CHAPTER 1

INTRODUCTION

Tropical cyclones are dangerous because they can bring strong winds, flooding rain, and destructive storm surges that can submerge low-lying coastal regions. Failure to forecast these cyclones could lead to more occurrences and create a warmer climate scenario. Therefore, it is crucial to improve cyclone prediction and estimate their impact.

Aircraft detection was first employed as a reliable method of keeping track of the location, intensity, and stage of development of TCs in the 1940s. The very first technique, known as the systematic approach, was developed by Dvorak and is still used as a basis in many operational centers. However, Dvorak's approach involves frequent human interactions and complicated decision rules for various physical and environmental conditions, which can limit its efficiency. The deviation-angle variance technique (DAVT) is another approach that has shown less accuracy and efficiency.

In contrast to the techniques, this project makes use of a CNN model in order to estimate the intensity of tropical cyclones (TCICENet) by making use of images from an infrared geostationary satellite. The TCICENet model takes on infrared satellite images as its input. This model comprised two modules, TC intensity classification(TCIC) module and the TC intensity estimation(TCIE) module, which performs a regression task for three TC categories.

Integration of IoT-Based Real-Time Flood Alert Subsystem

In addition to cyclone intensity estimation using image processing and deep learning techniques, the project has been enhanced with an IoT-based real-time flood alert subsystem. This subsystem uses a compact hardware model built using ESP8266 Wi-Fi module, a water-level sensor, a buzzer, and an LED indicator. The objective of integrating the IoT unit is to provide an early warning mechanism when sudden water level rise or localized flooding is detected.

The ESP8266 continuously monitors the sensor data and communicates with the cloud using MQTT protocol. When a predefined flood threshold is crossed, the system immediately sends an alert command to the hardware to activate the buzzer and LED, ensuring local on-ground warning. Additionally, a parallel alert notification is pushed to a mobile application developed using Flutter, allowing remote users or authorities to receive alerts instantly.

This integration ensures that along with large-scale cyclone prediction, localized flood events can also be detected and reported in real time, thereby improving overall disaster response capability.

1.1 Motivation

Tropical cyclones, also known as hurricanes or typhoons, can cause significant damage and loss of life when they make landfall, making accurate classification and intensity estimation critical for disaster preparedness and response. Currently, tropical cyclones are classified and their intensities estimated based on satellite imagery, which requires human analysis and interpretation. This process can be time-consuming and prone to error, particularly in areas with limited access to trained meteorologists. By developing an automated system that can accurately classify and estimate the intensity of tropical cyclones, we can improve our ability to respond to these natural disasters and potentially save lives.

1.2 Aim of our Project

The aim of the project is to develop a system that can automatically classify and estimate the intensity of tropical cyclones using image processing techniques and convolutional neural networks (CNNs). The specific objectives of the project may include:

- Collecting and preprocessing satellite imagery data of tropical cyclones
- Designing and training a CNN model to classify tropical cyclones into different categories (such as tropical depression, tropical storm, hurricane, etc.)
- Designing and training a CNN model to estimate the intensity of tropical cyclones
- Evaluating the performance of the developed models on a test dataset and comparing it with existing methods
- Developing a user-friendly interface for the system to allow easy access for meteorologists and disaster response personnel

1.3 Need for the proposed system

The need for the proposed system arises from the fact that accurate classification and intensity estimation of tropical cyclones is crucial for disaster management and response. The current methods of classification and intensity estimation rely heavily on human interpretation of satellite imagery, which can be time-consuming and prone to errors.

CHAPTER 2

LITERATURE SURVEY

Wangdi Du et al., [1] presented a tropical cyclone risk estimation on the basis of information diffusion theory. The theory of probability statistics, which was previously used to assess disaster risk, was proven to be ineffective and mostly focused on statistical risk. The data analysis is fairly simple, and the risk assessment methodology which is based on information diffusion theory used in this work requires less data than previous methodologies. By making use of the best track data on CMA tropical storms, the total number of tropical cyclones which occurred in every Chinese province from 1949 to 2013 was calculated. Data sets were obtained from the CMA tropical cyclone information centre. The information diffusion theory was used in this work to directly estimate the probability of tropical cyclone occurrence in each province, yielding a risk assessment value for the probability of tropical cyclone occurrence in each province. Using this risk assessment analysis approach with that of genetic analysis to enhance the degree of risk analysis can help in achieving the objective of disaster reduction.

A. Fore et al.,[2] proposed SMAP mission's objective to find soil moisture on land. With a 9 km resolution and an 8-day return duration, the SMAP mission's hybrid active and passive L-band microwave instrument measures soil moisture across land. In this work, the Automated Tropical Cyclone Forecasting (ATCF) system B-deck files are compared with the SMAP cyclone sizes in order to update the SFMR analysis. First, the study validates the size of the cyclone and then validates the intensity. The magnitude of tropical cyclones is estimated using SMAP data using an algorithm that exclusively uses best-track sites. Once the cyclone's size has been confirmed, in order to improvise the hurricane track predictions and forecast the storms, the SFMR airborne instrument is used. The dropsondes are being used to confirm the speeds of the wind up to category 5 wind speed (70 m/s). The SMAP wind speeds eventually closely resemble the SFMR wind speeds, with an STD between 3 and 5 m/s and a minor positive bias of roughly 1 m/s. The study's findings show that SMAP can offer useful data on size and intensity for operational tropical cyclone forecasting.

Chong Wang et al.,[3] proposed Satellite-based technologies that use a neural network to predict the direction of a typhoon in a matter of seconds. The CNN model was trained using roughly 2250 infrared pictures from 97 typhoon cases between 2015 and 2018. Convolutional, pooling, and fully linked layers are all included in the CNN model, which gives it objectivity and speed. The fully connected layers learn picture features after they have been entirely lowered by the pooling layers and extracted by the convolutional layers. The Himawari-8 geostationary satellite's Advanced Himawari Imager (AHI) on Channels 13 and 15 was used in the study. Every ten minutes, AHI is able to deliver infrared and visible images thanks to its 16 spectral channels. Using information on the optimal course for a typhoon provided by the Chinese Meteorological Agency, typhoon photographs were labelled and their future movement direction was predicted(CMA).

Hao Hu and Fuzhong Weng [4], having accurate data on the depth and region of tropical cyclones (TCs) is crucial for climate prognostication and warning. This study aims to forecast the position and depth of two storms that occurred in 2018. It does this by using the hydrostatic stability equation and the thermal data recovered from microwave sounding equipment. Four tests were done to see how well the algorithm worked with different instruments. The results demonstrated that the SD1DVAR retrieval technique significantly reduced the area and depth estimation error by approximately 52.79% and 36.91% in comparison to that of MIRS products while using the same instruments. Additionally, including 118 GHz channels led to a reduction in depth errors by approximately 37.2%. Moreover, this SD1DVAR-based method performed consistently between ATMS and CMWS [16]. Further effort to develop a worldwide TC region and depth detection dataset with greater temporal resolution will concentrate on additional devices.

Yu Xie et al., [5] have proposed a MVFF network to estimate the TC intensity. The MVFF network is trained and calibrated using the TC warm core temperature anomalies obtained from the S-NPP ATMS. The MVFF employs a multi-view structure that can make full use of the complementary information of different views, and also uses FEF Chong Wang et al.,[3] proposed Satellite-based technologies that use a neural network to predict the direction of a typhoon in a matter of seconds. The CNN model was trained using roughly 2250 infrared pictures from 97 typhoon cases between 2015 and 2018. Convolutional, pooling, and fully linked layers are all included in the CNN model, which gives it objectivity and speed. The fully connected layers learn picture features after they have been entirely lowered by the pooling layers and extracted by the convolutional layers. The Himawari-8 geostationary satellite's

Advanced Himawari Imager (AHI) on Channels 13 and 15 was used in the study. Every ten minutes, AHI is able to deliver infrared and visible images thanks to its 16 spectral channels. Using information on the optimal course for a typhoon provided by the Chinese Meteorological Agency, typhoon photographs were labelled and their future movement direction was predicted(CMA).

Hao Hu and Fuzhong Weng [4], having accurate data on the depth and region of tropical cyclones (TCs) is crucial for climate prognostication and warning. This study aims to forecast the position and depth of two storms that occurred in 2018. It does this by using the hydrostatic stability equation and the thermal data recovered from microwave sounding equipment. Four tests were done to see how well the algorithm worked with different instruments. The results demonstrated that the SD1DVAR retrieval technique significantly reduced the area and depth estimation error by approximately 52.79% and 36.91% in comparison to that of MIRS products while using the same instruments. Additionally, including 118 GHz channels led to a reduction in depth errors by approximately 37.2%. Moreover, this SD1DVAR-based method performed consistently between ATMS and CMWS [16]. Further effort to develop a worldwide TC region and depth detection dataset with greater temporal resolution will concentrate on additional devices.

Yu Xie et al., [5] have proposed a MVFF network to estimate the TC intensity. The MVFF network is trained and calibrated using the TC warm core temperature anomalies obtained from the S-NPP ATMS. The MVFF employs a multi-view structure that can make full use of the complementary information of different views, and also uses FEF module to realize effective fusion of features of different scales. Simulations showed that the MVFF network provides better intensity estimation in term of both MAE and RMSE.

Fan Meng et al., [6] have designed a two-branch convolutional neural network model to explore the role of deep learning in cyclone recognition. A total of 10 types of cyclone phenomena are involved. The water vapor map and the lowest sea level pressure are used as model inputs. The results reveal that the model can accurately identify various types of cyclones, and can also learn the characteristics of different cyclones and classify them. However, it is undeniable that our model is not very accurate in classifying different cyclones, and it is easy to be confused with adjacent categories. The confusion between the types of cyclones is firstly related to the similar characteristics of adjacent types of cyclones, and it is also difficult for human experts to distinguish accurately. Secondly, this is related to the imbalance of the number of samples of different categories in the data set used.

Shuhan Yang et al., [7] have made use of the Bagging-IBK & LMT models in the hurricane intensity forecast model. The present strength forecasts, which are based on statistical techniques and storm track patterns, are inadequate. Therefore, data mining methods can be utilized to forecast changes in hurricane intensity, and several researchers have employed dynamic statistical models and prediction models with neural networks. Although tropical cyclone prediction models apply data mining methods to hurricane intensity, they cannot meet actual application requirements. Yang used the RI technique (Rapid Intensification) to analyse this problem and evaluate whether hurricanes were fast intensifying. To get beyond the limitations of a single classifier, the Bagging approach was utilised to combine numerous weak classifiers. The selected classifiers expanded the model into Bagging classifier sequences with the number which is the same, and the model uses techniques like REPTree, Logistic Model Tree, C4.5, and KNN. The training method was randomly sampled using the Bootstrap method. Once prediction functions were graded based on the classified instance, the top m prediction functions were chosen to be voted as reliable prediction outcomes.

Ana Raquel Carmo et al.,[8] have proposed a system based on topological patterns. The experimental findings show that the proposed method not only successfully identifies TCs but also has the capacity to precisely pinpoint their centres and compete with current manual and automated intensity estimation methods. Convolutional neural networks are the typical DL network model used with picture data. In this work, two alternative CNNs trained well on the datasets of ImageNet, ResNet50 and MobileNetV2, are put to the test. Gradient-based class activation maps have been used to display the internal depictions that the CNN's have learned and for the better understanding of their characteristics (Grad-CAM).

Guangchen Chen et al., [9] proposed a new semisupervised deep learning framework for accurate tropical cyclone classification, which only requires a small number of labeled samples. The proposed method updates training set by selecting the samples with reliably predicted labels based on a specially designed hybrid similarity measurement. It trains two CNN's in a semisupervised and iterative manner. Experiments show that the proposed method is significantly better than several popular classification methods.

Chong Wang et al., [10] proposed a system to gauge the strength of tropical cyclones in the Northwestern Pacific by using high-frequency brightness temperature information from

Himawari-8 satellite.Based on high-frequency Himawari-8 brightness temperature dattyphoonsintensity of typhoon over the Northwestern Pacific was successfully estimated using the deep convolutional neural network. The best accuracies of top-1 and top-2 classification are 81.4% and 93.3%, respectively. More typhoon data are necessary to balance the number of each category samples. In the present study, they only use the bright temperature data from only one infrared band with resolution of 8 km. In the near future, satellite data from different bands with higher spatial resolution should be used to optimize the neural network structures.

Xingxing Yu et al.,[11] proposed a system to estimate the TC wind speed by classification and successive regression. The Tensor CNN provides a unitary and concise representation of CNN and Tucker decomposition, combines their strengths and classifies the cyclones. The class-wise regression model makes the most of the Tensor CNN and estimates the TC wind speed accurately. The experiments show that the proposed framework achieves the state-of- the-art performance on TC wind speed estimation.

Zhao Chen et al., [12] proposed a system FY-4 MSI to approximate the intensity of tropical cyclones (TC) by making use of machine learning-based methodology. The three essential components of the framework are fusion, band-wise classification, and usable band determination. The authors have chosen three well-known machine learning techniques for band-wise classification, which are Multinomial Logistic Regression (MLR), Support Vector Machine (SVM), and Back-Propagation Neural Network (BPNN). The first to sixth bands, which are largely in the visible and near-infrared spectrum, communicate a small amount of information for classification when collected at night along the southeast coast of China and the western Pacific. Both during the day and at night, the remaining eight infrared channels perform admirably. This suggests that a distinct category should be used to group the band pictures. Because of this, the authors employ selectors created especially for the FY-4 data. To decide if the band picture may be supplied to the appropriate band classifier, one selector is utilized for each band. Fusion is used to combine the outcomes from each classifier following band-wise classification. The features of the FY-4 MSI, according to the authors, automatically evaluate the cyclone intensity and perform band-wise classification.

CHAPTER 3

SYSTEM REQUIREMENTS AND SPECIFICATION

3.1 Software Requirement and Specification

Software Requirements Specification (SRS) provides an overview of the entire SRS with purpose, scope, definitions, acronyms, abbreviations, references and overview of the SRS. A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform the various gestures and determining its accuracy. The SRS is a requirements specification for a software system, is a description of the behavior of a system to be developed and may include a set of use cases that describe interactions the users will have with the software. In addition it also contains non- functional requirements. Nonfunctional requirements impose constraints on the design or implementation. Software requirements specification establishes the basis for agreement between customers and contractors or suppliers on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. An SRS minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations. Parameters such as operating speed, response time, availability, portability, maintainability, footprint, security and speed of recovery from adverse events are evaluate.

3.2 Functional Requirements

A functional requirement defines a function of a software system or its components. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability).

3.3 Non-Functional Requirements

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. Non-functional requirements define how a system is supposed to be. Non-functional requirements are in the form of "system shall be <requirement>", an overall property of the system as a whole or of a particular aspect and not a specific function. The system's overall properties commonly mark the difference between whether the development project has succeeded or failed. Non-functional requirements are often called “quality attributes” of a system.

- **Security**

  The system should have security measures in place to ensure the confidentiality, integrity, and availability of sensitive data, such as user data and system logs.

- **Concurrency and Capacity**

  System should be able to handle multiple computations executing simultaneously, and potentially interacting with each other.

- **Performance**

  Performance is generally perceived as a time expectation. This is one of the most important considerations especially when the project is in the architecture phase.

- **Reliability**

  It is necessary to ensure and notify about the system transactions and processing as simple as keep a system log will increase the time and effort to get it done from the very beginning. Data should be transferred in a reliable way and using trustful protocols.

- **Maintainability**

  Well-done system is meant to be up and running for long time. Therefore, it will regularly need preventive and corrective maintenance. Maintenance might signify scalability to grow and improve the system features and functionalities.

- **Usability**

- End user satisfaction and acceptance is one of the key pillars that support a project success. Considering the user experience requirements from the project conception is a win bet, and it will especially save a lot of time at the project release, as the user will not ask for changes or even worst misunderstandings.
- **Documentation**

  All projects require a minimum of documentation at different levels. In many cases the users might even need training on it, so keeping good documentation practices and standards will do this task spread along the project development; butas wel this must be established since the project planning to include this task in the list.

3.3.1 Software Requirements

- **Jupyter Notebook:** Jupyter Notebook is an open-source software program this is an interactive computational environment, wherein you could integrate code execution, wealthy text, mathematics, plots, and wealthy media, it's far used for modifying and walking the application, additionally it turned into first-rate appropriate for us to broaden our challenge
- **VSCode:** Visual Studio Code is a code editor that may be used with a whole lot of programming languages, it's far a code editor made with the aid of using Microsoft for Windows, Linux, and macOS. Features encompass guide for debugging, syntax highlighting, sensible code completion, snippets, code refactoring, and embedded Git
- **Anaconda:** Anaconda is an open-source distribution of Python and R programming languages for scientific computing, data science, and machine learning. It provides an integrated environment for data analysis and includes over 1,500 data science packages out-of-the-box. It is a powerful tool for data scientists and machine learning practitioners, providing a complete and integrated environment for developing, testing, and deploying machine learning models
- **Streamlit:** It is a Python framework that enables developers to build web applications for machine learning and data science. Streamlit allows developers to create interactive web applications with minimal effort by providing easy-to-use widgets for user input and output display. It also enables integration with popular machine learning and data science libraries like TensorFlow, PyTorch, and Pandas, among others.

3.3.2 Hardware Requirements

- CPU: A multi-core CPU with at least 4 cores, such as an Intel Core i5 or i7
- GPU: A graphics processing unit (GPU) can significantly accelerate the training process for deep learning models. A mid-range GPU with at least 4GB of VRAM, such as an Nvidia GTX 1660, may be sufficient.
- RAM: Atleast 4GB of RAM is recommended
- HDD: A minimum of 10GB is recommended.

CHAPTER 4

SYSTEM ANALYSIS

4.1Existing System

Since 1940, a lot of experiments and systems that have been proposed to estimate the intensity of a tropical cyclone. Existing TC intensity estimation models rarely consider the intensity grade classification before intensity estimation, although some researchers have recently divided the TC intensity grades into many categories using CNNs. Due to the high misclassification probability when directly dividing TC intensity grades into too many categories, the accuracy of the intensity estimation is bound to be affected. Some of the existing system includes:

- Microwave data from polar-orbiting and geostationary satellite imagery are currently used for TC intensity estimation. Geostationary satellites have a higher temporal resolution and stable image quality, but cannot obtain near-surface structure of TC.

- The Dvorak technique and its improved versions are commonly used for TC intensity estimation using infrared satellite images, but they rely on the experience of the forecaster and may not be suitable for estimating the intensity of weak TCs.

- Traditional machine learning techniques such as multivariate linear regression, K-nearest neighbors algorithm, multilayer perceptron, SVM, and RVM have been used to estimate TC intensity, but they mainly focus on the manual extraction of statistical or structural features that are largely dependent on human subjectivity and experience.

- Convolutional neural networks (CNNs) of deep learning have been used to extract deep features directly from infrared satellite images without the need for manual feature extraction. Recent studies have used CNN-based models for TC intensity estimation as a regression task, but these single models may not fully cover TC changes, especially when the TC samples of different intensities are not balanced.

4.2 Proposed System

The proposed system, called TCICENet, is a CNN model designed for estimating the intensity of tropical cyclones (TCs) from infrared geostationary satellite images. It consists of two modules: the TC intensity classification (TCIC) module and the TC intensity estimation (TCIE) module. The TCIC module performs the classification of TC intensity grade, which is a challenging task because TCs are not rigid bodies, and their structure changes greatly in different development stages with continuous rotation and translation. The TCIE module estimates the TC intensity from three TC categories as a regression task.

IoT-Based Flood Detection Subsystem

The enhanced system now includes an IoT module responsible for detecting sudden water level rise. The subsystem consists of the following components:

1. **ESP8266 NodeMCU** – acts as the central controller and MQTT client.
2. **Water-Level / Ultrasonic Sensor** – measures the distance between sensor and water surface.
3. **Buzzer** – provides audible alert during flood condition.
4. **LED Indicator** – provides visual alert for quick identification.
5. **MQTT Cloud Broker** – facilitates communication between ESP8266, backend system, and mobile app.
6. **Flutter Mobile Application** – receives push notifications when flood level crosses the threshold.

Working Principle

- Sensor continuously measures water level.
- ESP8266 publishes sensor data to an MQTT topic at regular intervals.
- When threshold is crossed, deep learning subsystem or local logic sends MQTT command:
  "FLOOD=1"
- ESP8266 receives the command and immediately turns ON buzzer and LED.
- A notification is also sent to the Flutter app to alert the user.

This IoT unit acts as a real-time extension to cyclone intensity estimation.

CHAPTER 5

SYSTEM DESIGN

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

5.1 High Level Design

High-level design (HLD) explains the architecture that would be used for developing a software product. The architecture diagram provides an overview of an entire system, identifying the main components that would be developed for the product and their interfaces. The HLD uses possibly nontechnical to mildly technical terms that should be understandable to the administrators of the system.

5.2 System Architecture

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

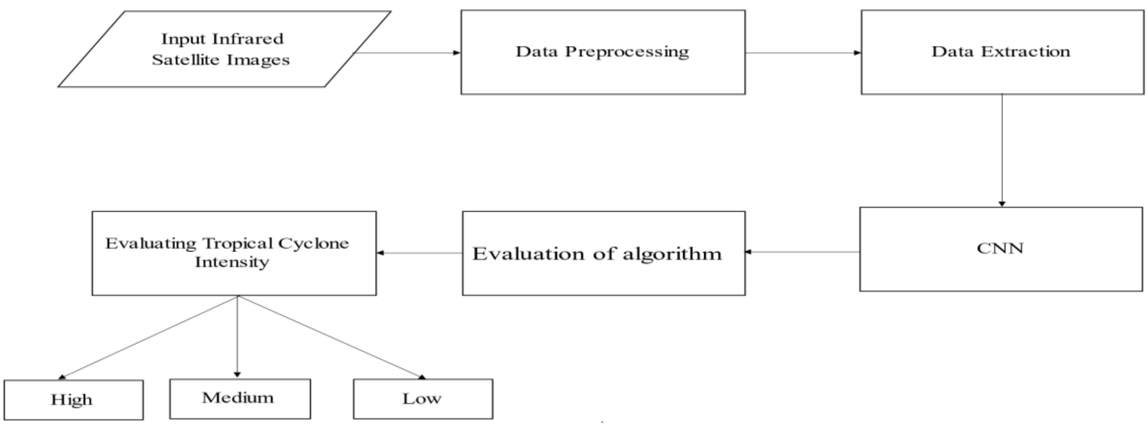


Fig: 5.1: System Architecture using CNN

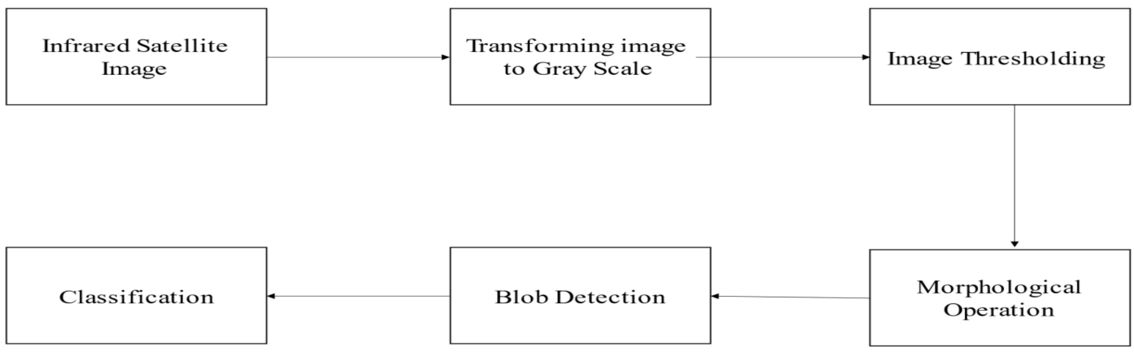


Fig: 5.2: System Architecture using Image Processing

The architecture of the proposed system is shown in the Fig 5.2.1 and 5.2.2. The system architecture can be divided into two different architecture,

According to the deep learning architecture,

- Image acquisition: It is the very first step that requires capturing an image with the help of camera.
- Data Collection: Collect a large dataset of high-resolution infrared satellite images of tropical cyclones with corresponding intensity labels.
- Data Preprocessing: Preprocess the data to remove any noise, artifacts, or other inconsistencies in the images. This step may involve applying filters, normalization, and scaling to the data.
- Feature Extraction: Use a convolutional neural network (CNN) to extract features from the preprocessed images. CNNs are particularly good at learning spatial features in images, which makes them well-suited for this task
- Model Training: Train a deep learning model, such as a neural network, on the extracted features to predict the intensity of tropical cyclones from the infrared satellite images. This step may involve using various algorithms such as regression, classification, or ensemble methods.
- Model Evaluation: Evaluate the performance of the trained model using a different evaluation metrics.
- Evaluating Intensity: Estimating the intensity and categorizing it. According to the deep learning architecture,

- Image acquisition: It is the very first step that requires capturing an image with the help of camera.
- Transforming an image to grayscale: Grayscale images have only one color channel, instead of three (red, green, and blue) in a typical color image. This can simplify the image processing task and reduce the computational complexity, while still retaining the relevant information in the image.
- Image thresholding: It is a step in image processing that involves segmenting an image into foreground and background regions based on the intensity values of the pixels. The basic idea behind thresholding is to convert a grayscale or color image into a binary image, where each pixel is either black or white, depending on whether its intensity value is above or below a threshold value.
- Morphological operations: These are a class of image processing techniques that operate on the shape and structure of objects in an image. These operations are commonly used for tasks such as noise reduction, edge detection, segmentation, and feature extraction.
- Blob detection: Used to identify and locate regions of an image that have a similar intensity or color value. These regions are referred to as "blobs" or "connected components". Blob detection can be used for various image analysis tasks, such as object recognition, tracking, and segmentation.
- Classification: This step involves assigning a label or category to each pixel or region in an image based on its characteristics or features. Image classification can be used for various applications, such as object recognition, scene analysis, and image retrieval.

5.3 Use Case Diagram

Figure 5.3.1 shows the graphic depiction of the interactions among the elements of a system. Use cases will specify the expected behavior, and the exact method of making it happen. Use cases once specified can be denoted both textually and visual representation.

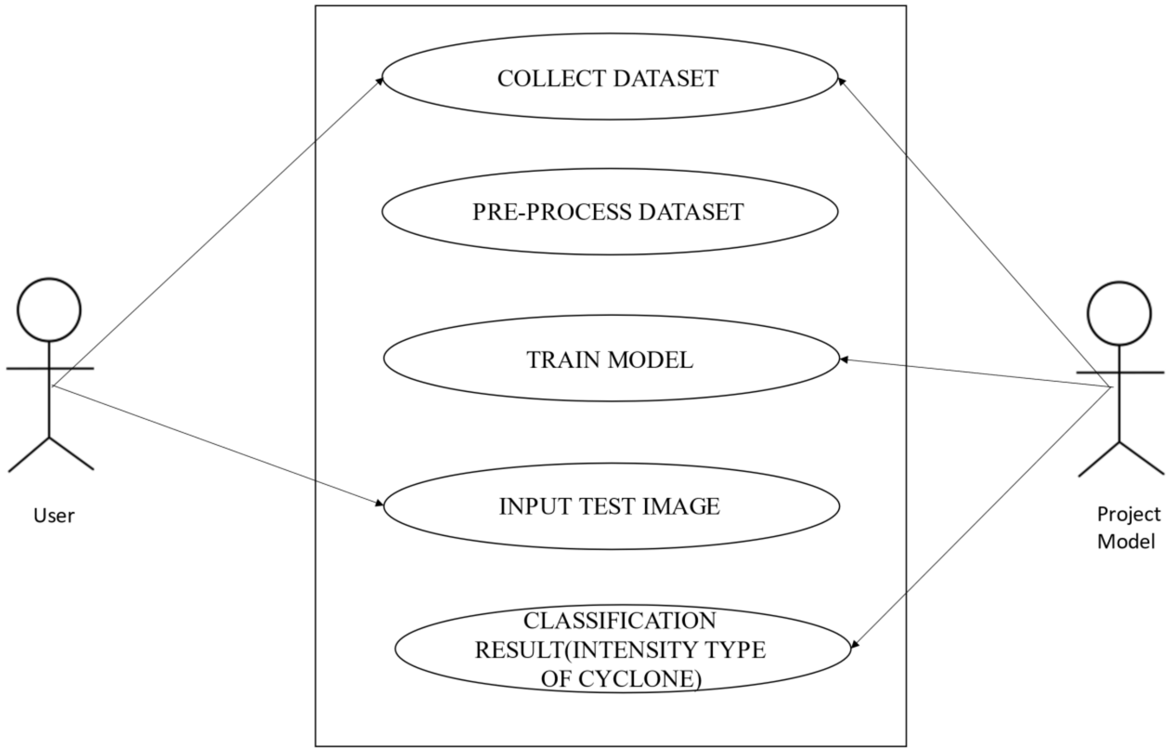


Fig: 5.3: Use Case Diagram

Use case diagram are used to specify:

- Requirements(external), requirement usages of a system under design or analysis to capture what the system is supposed to do.
- The functionality offered by a subject – what the system can do
- Requirements the specified subject poses on its environment – by defining how environment should interact with the subject so that it will be able to perform its services.

5.4 Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as in the operation of the system. The control flow is drawn from one operation to another.

In the "Image Processing" path, several image processing techniques are applied to the input image to extract relevant features for classification. These techniques include converting the image to grayscale, thresholding, morphological operations, and clearing border regions. These steps are represented by their respective nodes in the diagram.

After the image processing steps, the process continues with the "Connected Component Segmentation" node. This step segments the image into connected components, representing potential cyclone regions.

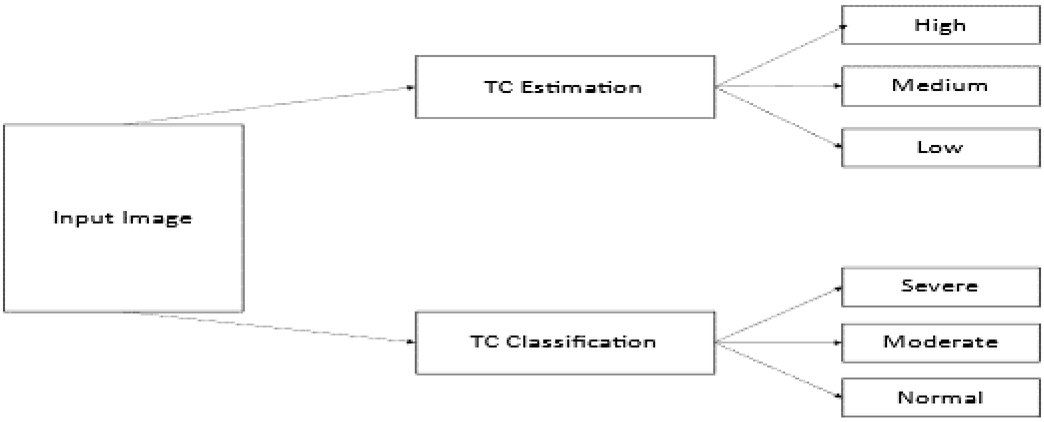


Fig: 5.4: Activity Diagram

Next, the process branches into the "Size Classification" node, where the size of each connected component is determined. Based on predefined thresholds or criteria, the connected components are classified into different size categories.

Following the "Size Classification," the process moves to the "Intensity Classification" node. Here, the intensity of each cyclone region is classified as either "High Intensity" or "Low/Medium Intensity" based on predefined criteria. This node represents a branching condition in the diagram.

If a cyclone region is classified as "High Intensity," the process follows the "High Intensity Classification" path, which may involve additional analysis or actions specific to high-intensity cyclones.

If a cyclone region is classified as "Low/Medium Intensity," the process follows the "Low/Medium Intensity Classification" path, which represents the handling for cyclones with lower intensity. This branch may have its own set of actions or analysis specific to low and medium-intensity cyclones.

5.5 Data Flow Diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Fig 5.5.1 is a data flow diagram and graphical representation of the flow of the data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated.

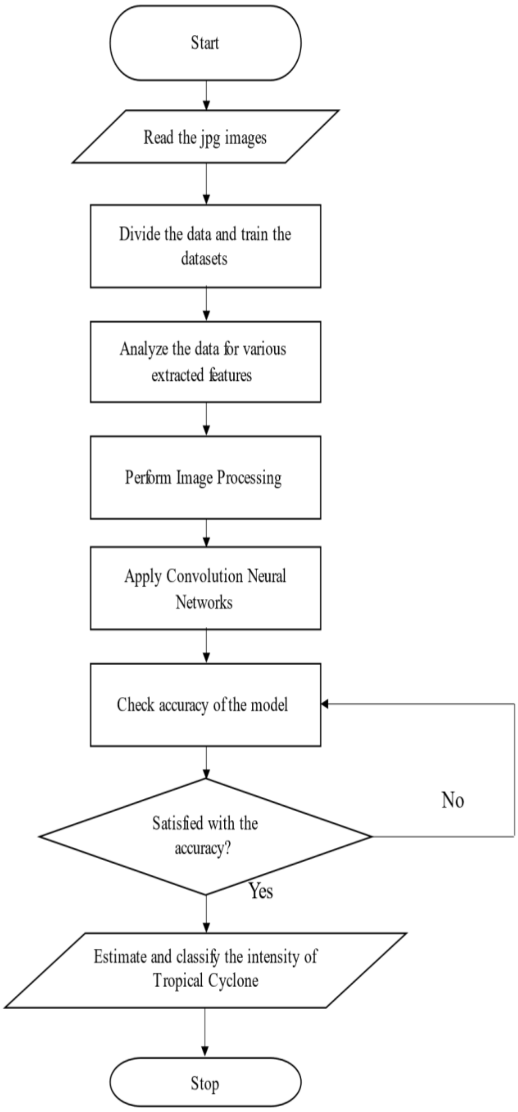


Fig: 5.5: Data Flow Diagram

5.6 Low Level Design

Low-level design (LLD) is a component-level design process that follows a step-by- step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms.

IoT Flood Alert System Architecture

This section describes the architecture of the IoT subsystem added to the existing TCICENet system.

Components

- ESP8266 Wi-Fi module
- Water-level sensor
- Buzzer
- LED
- MQTT broker (broker.emqx.io or similar)
- Flutter mobile alert app

Flow

1. Sensor measures water level.
2. Data sent to ESP8266.
3. ESP8266 publishes value to MQTT.
4. Backend or threshold logic checks if flood alert is required.
5. If alert is triggered, command sent back to ESP8266: "FLOOD_ALERT"
6. ESP8266 activates buzzer + LED.
7. Alert also sent to mobile application.

5.7 Pseudo Code

Pseudo code is an informal low - level description of the operating principle of a computer program or other algorithm. It uses the structural conventions of a normal programming language, but is intended for human reading rather than machine reading.

```
def detect_cyclone(img): st_images = [] st_caption = []

input_image = np.array(img)
img_fastai = Image(pil2tensor(input_image, dtype=np.float32).div_(255)) pred,idx,outputs = model.predict(img_fastai)

# Convert the image to Grayscale img = img.convert('L')
img = np.array(img)
orig = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR) st_images.append(img.copy()) st_caption.append('Grayscale Image')
```

Image Thresholding img = img > threshold

img = np.array(img) * 255 st_images.append(img.copy()) st_caption.append('Image Thresholding')

Morphological operations footprint = disk(2)

img = erosion(img, footprint) # Noise removal

footprint = square(4)

img = opening(img, footprint) # Opening operation on binary image

footprint = square(4)

img = closing(img, footprint) # Closing operation on binary image

footprint = disk(12)

img = dilation(img, footprint) # Dilation oparation on binary image st_images.append(img.copy())

st_caption.append('Morphological Operations')

Clear border region img = clear_border(img)

img = img.astype(np.uint8) *255 img = np.array(img) * 255 st_images.append(img.copy())

st_caption.append('Clear Border regions')

Blob Detection

num_labels, labels, stats, centroids = cv2.connectedComponentsWithStats(img) min_area = 100 # Small object removal

filtered_labels = []

for label in range(1, num_labels):

if stats[label, cv2.CC_STAT_AREA] >= min_area: filtered_labels.append(label)

class_label = "" sizes = [0]

Loop over the connected components and find the center and radius of each component for label in filtered_labels:

Create a binary image containing only the current component component_image = (labels ==

```
label).astype('uint8')

# Find the contours of the component Contours=cv2.findContours(component_image,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
center, size = cv2.minEnclosingCircle(contours[0]) size = int(size)

dist=cv2.norm(np.array((int(center[0]),int(center[1]),np.array([img.shape[0]//2,img.shape[1]//
2])

# Filter Blobs and draw circle around the cyclone if dist <= size:
# Draw a circle around the component if size < 45:
class_label = "normal"
cv2.circle(orig, (int(center[0]), int(center[1])), size, (135, 206, 235), thickness=3, lineType=8,
shift=0)
elif size >= 45 and size <= 75: class_label = "moderate"
cv2.circle(orig, (int(center[0]), int(center[1])), size, (252, 245, 95), thickness=3, lineType=8,
shift=0)
elif size > 75: class_label = "severe"
cv2.circle(orig, (int(center[0]), int(center[1])), size, (255, 87, 51), thickness=4, lineType=8,
shift=0)
col1, col2, col3, col4, col5 = st.columns(5) col1.metric("Horizontal Position",
f"{round(center[0], 2)}") col2.metric("Vertical Position", f"{round(center[1], 2)}")
col3.metric("Size", f"{size}") sizes.append(size)
if class_label == "normal":
col4.write(f'Size Classification <p><b style="font-size:26px; color:rgb(135, 206,
235);">{class_label}</b></p>', unsafe_allow_html=True)
elif class_label == "moderate":
col4.write(f'Size Classification <p><b style="font-size:26px; color:rgb(252, 245,
95);">{class_label}</b></p>', unsafe_allow_html=True)
```

CHAPTER 6

IMPLEMENTATION

An implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through computer programming and deployment. Many implementations may exist for a given specification or standard. For example, web browsers contain implementations of World Wide Web Consortium-recommended specifications, and software development tools contain implementations of programming languages.

6.1 Tropical Cyclone Intensity Classification Module

In this module the classification of intensity is done using a convolutional neural network (CNN) that has been implemented using the fastai library. The CNN is trained on the image dataset to predict the intensity of new images.

The CNN is trained using transfer learning, where a pre-trained model is used as the base model and then trained on the new dataset. The pre-trained model used in this code is the VGG19_bn and ResNet101 models. These models have been trained on the ImageNet dataset and have been shown to work well on image classification tasks.

The CNN model is first trained on the dataset using the VGG19_bn model architecture. The CNN model is trained for 11 epochs with a learning rate of 3e-3 using the fit_one_cycle() method. The accuracy and error rate metrics are used to evaluate the performance of the model during training.

After training the model with VGG19_bn, the same model is trained on the dataset using the ResNet101 architecture. The CNN model is trained for 10 epochs with a learning rate of 3e-3 using the fit_one_cycle() method. The accuracy and error rate metrics are used to evaluate the performance of the model during training. It was observed that there was not much change in the accuracy when compared to VGG19_bn.

6.1.1 Libraries

- **Fastai:** Fastai is an open-source library built on top of PyTorch that provides high-level APIs for training and deploying deep learning models. It is designed to make the process of deep learning more accessible and easier for researchers, developers, and practitioners of all levels of expertise. Fastai offers a variety of pre-built models, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers, which can be easily customized to suit the specific needs of a project. It also includes modules for natural language processing (NLP) and computer vision (CV) tasks. One of the key features of Fastai is its ability to automatically apply various data preprocessing techniques, such as data augmentation and normalization, which helps improve the accuracy of the models. It also includes an extensive set of visualizations and interpretation tools that can help in understanding the models' behavior and performance.

- **Torch:** Torch is an open-source machine learning library that is widely used in the field of artificial intelligence and deep learning. Torch is designed to provide fast and flexible experimentation with neural networks, and it supports a wide range of algorithms and models for building and training deep learning models. The library includes modules for data manipulation, optimization, model creation, and a range of built-in tools for visualization and debugging. Torch has been used in a wide range of research applications, including image and speech recognition, natural language processing, and game AI.
- **TorchVision:** TorchVision is a PyTorch package that consists of popular datasets, model architectures, and image/video transforms for computer vision tasks. TorchVision is designed to make it easy for researchers and practitioners to use PyTorch for image and video classification, segmentation, and detection tasks. TorchVision includes a number of standard computer vision datasets, such as CIFAR10, CIFAR100, and ImageNet, as well as tools for loading and preprocessing these datasets. TorchVision also includes a large number of pre-trained model architectures that can be used as the starting point for building more complex models. These pre-trained models are available in various forms, such as the weights of the trained models, the model architecture, or both.

6.1.2 Data

The infrared satellite images of Tropical Cyclones from 1981 to 2019, comprising four categories- Tropical Depression, Tropical Storm, Hurricane and Major Hurricane, were obtained from the National Institute of Informatics of Japan (http://agora.ex.nii.ac.jp/digital-typhoon/year/wnp/), and were then used to verify the performance of the proposed model. The infrared satellite images used in this project derive from the GMS1–5, GEO 9, MTSAT-1R, MTSAT-2, and Himawari 8 satellites in the northwest Pacific Ocean basin. The images were divided into 3 classes of intensity high, medium and low. A total of 3000 satellite images of Tropical Cyclones from 1981 to 2019 were selected to train and validate the proposed model. Out of which 80% were used for training and 20% for validation. Fig.6.1.2.1 shows sample infrafred satellite images.

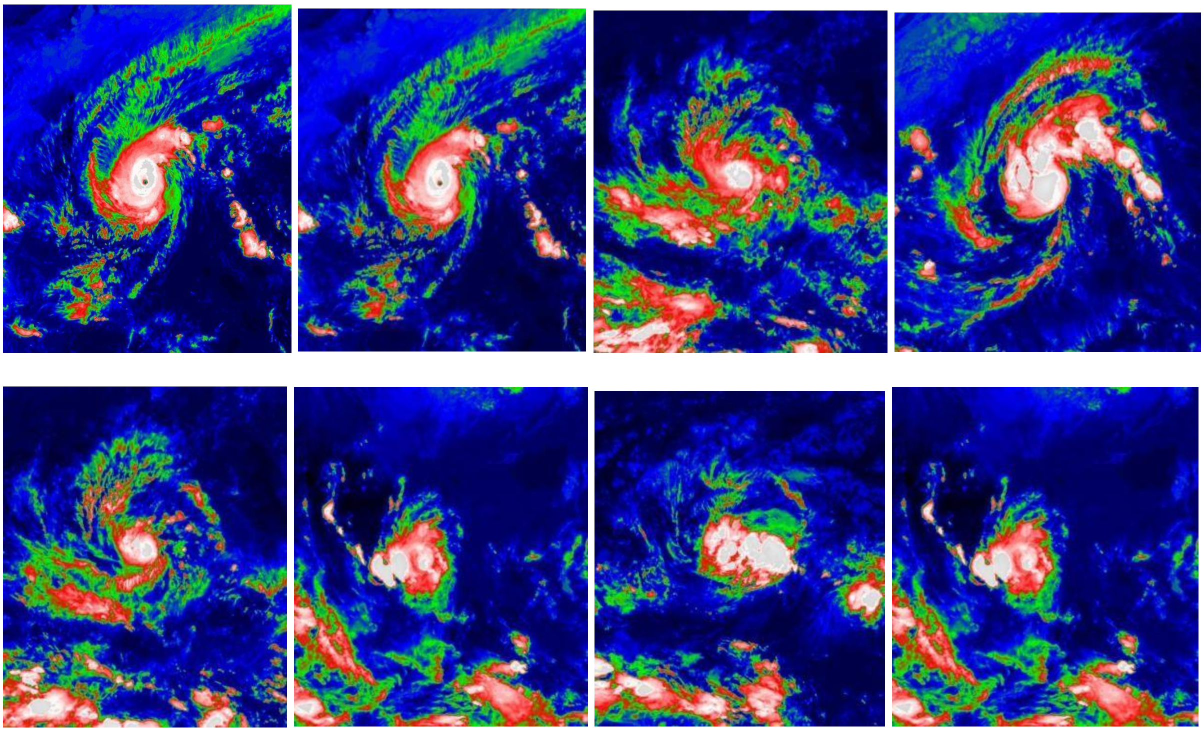


Fig: 6.1: Sample Infrared Satellite Images.

6.1.3 Data Preprocessing

Data preprocessing plays a vital role in developing an accurate and efficient deep learning model for infrared satellite image analysis. To achieve this, there are various techniques used in data preprocessing for infrared satellite images. One such technique is image normalization, which rescales the pixel values of an image to a standard range, thereby reducing the effect of variations in illumination and contrast of the input images. Another technique is image resizing, which ensures that all the input images have the same size and aspect ratio, thereby simplifying the training process. Image augmentation is another technique, which involves applying a variety of random transformations to the input images, such as rotation, flipping, and cropping, to increase the size of the training dataset and reduce overfitting. Image cropping is also used to remove unwanted regions, such as borders or regions with low contrast, to reduce the dimensionality of the input data and improve the accuracy of the deep learning model. Lastly, image enhancement is used to adjust the brightness and contrast levels of infrared satellite images, making the important features more prominent and improving the accuracy of the deep learning model.

Before feeding the infrared satellite images to the deep learning model, data preprocessing is necessary to improve the quality and accuracy of the results. The first step is to remove any corrupted or low-quality images from the dataset using image verification techniques. Then, the dataset is split into training and validation sets, with a validation percentage of 30% to evaluate the performance of the model. The data is transformed using data augmentation techniques such as resizing, flipping, and rotation to increase the diversity of the dataset and prevent overfitting. The images are normalized using the ImageNet statistics to ensure that the input values are in the same range as the pre-trained models. The final step is to create an ImageDataBunch object, which is a class in the fastai library that stores the dataset and its associated transformations. This object is passed to the CNN learner to train the deep learning model. By performing these preprocessing steps, the accuracy and performance of the model can be significantly improved.

6.1.4 Convolutional Neural Network (CNN)

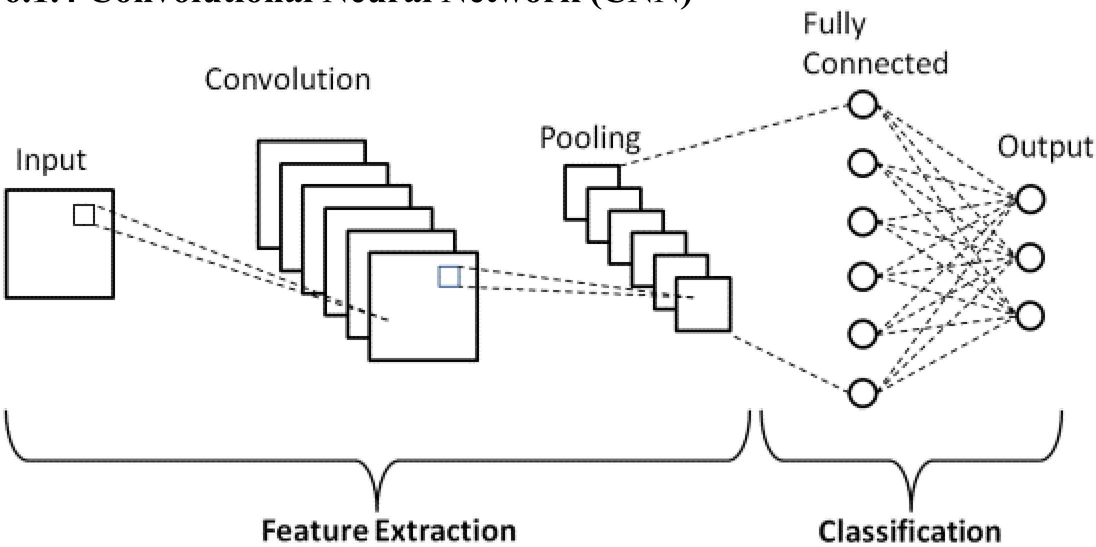


Fig:6.2: CNN architecture.

A convolutional neural network (CNN) is a type of artificial neural network used primarily in image processing and computer vision. CNNs are composed of multiple layers, each layer performing a specific function to extract increasingly complex features from the input image.

There are many CNN layers as shown in Fig 6.1.4. There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function.

1. **Input layer:** The input layer of a CNN receives the image data and passes it to the next layer.
2. **Convolutional layer:** This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.
3. **Pooling Layer:** In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations. It basically summarises the features generated by a convolution layer. In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer.
4. **Fully Connected Layer (FC):** The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.In this, the input image from the previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place. The reason two layers are connected is that two fully connected layers will perform better than a single connected layer. These layers in CNN reduce the human supervision.
5. **Dropout:** Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model’s performance when used on a new data.To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in

reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.Dropout results in improving the performance of a machine learning model as it prevents overfitting by making the network simpler. It drops neurons from the neural networks during training.

6. **Activation Functions:** one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network. It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred and for a multi-class classification, generally softmax is used. In simple terms, activation functions in a CNN model determine whether a neuron should be activated or not. It decides whether the input to the work is important or not to predict using mathematical operations.

CNN architectures can have different numbers of convolutional, pooling, and fully connected layers, depending on the complexity of the problem being solved. The number of neurons in each layer can also vary, depending on the size of the input image and the number of classes being classified. Overall, CNNs have proven to be highly effective in image processing and computer vision tasks and are widely used in a variety of applications. In our project we have made use of vgg19_bn and ResNet-101 architecture.

After training the model with VGG19_bn, the same model is trained on the dataset using the ResNet101 architecture. The CNN model is trained for 10 epochs with a learning rate of 3e-3 using the fit_one_cycle() method. The accuracy and error rate metrics are used to evaluate the performance of the model during training. It was observed that there was not much change in the accuracy when compared to VGG19_bn.
The number of neurons in each layer can also vary, depending on the size of the input image and the number of classes being classified. Overall

6.1.5 Deep Learning Algorithms for TC intensity Classification

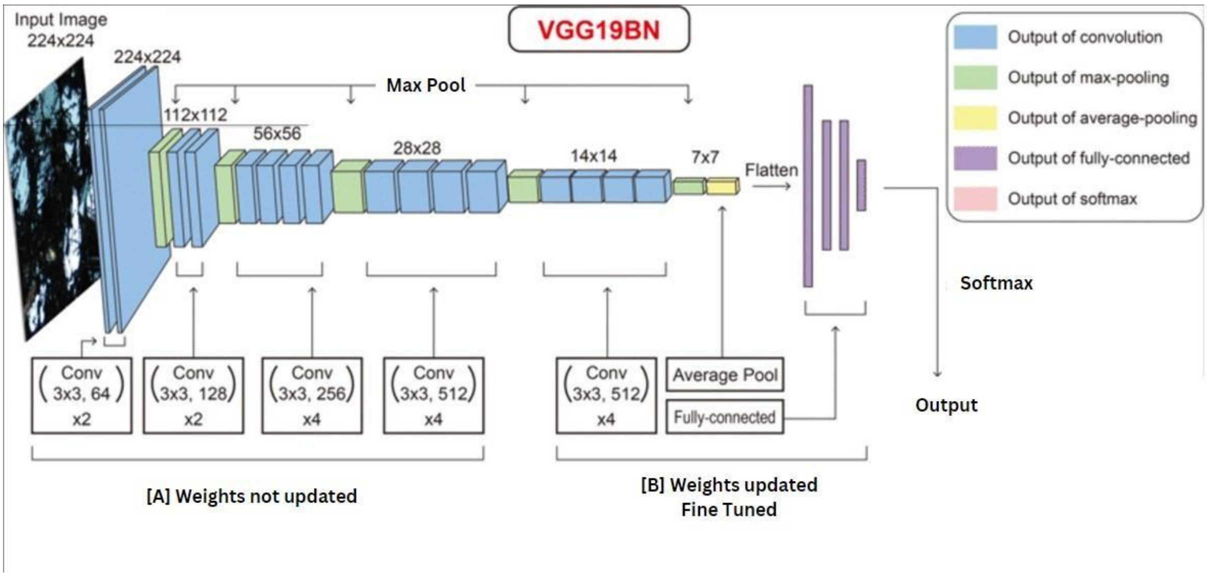


Fig:6.3: VGG19_bn architecture

The VGG19_bn (VGG stands for Visual Geometry Group) is a deep convolutional neural network model that was introduced in 2014 by Karen Simonyan and Andrew Zisserman. It is widely used for image classification and feature extraction. The VGG19_bn architecture consists of 19 layers, which are divided into convolutional layers, max pooling layers, and fully connected layers.

- The first two layers of VGG19_bn are convolutional layers with 64 filters each. The size of each filter is 3x3, and the padding is set to 1. The input image passes through these two layers, and the output of each layer is a feature map of size (224x224x64).
- The third layer is a max-pooling layer with a pool size of 2x2 and a stride of 2. This layer reduces the size of the feature map by half and increases the receptive field of the next layer.
- The fourth and fifth layers are convolutional layers with 128 filters each, followed by a max-pooling layer with a pool size of 2x2 and a stride of 2. These layers further extract more features from the input image.
- The sixth to ninth layers are convolutional layers with 256 filters each, followed by a max-pooling layer with a pool size of 2x2 and a stride of 2. These layers extract even more complex features from the input image.

- The tenth to thirteenth layers are convolutional layers with 512 filters each, followed by a max-pooling layer with a pool size of 2x2 and a stride of 2. These layers extract highly complex features from the input image.
- The fourteenth to seventeenth layers are convolutional layers with 512 filters each, followed by a max-pooling layer with a pool size of 2x2 and a stride of 2. These layers extract even more complex features from the input image.
- The eighteenth layer is a fully connected layer with 4096 neurons, which processes the output of the previous layer and outputs a feature vector of size 4096.
- The nineteenth layer is another fully connected layer with 4096 neurons, which further processes the output of the previous layer and outputs a feature vector of size 4096.
- The final layer is a softmax layer that produces the probability distribution over the different classes. In the case of image classification, the number of output neurons in the softmax layer is equal to the number of classes in the dataset.

The VGG19_bn architecture is similar to the VGG16 architecture, but it includes batch normalization layers after each convolutional layer. Batch normalization layers help in reducing the internal covariate shift, which is the change in the distribution of the input to a learning algorithm that slows down the learning process.

Residual Network (ResNet):

A residual neural network (ResNet) is an artificial neural network (ANN). It is also used for Control Neural Network. It is a gateless or open-gated variant of the HighwayNet, the first working very deep feedforward neural network with hundreds of layers, much deeper than previous neural networks.

After the first CNN-based architecture (AlexNet) that win the ImageNet 2012 competition, every subsequent winning architecture uses more layers in a deep neural network to reduce the error rate. This works for a smaller number of layers, but when we increase the number of layers, there is a common problem in deep learning associated with that called the Vanishing/Exploding gradient. This causes the gradient to become 0 or too large. Thus when we increases number of layers, the training and test error rate also increases.

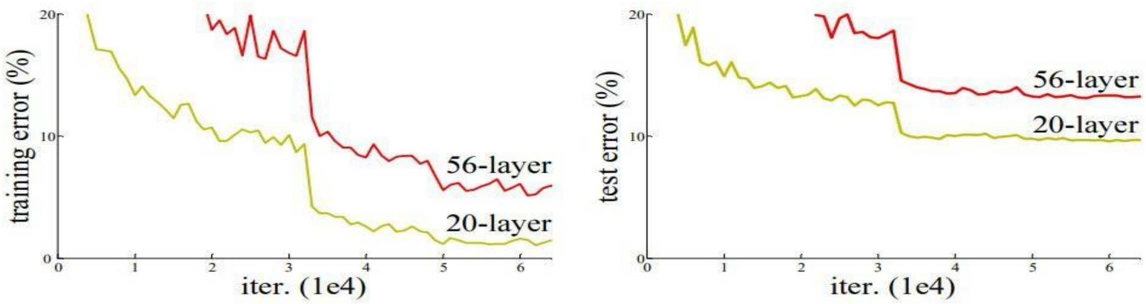


Fig:6.4 Comparison of 20-layer vs 56-layer architecture.

In the above plot, we can observe that a 56-layer CNN gives more error rate on both training and testing dataset than a 20-layer CNN architecture. After analyzing more on error rate the authors were able to reach conclusion that it is caused by vanishing/exploding gradient. ResNet, which was proposed in 2015 by researchers at Microsoft Research introduced a new architecture called Residual Network.

6.2 Image Processing

6.2.1 What is Image

An image is an array, or a matrix, of square pixels (picture elements) arranged in columns and rows. In a (8-bit) greyscale image each picture element has an assigned intensity that ranges from 0 to 255. A grey scale image is what people normally call a black and white image, but the name emphasizes that such an image will also include many shades of gray.

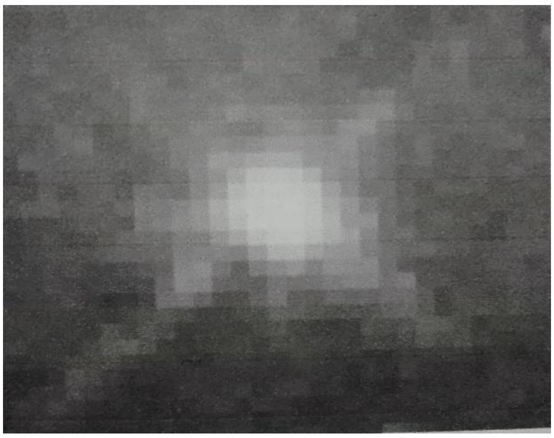


Fig: 6.5: An image – an array or a matrix of pixels arranged in columns and rows.

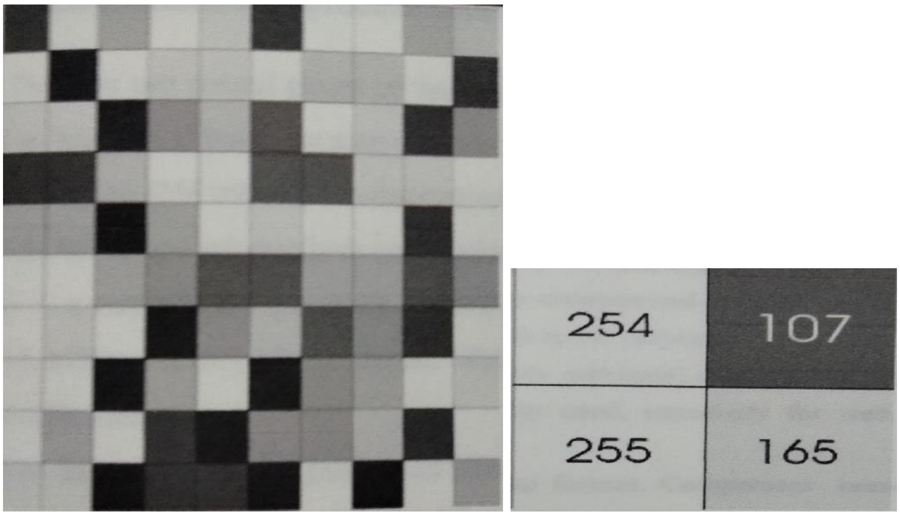


Fig:6.6 : Each pixel has a value from 0 (black) to 255 (white)

The possible range of the pixel values depend on the colour depth of the image, here 8 bit = 256 Enes or greyscales. A normal greyscale image has 8 bit colour depth = 256 greyscales. A "true colour" image has 24 bit colour depth = 8 * 8 * 8 bits = 256 * 256 * 256 colors = ~16 million colors.

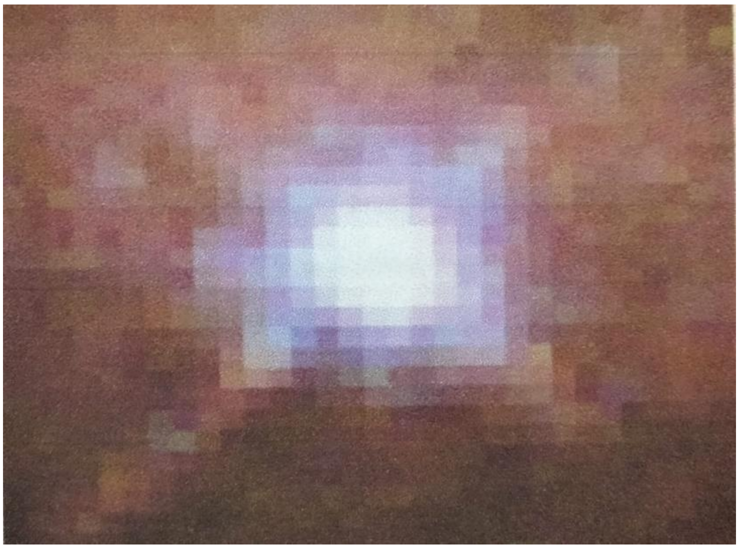


Fig: 6.7: A true-color imager assembled from three grayscale images colored RGB

Three greyscale images can be combined to form an image with 281,474,976,710,656 greyscale images have more greyscales, for instance 16 bit = 65536 greyscales. There are two general groups of 'images': vector graphics (or line art) and bitmaps (pixel-based or 'images'). Some of the most common file formats are:

- GIF - an 8-bit (256 colour), non-destructively compressed bitmap format. Mostly used for web. Has several sub-standards one of which is the animated GIF.
- JPEG - a very efficient (i.e. much information per byte) destructively compressed 24 bit (16 million colours) bitmap format. Widely used, especially for web and Internet (bandwidth-limited).
- TIFF - the standard 24 bit publication bitmap format. Compresses non-destructively with, for instance, Lempel-Ziv-Welch (LZW) compression.
- PS - Postscript, a standard vector format. Has numerous sub-standards and can be difficult to transport across platforms and operating systems.
- PSD - a dedicated Photoshop format that keeps all the information in an image including all the layers.

6.3 Images and Pictures

As we mentioned in the preface, human beings are predominantly visual creatures: we rely heavily on our vision to make sense of the world around us. We not only look at things to identify and classify them, but we can scan for differences, and obtain an overall rough feeling for a scene with a quick glance. Humans have evolved very precise visual skills: we can identify a face in an instant; we can differentiate colors; we can process a large amount of visual information very quickly.

However, the world is in constant motion: stare at something for long enough and it will change in some way. Even a large solid structure, like a building or a mountain, will change its appearance depending on the time of day (day or night); amount of sunlight (clear or cloudy), or various shadows falling upon it. We are concerned with single images: snapshots, if you like, of a visual scene. Although image processing can deal with changing scenes, we shall not discuss it in any detail in this text. For our purposes, an image is a single picture which represents something. It may be a picture of a person, of people or animals, or of an outdoor scene, or a microphotograph of an electronic component, or the result of medical imaging. Even if the picture is not immediately recognizable, it will not be just a random blur.

Image processing involves changing the nature of an image in order to either

1. Improve its pictorial information for human interpretation,
2. Render it more suitable for autonomous machine perception.

We shall be concerned with digital image processing, which involves using a computer to change the nature of a digital image. It is necessary to realize that these two aspects represent two separates but equally important aspects of image processing. A procedure which satisfies condition, a procedure which makes an image look better may be the very worst procedure for satisfying condition. Humans like their images to be sharp, clear and detailed; machines prefer their images to be simple and uncluttered.

6.4 Images and Digital Images

Suppose we take an image, a photo, say. For the moment, let's make things easy and suppose the photo is black and white (that is, lots of shades of grey), so no colour. We may consider this image as being a two-dimensional function, where the function values give the brightness of the image at any given point. We may assume that in such an image brightness value can be any real numbers in the range (black) (white).

A digital image from a photo in that the values are all discrete. Usually, they take on only integer values. The brightness values also ranging from 0 (black) to 255 (white). A digital image can be considered as a large array of discrete dots, each of which has a brightness associated with it. These dots are called picture elements, or more simply pixels. The pixels surrounding a given pixel constitute its neighborhood. A neighborhood can be characterized by its shape in the same way as a matrix: we can speak of a neighborhood, Except in very special circumstances, neighborhoods have odd numbers of rows and columns; this ensures that the current pixel is in the center of the neighborhood.

6.5 Color Scale

The Two main color spaces are **RGB** and **CMYK**.

6.6.1 RGB

The RGB color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. RGB uses additive color mixing and is the basic color model used in television or any other medium that projects color with light. It is the basic color model used in computers and for web graphics, but it cannot be used for print production. This is shown in fig 6.5.3.2.

The secondary colors of RGB cyan, magenta, and yellow - are formed by mixing two of the primary colors (red, green or blue) and excluding the third color. Red and green combine to make yellow, green and blue to make cyan, and blue and red form magenta. The combination of red, green, and blue in full intensity makes white. These are depicted as in Fig 6.5.3.2.

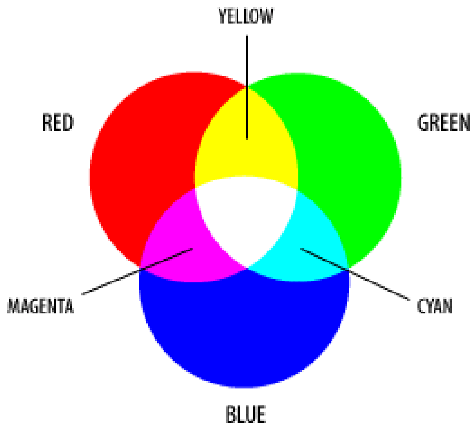


Fig: 6.8: The additive model of RGB

To see how different RGB components combine together, here is a selected repertoire of colors and their respective relative intensities for each of the red, green, and blue components:

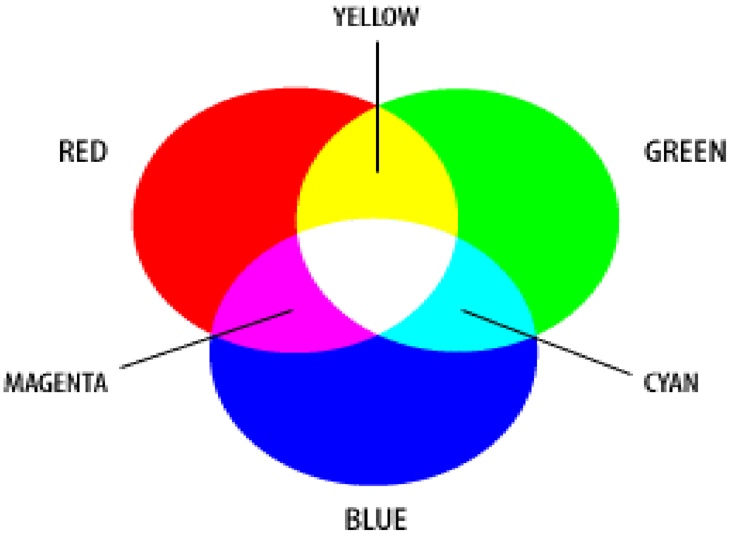


Fig:6.9: Components of RGB

6.6 Aspects of Image Processing

It is convenient to subdivide different image processing algorithms into broad sunclasses. There are different algorithms for different tasks and problems, and often we would like to distinguish the nature of task at hand.

Image enhancement: This refers to processing an image so that the result is more suitable for particular application.

Example include:

- Sharpening or de-blurring an out of focus image
- Highlighting edges
- Improving image contrast, or brightening an image
- Removing noise

Image restoration: This may be considered as reversing the damage done to an image by a known cause, for example:

- Removing of blur caused by linear motion
- Removal of optical distortions
- Removing periodic interference

Image segmentation: This involves subdividing an image into constituent parts, or isolating certain aspects of an image:

- Circles, or particular shapes in an image
- In an aerial photograph, identifying cars, trees, buildings, or roads

These classes are not disjoint; a given algorithm may be used for both image enchancement or for image restoration. However, we should be able to decide what it is that we are trying to do with our image: simply make it look better (enhancement), or remove damage (restoration).

6.6.1 An Image Processing Task

We will look into some details at a particular real-world task, and see how the above classes may be used to describe the various stages in performing this task. The job is to obtain, by an automatic process, the postcodes from envelopes. Here is how this may be accomplished:

- **Acquiring the image:** First we need to produce a digital image from a paper envelope. This can be done using either CCD camera, or a scanner.
- **Preprocessing:** This is the step taken before the major image processing task. This problem here is to perfrom some basic tasks in order to render the resulting image more suitable for the job to follow. In this case it may involve enhancing the contrast, removing noise, or identifying regions likely to contain the postcode.
- **Segmentation:** Here is where we actually get the postcode; in other words we extract from the image that part of it which contains just the postcode.
- **Representation and description:** These terms refer to extracting the particular features which allow us to differentiate between objects. Here we will be looking for curves, holes and corners which allow us to distinguish the different digits which constitute a postcode.
- **Recognition and interpretation:** This mean assigning labels to objects based on their descriptions (from the previous step), and assigning meanings to those labels. So we identify particular digits, and we interpret a string of four digits at the end of the address as the postcode.

6.7 Tropical Cyclone Estimation based on core size using Image Processing

6.7.1 Libraries

- **Streamlit:** Streamlit is a powerful Python library that enables developers to create interactive and customizable web applications for data science and machine learning projects. It simplifies the creation of user interfaces and data visualization, allowing developers to focus on the core functionality of their application. Streamlit provides a simple and intuitive API for creating web applications and data visualizations using Python, and is compatible with a wide range of popular data science and machine learning libraries such as Pandas, NumPy, Matplotlib, and TensorFlow.
- **PIL:** PIL (Python Imaging Library) is a library for working with images in Python. It provides a wide range of image processing functions, including image loading and saving, basic image manipulation, filtering and transformation, color manipulation, and more.PIL can read and write various image file formats, including BMP, GIF, JPEG, PNG, TIFF, and others. It also supports working with images in different color modes, such as grayscale, RGB, and CMYK. Some of the functions available in PIL include resizing and cropping images, applying various filters to images such as blur or edge detection, converting between different image modes, adding text or drawing shapes to images, and more. PIL is an open-source library that is available under a BSD-style license, making it free to use and distribute. It has been widely used in various applications, such as web development, scientific research, and data visualization.
- **CV2:** OpenCV (Open Source Computer Vision Library) is a popular computer vision library that is widely used for various applications such as object detection, facial recognition, image and video processing, and more. It is an open-source library that contains a large collection of tools and functions for computer vision and machine learning tasks. One of the most important features of OpenCV is its ability to work with images and video data from various sources such as cameras, files, and network streams. It provides many algorithms and functions for image processing, such as image filtering, edge detection, image segmentation, feature detection, and more. OpenCV is written in C++ and has interfaces for several programming languages, including Python, Java, and MATLAB. It is widely used in industry and academia for computer vision applications and research.
- **Math:** The math library is a built-in Python library that provides various mathematical functions and constants. It includes functions for performing basic mathematical

operations such as trigonometry, logarithms, exponents, and factorials. The library also provides constants such as pi and e, which are frequently used in mathematical calculations.The math library can be used by importing it at the beginning of a Python script. Once imported, the functions and constants can be called using their respective names. For example, the sin function can be called using math.sin(x) where x is the value in radians.In addition to these basic mathematical operations, the math library also provides functions for more complex mathematical operations such as trigonometric and hyperbolic functions, special functions like gamma and beta functions, and statistical functions like mean, variance, and standard deviation.the math library is a powerful tool for performing mathematical calculations in Python, and it can be used in a wide variety of applications such as data analysis, machine learning, and scientific computing.

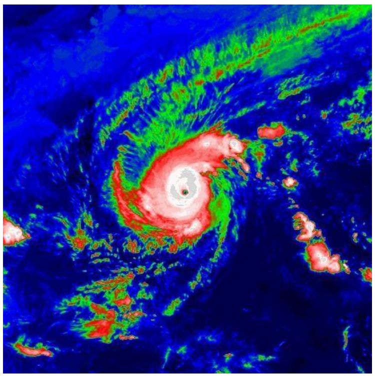
- **Pandas:** Pandas is a powerful and popular open-source data analysis library for Python. It provides data structures for efficiently storing and manipulating large datasets, as well as tools for data cleaning, aggregation, and analysis.The primary data structures in Pandas are the DataFrame and the Series. A DataFrame is a two-dimensional table that stores data in rows and columns, similar to a spreadsheet or SQL table. A Series is a one-dimensional array-like object that can hold various data types, including numerical, categorical, and text data.Pandas offers a vast array of functionality for data manipulation and analysis, including merging and joining datasets, reshaping data, handling missing data, and working with time series data. It also provides a powerful set of data visualization tools that make it easy to create high-quality charts and graphs.In addition to its core data analysis capabilities, Pandas can also interface with a wide range of other data sources and formats, including CSV files, SQL databases, Excel spreadsheets, and JSON data. This makes it a valuable tool for a variety of data science and machine learning applications.
- **Numpy:** NumPy is a Python library that provides a powerful array and matrix computation functionality for numerical operations. It is widely used in scientific and data-related applications due to its ability to efficiently perform complex mathematical operations on large multidimensional arrays and matrices. NumPy provides a wide range of mathematical functions, such as trigonometric functions, exponential and logarithmic functions, and linear algebra operations like matrix multiplication, inverse, determinant, and eigenvalues. NumPy also offers tools for data manipulation, including

indexing, slicing, sorting, filtering, and reshaping of arrays. It is widely used in fields such as machine learning, data analysis, scientific computing, and engineering. NumPy is an essential library in the Python ecosystem, and its functions are often integrated into other scientific and data-related libraries.

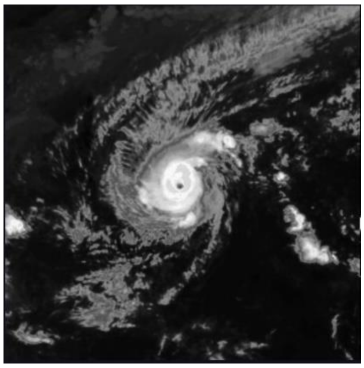
- **Matplotlib.pyplot:** Matplotlib.pyplot is a plotting library in Python that provides an interface similar to that of MATLAB. It is a powerful tool for creating a wide range of static, animated, and interactive visualizations in Python. pyplot is a collection of command-style functions that make matplotlib work like MATLAB. It is a part of the larger matplotlib library, which provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, and GTK.With pyplot, you can create line plots, scatter plots, bar plots, histograms, 3D plots, and many other types of visualizations. It provides a flexible and customizable framework for creating and editing plots with a wide range of options for controlling colors, line styles, labels, and axis formatting. It also provides support for adding legends, annotations, and text to your plots.One of the strengths of matplotlib.pyplot is its ability to handle large datasets efficiently. It can work with data in a variety of formats, including NumPy arrays, Pandas DataFrames, and Python lists. Additionally, pyplot integrates seamlessly with other Python libraries such as NumPy, SciPy, and Pandas, making it a popular choice for data visualization and analysis.
- **Skimage:** The Scikit-Image (skimage) is an open-source image processing library built on the top of NumPy, SciPy, and matplotlib. It is designed to be a user-friendly library for image processing and computer vision tasks. The skimage library provides a comprehensive set of functions for image manipulation, processing, analysis, and visualization. These functions include image filtering, segmentation, feature detection, and more. The library is especially useful for its ability to handle multi-dimensional arrays and its seamless integration with other scientific Python libraries. Skimage has a large and active community, making it easy to find support, examples, and tutorials online.
- **Statistics:** Statistics is the branch of mathematics that deals with the collection, analysis, interpretation, presentation, and organization of data. It provides tools for summarizing and describing large datasets, identifying patterns and relationships within the data, and making predictions based on the available information. Statistics can be divided into two main categories: descriptive statistics and inferential statistics.

Descriptive statistics involves the methods used to summarize and describe data, such as measures of central tendency (mean, median, mode) and measures of dispersion (variance, standard deviation). Inferential statistics involves using sample data to make inferences about a larger population, such as hypothesis testing and confidence intervals. Statistics is used in a wide range of fields, including business, medicine, engineering, social sciences, and many others.

6.7.2 Grayscale Conversion



(a) Infrared Image



(b) Grayscale Image

Fig: 6.10: Conversion from Infrared to Grayscale image

Grayscale conversion is a common step in image processing. A grayscale image is a black-and-white image where the intensity of each pixel is represented by a single value between 0 and 255. In a color image, each pixel is composed of three color channels - Red, Green, and Blue (RGB). In contrast, in a grayscale image, each pixel has only one channel, which represents the intensity of the color.

The grayscale conversion step involves converting a color image into a grayscale image by removing the color information and representing each pixel as a single value. There are several methods to convert a color image to grayscale, including:

- **Luminance method:** This method calculates the intensity of each pixel by taking the weighted average of the RGB channels. The weights are based on the human eye's sensitivity to different colors, with green having the highest weight and blue having the lowest weight.
- **Average method:** This method calculates the intensity of each pixel by taking the average of the RGB channels.
- **Lightness method:** This method calculates the intensity of each pixel by taking the maximum and minimum values of the RGB channels and averaging them.

6.7.3 Image Thresholding

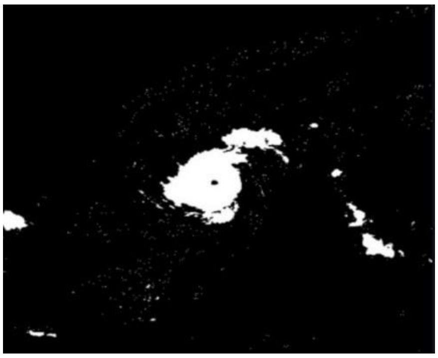


Fig: 6.11: Tropical Cyclone image after performing Thresholding operation

Image thresholding is a technique used in image processing to convert an image into a binary image. In a binary image, each pixel is either black (0) or white (255) based on a threshold value. This threshold value is used to separate the foreground pixels from the background pixels in an image.

There are different methods for thresholding an image, such as global thresholding, adaptive thresholding, and Otsu's thresholding. Global thresholding involves selecting a single threshold value that is applied to the entire image. Adaptive thresholding takes into account the local variations in image intensity and applies a threshold that varies with the local intensity. Otsu's thresholding is a statistical approach that calculates an optimal threshold value based on the image histogram.

Once the threshold value is selected, each pixel in the image is compared to this value. If the pixel value is greater than the threshold, it is set to white (255), otherwise, it is set to black (0). This results in a binary image where the foreground pixels are white and the background pixels are black.

The second step in our model is converting greyscale image into binary image after performing image thresholding. Image thresholding is performed using the cv2.threshold function. the cv2.threshold function is used to convert the grayscale image obtained in the previous step into a binary image. The function takes the grayscale image as input and applies a threshold value to each pixel in the image. Pixels with intensity values above the threshold value are classified as foreground (white pixels) while pixels with intensity values below the threshold value are classified as background (black pixels). The function returns two outputs: the thresholded binary image and a threshold value. The thresholded binary image of the Tropical cyclone is shown in fig. 6.11.

6.7.4 Morphological Operation

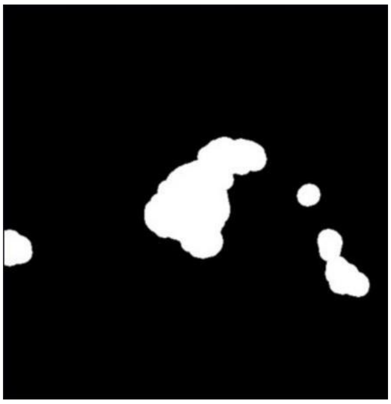


Fig: 6.12: Image after performing morphological operations on binary image

Morphological operations are a fundamental set of image processing techniques used to analyze and manipulate the geometric structure of objects in digital images. These operations are based on the shapes and sizes of the objects in the image and the way they are arranged.

The main purpose of morphological operations is to extract information about the shape and structure of objects, as well as to enhance or remove specific features from the image. The two most common morphological operations are erosion and dilation. Erosion is a process that removes pixels from the boundary of an object, which can be used to remove small objects or to separate objects that are close together. Dilation, on the other hand, adds pixels to the boundary of an object, which can be used to fill in gaps or holes in an object or to connect objects that are close together. Other morphological operations include opening and closing, which are combinations of erosion and dilation, and top-hat and bottom-hat transforms, which are used to extract features that are smaller or larger than a given size. Morphological operations are widely used in image processing applications, such as image segmentation, object recognition, and feature extraction.

In our model the erosion and dilation morphological operations are applied on the thresholded image using the OpenCV library functions cv2.erode() and cv2.dilate(). These operations can help remove noise and unwanted small objects from the binary image, and can also help connect or separate different regions of the image. Some of the morphological operations are:

- **Erosion:** erosion operation is performed using cv2.erode() function. Erosion is a morphological operation that shrinks the foreground pixels and expands the background pixels, thereby removing small regions or thin lines from the image. It is achieved by sliding a structuring element (kernel) over the image and replacing each pixel with the minimum value of the pixels in the kernel. Here we have used a rectangular kernel of size (5,5) is used for erosion.

- **Dilation:** dilation operation is performed using cv2.dilate() function. Dilation is the opposite of erosion, it expands the foreground pixels and shrinks the background pixels, thereby filling in gaps and making the object boundaries more defined. It is achieved by sliding a structuring element (kernel) over the image and replacing each pixel with the maximum value of the pixels in the kernel.
- **Opening:** The opening operation is a combination of erosion followed by dilation. It is used to remove small objects and smooth the edges of larger objects. In the code, we use a 3x3 rectangular kernel for opening. The cv2.morphologyEx() function takes three arguments: the input image, the type of morphological operation and the structuring element (kernel). The output image is the opened version of the input image.

By applying these morphological operations in sequence, we can obtain a more refined and cleaned binary image as shown in Fig 6.12.

6.7.5 Clear border regions

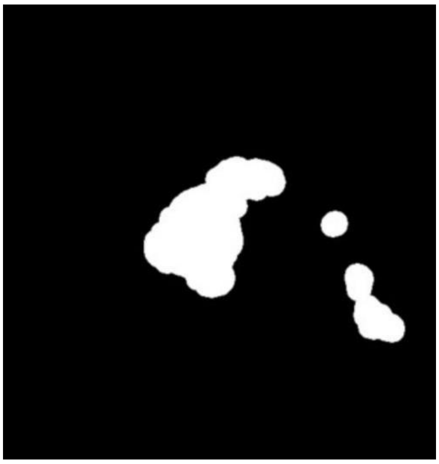


Fig: 6.13: Image after clearing border regions

In image processing, the "clear border region" refers to the process of removing or erasing the objects or features that are touching or intersecting with the image's border or edges. This process is often performed to separate the foreground objects or features from the background and to remove the artifacts that can interfere with the analysis or interpretation of the image.

Clearing the border region can be achieved using several techniques such as morphological operations, image erosion, and image dilation. These operations help to remove the objects or features that are partially or completely connected to the border of the image. In many cases, clearing the border region can also help to reduce the processing time and improve the accuracy of image analysis algorithms by removing irrelevant information from the image.

In our model the "clear_border" function from the "skimage.segmentation" module is used to remove objects that touch the border of the image. This is done because objects that touch the border may not be fully captured in the image, which can lead to errors in subsequent processing steps.

The "clear_border" function takes the binary image as input and returns a new binary image as shown in Fig 6.8.4.1. where all objects that touch the border have been removed. It does this by first identifying the objects that touch the border using the "find_contours" function from the "skimage.measure" module. It then creates a mask where all pixels that are part of an object touching the border are set to 0 (black). Finally, it applies this mask to the input image using element-wise multiplication, effectively removing the objects touching the border.

6.7.6 Blob Detection

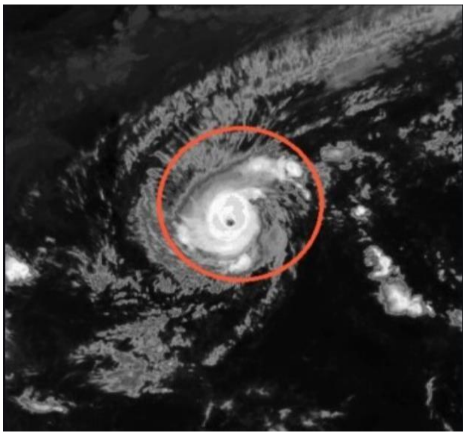


Fig 6.14: Blob detection

Blob detection is a fundamental operation in computer vision that involves identifying and locating regions or objects in an image that have homogeneous pixel values or color. Blobs can be described as connected components or regions that are distinguished from their surroundings by a significant difference in intensity, color, or texture. Blob detection can be used for a wide range of applications, including object recognition, tracking, and image segmentation.

The process of blob detection involves thresholding an image to produce a binary image, followed by grouping adjacent pixels or regions into connected components. The size, shape, and other features of each blob can then be analyzed and used for further processing or decision- making. One commonly used approach for blob detection is the Laplacian of Gaussian (LoG) method, which involves convolving an image with a Gaussian filter to smooth the image and then applying a Laplacian filter to highlight regions of high intensity variation. Other methods for blob detection include the Difference of Gaussian (DoG) method, the Hessian matrix method, and the Maximally Stable Extremal Regions (MSER) method.

In our model blob detection is performed using the cv2.findContours() function. This function finds contours (i.e., the boundaries) of white objects in a binary image. In this case, the binary image is the output of the thresholding step performed earlier.

After the contours are found, the area of each contour is calculated using the cv2.contourArea() function. Contours with an area below a certain threshold (specified by the min_area variable) are ignored, as they are likely to be noise rather than actual blobs.

The remaining contours are assumed to be blobs and are drawn on the original grayscale image using the cv2.drawContours() function. The function takes the original image, the list of contours, the index of the contour to draw (-1 to draw all contours), the color of the contour, and the thickness of the contour as inputs.

Finally, the number of detected blobs is printed to the console and based on the core size of the Tropical Cyclone the severity of the cyclone is decided.

Implementation of IoT Flood Detection Module

6.9.1 Hardware Setup

The hardware model consists of:

- NodeMCU ESP8266
- Water level sensor
- Active buzzer
- LED indicator
- USB power supply

6.9.2 MQTT Communication Logic

ESP8266 connects to Wi-Fi and subscribes to a specific MQTT topic, for example:

Topic: floodsystem/alert

When the flood threshold is crossed, the backend publishes:

Payload: "1"

On receiving this payload, ESP8266 triggers:

- digitalWrite(buzzer, HIGH)
- digitalWrite(LED, HIGH)
- digitalWrite(LED, LOW)

6.9.3 Flutter Mobile Alert Integration

The mobile application subscribes to the same MQTT topic or receives REST-based push notification.

The alert screen displays:

- Flood detection timestamp
- Water level
- Suggested action
- Emergency contact button

6.9.4 ESP8266 Sample Code Snippet

```
if (message == "1") {
  digitalWrite(buzzer, HIGH);
  digitalWrite(led, HIGH);
}
```

This ensures instant on-ground alert.

CHAPTER 7

RESULTS AND ANALYSIS

7.1Results

The results provides an opportunity to demonstrate the effectiveness of the developed model. It includes metrics such as accuracy, precision, and others that provide a quantitative measure of the performance of the model. It provides a means to verify the validity of the approach taken in developing the model. By analyzing the results, it may become clear that certain features or parameters could be modified to improve the performance of the model.

7.2Validation

Validation is a crucial step in the development of deep learning models. It involves assessing the accuracy and reliability of the model, and ensuring that it performs well on a variety of data. The first step in validating a deep learning model is to prepare the data. Once the data is prepared, the model is trained using the training set. After training, the model is evaluated using the validation set. Finally, the model is evaluated using the test set.

7.3Confusion Matrix

A confusion matrix is a table that is often used to evaluate the performance of a deep learning model in classification tasks. It compares the predicted class labels with the actual class labels to calculate various performance metrics.

- True Positive (TP): This is when the model correctly predicts the positive class for a given sample. For example, if the actual class is high, and the model correctly predicts high, it would be a true positive for High.
- False Positive (FP): This is when the model predicts the positive class for a sample, but the actual class is negative. For example, if the actual class is Low, but the model predicts High, it would be a false positive for High.
- False Negative (FN): This is when the model predicts the negative class for a sample, but the actual class is positive. For example, if the actual class is High, but the model predicts Low, it would be a false negative for High.

- True Negative (TN): This is when the model correctly predicts the negative class for a given sample. For example, if the actual class is Low, and the model correctly predicts Low, it would be a true negative for High.

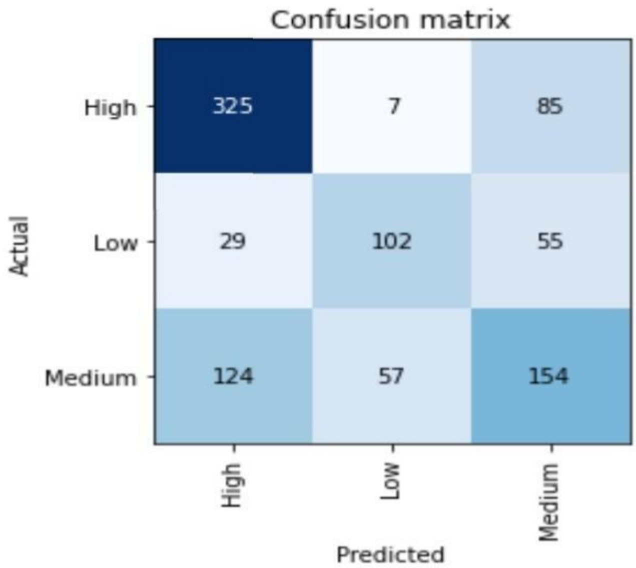


Fig 7.1: Confusion Matrix.

7.4Learning Rate and Accuracy curve

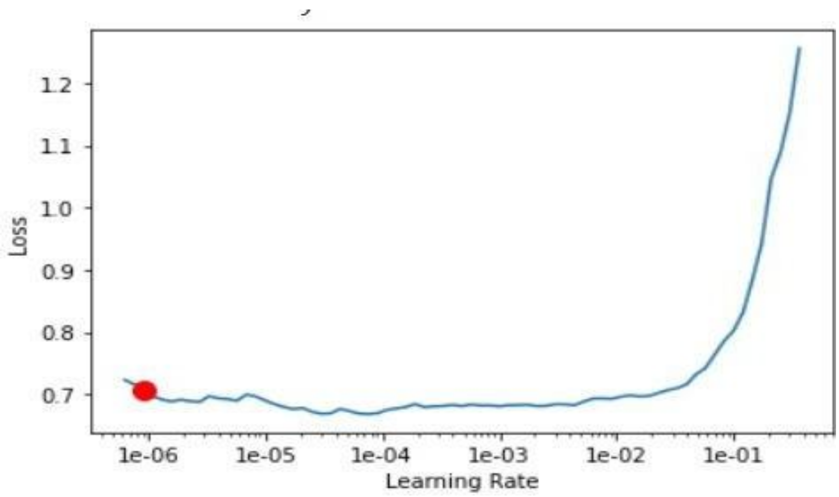


Fig 7.2: Learning rate

In machine learning and deep learning, the learning rate is a hyperparameter that controls how much the model adjusts its parameters in response to the error calculated during training. The learning rate is one of the most important hyperparameters to tune when training a neural network or other machine learning models.

If the learning rate is too high, the model may overshoot the optimal solution and fail to converge. On the other hand, if the learning rate is too low, the model may take too long to converge, or it may get stuck in a suboptimal solution.

The choice of learning rate often depends on the specific problem and the architecture of the model. Generally, it is recommended to start with a small learning rate and gradually increase it until the model converges. If the model fails to converge, the learning rate should be reduced. There are also various techniques, such as learning rate schedules and adaptive learning rate methods, that can help to optimize the learning rate during training.

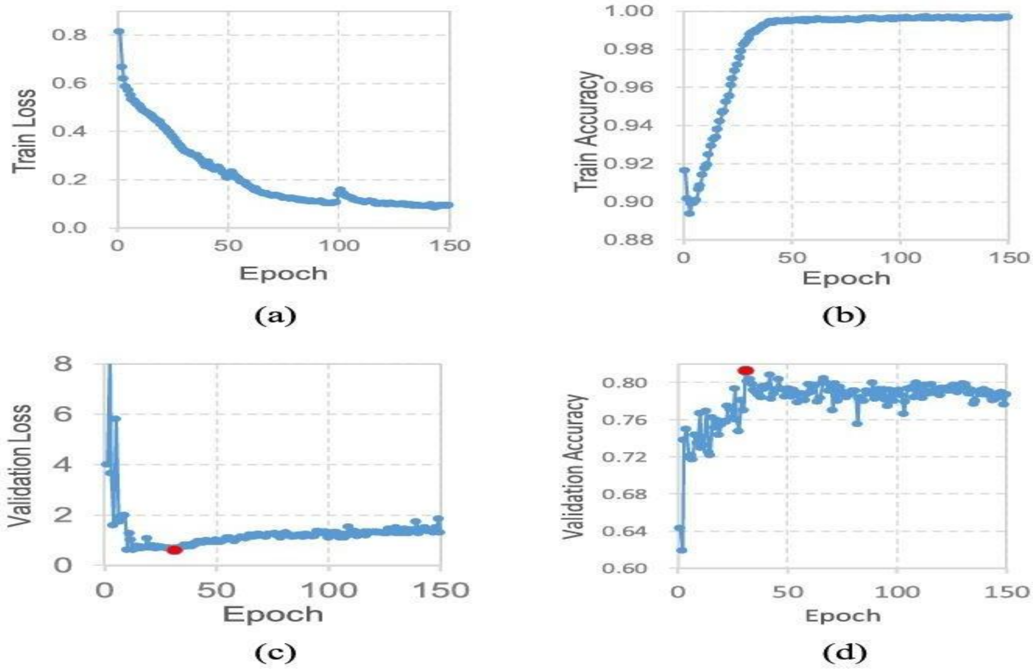


Fig: 7.3 :Accuracy curve and loss function curve for the TCIC module.

a. loss function curve
b. Training accuracy curve
c. Validation loss function curve
d. Validation accuracy curve

CHAPTER 8

SNAPSHOTS

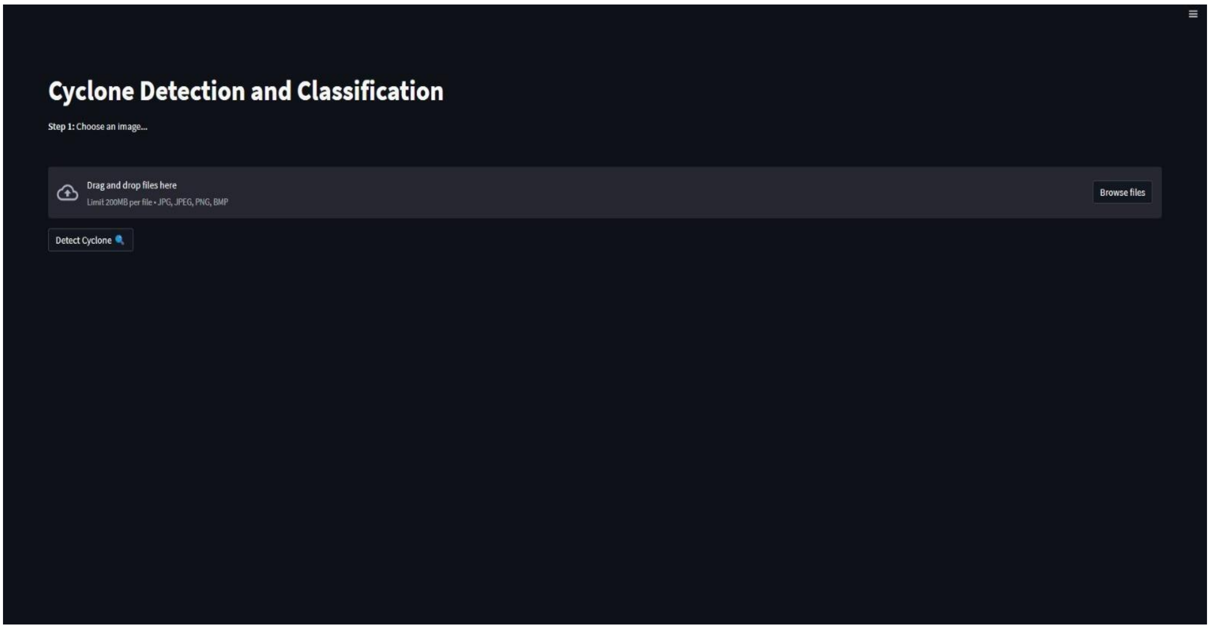


Fig 8.1: Browse files to detect Cyclone.

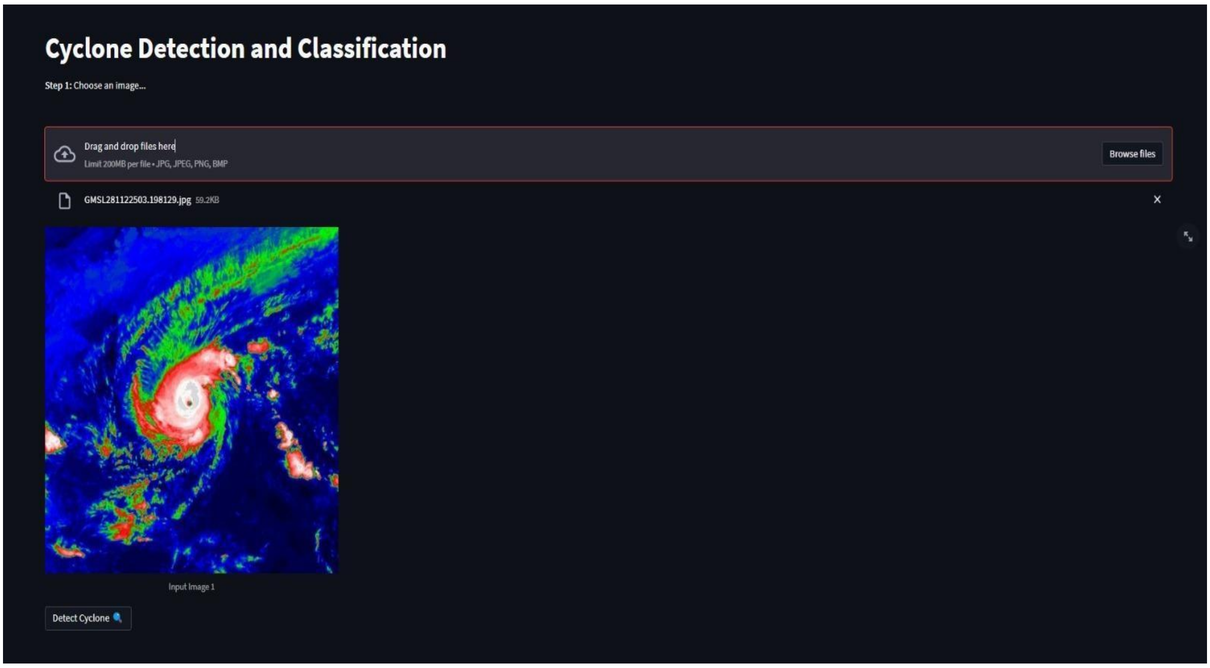


Fig 8.2: Selected a Tropical Cyclone to detect cyclone.

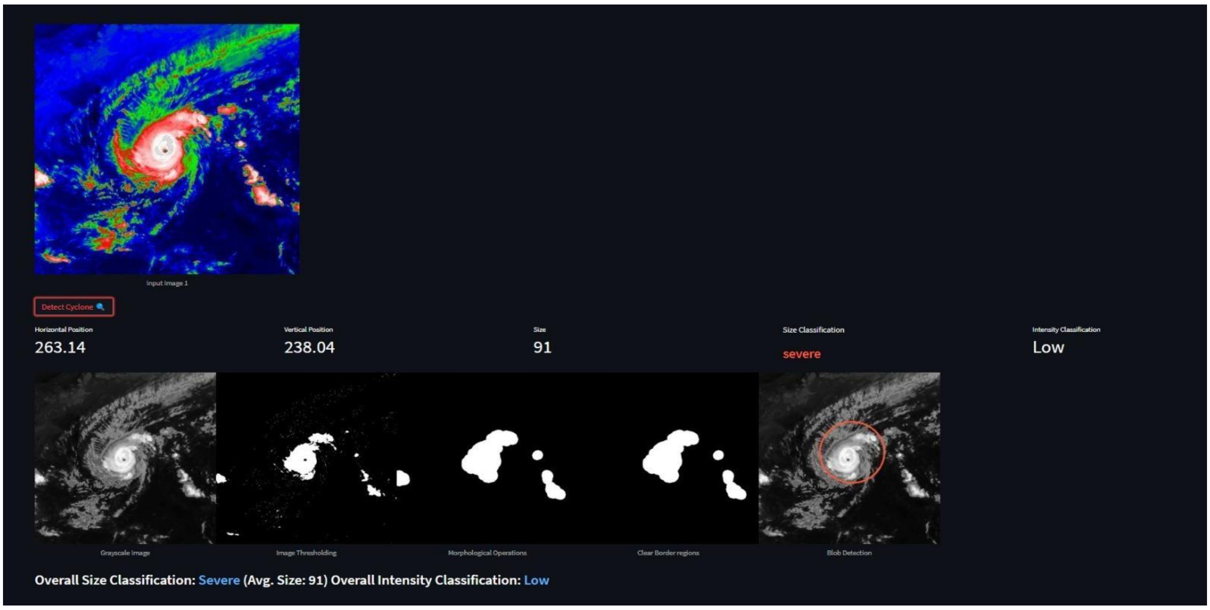


Fig 8.3: Intensity Classification of a single Tropical Cyclone.

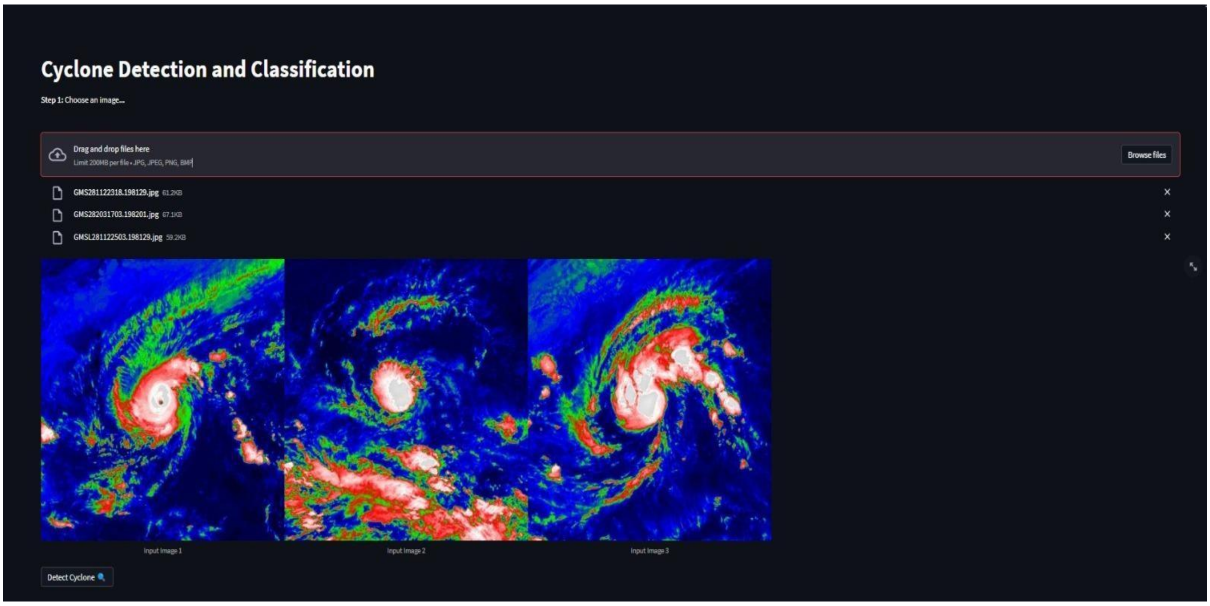


Fig 8.4: Selecting multiple Tropical Cyclones at a time for classification.

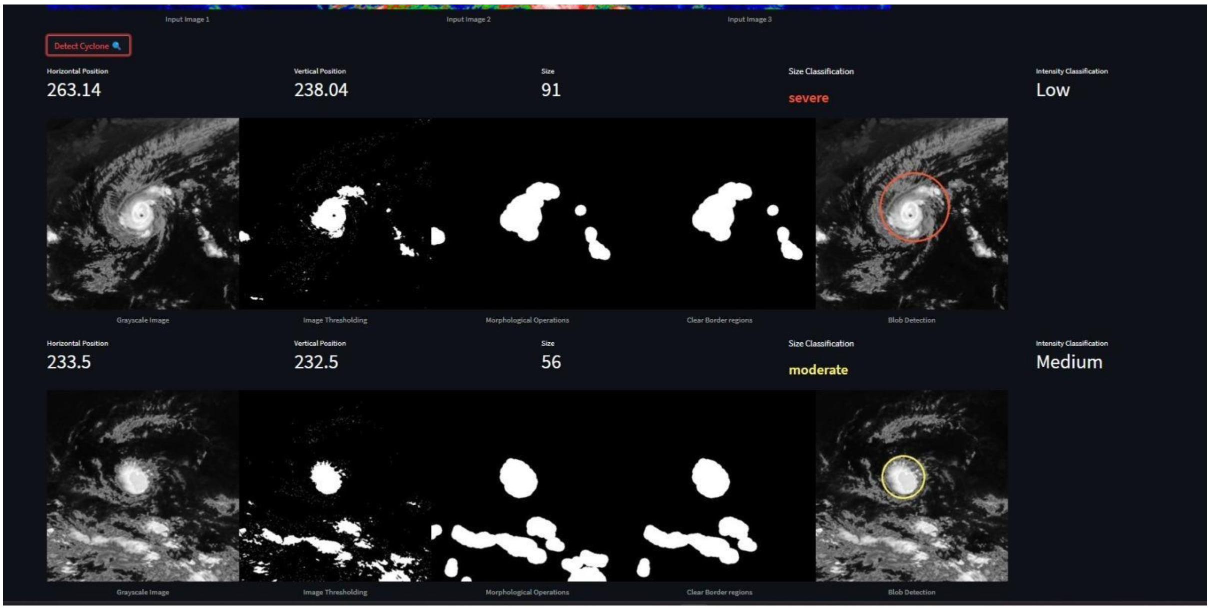


Fig 8.5: Classification of the first to Tropical cyclones.

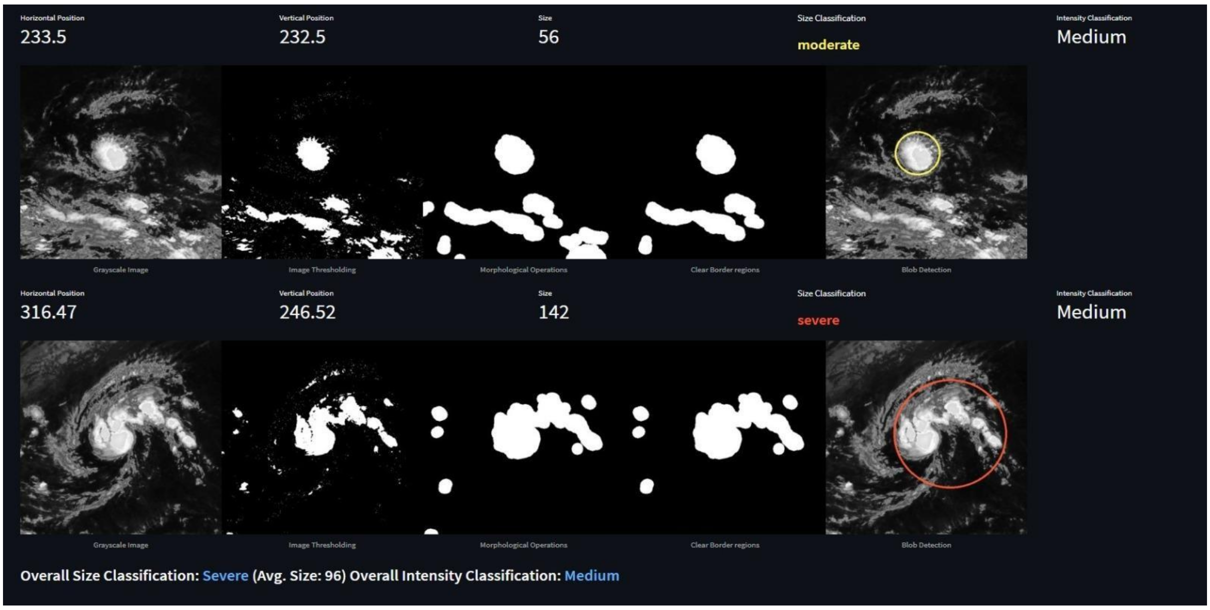


Fig 8.6: Taking average of all three to classify their intensity.