ii) Grayscale Conversion:

Transforming images to grayscale is a fundamental step that is applied adaptively throughout the lifecycle of flood images. For models and analyses where color doesn't significantly affect the results, transforming RGB images to grayscale greatly reduces processing demands on power, storage, and bandwidth. It changes the data structure in a more succinct, efficient, and streamlined way, making it easier for artificial intelligence systems to interpret visual data.

Why convert RGB images into grayscale for the flood image

Grayscale in flood image analysis has many benefits. First, they provide a computational efficiency advantage: they use only a single intensity channel, rather than RGB's red, green, and blue channels. This, in turn, reduces memory use and processing time, which is very useful for large datasets. Second, greyscale images emphasize intensity, which is often enough to detect flood patterns, extent, and severity, eliminating the need for colour information. Third, grayscale simplifies models by removing color, which in turn makes it easier for machine learning algorithms, particularly early-stage models, to identify key features at the pixel level. Fourth, grayscale works well with pre-trained models such as VGG16 or ResNet, which are mostly trained on RGB data, and they still perform well in a grayscale environment. Finally, greyscale images speed up inference by allowing models to focus on intensity-based features such as edges, textures, and shapes, which are critical to flood analysis. Greyscale conversion is accomplished by calculating a weighted average of the RGB colour channels to create a single channel representing light intensity. The most widely used formula for converting an RGB image to greyscale is defined as (35):

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \qquad (35)$$

In this formulation, Y denotes each pixel's grayscale value (luminance). Figure 5 shows the values of R, G, and B as the red, green, and blue components of the pixel with a range of 0 to 255. The constants 0.299, 0.587, and 0.114 are derived from the dimensions of human vision, reflecting that green is brighter than red and blue, and thus contributes more to brightness in grayscale.

Grayscale Conversion Process

First, extract the RGB values for each pixel in the image to convert the image to greyscale. After obtaining these values, a weighted sum is calculated using the greyscale conversion formula, which assigns weights to the red, green, and blue components to determine the greyscale intensity. Finally, the greyscale image is created by replacing each pixel's original RGB channels with the computed greyscale intensity value. Greyscale conversion is a simple but powerful preprocessing technique, particularly in flood image analysis, where the focus is frequently on intensity-based features rather than colour. By reducing the 3 RGB channels to a single grayscale channel, we improve the efficiency of computer resources while preserving the main information for flood detection. The math behind the gray scale transform is a weighted sum of the red, green, and blue channels, representing each color's perceived brightness. This makes greyscale conversion especially useful when working with large datasets, pretrained models, or when colour does not significantly contribute to the analysis.

(iii) <u>Normalization - Scale Pixel Values to a Standard Range for Flood Images</u>

Normalization standards input ranges and speeds up learning convergence by scaling pixel values, usually from 0–255 to 0–1 or -1 to 1. By applying random transformations such as rotation, flipping, zooming, translation, and cropping, data augmentation artificially expands the training dataset's size and diversity, improving the model's performance and generalization. Normalization is a key step in image processing, particularly when dealing with flood images. It requires transforming pixel values to a standardized range, typically [0, 1] or [-1, 1], to ensure data consistency and make it ready for machine learning analysis. Normalization boosts model performance by streamlining and stabilizing the optimization process.

1. Consistency: Normalization brings consistency to pixel values across images, despite differences in capture devices, formats, or lighting conditions.

2. Improved Convergence in Neural Networks: Scaling input data to a consistent range enhances the performance of neural networks and intense learning models by facilitating faster convergence and reducing issues with large gradients.

3. Avoiding Saturation: Normalizing pixel values helps prevent numerical instability and learning issues in models like CNNs by keeping values within a manageable range, avoiding saturation and extreme value problems.

4. Uniformity with Pertained Models: To effectively utilize pertained models like VGG16 or ResNet, it's crucial to normalize pixel values in the same range used during their training, typically [0, 1] or [-1, 1], to ensure consistency.

5. Improved Image Analysis: Normalization stabilizes pixel intensity in flood images, which can vary greatly due to lighting, time of day, and environmental factors. Thus, normalization enhances the model's ability to detect key features like water boundaries, flood extent, and vegetation changes.

Normalization typically involves transforming pixel values from their initial range to a desired range. The most routine approaches are:

1. Min-Max Normalization: Rescaling pixel values to a range of [0, 1] or [-1, 1]. A flood image with intensity values ranging from 0 to 255 (the standard for 8-bit images) is resized to (0, 1).

$$Xnorm = \frac{X - Xmin}{Xmax - Xmin} \quad (36)$$

Where, in equation (36) X represents the original pixel value, $Xmin$ and Xmax denote the image's minimum and maximum pixel values, respectively, and $Xnorm$ represents the normalized pixel value, typically scaled to the [0, 1] range.

$$Xnorm = 2.\frac{X - Xmin}{Xmax - Xmin} - 1 \quad (37)$$

This transforms the pixel values to the range [-1, 1]. This method is used in situations where models benefit from pixel values centered on zero, which can help with the intersection of algorithmic optimization. A flood image where a pixel intensity range from 0 to 255 is first normalized to (0, 1) and then shifted and scaled to [-1, 1].

2. Z-score Normalization (Standardization): Pixel values are normalized through standardization, where the mean is subtracted, and the result is divided by the standard deviation.

$$Xnorm = \frac{X - \mu}{\sigma} \quad (38)$$

Where, in equation (38), μ represents the mean of the pixel values in the image, μ denotes the standard deviation of the pixel values in the image, and Xnorm is the normalized pixel value. Z-score normalization is applied when there is a necessity to standardize with a mean of zero and a standard deviation of one. This method is particularly useful for algorithms sensitive to the scale of input data. data. Flood image intensity values are systematic with respect to the image's mean and standard deviation, resulting in a distribution of pixel values with a mean of 0 and a standard deviation of 1.

3. **Dividing by a Constant:** For 8-bit images with pixel values ranging from 0 to 255, a straightforward normalization technique involves dividing the values by 255, thereby scaling them to the range [0, 1].

$$Xnorm = \frac{X}{255} \quad (39)$$

Equation (39), maps pixel values from [0, 255] to [0, 1].

(iv) **Image Data Augmentation:**

Data augmentation is an essential preprocessing technique for image-based applications such as flood detection, segmentation, and classification. It includes making several changes to the images to improve the model's generalization, reduce overfitting, and artificially enhance the training dataset. To help the model become invariant to these transformations and improve its generalization to unseen data, data augmentation exposes the model to a wide range of variants of the same image, such as rotations, flips, zooms, and crops.

Benefits of Applying Data Augmentation for Flood Images

Images of flood events may show a wide range of changes due to environmental and technical factors, such as varying light conditions, changing weather, different camera angles, and varied geography. These issues can present a great challenge to the performance of machine learning models. Data augmentation helps address this issue by adding variety to the data the model is trained on, which in turn creates better, more flexible models that perform well in the real world. Also, there are a number of very good reasons that data augmentation is so important in the field of flood image work:

1. **Overfitting Prevention:** When the models are trained on small sets of data, they tend to memorize certain examples, which in turn causes them to perform well on that narrow set of

data at the expense of more general features. This is what is termed as overfitting. Also, by creating multiple variations of the same input through data augmentation, the model learns to recognize which features are consistent across inputs, which in turn improves its performance on novel data.

2. **Improved Robustness**: Images vary in quality, size, and orientation by source. The introduced augmentation will bring out these variations, which in turn make the model less sensitive to such changes. As a result, the model becomes more robust and accurate at processing images taken under different conditions.
3. **Simulating Different Perspectives:** Flood images are captured from different perspectives, aerial and ground shots, at different times of the day. Including such variations during model training helps them better understand floods and become more useful for a wider range of real-world tasks.
4. **Artificially Increase Dataset Size:** In most scenarios, including emergencies and natural disasters, collecting a vast number of images and labelling them is a challenging task. Data augmentation helps tackle these problems by expanding the training dataset using pre-existing labelled data without additional capture, thereby reducing data collection effort while enhancing model performance.

Common Data Augmentation Techniques for Flood Images

Numerous image augmentation techniques are performed to increase the performance of flood detection models. These methods increase dataset diversity and enhance model adaptability to multiple visual situations. One of the primary methods includes, but is not limited to:

1. Rotation

This method consists of rotating an image in a specific direction (clockwise or counterclockwise) by a particular angle. Doing this helps the model identify flood patterns regardless of image orientation. This is particularly helpful as real-life images of floods are often captured through different devices and might be mounted on the cameras at odd angles.

Mathematical Model:

For an image with pixel coordinates (x,y), a rotation by angle θ is performed as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (40)$$

For the above equation (40), (x',y') have been the coordinates of the rotated pixel and θ is the rotation angle (in radians).

2. Flipping (Horizontal and Vertical)

Fluctuating between different axis orientations is a common image augmentation practice. The presentation of flood events includes the sides of a structure that are affected by the water and the direction in which the flood is presented to the model. This is done to reduce the model's attention to the direction and position of flood-related elements. It increases the model’s flexibility in identifying floods across a variety of spatial orientations.

Mathematical Model:

For horizontal flipping, the transformation is defined as (41):

$$x' = W - x \quad (41)$$

For the above equation (41), W represents the width of the image, x denotes as x-coordinate of the pixel, and x' denotes as the new x-coordinate after flipping.

For vertical flipping:

$$y' = H - y \quad (42)$$

For the above equation (42), H represents the height of the image, y denotes the y-coordinate of the pixel, and y' denotes the new y-coordinate after flipping.

3. Zooming

Here, it can zoom in for a closer look at fine details or zoom out to see the bigger picture. This is useful when it contains flood images from different distances or resolutions. Exposing the model to flood elements at multiple scales helps it generalize more efficiently.

Mathematical Model:

Zooming in on an image is done by scaling the pixel coordinates by a factor s, where s > 1 for zoom-in and s < 1 for zoom-out. The new coordinates are given by (43) and (44):

$$x' = s \cdot x \quad (43)$$

$$y' = s \cdot \mathrm{y} \quad (44)$$

Here, the x' and y' are denoted as zoomed-in coordinates.

4. Cropping

Cropping is the practice of using an "as seen at point X" approach with a given picture. It will choose and present only part of the whole, sometimes a random segment. This also allows the model's attention to be focused on very local areas of the image that may contain telltale signs of a flood, and which may appear only in some sections.

Mathematical Model:

Given an image of size W×H, cropping can be defined by selecting a region defined by (45):

$$Crop_{x_{start,}x_{end,}y_{start,}y_{end}} \quad (45)$$

Here, $x_{start,}x_{end,}y_{start,}y_{end}$ define the region of interest for cropping. The coordinates are then restricted to the region, which is defined as (46):

$$Cropped\ Image = I(x_{start}{:}\,x_{end}, y_{start}{:}\,y_{end}) \quad (46)$$

This technique ensures that the model can learn and detect flood-related features across various regions of the image.

5. Brightness and Contrast Adjustment

Here it utilizes adjustment of the brightness and contrast in these images to represent various lighting conditions, which in turn helps us out a lot in what may be very different environmental settings, from overcast days to very bright sun. By doing so, it will improve the model's performance by grading images more accurately regardless of lighting conditions.

Mathematical Model:

To adjust the brightness, add a constant value c to the pixel intensities is defined as (47):

$$I^{'}(x, y) = I(x, y) + c \quad (47)$$

Where $I(x, y)$ is the original pixel intensity, and $I^{'}(x, y)$ is the adjusted pixel intensity.

To adjust the contrast, multiply the pixel intensities by a constant factor f:

$$I^{'}(x, y) = f \times \ (I(x, y) - \mu) + \mu \quad (48)$$

In equation (48) μ denotes the mean pixel intensity of the image.

6. Shearing

Shearing is a process that moves the image along a horizontal or vertical axis, thus producing a skewed or slanting effect. This transformation helps the model adapt to how floods appear in photos taken from various angles, which are also slightly distorted. It improves the model's generalization performance across variable perspectives in the real world.

Mathematical Model:

For horizontal shearing, the transformation is defined as (49):

$$x^{'} = x + k \cdot y \quad (49)$$

Here, K represents the shearing factor, x, y are denoted as the coordinates of the original pixel, and $x^{'}$ denotes the new x-coordinate after shearing.

Using data augmentation is a great way to add variety to a flood image set when limited data is only available. It will use techniques such as rotation, flipping, cropping, and zooming to simulate different visual settings and points of view. This, in turn, helps the model recognize a wider range of flood-related patterns. Here, it created artificial variations that, over time, deepened the training process and improved the model's performance on new or real-world flood images. Also, this practice reduces the risk of overfitting, improves the model's reliability, and, at the same time, delivers better performance in diverse and unexpected situations such as scale changes, angle shifts, or variations in environmental lighting.

(v) **<u>Image Data Noise Reduction</u>**

Flood images obtained from real-world environments—such as those captured by drones, satellites, or surveillance systems—are often affected by various types of noise. This noise can arise from poor lighting conditions, sensor imperfections, image compression, or environmental factors such as rain, fog, and reflections. The goal of noise reduction is to smooth these images while preserving essential visual elements, such as water boundaries and flooded areas, and removing irrelevant pixel-level fluctuations that could interfere with analysis.

Applying noise reduction techniques to flood images is crucial for several reasons. It enhances overall image clarity, which benefits downstream computer vision tasks such as image segmentation, classification, and object detection. Reducing noisy edges or artifacts minimizes the likelihood of false detections. Additionally, noise reduction helps maintain spatial coherence in water-affected regions, making it easier to delineate the extent of flooding.

One widely used method for smoothing is the Gaussian Blur filter. This linear smoothing technique applies a Gaussian function to assign different weights to the neighboring pixels of each target pixel. Doing so suppresses high-frequency components—typically associated with noise—while retaining low-frequency features that represent meaningful information, such as flood zones, as expressed in (50).

$$G\,(x, y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{X^2 + y^2}{2\sigma^2}} \qquad (50)$$

The two-dimensional Gaussian function is defined mathematically, where the standard deviation (σ) controls the amount of blurring. The image is then convolved with a Gaussian kernel of the specified size (e.g., 5×5). This results in a smoothed image with significantly reduced noise, while important flood features are preserved.

Gaussian Blur offers several benefits in flood image analysis. It smooths the surface texture of water bodies, making the water boundaries more distinct. It also eliminates distracting background elements such as debris or vegetation and improves the accuracy of flood area segmentation models by reducing pixel-level inconsistencies.

$$I_{blurred}\,(i,j) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} G(m,n) \cdot I(i+m, j+n) \qquad (51)$$

Where, in equation (51), i and j denote the coordinates of the pixel being processed, k denotes the radius of the Gaussian kernel, and $I_{blurred}$ (i,j) denotes the new smoothed value at location (i,j).

In addition to Gaussian Blur, alternative smoothing techniques are useful depending on the type and intensity of the noise. The Median Filter, for instance, replaces each pixel with the median of its neighborhood and is particularly effective against salt-and-pepper noise. The Bilateral Filter offers edge-preserving smoothing, which is useful when retaining detailed edges in flood maps is important. Another advanced method is the Non-Local Means filter, which considers pixel similarity across larger regions. Although computationally intensive, it is highly effective for high-fidelity noise reduction. Overall, using noise reduction filters, particularly Gaussian Blur, significantly prepares flood images for accurate analysis by improving clarity, preserving critical features, and enhancing model performance.

(vi) Contrast Enhancement using CLAHE

To improve feature clarity, noise reduction uses filtering techniques such as median filtering and Gaussian blurring to smooth the image and reduce pixel-level noise. Techniques for obtaining different photos, such as histogram equalization or CLAHE (Contrast Limited Adaptive Histogram Equalization), enhance the visibility of features in low-contrast or poorly lit images. Flood images often suffer from poor visibility due to factors like cloudy weather, nighttime conditions, or camera limitations. This reduced contrast can obscure critical flood-related features such as the spread of water, submerged roads, or damaged buildings. To address this, contrast enhancement techniques such as Histogram Equalization and CLAHE (Contrast Limited Adaptive Histogram Equalization) are used to improve visual clarity. These methods not only enhance human interpretation during emergency response but also assist machine learning models in accurately detecting relevant patterns and structures in the image.

Histogram Equalization is a global contrast enhancement technique that redistributes pixel intensity values across the full brightness range (typically 0–255 for 8-bit images). This process involves computing the histogram of the grayscale image, deriving its normalized version (probability distribution function), and then calculating the cumulative distribution function (CDF). Based on this CDF, pixel intensities are then mapped to new values, effectively spreading out frequently occurring intensity levels and enhancing overall contrast. This method is beneficial for flood images with uniform lighting, such as overcast skies or foggy scenes, where global contrast is uniformly low.

Compute normalized histogram: (PDF) is defined as (52):

$$P(i) = \frac{h(i)}{N} \quad (52)$$

The cumulative distribution function (CDF), is defined as (53):

$$c(i) = \sum_{j=0}^{i} p(j) \quad (53)$$

Map original pixel values, which is defined as (54):

$$T(i) = round\big(c(i) \cdot (L-1)\big) \quad (54)$$

On the other hand, CLAHE enhances contrast locally by dividing the image into smaller regions called tiles (e.g., 8x8 pixel blocks). For each tile, a localized histogram and CDF are computed. To avoid amplifying noise in flat areas, the histogram is clipped if any bin exceeds a certain threshold, and excess pixels are redistributed. Each pixel is then adjusted based on its tile's local histogram. Finally, interpolation is applied between tiles to ensure smooth transitions across borders and prevent artifacts. CLAHE is especially effective for flood images with varying lighting conditions, such as those taken by drones or in mixed weather, because it adapts to localized image characteristics.

In practice, both techniques serve essential roles in flood image processing. Histogram Equalization is best suited for images with consistent lighting, providing an overall contrast boost. CLAHE, by contrast, excels in environments with mixed or complex lighting, offering nuanced enhancement without excessive noise amplification. These methods are used to enhance satellite images for accurate flood zone mapping, improve drone footage captured in low-light conditions, and preprocess datasets for deep learning models such as CNNs. Ultimately, enhancing contrast in flood imagery significantly boosts both automated analysis and manual inspection, contributing to more effective disaster management and response.

(vii) <u>Image centering and padding:</u>

The size and aspect ratio of flood images vary frequently; they are collected from various devices such as drones, satellites, and CCTV. Image resizing is done using image centering and padding without distortion, ensuring the object of interest, such as flood areas, is centered and aligned. This is an essential process for storing images in a CNN or other DL-based model that requires a standard input dimension, such as 224 x 224 pixels; this value is the same for all inputs to the model.

Image centering and padding are important for flood image processing because training and testing require standardized image dimensions to prevent distortion from naïve resizing, keep the flooded region centered to enhance the model's focus, and preserve spatial context such as buildings, roads, and rivers during the detection task.

Let the dimensions of the original input image for height and width is defined as (55) and (56):

$$Height = H_o \qquad (55)$$

$$Width = W_o \qquad (56)$$

Let the target dimension of the input flood image is defined as (57) and (58):

$$Height = H_t \qquad (57)$$

$$Width = W_t \qquad (58)$$

Step 1: Resize Proportionally

Initially, the input image is resized until it fit into target size image aspect ratios is expressed as (59), (60), and (61):

$$s = min\left(\frac{W_t}{W_o}, \frac{H_t}{H_o}\right) \qquad (59)$$

$$W_r = round(W_o \, . s), \qquad (60)$$

$$H_r = round(H_o \, . s) \qquad (61)$$

Now resize the input image to the size of W_r, H_r aspect ratio.

Step 2: Computing padding amounts

The resized image needs to be padded to fit the target size image aspect ratio is defined as (62) and (63):

$$pad_x = W_t - W_r, (62)$$

$$pad_y = H_t - H_r \qquad (63)$$

Now, split the padding equally between both sides of the image is defined in (64), (65), and (66):

$$left = \left\lfloor \frac{pad_x}{2} \right\rfloor, \ (64)$$

$$right = pad_x - left, \ (65)$$

$$top = \left\lfloor \frac{pad_y}{2} \right\rfloor, \qquad (66)$$

Zero padding is used for neural backgrounds; they are generally black. If the flood textures are sensitive, sharp or harsh edges are avoided using repeated or reflective padding. Centering is applied regularly when the flood region is the primary object of interest. Even input shapes and their primarily dependent continuous spatial patterns are more important for a DL-based model. All flood input images must be the same size, and image centering and padding ensure the original aspect ratio is preserved. During naïve resizing, this proposed methodology enhances generalization and prevents artifacts that occur when stretching or squashing.

(viii) <u>Color Space Conversion</u>

In image preprocessing, color space conversion is a critical step. The primary objective of this step is to extract or highlight essential features in the input images, such as vegetation, water, and man-made color properties. The model's prediction accuracy will be further improved by converting flood images from the RGB color space to CIELAB or HSV. The RGB color images failed to depict the hidden details due to inconsistent lighting or shadowing. Whereas, HSV- or CIELAB-based images accurately depict every edge and isolate intensity and color. Table-2 depicts the advantage of color space conversion.

Table-2: Benefits of Color Space Conversion

Color space	**Advantage**
HSV	Isolated chromatic details from lighting
LAB	Isolated intensity and color
RGB	Easy to capture details from poor images

Using the Following expressions, the input RGB triplet $R, G, B \in [0,1]$ the image is converted into HSV. For this, initially, the maximum and minimum RGB values are evaluated as (66) and (67):

$$C_{max} = \max(R, G, B) \quad (66)$$

$$C_{min} = \max(R, G, B) \quad (67)$$

Then, the predicted max and min values are computed as (68):

$$\Delta = C_{max} - C_{min} \quad (68)$$

Now the value of Hue (H) is computed as (69):

$$\text{H} = \begin{cases} 0^{\circ} \quad \Delta = 0 \\ 60^{\circ} \times \left(\frac{G-B}{\Delta} \ mod\ 6\right), C_{max} = R \\ 60^{\circ} \times \left(\frac{B-R}{\Delta} + 2\right), C_{max} = G \\ 60^{\circ} \times \left(\frac{R-G}{\Delta} + 4\right), C_{max} = B \end{cases} \quad (69)$$

The value of Saturation (s) is defined as (70):

$$S = \begin{cases} 0, & C_{max} = 0 \\ \dfrac{\Delta}{C_{max}}, & otherwise \end{cases} \quad (70)$$

Finally, the value of V is calculated by $V = C_{max}$.

RGB to LAB Conversion

Convert the input RGB to linear XYZ using a matrix transformation, and normalize it to the white point. The LAB conversion is using the following expressions as (71), (72), (73), and (74):

$$L^{*} = 116f\left(\frac{y}{y_n}\right) - 16 \quad (71)$$

$$A^* = 500\left[f\left(\frac{X}{X_n}\right) - f\left(\frac{y}{y_n}\right)\right] \quad (72)$$

$$B^* = 200\left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right] \quad (73)$$

Where,

$$f(t) = \begin{cases} t^{1/3} & t > \delta \\ \frac{1}{3\delta^2} + \frac{4}{29}, & otherwise \end{cases} , \delta = \left(\frac{6}{29}\right)^3 \quad (74)$$

Color conversion is helpful during flood analysis, especially when using input RGB images. Converting RGB to LAB and HSV is more helpful for analyzing the features of input images under various environmental conditions.

(ix) Edge Detection

Among preprocessing techniques, edge detection systems extract structural outlines, boundaries, and significant transitions from input images. The edge detection step in the input flood images detects roads, bridges, land areas, and water. Discontinuities in the input image are identified using an edge-detection algorithm to detect floods. The most commonly used operator for edge detection is the Sobel operator. The edges are detected using the Sobel operator, which computes the horizontal and vertical components. It is expressed as (75), (76), (77), and (78):

$$Horizontal\ (GX)\colon\ G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I \quad (75)$$

$$Vertical\ (GY)\colon\ G_Y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I \quad (76)$$

$$Gradient\ Magnitude : G = \sqrt{G_x^2 + G_y^2} \quad (77)$$

$$Direction\ (Optinal)\colon \theta = tan^{-1}\left(\frac{G_Y}{G_x}\right) \quad (78)$$

The other operator used for edge detection is the Canny operator. A Gaussian blur is applied to remove noise from the input images. The gradient magnitude is computed by taking the magnitude of the gradient. Based on the maximum and minimum threshold values, the input images' strong, weak, and non-edges are detected. Table-3 presents the advantages of various edge detection operators.

Table-3: Advantages of edge detection operator

Operators	Pros
Sobel	Superior, fast, and good for gradient detection than other models
Canny	Robust, clean, and multi-stage edge detection.
Laplacian	Depicts all the intensity changes in the input images.

Edge detection is a powerful tool for predicting floods from input environmental data. Operators such as Sobel and Canny are more effective at isolating unwanted regions and extracting essential features from the input data. This step enhances the model's efficiency in predicting floods from the images.

(x) <u>Channel Standardization</u>

Lastly, channel standardization is practiced in deep learning pipelines (specifically with pretrained convolutional neural networks) to ensure input distributions match the data the models were initially trained on. It ensures subtracting the mean and dividing by the standard deviation for each color channel. The foundation of multimodal data handling includes several preparation stages that ensure time-series and image data are appropriately prepared for analytical or predictive modeling.

Channel standardization is performed during prediction, also known as per-channel normalization. The input image channels are standardized by subtracting the mean and dividing by the standard deviation of the input RGB image channels. The input RGB images are collected from various sources such as ground photography, drones, or satellites. These images are standardized by setting the mean to 0 and the variance to 1. The primary objective of this step is to analyse the flood images from the water images with complex light reflections. The channel standardization technique is applied to adjust color contrast, eliminate sharp edges, prevent channel-wise gradient loss, and improve model accuracy. It is defined as:

$$Let\ I \in R^{H\times W\times C} \text{ is a color image,}$$

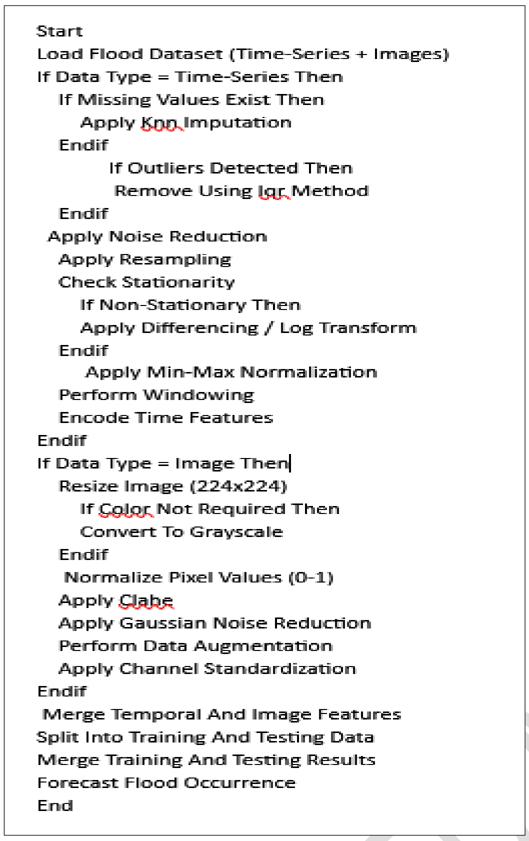
Where H and W are the height and width of the input images, and C represents the 3 channels (RGB) of the input images.

Each channel $C \in \{R, G, B\}$ is standardized as:

$$\grave{I}_{x,y,c} = \frac{I_{x,y,c} - \mu_c}{\sigma_c} \quad (79)$$

Where in equation (79), $I_{x,y,c}$ represent the pixel value of the input images at (x, Y) position in channel C, μ_c represent the mean pixel intensity value of channel C, σ_c represent the standard deviation value, and $\grave{I}_{x,y,c}$ represent the standardized pixel value. Channel standardization is a critical step in an ML- or DL-based flood prediction system, ensuring a zero-mean, unit-variance distribution. This method is more useful when large image datasets are analysed under various environmental conditions. The overall structure of the proposed approach is discussed in pseudocode 1.

Pseudocode_1:



```
Start
Load Flood Dataset (Time-Series + Images)
If Data Type = Time-Series Then
  If Missing Values Exist Then
    Apply Knn Imputation
  Endif
      If Outliers Detected Then
       Remove Using Iqr Method
  Endif
 Apply Noise Reduction
  Apply Resampling
  Check Stationarity
    If Non-Stationary Then
    Apply Differencing / Log Transform
  Endif
    Apply Min-Max Normalization
  Perform Windowing
  Encode Time Features
Endif
If Data Type = Image Then
  Resize Image (224x224)
    If Color Not Required Then
    Convert To Grayscale
  Endif
   Normalize Pixel Values (0-1)
  Apply Clahe
  Apply Gaussian Noise Reduction
  Perform Data Augmentation
  Apply Channel Standardization
Endif
 Merge Temporal And Image Features
Split Into Training And Testing Data
Merge Training And Testing Results
Forecast Flood Occurrence
End
```

Experimental Setup

The overall input flood data are analysed using the simulation software installed on a system with 8GB RAM, an Intel i3 11th Gen Processor, 512 GB ROM, and a 512GB SSD. By deploying various processing techniques to pre-process and forecast the possibility of flooding from the input real-time flood prediction data. For this, 80% of the data is used for training and 20% for testing. It improves the proposed model's flood prediction accuracy.

4. Results and Discussion

This section presents detailed simulation results for the proposed model. It compares the performance of the proposed and traditional learning models in predicting floods from input images and time-series data, both before and after preprocessing, and clearly discusses the results. The input flood data are collected using multiple sensors from the coastal region at different locations, times, and environmental conditions. Before predicting model efficiency, the overall input data is split into two phases: training and testing. That is, 80% of the data is used for training and 20% for testing. It improves the proposed model's flood prediction accuracy. In terms of accuracy, the model's performance before and after pre-processing is analysed to evaluate model efficiency. Then, the results are compared with those of other learning models, demonstrating the efficiency of the DL-based approach.

Performance Evaluation

Model performance is verified by comparing the truth flood extent maps presented by government agencies responsible for disaster management. Flood prediction is compared to reference labels and a confusion matrix. It uses several performance metrics, such as Accuracy, Precision, Recall, and F1-score, to validate overall efficiency. Temporal validation is also achieved by testing the model on unseen floods. Paired t-tests are used to assess statistical significance between the pre-processed and non-pre-processed datasets. This validation framework will help ensure some certainty about how preprocessing improves classification and segmentation performance.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

$$Precision = \frac{TP}{(TP + FP)}$$

$$F1 - \ Score = \ 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)}$$

$$Recall = \frac{TP}{(TP + FN)}$$

$$Specificity = \frac{TN}{(TN + FP)}$$

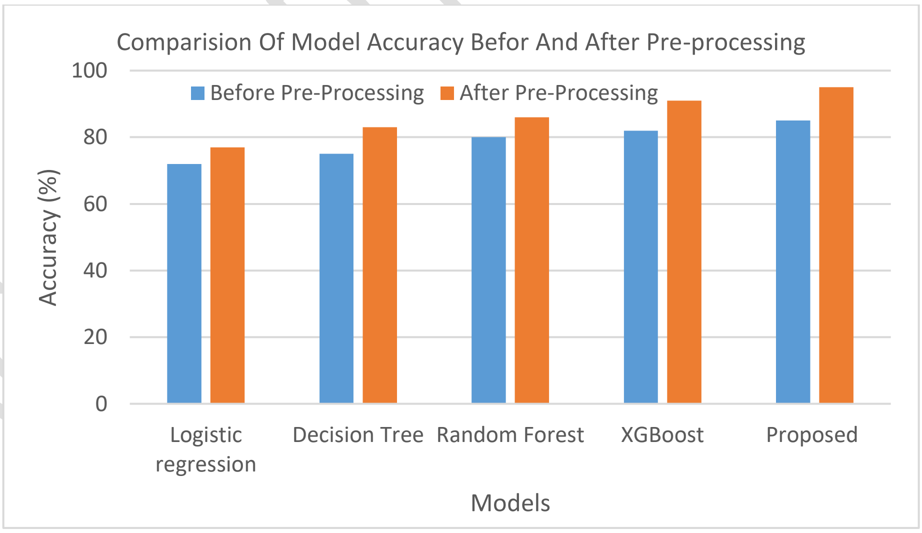


Figure 2: Performance Comparison

Figure-2 shows the overall accuracy comparison of several models, including LR, RF, DT, XGBoost, and the proposed model, for flood prediction. The LR model has attained only a 6% gain in overall performance. Whereas tree-based models, such as RF and DT, have improved performance by 8% in flood prediction, the XGBoost model has increased accuracy by 9%, and the proposed model has further boosted performance by 10%, achieving an overall accuracy of 95%. These results demonstrate that the DL model performs better with robust preprocessing, yielding more precise and reliable flood predictions.

Table-4: Performance of Different Pre-processing Techniques

Techniques	Accuracy	Result
Before Pre-processing	85%	Baseline
Missing data imputation (KNN)	89% (+4%)	Minor improvement
Outlier Removal (IQR)	91% (+6%)	Major Improvement
Feature Scaling (Min-Max)	92% (+7%)	Useful for a gradient-based model
Proposed combined technique	95% (+15%)	Optimal and superior model

The performance of the proposed combined preprocessing techniques and other simple preprocessing steps is analyzed, and the results are presented in Table 4. The analysis indicates that each preprocessing step uniquely improves model predictive performance. The missing data imputation step is crucial for addressing missing values in the input, boosting model accuracy to 89%. The outlier removal step removes unwanted regions from the input images using IQR filters, further improving model accuracy to 91%. The min-max normalization method is then applied to scale the features. This step ensures that features like rainfall and tide height contribute equally. It results in a 7% improvement, bringing the total to 92%. By combining these techniques, the proposed model has achieved 95% accuracy in predicting floods from the input time-series and image data.

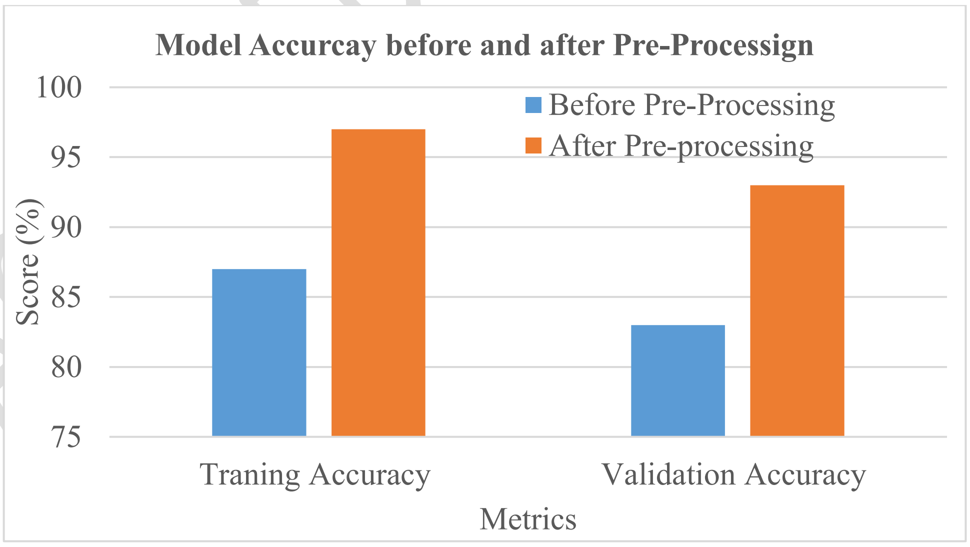
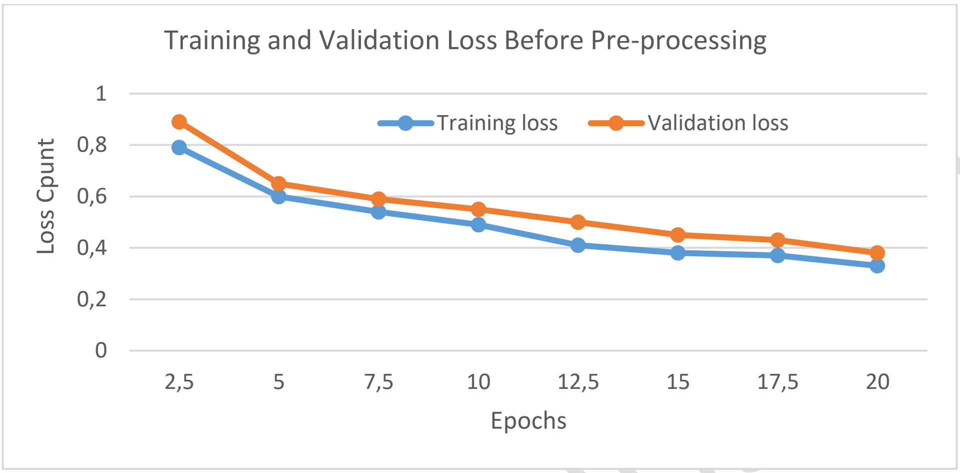
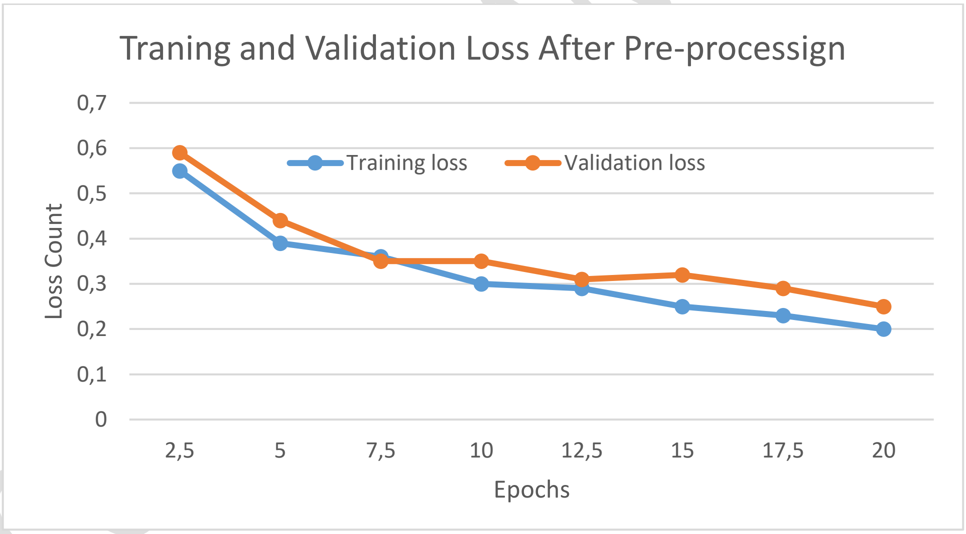


Figure-3 Model Accuracy Result

Figure-3 shows that, after applying preprocessing techniques, both the training and validation results of the proposed model improve. Before pre-processing, the model achieved a training accuracy of about 87% and a validation accuracy of around 83%. However, these accuracies were limited due to outliers, noise, and missing data, which affected generalization. After applying preprocessing methods such as feature scaling, KNN imputation, and IQR-based outlier removal, training and validation accuracies increased to approximately 97% and 93%, respectively. These results suggest that the proposed model not only learns from the training data but also generalizes well to new, unseen data, enhancing reliability and reducing overfitting. To stabilize the learning process and improve predictive accuracy, preprocessing is essential and yields consistent gains during model training and validation.



(A) Training and Validation Loss Before Pre-processing



(B) Training And Validation Loss After Pre-Processing

Figure-4: Training And Validation Loss Before and After Pre-Processing

Figure 4 (a) and (b) represent how training and validation losses are reduced by applying pre-processing steps, and also enhance the model stability. Before applying preprocessing, the training loss was around 1.2 and then declined slowly to approximately 0.35, whereas the validation loss fluctuated between 0.4 and 0.5; these values indicate noise and poor generalization. After applying the preprocessing techniques, the training loss initially drops to around 0.9 and then decreases to around 0.25; the validation loss follows a similar pattern and stabilizes at around 0.18. The results for the training and validation losses after pre-processing indicate that the model avoids overfitting, generalizes well, and converges quickly; moreover, it improves accuracy by around 95%.

Table 5: Accuracy Comparison

Author Name and Year	Model Used	Accuracy	Pre-processing Technique
Kumar et al. (2023)	SVM	76%	No
Patel and Singh (2023)	Random Forest	82%	Basic Scaling
Proposed	DL + Pre-processing	95%	KNN, IQR, Scaling

The accuracy comparison between the proposed and existing models is presented in Table 5. The comparison indicates that the proposed DL+ preprocessing technique outperforms other models, achieving 95% accuracy in predicting floods from input data collected in the coastal region. The proposed model has achieved ~13% Higher accuracy than existing models. The overall results show that the proposed model outperforms the other models in predicting floods.

Table 6: Performance Comparison

Author / Model	Accuracy	Precision	Recall	F1-Score
Kumar et al. (2023) – SVM	76%	0.75	0.76	0.75
Patel & Singh (2023) – Random Forest	82%	0.81	0.82	0.81
Proposed – DL + Pre-processing	95%	0.94	0.95	0.95

Table-6 compares the classification performance over three models using Accuracy, precision, recall, and F1-Score. The SVM model achieves moderate performance, with balanced precision and recall near 76%, indicating reasonable but limited predictive capability without preprocessing. The random forest model shows improved performance, achieving 82% accuracy, and reflects better classification stability and reduced variance. The proposed Deep learning model is combined with the preprocessing methods such as KNN imputation, IOR outlier removal, and scaling. This is a crucial performance of about 95% across analysis metrics. This high precision results in fewer false positives, and a high recall shows strong detection capability. The F1-score confirms balanced performance, demonstrating that combining preprocessing with deep learning improves classification efficiency and reliability compared to conventional machine learning approaches.

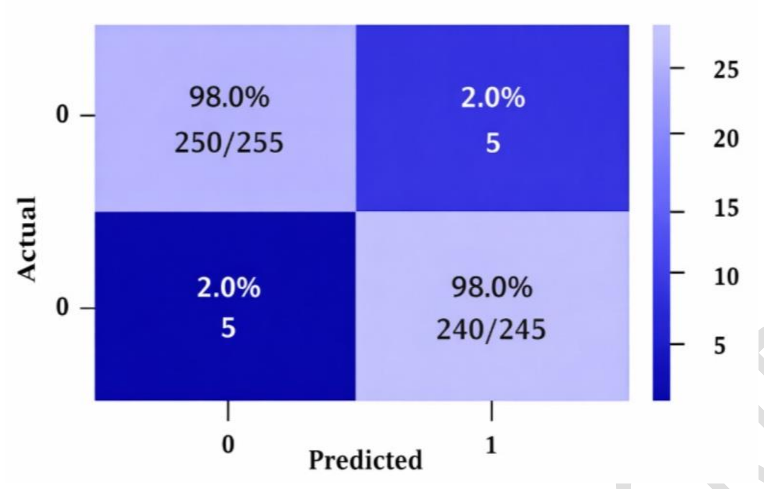


Figure-5 Confusion matrix

The confusion matrix shown in Figure-5 quantifies the string classification performance of the proposed model, as used in the percentage analysis. The top-left cell denotes the true negatives (98.0%), indicating that 98% of the actual negative samples are correctly classified. The bottom-right cell represents the true positive (98.0%), meaning 98% of actual positive cases were identified. The off-diagonal cells represent misclassifications, with 0.2% false positives and 2.0% false negatives. This is a low error, underscored by the model's strong dependence. This high true prediction percentage, which confirms excellent sensitivity and specificity, yields minimal false predictions and indicates minimized classification bias. Moreover, the matrix shows balanced performance across classes and supports the high accuracy, precision, recall, and F1-score reported by the deep learning model.

5. Conclusion

This paper presented a systematic preprocessing process tailored to multimodal flood datasets, incorporating spatial, temporal, and spatiotemporal dependencies derived from time-series and image data collected across various coastal regions in India. A separate set of preprocessing methods is used for spatial and temporal data, processed individually and sequentially to improve data quality. All techniques are implemented separately in Python using comprehensive datasets, and the results are verified. The results demonstrate that the methods significantly enhance data quality by converting raw data into a standardized format, thereby improving the performance of flood prediction and forecasting models. The performance is evaluated by comparing the prediction accuracy before and after preprocessing is applied to the data. The final data analytics process uses ML and DL algorithms. Security can be achieved with the proposed model by integrating multimodal preprocessing with a deep learning-based flood prediction framework. At the beginning of data security, data are enhanced through missing-value imputation (KNN), outlier removal (IQR), normalization, and stationarity checks to eliminate noise, inconsistencies, and abnormal spikes from IoT sensors. Image insecurity involves resizing, grayscale conversion, CLAHE, noise reduction, and channel standardization to prevent distorted or misleading visual patterns. The combination of preprocessing techniques yield 95% prediction accuracy, hence enabling secure, dependable, and timely flood risk detection for coastal regions. The proposed model achieves reliability through comprehensive preprocessing, including KNN imputations, IQR-based outlier removal, and min-max standardization, to improve data quality. This merging of multimodal data, such as time and image data, enhances the complementary information for flood prediction. The preprocessing and model achieve improved training (97%) and validation (93%) accuracy, indicating better generalization and reduced overfitting. Finally, the stable loss is minimized, achieving a final prediction accuracy of 95%. The proposed model provides reliable and strong flood forecasting.

Future research will involve extending the suggested preprocessing framework into a fully automated, adaptive system capable of processing heterogeneous, large-scale multimodal flood data in real time. They will also be supplemented with additional data sources to increase spatial and temporal coverage, such as IoT-based water-level sensors, UAV imagery, crowd-sourced reports, and high-resolution climate projections. To improve the framework's generalization across coastal and inland areas, dynamic feature selection and adaptive normalization will be employed. Also, explainable AI (XAI) methods, such as SHAP and LIME, will be used to measure the effects of preprocessing on prediction results. The model will also be tested under varying climatic conditions to support global flood predictions, multi-hazard risk appraisal, and early-warning decision support systems.

References

1. Tariq, M. A. U. R., & Van De Giesen, N. (2012). Floods and flood management in Pakistan. *Physics and Chemistry of the Earth, Parts A/B/C*, *47*, 11-20.
2. Tsakiris, G. J. N. H. (2014). Flood risk assessment: concepts, modelling, applications. *Natural Hazards and Earth System Sciences*, *14*(5), 1361-1369.
3. Balica, S. F., Wright, N. G., & Van der Meulen, F. (2012). A flood vulnerability index for coastal cities and its use in assessing climate change impacts. *Natural hazards*, *64*, 73-105.
4. Khan, A., & Chatterjee, S. (2018). *Coastal risk assessment: A comprehensive framework for the bay of bengal*. Springer International Publishing.
5. Small, C., & Nicholls, R. J. (2003). A global analysis of human settlement in coastal zones. *Journal of coastal research*, 584-599.
6. Sun, D., Yang, T., Li, S., Goldberg, M., Kalluri, S., Helfrich, S., ... & Miralles-Wilhelm, F. (2024). Hazard or Non-Hazard Flood: Post Analysis for Paddy Rice, Wetland, and Other Potential Non-Hazard Flood Extraction from the VIIRS Flood Products. *ISPRS Journal of Photogrammetry and Remote Sensing*, *209*, 415-431.
7. Tiwari, D., Bhati, B. S., Nagpal, B., Sankhwar, S., & Al-Turjman, F. (2021). An enhanced intelligent model: To protect marine IoT sensor environment using ensemble machine learning approach. *Ocean Engineering*, *242*, 110180.
8. Dai, W., Tang, Y., Zhang, Z., & Cai, Z. (2021). Ensemble learning technology for coastal flood forecasting in internet-of-things-enabled smart city. *International Journal of Computational Intelligence Systems*, *14*(1), 166.

9. Yang, G., Li, Y., Guo, X., Shi, B., Wu, W., & Cheng, X. (2025). Innovative dynamic evaluation and classification method for marine atmospheric corrosion based on corrosion sensors and machine learning. *Materials Today Communications*, *42*, 111558.
10. Gambín, Á. F., Angelats, E., González, J. S., Miozzo, M., & Dini, P. (2021). Sustainable marine ecosystems: Deep learning for water quality assessment and forecasting. *IEEE access*, *9*, 121344-121365.
11. Drogkoula, M., Kokkinos, K., & Samaras, N. (2023). A comprehensive survey of machine learning methodologies with emphasis in water resources management. *Applied Sciences*, *13*(22), 12147.
12. Karthik, S., Surendran, R., Kumar, G. V., & Srinivasulu, S. (2025). Flood prediction in Chennai based on extended elman spiking neural network using a robust chaotic artificial hummingbird optimizer. GLOBAL NEST JOURNAL, 27(4).
13. Babu, T., Selvanarayanan, R., Thanarajan, T., & Rajendran, S. (2024). Integrated early flood prediction using sentinel-2 imagery with VANET-MARL-based deep neural RNN. Global NEST Journal, 26(10).
14. Sundarapandi, A. M. S., Navaneethakrishnan, S. R., Hemlathadhevi, A., & Rajendran, S. (2024). A Light weighted dense and tree structured simple recurrent unit (LDTSRU) for flood prediction using meteorological variables. Global NEST Journal, 26(8).
15. Qiu, Y., Shi, X., & He, X. (2026). Enhancing flood prediction in the Lower Mekong River Basin by scale-independent interpretable deep learning model. Environmental Impact Assessment Review, 116, 108130.
16. ShravanKumar, S. M., Karthick, A., Priya, A. K., Mohanavel, V., & Muthusamy, S. (2026). Remote Sensing and Artificial Intelligence in Flood Prediction: Progress, Challenges, and Prospects. Archives of Computational Methods in Engineering, 1-32.
17. Zhu, M., Wang, J., Yang, X., Zhang, Y., Zhang, L., Ren, H., ... & Ye, L. (2022). A review of the application of machine learning in water quality evaluation. Eco-Environment & Health, 1(2), 107-116.
18. Tselemponis, A., Stefanis, C., Giorgi, E., Kalmpourtzi, A., Olmpasalis, I., Tselemponis, A., ... & Constantinidis, T. C. (2023). Coastal water quality modelling using E. coli, meteorological parameters and machine learning algorithms. *International Journal of Environmental Research and Public Health*, *20*(13), 6216.
19. Essamlali, I., Nhaila, H., & El Khaili, M. (2024). Advances in machine learning and IoT for water quality monitoring: A comprehensive review. *Heliyon*.
20. Chansi, Hadwani, K., & Basu, T. (2025). Artificial intelligence-, machine learning-and Internet of Things-based sensors for detection of marine pollutants. In *Sensors for Marine Biosciences: Next-generation sensing approaches* (pp. 6-1). Bristol, UK: IOP Publishing.
21. Mangukiya, N. K., & Sharma, A. (2022). Flood risk mapping for the lower Narmada basin in India: a machine learning and IoT-based framework. *Natural Hazards*, *113*(2), 1285-1304.
22. Bhardwaj, A., Dagar, V., Khan, M. O., Aggarwal, A., Alvarado, R., Kumar, M., ... & Proshad, R. (2022). Smart IoT and machine learning-based framework for water quality

assessment and device component monitoring. *Environmental Science and Pollution Research*, *29*(30), 46018-46036.

23. Alprol, A. E., & Khairy, H. M. (2025). Machine-Learning Application for Water Pollution Control and Water Treatment. In *Modelling and Advanced Earth Observation Technologies for Coastal Zone Management* (pp. 261-284). Cham: Springer Nature Switzerland.
24. Motta, M., de Castro Neto, M., & Sarmento, P. (2021). A mixed approach for urban flood prediction using Machine Learning and GIS. *International journal of disaster risk reduction*, *56*, 102154.
25. Ogbuene, E. B., Eze, C. A., Aloh, O. G., Oroke, A. M., Udegbunam, D. O., Ogbuka, J. C., ... & Okolo, O. J. (2024). Application of Machine Learning for Flood Prediction and Evaluation in Southern Nigeria. *Atmospheric and Climate Sciences*, *14*(3), 299-316.
26. Kumar, V., Azamathulla, H. M., Sharma, K. V., Mehta, D. J., & Maharaj, K. T. (2023). The state of the art in deep learning applications, challenges, and future prospects: A comprehensive review of flood forecasting and management. *Sustainability*, *15*(13), 10543.
27. Patel, A., & Yadav, S. M. (2023). Development of flood forecasting and warning system using hybrid approach of ensemble and hydrological model for Dharoi Dam. *Water Practice & Technology*, *18*(11), 2862-2883.