“Tropical cyclone intensity detection”

A Project Report Submitted to
Rajiv Gandhi Proudyogiki Vishwavidyalaya

Towards Partial Fulfillment for the Award of
Bachelor of Technology in Computer Science

Submitted by:

Harshwardhan Songirkar(0827CD201025)

Ankita Pandey(0827CD201008)

Vedant Singh(0827CD201062)

V Mohit Rao(0827CD201061)

Guided by:

Mr. Kamlesh Chopra
Professor, IT Department
AITR, Indore

Acropolis Institute of Technology & Research, Indore
July-Dec 2022

EXAMINER APPROVAL

The Project entitled **"Tropical Cyclone Intensity Detection using CNN"** *submitted* by **Ankita Pandey** (**0827CD201008**), **Harshwardhan Songirkar(0827CD201025), Vedant Singh(0827CD201062), V Mohit Rao** (**0827CD201061**) has been examined and is hereby approved towards partial fulfillment for the award of ***Bachelor of Technology degree in Computer Science*** discipline, for which it has been submitted. It understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it has been submitted.

(Internal Examiner) **(External Examiner)**

Date: **Date:**

GUIDE RECOMMENDATION

This is to certify that the work embodied in this project entitled "Tropical cyclone intensity detection" submitted by **Ankita Pandey** ,(**0827CD201008**) , **Harshwardhan Songirkar(0827CD201025) , Vedant Singh(0827CD201062) , V Mohit Rao**(**0827CD201061**) Is a satisfactory account of the bona fide work done under the supervision of ***Mr. Kamlesh Chopra,*** is recommended towards partial fulfillment for the award of the Bachelor of Engineering (Information Technology) degree by Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal.

(Project Guide) **(Project Coordinator)**

STUDENTS UNDERTAKING

This is to certify that project entitled *"*Tropical cyclone intensity detection*"* has developed by us under the supervision of ***Mr. Kamlesh Chopra***. The whole responsibility of work done in this project is ours. The sole intension of this work is only for practical learning and research.

We further declare that to the best of our knowledge; this report does not contain any part of any work which has been submitted for the award of any degree either in this University or in any other University / Deemed University without proper citation and if the same work found then we are liable for explanation to this.

Ankita Pandey (**0827CD201008**)

Harshwardhan Songirkar(0827CD201025)

Vedant singh(0827CD201062)

V Mohi Rao(0827CD201061)

ACKNOWLEDGEMENT

We thank the almighty Lord for giving me the strength and courage to sail out through the tough and reach on shore safely.

There are number of people without whom this projects work would not have been feasible. Their high academic standards and personal integrity provided me with continuous guidance and support.

We owe a debt of sincere gratitude, deep sense of reverence and respect to our guide and mentor **Prof. Kamlesh Chopra,** Professor, AITR, Indore for his motivation, sagacious guidance, constant encouragement, vigilant supervision and valuable critical appreciation throughout this project work, which helped us to successfully complete the project on time.

We express profound gratitude and heartfelt thanks to **Prof. Prashant Lakkadwala**, HOD IT, AITR Indore for his support, suggestion and inspiration for carrying out this project. I am very much thankful to other faculty and staff members of IT Dept, AITR Indore for providing me all support, help and advice during the project. We would be failing in our duty if do not acknowledge the support and guidance received from **Dr S C Sharma**, Director, AITR, Indore whenever needed. We take opportunity to convey my regards to the management of Acropolis Institute, Indore for extending academic and administrative support and providing me all necessary facilities for project to achieve our objectives.

We are grateful to **our parent** and **family members** who have always loved and supported us unconditionally. To all of them, we want to say "Thank you", for being the best family that one could ever have and without whom none of this would have been possible.

Ankita Pandey (0827CD201008), Vedant Singh (0827CD201062),

Harshwardhan Songirkar(0827CD201025), V MohitRao (0827CD201061)

Executive Summary

Tropical cyclone intensity detection

This project is submitted to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (MP), India for partial fulfillment of Bachelor of Engineering in Information Technology branch under the sagacious guidance and vigilant supervision of ***Prof. Kamlesh Chopra***

The project is based on Deep Learning, which is a sub field of machine learning, concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. In the project, TensorFlow and Keras is used, which is an library created for machine learning applications. It is used for detecting, identifying and tracking objects. The project uses a 2017 Tropical Cyclone Dataset in h5 extension, which contains approximately all common objects. The purpose of this project is to calculate the intensity of cyclone and classify them.

Key words : Deep Learning , AlexNet CNN, Tensorflow

“Where the vision is one

year, cultivate flowers;

Where the vision is ten years,

cultivate trees;

Where the vision is eternity,

cultivate people.”

- Oriental Saying

List of Figures

List of Tables

List of Abbreviations

Abbr1: CNN- Convolutional Neural Network

Abbr2: TC- Tropical Cyclone

Table of Contents

BIBLOGRAPHY…..

PROJECT PLAN…..

GUIDE INTERACTION SHEET….

SOURCE CODE……..

Chapter 1. Introduction

Introduction

Datasets of Cyclones captured by INSAT-3D over the Indian Oceans are available since 2012. These datasets can be used for training and testing of the Model. Traditional methods for Intensity estimation require accurate center determination for intensity estimation. Development of CNN based model for intensity estimation will be very useful during the initial stage of cyclone formation when determination of accurate center becomes difficult. A TC is said to be a high-speed rotating storm, characterized by a low-pressure centre with a closed low-level atmospheric movement of winds which produces heavy rain.

1.1 Overview

Development of a deep Convolutional Neural Network for Tropical Cyclone intensity estimation using INSAT-3D IR Images and development of a web application for visualization of the cyclonic images.

1.2 Background and Motivation

India currently has INSAT-3D Satellite that hovers on the Indian Ocean.

Every year, 1000s of families and their homes are washed away because of the inaccurate prediction of forecasting and intensity of the cyclones.

Since ISRO has given the challenge to predict the intensity of the upcoming cyclone of the future, given a satellite image captured by INSAT-3D, our model will use deep CNN and AlexNET to predict the intensity of the cyclone and the results can be used to mitigate the effects of the cyclone. AlexNET is a pre-defined model used for regression and classification problems of image datasets.

Through this prediction we also aim to utilize the real time data from the images of the cyclones, that will be useful for the government.

1.3 Problem Statement and Objectives

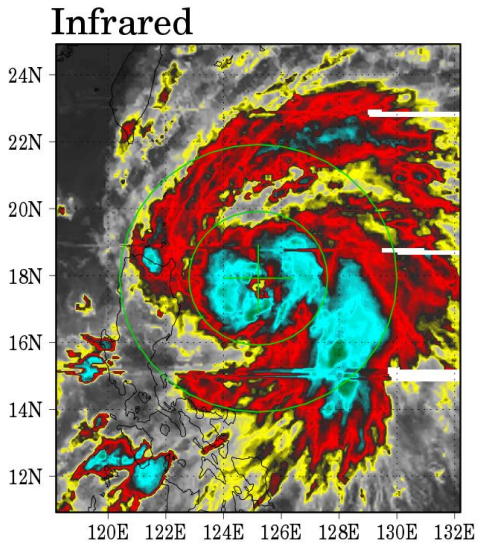
Development of a deep Convolutional Neural Network (CNN) for Tropical Cyclone intensity estimation using half-hourly INSAT-3D IR Images and development of a web application for visualization of the cyclonic images. INSAT3D/3DR observations are available at every 15-minute interval and these observations are very useful in understanding the instantaneous structural changes during evolution, intensification, and landfall of Tropical Cyclones. Datasets of Cyclones captured by INSAT-3D over the Indian Oceans are available since 2014. These datasets can be used for training and testing of the Model. Traditional

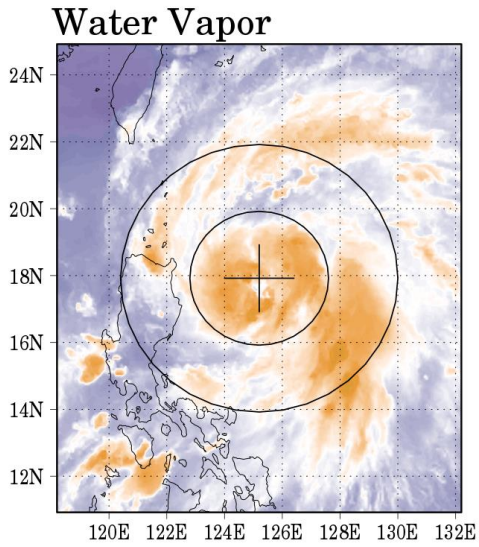
methods for Intensity estimation require accurate center determination for intensity estimation. Development of CNN based model for intensity estimation will be very useful during the initial stage of cyclone formation when determination of accurate center becomes difficult.

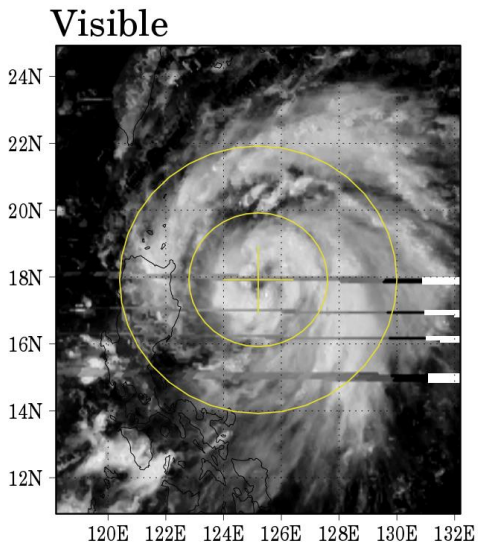
1.4 Scope of the Project

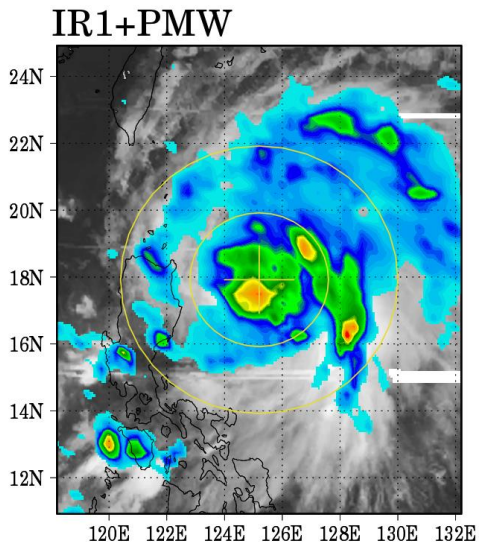
India will be able to predict the intensity of the cyclones which will in turn help in finding the places which will be adversely affected. As of now India is relying on the information provided by foreign satellites but with the help of our proposed solution Indian government will have their own prediction system. We can also fetch real time data if needed and will be able to get the intensities in the real time.

1.5 TCIR collects TC data from 4 channels of satellite images. TCIR aims to act as a benchmark dataset to help data scientists fairly evaluate the performance of TC intensity prediction models.









// global attributes:
:TC_ID = "200815W" ;
:TC_lon\ \(deg\) = 125.2 ;
:TC_lat\ \(deg\) = 17.9 ;
"2008090906" (UTC)
65. (kt)
56.2 (nmi)
974. (hPa)

1.6 Team Organization

- **Vedant Singh :**

  Along with doing preliminary investigation and understanding the limitations of current system, I studied about the topic and its scope and surveyed various research papers related to the object detection and the technology that is to be used.

- **V mohit Rao:**

  I also worked on the implementation of tensorflow framework and the working of counting of objects in the project.

  Worked on predefined datatype like h5 file . Documentation is also a part of the work done by me in this project.

- **Harshwardhan Songirkar:**

  I investigated and found the right technology and studied in deep about it. For the implementation of the project , I collected the object data and trained the model for it.

- **Ankita Pandey:**

  I have contributed in Implementation logic for the project objective and coding of internal functionalities. I have contributed in creation of the research paper , poster , report , also is also done by me.

1.7 Report Structure

The project ***Tropical cyclone intensity estimation*** is primarily concerned with the **Image processing in real-time** and the whole project report is categorized into five chapters.

Chapter 1: Introduction- introduces the background of the problem followed by rationale for the project undertaken. The chapter describes the objectives, scope and applications of the project. Further, the chapter gives the details of team members and their contribution in development of project which is then subsequently ended with report outline.

Chapter 2: Review of Literature- explores the work done in the area of Project undertaken and discusses the limitations of existing system and highlights the issues and challenges of project area. The chapter finally ends up with the requirement identification for present project work based on findings drawn from reviewed literature and end user interactions.

Chapter 3: Proposed System - starts with the project proposal based on requirement identified, followed by benefits of the project. The chapter also illustrate software engineering paradigm used along with different design representation. The chapter also includes block diagram and details of major modules of the project. Chapter also gives insights of different type of feasibility study carried out for the project undertaken. Later it gives details of the different deployment requirements for the developed project.

Chapter 4: Implementation - includes the details of different Technology/ Techniques/ Tools/ Programming Languages used in developing the Project. The chapter also includes the different user interface designed in project along with their functionality. Further it discuss the experiment results along with testing of the project. The chapter ends with evaluation of project on different parameters like accuracy and efficiency.

Chapter 5: Conclusion - Concludes with objective wise analysis of results and limitation of present work which is then followed by suggestions and recommendations for further improvement.

Chapter 2 . Review of Literature

Review of Literature

India currently has INSAT-3D Satellite that hovers on the Indian Ocean.Every year, 1000s of families and their homes are washed away because of the inaccurate prediction of forecasting and intensity of the cyclones.

Since ISRO has given the challenge to predict the intensity of the upcoming cyclone of the future, given a satellite image captured by INSAT-3D, our model will use deep CNN and AlexNET to predict the intensity of the cyclone and the results can be used to mitigate the effects of the cyclone. AlexNET is a pre-defined model used for regression and classification problems of image datasets.

Through this prediction we also aim to utilize the real time data from the images of the cyclones, that will be useful for the government.

2.1 Preliminary Investigation

2.1.1 Current System

Applications of satellite remote sensing from geostationary (GEO) and low earth orbital (LEO) platforms, especially from passive microwave (PMW) sensors, are focused on TC detection, structure, and intensity analysis as well as precipitation patterns. The impacts of satellite remote sensing on TC forecasts are discussed with respect to helping reduce the TC's track and intensity forecast errors. Finally, the multi-satellite-sensor data fusion technique is explained as the best way to automatically monitor and track the global TC's position, structure, and intensity.

2.2 Limitations of Current System

The limitations of these are as follows :

- Limitations in satellite IR/VIS observations at early stages of storm development when its center is not obvious or overcast..

2.3 Requirement Identification and Analysis for Project

The paper brings the idea about the Tropical cyclones (TCs) usually bring substantial economic losses and casualties through strong winds, heavy rainfall, and storm surge during and after TCs landfall, especially in the coastal area which is vulnerable to TCs (Zhang et al., 2009). Therefore, an accurate and timely forecast of TCs track and intensity is crucial for local disaster mitigation. During the past century, scientists and meteorological administrations tried to improve their ability in TCs observation technology, forecasting techniques, and understanding TCs intensification mechanism (Emanuel, 2018). In the past 30 years, with the application of meteorological satellites and the popularity of ensemble forecasts for TCs track, the track prediction has been greatly improved (Zhang and Krishnamurti, 1997; Fraedrich et al., 2003; Langmack et al., 2012; Jun et al., 2017). However, the intensity forecast is still poor compared with the track forecast and is a huge challenge around the world (DeMaria et al., 2014).

TCs intensity is influenced by two main physical processes (Wang and Wu, 2004; Elsberry et al., 2013), which are synoptic variables such as vertical wind shear, humidity, sea surface temperature, water vapor, divergency (DeMaria, 1996; Ge et al., 2013; Gao et al., 2016; Mercer and Grimes, 2017), and climatological and persistent variables such as latitude, longitude, Julian day, and sea-land ratio (SL ratio) (DeMaria and Kaplan, 1994a; Gao et al., 2016; Li et al., 2018). Statistical-dynamical models were used to predict TCs intensity and rapid intensification probability and outperformed the forecasts of individual physics-based dynamical models (Knaff et al., 2005; Kaplan et al., 2010; Gao and Chiu, 2012). For decades,

scientists have dedicated themselves to improving the skill of TCs intensity forecast. Jarvinen and Neumann (1979) proposed a statistical regression equation (Statistical Hurricane Intensity FORecast, SHIFOR) using predictors derived from climatic and persistent variables to predict TCs intensity changes over the North Atlantic Basin for the future 72 h. DeMaria and Kaplan (1994a) proposed a Statistical Hurricane Intensity Prediction Scheme (SHIPS), which considered more synoptic predictors to predict the changes in TCs intensity over the Atlantic Ocean Basin. The results showed that the average errors by SHIPS were 10%–15% smaller than the errors by the SHIFOR model that used only climatic and persistent variables. The storm decay over land was further considered by SHIPS (DeMaria et al., 2005). Besides the conventional synoptic and climatological variables, Li et al. (2018) paid close attention to the land effect on TCs intensity change by proposing a new factor involving the ratio of seawater area to land area (SL ratio) in the statistical regression model. TCs intensity changes over the entire TCs life span, including over the ocean basin, near the coast, and after landfall, were considered in the model (Li et al., 2018). Intensity forecasting accuracy for TCs near the coast and over land was improved with the addition of the SL ratio, compared with that of the models that did not consider the index of SL ratio (Li et al., 2018).

Based on previous observations and studies, in most cases, intensity changes are usually slow and steady over a certain period. However, in some situations, the TCs intensity may vary rapidly. Predicting these rapid intensifications is very challenging (Emanuel and Zhang, 2016). Many factors, such as the complex and chaotic energy exchange process between the sea and the atmosphere, and the imperfection of real-time data collection, may influence the prediction of TCs intensity change. These rapid intensification processes are challenging to explain, and therefore, neither numerical nor statistical-dynamic models can predict TCs intensity changes accurately (Chen et al., 2020). A vital weakness for numerical methods is insufficient representation of the complicated dynamical process; however,

increasing the number of variables or equations would exponentially demand the computation (Gao et al., 2016). On the other hand, statistical methods, usually based on regression and lower computational costs, may not be effective in capturing nonlinear relationships. Therefore, their forecast results need to be further improved (Lin et al., 2009; Sandery et al., 2010).

To solve these problems, scientists began to use machine learning (ML) to predict the TCs intensity in recent years (Gao et al., 2016; Cloud et al., 2019; Chen et al., 2020; Xu et al., 2021). Some ML algorithms, such as support vector machine, artificial neural network, decision tree, and random forest, have been gradually introduced into meteorology (Breiman, 1996; Mas and Flores, 2008; Behrangi et al., 2009; Mountrakis et al., 2011; Pan et al., 2019). ML is applicable to high-dimensional datasets and can deal with nonlinear relationships between predictors and predictand. It is used to explore satellite, radar, and in-situ data to improve the TCs intensity forecast skills (Gao et al., 2016; Zhang et al., 2016; Griffin et al., 2017; Jin et al., 2020). Generally, ML models combine historical storm characteristics, physically motivated information, and storm-specific features to make predictions (Xu et al., 2021).

The gradient boosted regression tree (GBRT) is an ensemble ML algorithm consisting of multiple decision trees, which is robust in model training (Zhang et al., 2020). First, the decision tree algorithm is a classical ML method, and it can deal with inherent nonlinear relationships in variables and missing values (Quinlan, 1987; Fayyad and Stolorz, 1997). In addition, the algorithm can quantify the relative importance of variables and establish decision rules for prediction (Quinlan, 1987). It has been successfully applied in the analyses of the re-curvature, landfall, and intensity change of TCs in the Western North Pacific Ocean (Zhang et al., 2013; Gao et al., 2016). Second, like most ensemble methods, a combination of decision trees can provide more robust and accurate regressions than a

single one. The greater-than and less-than structures of the tree module make the GBRT less affected by outliers (Bishop and Nasrabadi, 2006; Ma et al., 2018). Furthermore, GBRT is able to capture the complicated and nonlinear relationships between TCs intensity change and other related features. Compared with a single decision tree, GBRT pays more attention to the regression errors with less calculation time for high-dimension data (Xie and Coggeshall, 2010; Ding et al., 2016; Yang et al., 2016; Ma et al., 2017; Ma et al., 2018). Therefore, GBRT is used in this study to predict the future change of TCs intensity in the Western North Pacific Ocean.

Most of the studies mainly focused on TCs intensity change over the open ocean. However, TCs that make landfall or approach the coast usually cause most of the loss of life and damage. Forecasting the intensity of TCs near the coast and over land should be, therefore, more critical than forecasting TCs intensity over the open ocean (Li et al., 2018). Li et al. (2018) explored the TCs intensity change over the entire TCs life span by considering a new factor, the "ratio of seawater area to the land area," in the multiple linear regression model (MLR). However, MLR usually performs poorly in handling the nonlinear relationship between predictors and predictand. In this study, the GBRT is proposed to forecast future TCs intensity at 12-, 24-, 36-, 48-, 60-, and 72-h forecasting lead time in the Western North Pacific region for different TCs life spans. GBRT's performance in predicting TCs intensity change over the Western North Pacific is compared with the performance of the MLR model used by Li et al. (2018). The remaining of the paper is as follows. The data and methodology are described in Section 2. Section 3 contains the results and discussion. In the final section, summaries are presented.

2.3.1 Conclusion

This chapter reviews the literature surveys that have been done during the research work. The related work that has been proposed by many researchers has been discussed. The research papers related to Tropical Cyclone Intensity Prediction from 2015 to 2017 have been shown which discussed about different methods and algorithm to identify objects.

Chapter 3 . Proposed System

Proposed System

3.1 The Proposal

India currently has INSAT-3D Satellite that hovers on the Indian Ocean.

Every year, 1000s of families and their homes are washed away because of the inaccurate prediction of forecasting and intensity of the cyclones.

Since ISRO has given the challenge to predict the intensity of the upcoming cyclone of the future, given a satellite image captured by INSAT-3D, our model will use deep CNN and AlexNET to predict the intensity of the cyclone and the results can be used to mitigate the effects of the cyclone. AlexNET is a pre-defined model used for regression and classification problems of image datasets.

Through this prediction we also aim to utilize the real time data from the images of the cyclones, that will be useful for the government

3.2 Benefits of the Proposed System

- The advantages of geostationary satellites (GEO) are frequent observations over a large domain

  Applications of satellite remote sensing from geostationary (GEO) and low earth orbital (LEO) platforms, especially

  from passive microwave (PMW) sensors, are focused on TC detection, structure, and intensity analysis as well as

  precipitation patterns. The impacts of satellite remote sensing on TC forecasts are discussed with respect

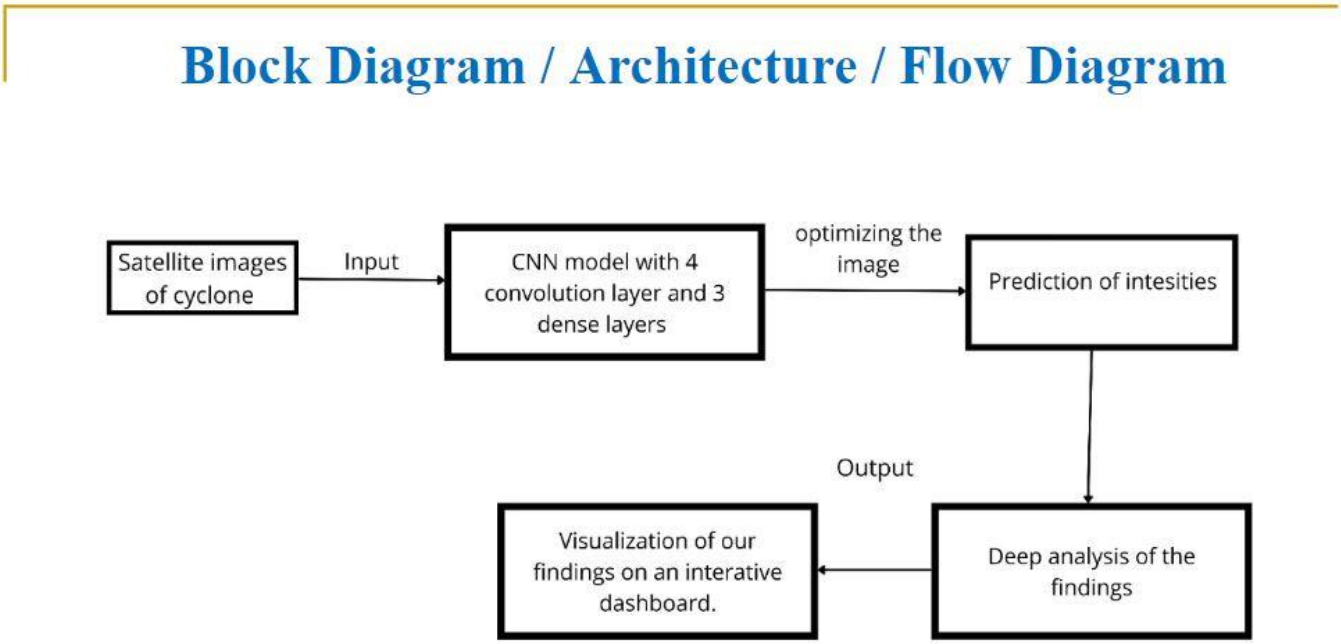
  to helping reduce the TC's track and intensity forecast errors. Finally,

the multi-satellite-sensor data fusion technique

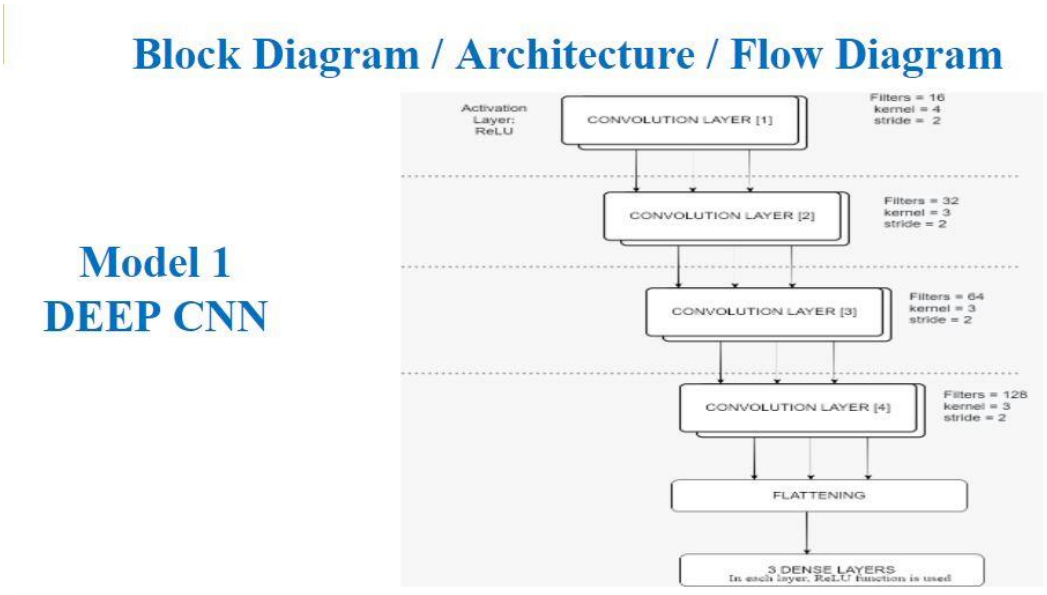
is explained as the best way to automatically monitor and track the global TC's position, structure, and intensity.

3.3 Block Diagram

Block Diagram of overall model

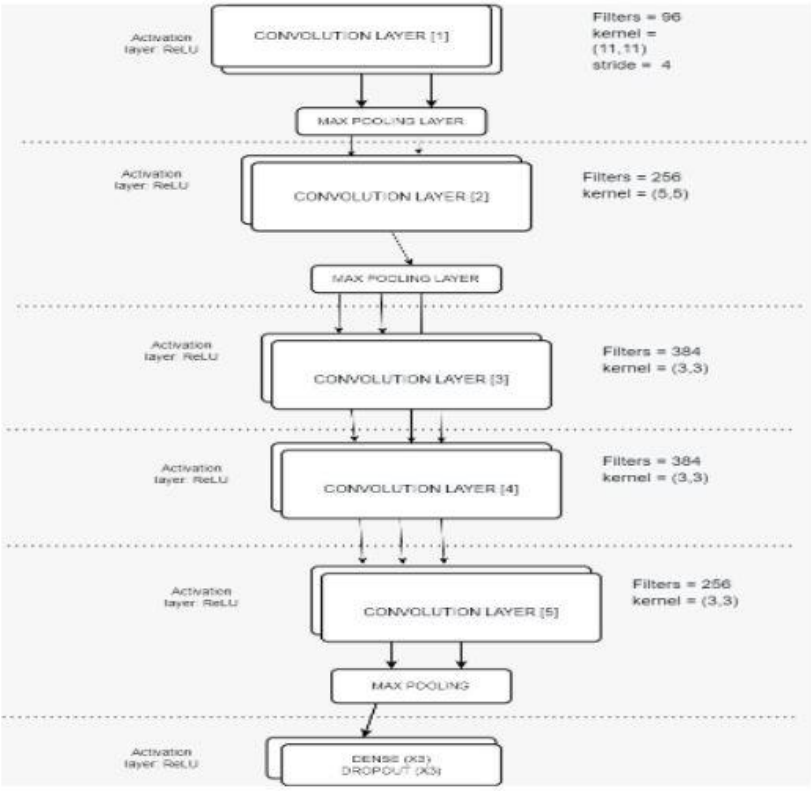


Block Diagram of Deep Learning mechanism:



Block Diagram of AlexNet CNN :

Model 2
Alex NET



3.4 Feasibility Study

A feasibility study is an analysis of how successfully a system can be implemented, accounting for factors that affect it such as economic, technical and operational factors to determine its potential positive and negative outcomes before investing a considerable amount of time and money into it.

3.4.1 Technical

For getting the dataset to run in the model we need the INSAT-3D IR Images (from the satellite) , there is a need to process images from the dataset . For this, the kind of framework used must be the one that is capable of extracting those objects from the images easily and accurately in real-time. The framework used in this is Tensorflow, which is a framework designed by Google for efficiently dealing with deep learning and concepts like neural networks , making the system technically feasible.

The system once set up completely, works automatically without needing any person to operate it. The result (intensity and classification)of the cyclone will automatically displayed For making the system technically feasible, there is a requirement of GPU built system with high processor for better performance.

.

3.4.2 Economical

For smooth working of the model the satellite needs to capture 4 different types of observation every 15 minutes

1. Infrared Image
2. Water vapor intensity image
3. Visible image
4. Passive microwave image

The TensorFlow framework used in the system works great with GPU built systems, which are a little on the expensive side.

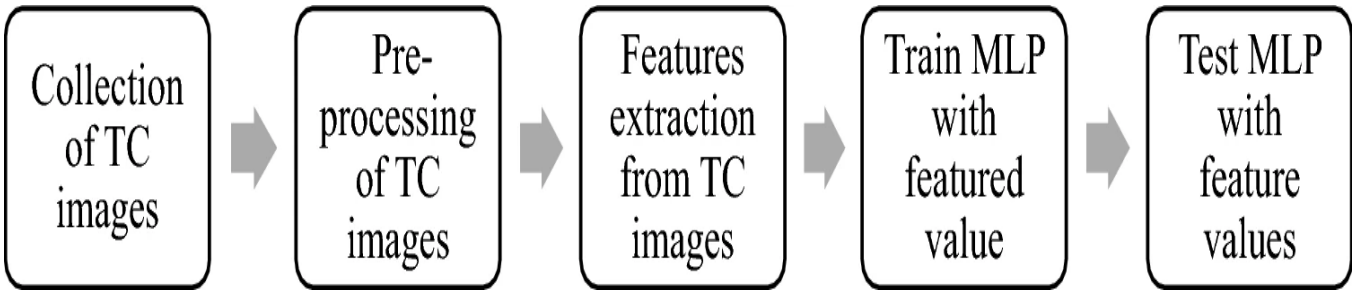
Since the system uses high performance processors continuously, so to save any disaster from occurring due to very high temperatures, there is a requirement of a cooling system in the environment where it is implemented.

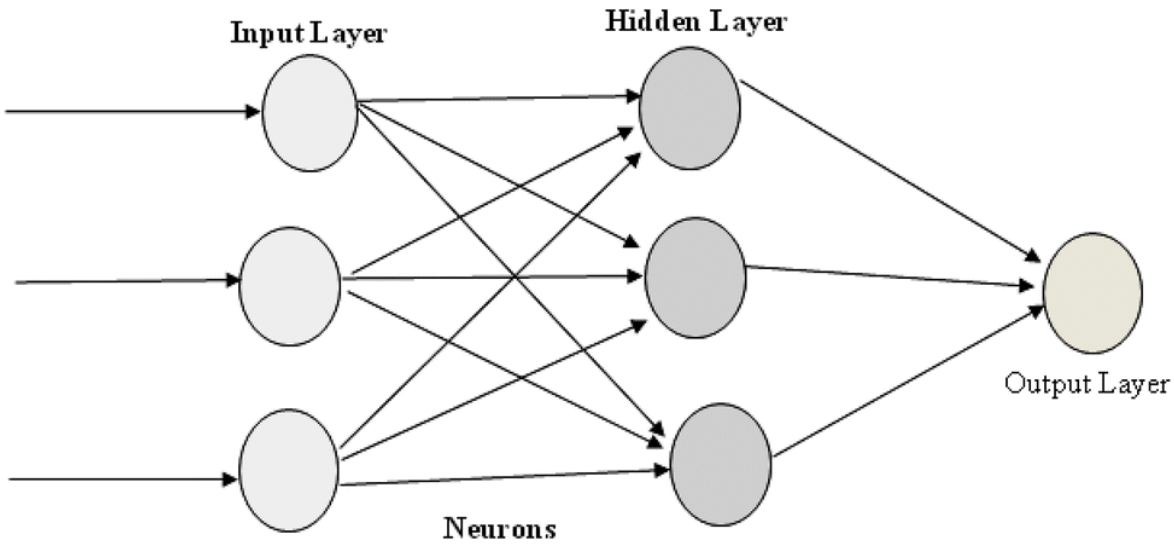
3.4.3 Operational

The main motto of our system is predict the intensity and classify the tropical cyclone and reduce the natural calamity damage caused by the cyclone .

The system is able to do that accurately and efficiently making the system operationally feasible.

3.5 Design Representation





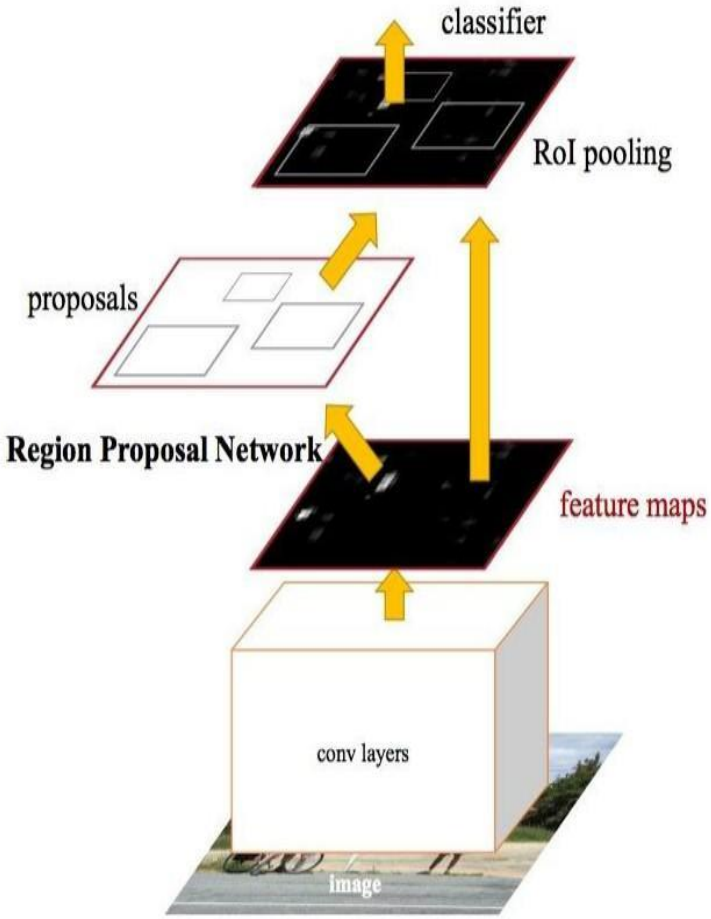


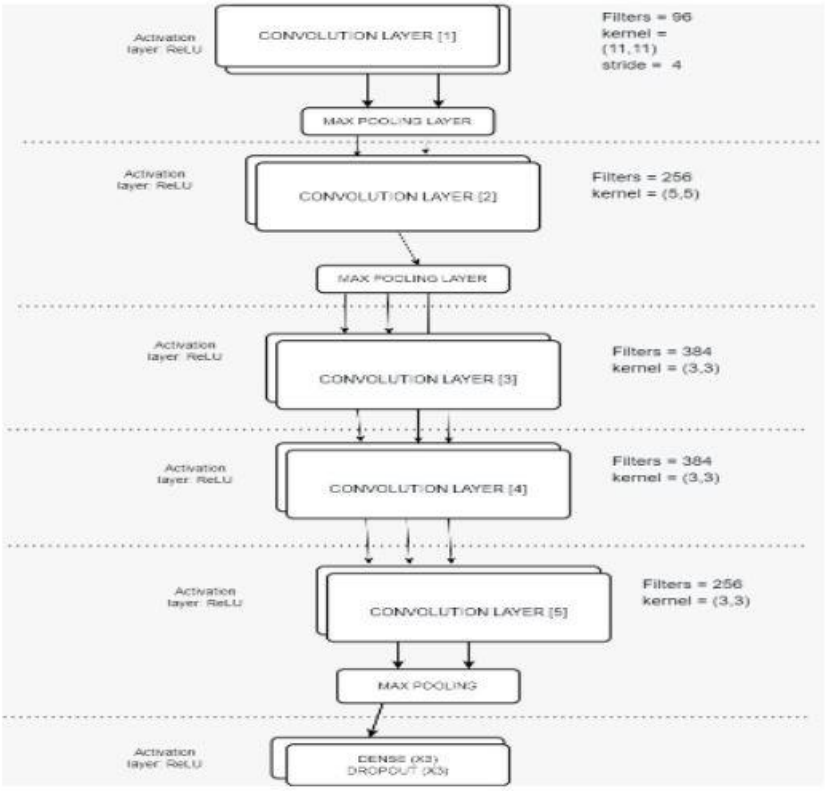
Figure 3-5 : Faster CNN Architecture

Used data type :

ID	data_set	lon	lat	time	Vmax	R35_4qAVG	MSLP
201701E	21	21	21	21	21	21	21
201701I	23	23	23	23	23	23	23
201701L	53	53	53	53	53	53	53
201701S	47	47	47	47	47	47	47
201701W	67	67	67	67	67	67	67

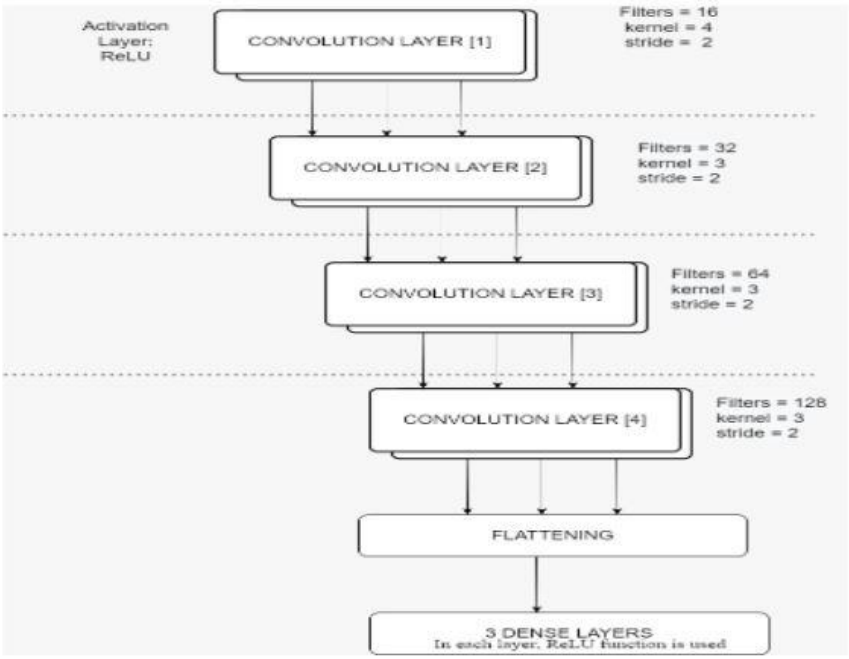
Table 1 : Types of Datasets

Model 2
Alex NET

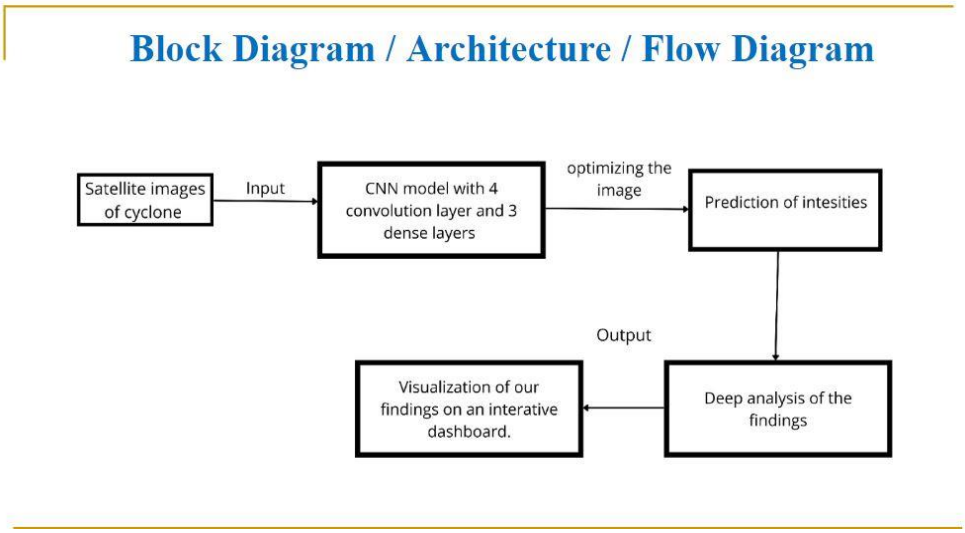


Block Diagram / Architecture / Flow Diagram

Model 1
DEEP CNN



3.5.1 Data Flow Diagrams



3.5.2 Database Structure

TC images are collected from Institute of Meteorological Satellite Studies and IMD. The images were taken from satellite known as INSAT 3D, Kalpana1 and Meteosat-7. The complete dataset is divided into the three categories based on the intensity

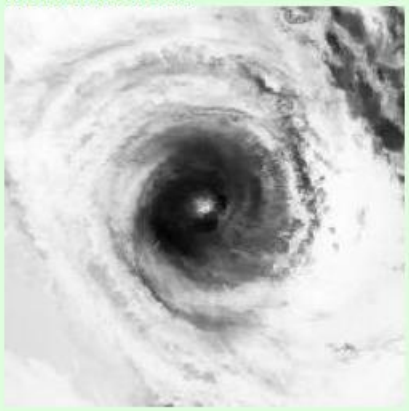
S. no.	Name given due to intensity	Intensity
1	Depression	31–49 km/h (17–27 knots)
2	Deep depression	50–60 km/h (28–33 knots)
3	Cyclonic storm	62–87 km/h (34–47 knots)
4	Severe cyclonic storm	88–117 km/h (48–63 knots)
5	Very severe cyclonic storm	118–166 km/h (64–90 knots)
6	Extremely severe cyclonic storm	167–221 km/h (91–119 knots)

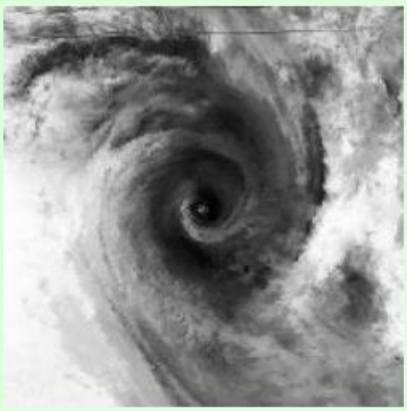
Data Statistics

Region	#TCs	#Frames
Atlantic	235	13707
West Pacific	379	20061
East Pacific	247	13615
Central Pacific	19	1479
Indian Ocean	75	3205
Southern Hemisphere	330	18434
Total	***1285***	***70501***

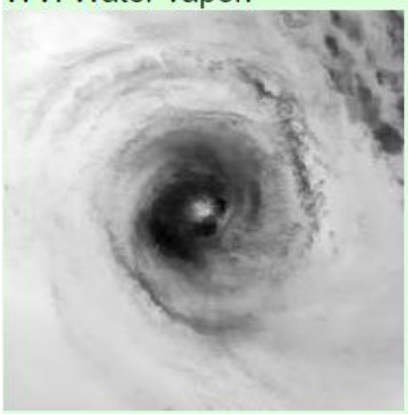
Frames :

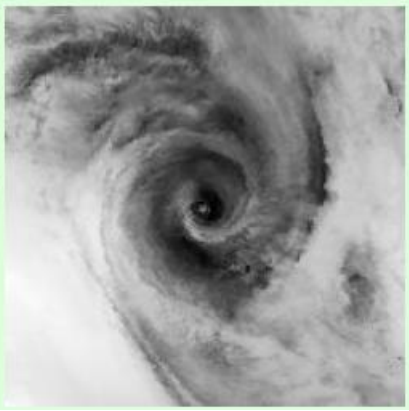
- IR1: Infrared.



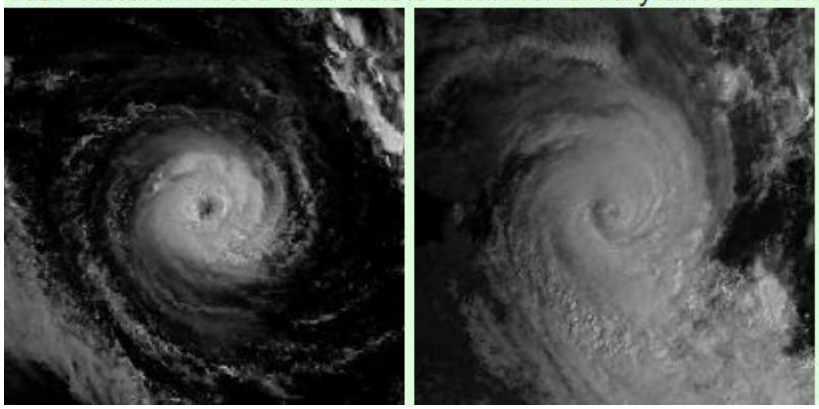


WV: Water vapor.

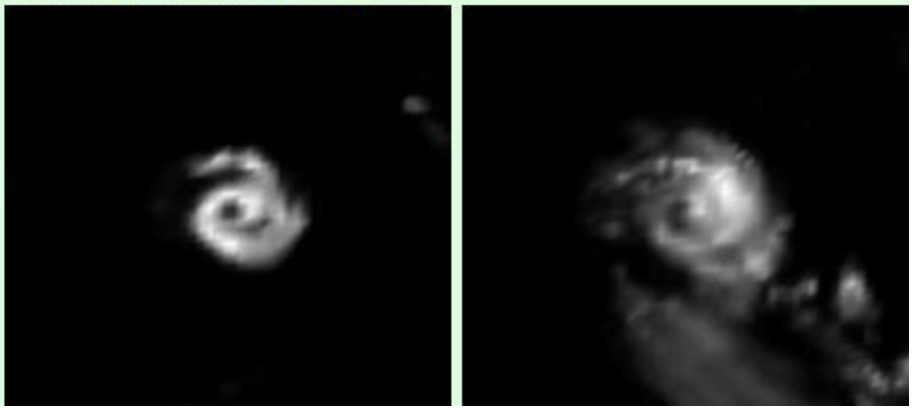




VIS: Visible. Noted that visible channel is very unstable because of the daylight.



PMW: Passive microwave.



3.6 Deployment Requirements

There are various requirements (hardware, software and services) to successfully deploy the system. These are mentioned below :

3.6.1 Hardware

- 64-bit, x86 Processing system
- Windows 10 or later operating system
- High processing computer system with GPU(high performance)
- 512 – GB SDD for virtual memory
  32 GB Ram or greater

3.6.2 Software

- Keras
- Python and its supported libraries
- Tensor Flow
- If Installing Tensorflow in GPU systems :
    1. CUDA® Toolkit 9.0.
    2. The NVIDIA drivers associated with CUDA Toolkit 9.0. cuDNN v7.0.
    3. GPU card with CUDA Compute Capability 3.0 or higher

Chapter 4 . Implementation

Implementation

The proposed model here is used to train and test the feature values of TC images through multilayer perceptron (MLP). MLP is considered to be a supervised learning technique. MLP is a class of feed forward artificial neural network and it consists of input layer, output layer and at least one hidden layer. In MLP each node treated as a neuron which follows a non-linear activation function except for input nodes. It uses a supervised learning procedure called backpropagation algorithm and used for training and testing of the model.In this paper Sect. 2 describes the dataset. Section 3 describes the methods and implementation details of the proposed algorithm. Section 4 is devoted to results and discussion and Sect. 5 concludes the proposed work.

4.1 Technique Used

4.1.1 Deep- Learning

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised. Deep learning models are loosely related to information processing and communication patterns in a biological nervous system, such as neuralcoding that attempts to define a relationship between various stimuli and associated neuronal responses in the brain.

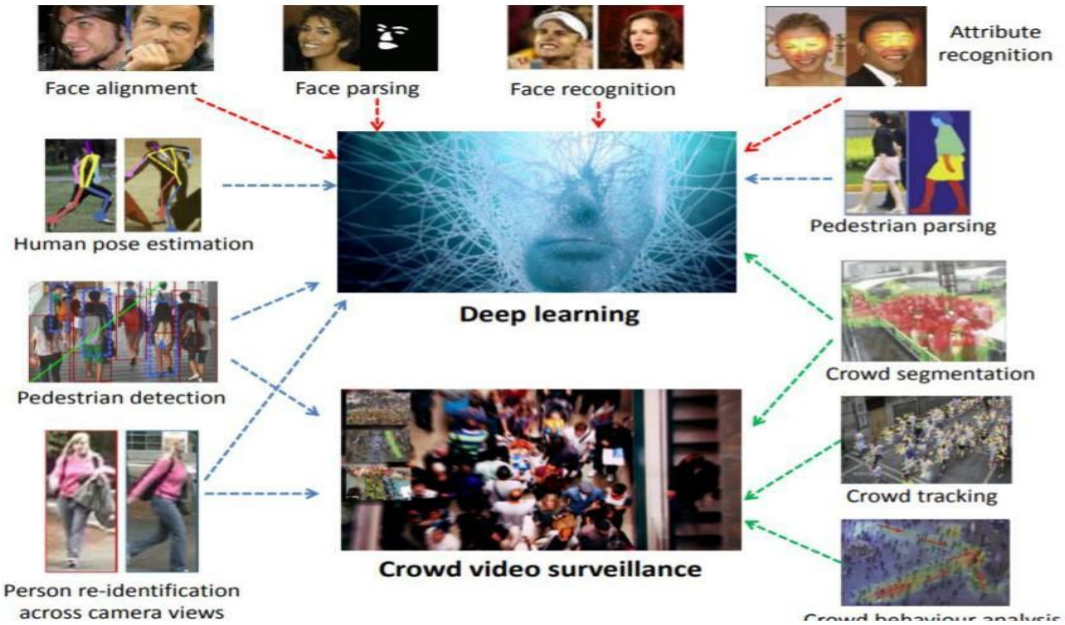


Figure 4-1 : Deep Learning

Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics and drug design, where they have produced results comparable to and in some cases superior to human experts.

4.1.2 Neural Networks :

In machine learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery.

CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared- weights architecture and translation invariance characteristics.

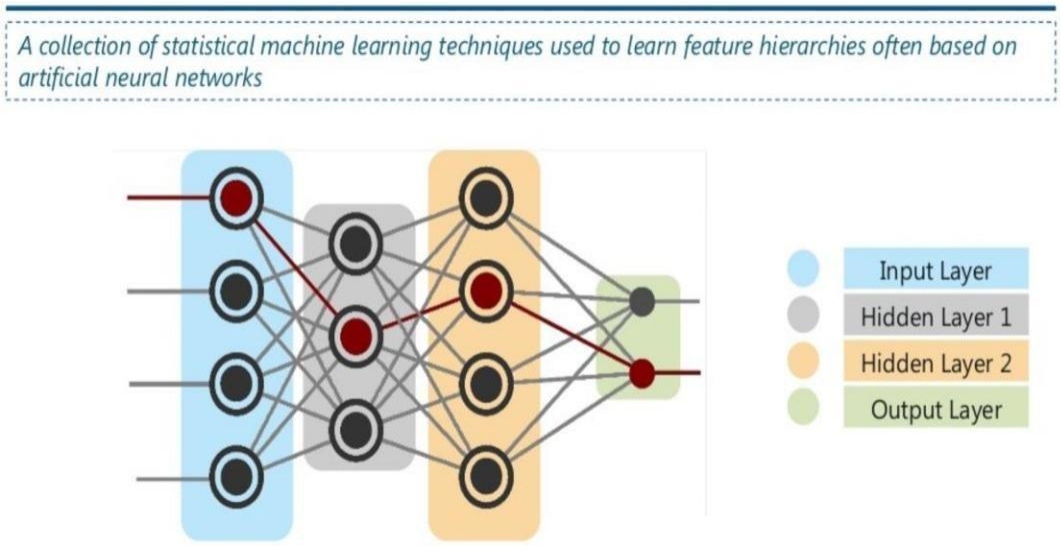


Figure 4-2 : Neural Networks

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage. They have applications in image and video recognition, recommender systems and natural language processing.

4.2 Tools Used

4.2.1 Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML.

Keras in general accepts three types of inputs — NumPy arrays, TensorFlow Dataset objects, and Python generators. A NumPy array pertains to a low-level array representation of the data, a Dataset object produces a high-level representation, and a Generator pertains to a batch of data with certain properties.

Keras is used by CERN, NASA, NIH, and many more scientific organizations around the world (and yes, Keras is used at the LHC). Keras has the low-level flexibility to implement arbitrary research ideas while offering optional high-level convenience features to speed up experimentation cycles.

4.2.2 Tensor Flow

TensorFlow is an open-source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

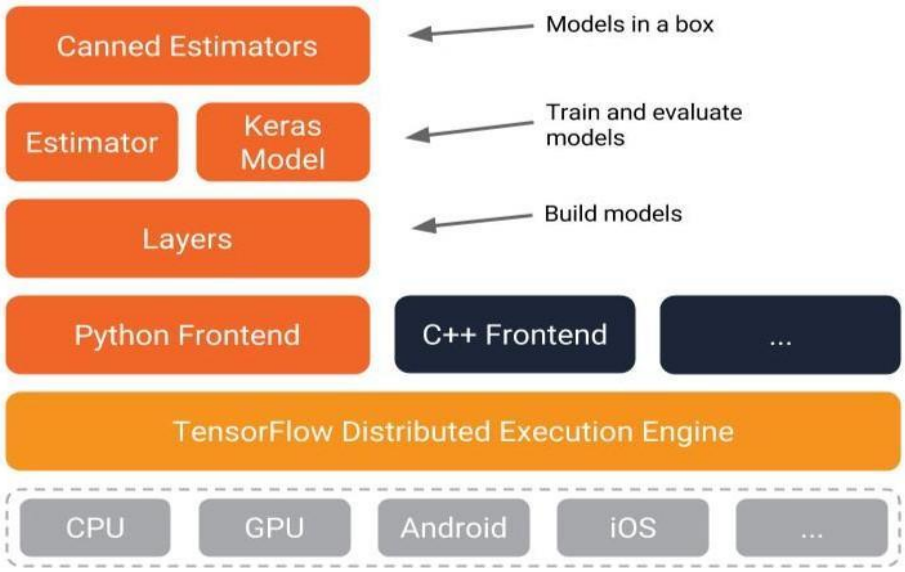


Figure 4-3 : TensorFlow Architecture

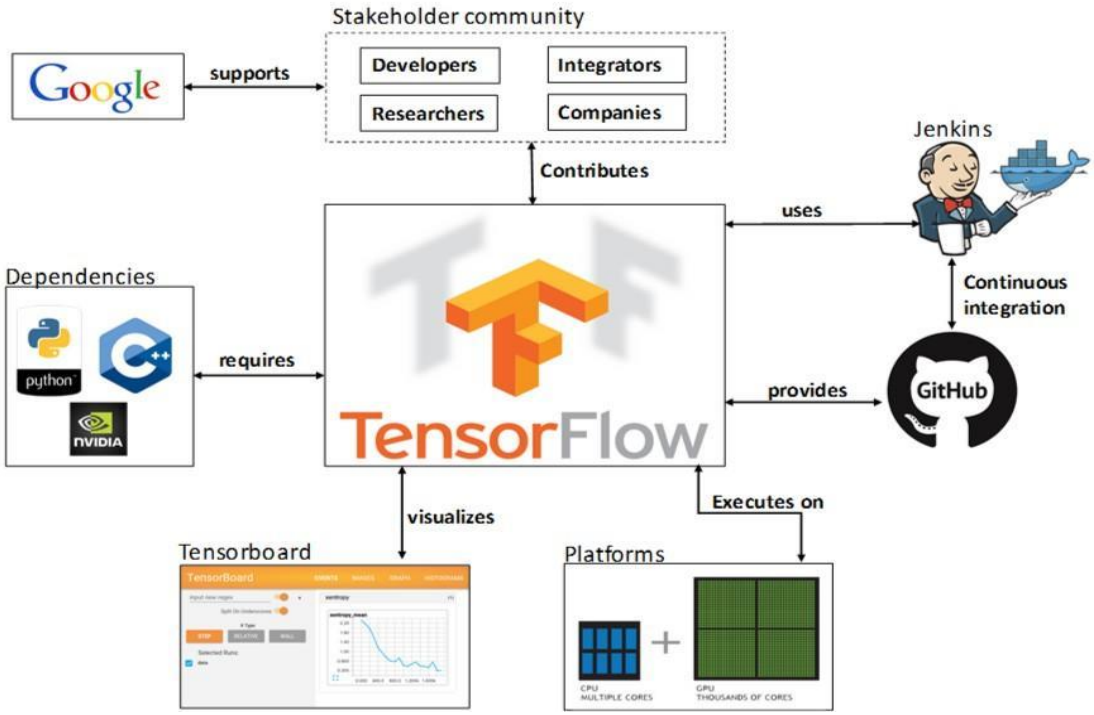


Figure 4-4 : TensorFlow Working

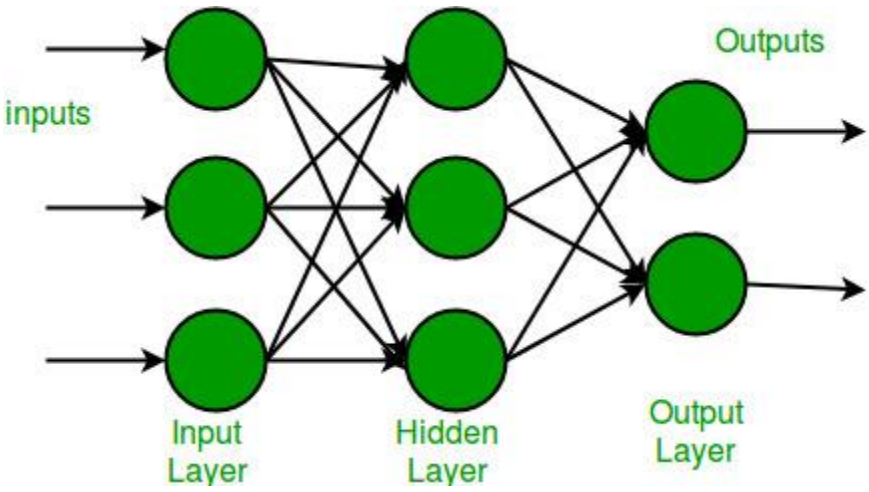
4.2.3 Models

The TensorFlow official models are a collection of example models that use TensorFlow's high-level APIs. They are intended to be well-maintained, tested, and kept up to date with the latest TensorFlow API. They should also be reasonably optimized for fast performance while still being easy to read.

PROPSED MODE l USES MULTILAYER -PERCEPTRON MODEL-

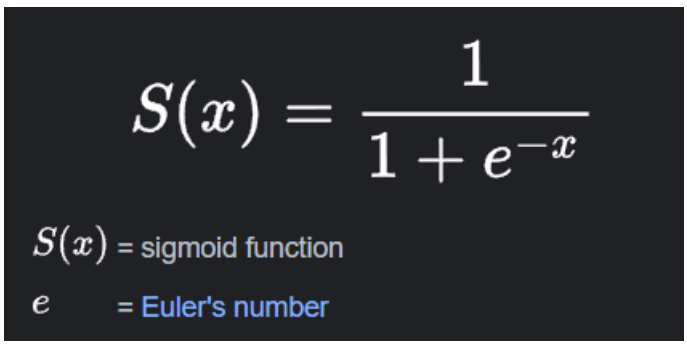
Multi-layer perception is also known as MLP. It is fully connected dense layers, which transform any input dimension to the desired dimension. A multi-layer perception is a neural network that has multiple layers. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons.

A multi-layer perceptron has one input layer and for each input, there is one neuron(or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes. A schematic diagram of a Multi-Layer Perceptron (MLP) is depicted below.



In the multi-layer perceptron diagram above, we can see that there are three inputs and thus three input nodes and the hidden layer has three nodes. The output layer gives two outputs, therefore there are two output nodes. The nodes in the input layer take input and forward it for further process, in the diagram above the nodes in the input layer forwards their output to each of the three nodes in the hidden layer, and in the same way, the hidden layer processes the information and passes it to the output layer.

Every node in the multi-layer perception uses a sigmoid activation function. The sigmoid activation function takes real values as input and converts them to numbers between 0 and 1 using the sigmoid formula



4.3 Language Used

Python language is used in the system due to the following Characteristics:

Simple:

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English (but very strict English!). This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the syntax i.e. the language itself.

Free and Open Source :

Python is an example of a FLOSS (Free/Libre and Open Source Software). In simple terms, you can freely distribute copies of this software, read the software's source code, make changes to it, use pieces of it in new free programs, and that you know you can do these things. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and improved by a community who just want to see a better Python.

Object Oriented :

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simple way of doing object-oriented programming, especially, when compared to languages like C++ or Java.

Extensive Libraries :

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, ftp, email, XML,

XML-RPC, HTML, WAV files, cryptography, GUI(graphical user interfaces) using Tk, and also other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the "batteries included" philosophy of Python.

4.4 Screenshots

The Following are the screenshots of the result of the project:

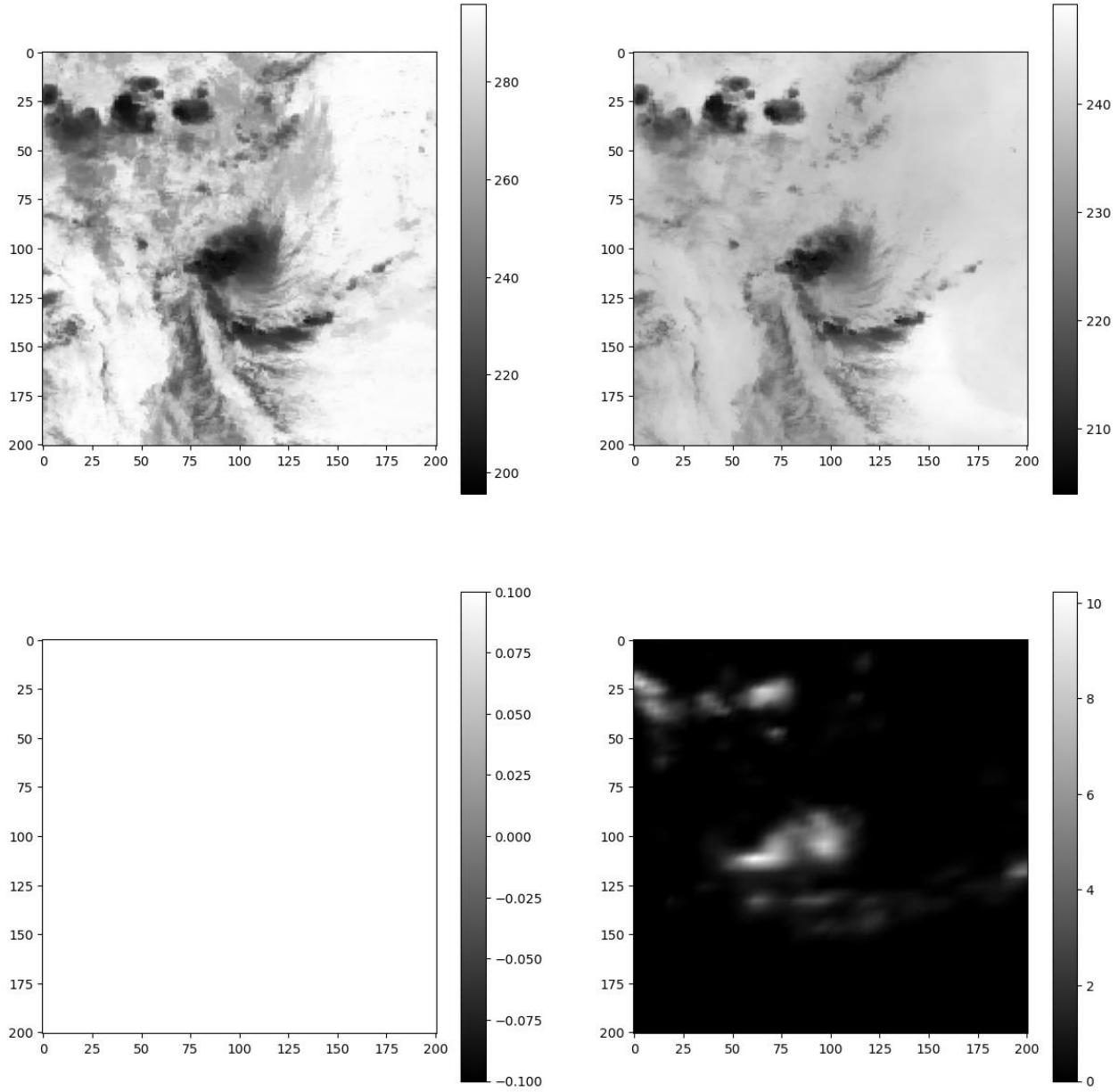


Figure 4-9 : Screenshot 1

	Intensity	Category
0	48	Tropical Storm
1	31	Tropical Depression
2	28	Tropical Depression
3	47	Tropical Storm
4	52	Tropical Storm
...	...	...
408	35	Tropical Storm
409	31	Tropical Depression
410	26	Tropical Depression
411	29	Tropical Depression
412	86	Typhoon

413 rows × 2 columns

Alexnet Output

	Intensity	Category
0	54	Tropical Storm
1	26	Tropical Depression
2	24	Tropical Depression
3	62	Tropical Storm
4	33	Tropical Depression
...	...	...
408	40	Tropical Storm
409	34	Tropical Storm
410	33	Tropical Depression
411	31	Tropical Depression
412	91	Typhoon

413 rows × 2 columns

Deeplearning Output

```
15/15 [==============================] - 36s 2s/step - loss: 50343.2461 - mean_squared_error: 50343.2461
Epoch 2/20
15/15 [==============================] - 37s 2s/step - loss: 902.4918 - mean_squared_error: 902.4918
Epoch 3/20
15/15 [==============================] - 36s 2s/step - loss: 854.9243 - mean_squared_error: 854.9243
Epoch 4/20
15/15 [==============================] - 35s 2s/step - loss: 853.6540 - mean_squared_error: 853.6540
Epoch 5/20
15/15 [==============================] - 34s 2s/step - loss: 835.4474 - mean_squared_error: 835.4474
Epoch 6/20
15/15 [==============================] - 35s 2s/step - loss: 829.1916 - mean_squared_error: 829.1916
Epoch 7/20
15/15 [==============================] - 34s 2s/step - loss: 817.2757 - mean_squared_error: 817.2757
Epoch 8/20
15/15 [==============================] - 36s 2s/step - loss: 812.0493 - mean_squared_error: 812.0493
Epoch 9/20
15/15 [==============================] - 35s 2s/step - loss: 828.4761 - mean_squared_error: 828.4761
Epoch 10/20
15/15 [==============================] - 36s 2s/step - loss: 795.8121 - mean_squared_error: 795.8121
Epoch 11/20
15/15 [==============================] - 35s 2s/step - loss: 784.3663 - mean_squared_error: 784.3663
Epoch 12/20
15/15 [==============================] - 35s 2s/step - loss: 746.6233 - mean_squared_error: 746.6233
Epoch 13/20
15/15 [==============================] - 36s 2s/step - loss: 739.4029 - mean_squared_error: 739.4029
...
Epoch 20/20
15/15 [==============================] - 34s 2s/step - loss: 460.9899 - mean_squared_error: 460.9899
15/15 [==============================] - 1s 55ms/step - loss: 393.0228 - mean_squared_error: 393.0228
Val Score:  [393.0228271484375, 393.0228271484375]
```

Alexnet Deep learning Training Cycles

```
Epoch 1/10

c:\Users\HP\anaconda3\lib\site-packages\keras\optimizers\optimizer_v2\adam.py:110: UserWarning: The `lr` argument
instead.
  super(Adam, self).__init__(name, **kwargs)

58/58 [==============================] - 4s 49ms/step - loss: 872.8423 - mean_squared_error: 872.8346
Epoch 2/10
58/58 [==============================] - 3s 47ms/step - loss: 700.3980 - mean_squared_error: 700.3914
Epoch 3/10
58/58 [==============================] - 3s 47ms/step - loss: 558.1287 - mean_squared_error: 558.1227
Epoch 4/10
58/58 [==============================] - 3s 46ms/step - loss: 501.4357 - mean_squared_error: 501.4298
Epoch 5/10
58/58 [==============================] - 3s 45ms/step - loss: 438.3375 - mean_squared_error: 438.3316
Epoch 6/10
58/58 [==============================] - 3s 46ms/step - loss: 381.7656 - mean_squared_error: 381.7597
Epoch 7/10
58/58 [==============================] - 3s 46ms/step - loss: 352.1645 - mean_squared_error: 352.1585
Epoch 8/10
58/58 [==============================] - 3s 47ms/step - loss: 328.6382 - mean_squared_error: 328.6321
Epoch 9/10
58/58 [==============================] - 3s 46ms/step - loss: 300.2810 - mean_squared_error: 300.2750
Epoch 10/10
58/58 [==============================] - 3s 46ms/step - loss: 280.8199 - mean_squared_error: 280.8138
15/15 [==============================] - 0s 9ms/step - loss: 279.3967 - mean_squared_error: 279.3906
Val Score:  [279.39666748046875, 279.3905944824219]
```

k-fold Training

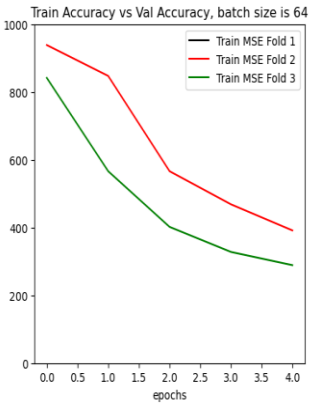
```
Training on Fold:  1
Epoch 1/5

c:\Users\HP\anaconda3\lib\site-packages\keras\optimizers\optimizer_v2\adam.py:110: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)

Output exceeds the size limit. Open the full output data in a text editor
96/96 [==============================] - 5s 28ms/step - loss: 2556.8972 - mean_squared_error: 2556.8933
Epoch 2/5
96/96 [==============================] - 3s 28ms/step - loss: 2556.8943 - mean_squared_error: 2556.8933
Epoch 3/5
96/96 [==============================] - 3s 28ms/step - loss: 2556.8933 - mean_squared_error: 2556.8933
Epoch 4/5
96/96 [==============================] - 3s 28ms/step - loss: 2556.8933 - mean_squared_error: 2556.8933
Epoch 5/5
96/96 [==============================] - 3s 28ms/step - loss: 2127.4778 - mean_squared_error: 2127.4778
48/48 [==============================] - 1s 10ms/step - loss: 1741.6434 - mean_squared_error: 1741.6433
Val Score:  [1741.6434326171875, 1741.643310546875]
================================================================================


Training on Fold:  2
Epoch 1/5
96/96 [==============================] - 4s 31ms/step - loss: 938.3958 - mean_squared_error: 938.3884
Epoch 2/5
96/96 [==============================] - 3s 29ms/step - loss: 847.3077 - mean_squared_error: 847.3015
Epoch 3/5
96/96 [==============================] - 3s 29ms/step - loss: 566.1224 - mean_squared_error: 566.1167
Epoch 4/5
96/96 [==============================] - 3s 30ms/step - loss: 468.4176 - mean_squared_error: 468.4116
Epoch 5/5
96/96 [==============================] - 3s 29ms/step - loss: 391.4813 - mean_squared_error: 391.4752
...
Val Score:  [383.0190734863281, 383.0136413574219]
================================================================================
```

k-fold output



MSE on validation sets: [1741, 344, 383]

mean MSE on validation set: 823

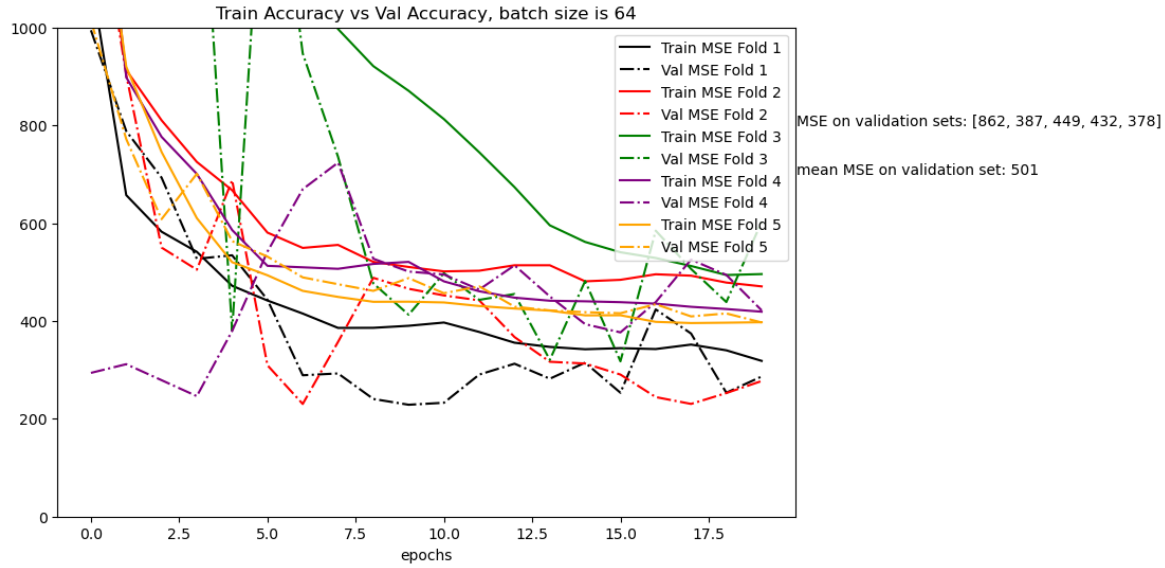
Output of eda (model building)

```
Training on Fold:  1

c:\Users\HP\anaconda3\lib\site-packages\keras\optimizers\optimizer_v2\adam.py:110: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)

Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/20
52/52 [==============================] - 6s 94ms/step - loss: 1102.1864 - mean_squared_error: 1102.1787 - val_loss: 994.2404 - val_mean_squared_error: 994.2334
Epoch 2/20
52/52 [==============================] - 4s 84ms/step - loss: 657.2084 - mean_squared_error: 657.2021 - val_loss: 788.1802 - val_mean_squared_error: 788.1741
Epoch 3/20
52/52 [==============================] - 4s 73ms/step - loss: 582.4577 - mean_squared_error: 582.4519 - val_loss: 693.0638 - val_mean_squared_error: 693.0584
Epoch 4/20
52/52 [==============================] - 4s 81ms/step - loss: 541.3077 - mean_squared_error: 541.3024 - val_loss: 527.0383 - val_mean_squared_error: 527.0330
Epoch 5/20
52/52 [==============================] - 4s 84ms/step - loss: 471.8490 - mean_squared_error: 471.8440 - val_loss: 534.2418 - val_mean_squared_error: 534.2367
Epoch 6/20
52/52 [==============================] - 4s 75ms/step - loss: 441.7766 - mean_squared_error: 441.7714 - val_loss: 443.4190 - val_mean_squared_error: 443.4138
Epoch 7/20
52/52 [==============================] - 4s 72ms/step - loss: 415.1017 - mean_squared_error: 415.0965 - val_loss: 288.9467 - val_mean_squared_error: 288.9414
Epoch 8/20
52/52 [==============================] - 4s 72ms/step - loss: 385.7546 - mean_squared_error: 385.7492 - val_loss: 292.6857 - val_mean_squared_error: 292.6804
Epoch 9/20
52/52 [==============================] - 4s 73ms/step - loss: 385.9480 - mean_squared_error: 385.9427 - val_loss: 240.5040 - val_mean_squared_error: 240.4986
Epoch 10/20
52/52 [==============================] - 4s 70ms/step - loss: 390.1136 - mean_squared_error: 390.1082 - val_loss: 228.6532 - val_mean_squared_error: 228.6478
Epoch 11/20
52/52 [==============================] - 4s 71ms/step - loss: 396.7584 - mean_squared_error: 396.7529 - val_loss: 232.7393 - val_mean_squared_error: 232.7339
Epoch 12/20
52/52 [==============================] - 4s 73ms/step - loss: 377.4651 - mean_squared_error: 377.4597 - val_loss: 290.3572 - val_mean_squared_error: 290.3518
Epoch 13/20
...
Val Score:  [378.0522766113281, 378.046173095703]
================================================================================
```

Testing strategy used (mean square error)



We have used MSE(Mean Square Error) to test our model at every cycle of running processes (epouch) the model at every cycle tends to reduce the mean Square Error and at last cycle the best value we get for the mean square value is taken as the final consideration .

4.5 Testing

Testing is the process of evaluation of a system to detect differences between given input and expected output and also to assess the feature of the system. Testing assesses the quality of the product. It is a process that is done during the development process. .

4.5.1 Strategy Used

Multilayer perceptron for Testing .

MLP is an artificial neural network which is used to train and test the proposed model. MLP is a feedforward network with one or more hidden layers [31]. It uses backpropagation algorithm for finding the

gradient (Fig. 4)

In MLP the gradient is determined by the backpropagation algorithm. The change in weight is calculated by multiplying the gradient with the learning rate and added to the previous change in the weight multiplied by momentum. The standard formula for perceptron learning is given below:

$$S(x) = \frac{1}{1 + e^{-x}}$$

$S(x)$ = sigmoid function

e = Euler's number

The above-mentioned sigmoid function is used as an activation function where $0 < \upsilon(x) < 1$, a is a constant to control the activation function.

The error function for each unknown training pattern p is calculated based on following formula:

$$\mathrm{r_p} = \sum_{i=1}^{n} (o_{ip} - t_{ip})^2$$

where rp is the error calculated based on output oip and target tip.

The purpose of backpropagation algorithm is to minimize the error gradually by adjusting the weight values.

$$\mathrm{w_{(next)}} = \mathrm{w} + \Delta \mathrm{w}$$

$$\Delta \mathrm{w} = \eta \times \mathrm{G} + \alpha \times \Delta \mathrm{w_{(previous)}}$$

Here, η is learning rate $0 < \eta < 1$, G is gradient and α stands for momentum factor $0 < \alpha < 1$.

4.5.2 Test Case and Analysis

TEST CASE: 1

Attributes	Test mode	Values
		MLP
Correctly classified instances	Tenfold cross-validation	83.6412%
Incorrectly classified instances		16.3588%
Kappa statistic		0.734
Mean absolute error		0.1392
Root mean squared error		0.2895
Relative absolute error		33.997%
Root relative squared error		64.0036%
Correctly classified instances	Split 66.0% train, remainder test	82.9457%
Incorrectly classified instances		17.0543%
Kappa statistic		0.7255
Mean absolute error		0.142
Root mean squared error		0.3034
Relative absolute error		34.5189%
Root relative squared error		66.518%

Table 4 (a) Summary of test results confusion matrix for test mode: tenfold cross-validation, (b) confusion matrix for test mode: 66% split

Name of the classifier	Classified as			Classes
	ESCS	CS	D	
(a)				
MLP	64	9	0	ESCS
	19	167	8	CS
	4	22	86	D
(b)				
MLP	24	3	0	ESCS
	6	55	3	CS
	1	9	28	D

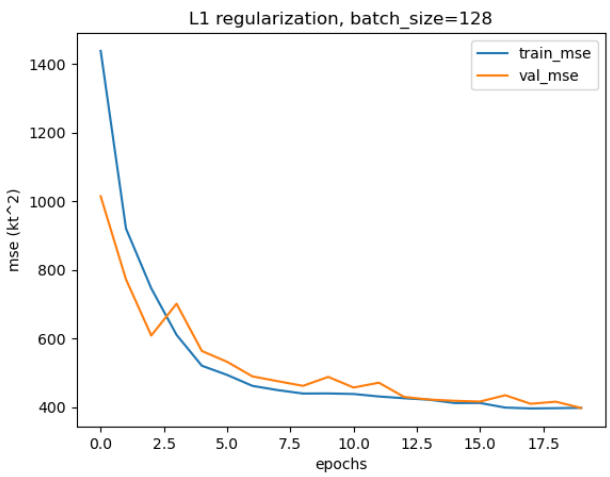
Chapter 5 . Conclusion

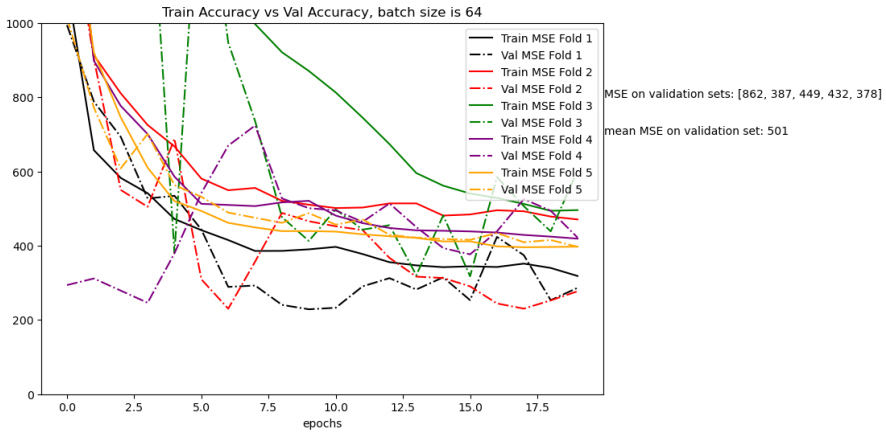
Conclusion

5.1 Conclusion

The MLP technique with geometric features turned out to be useful for the detection of cyclone intensity. In comparison with the conventional method like Dvorak Technique and other standard methods, this method works fine on ESCS, CS and D cyclones. The proposed method is completely based on images and its geometric property with accuracy around 84%.

The Proposed model uses Mean Square Error for justification for training accurate about testing accuracy





The other methods compared here are also tested on similar type of images and environment. Moreover, the number of images used here is greater than the work done in . The proposed method also compared with the work done by Pradhan et al. , where the performance of deep convolution network is varying from 80.66 to 95.47% and 76.91 to 92.55% depending upon datasets. The number of cyclone images of ESCS, CS and D classes is 379. The proposed work is restricted on ESCS, CS, and D type of images and to be extended further with other classes such as DD, ECS, and VSCS,

5.2 Limitations of the Work

- The working of this project would be a little slow because framework like TensorFlow, & deep learning need high-processing hardware and GPU(graphical processing unit) systems but we are using CPU only.
- The models that we are using for identifying the objects are pre-trained models. So, if we want to train our own model, it takes a lot of time and processing.

5.3 Suggestion and Recommendations for Future Work

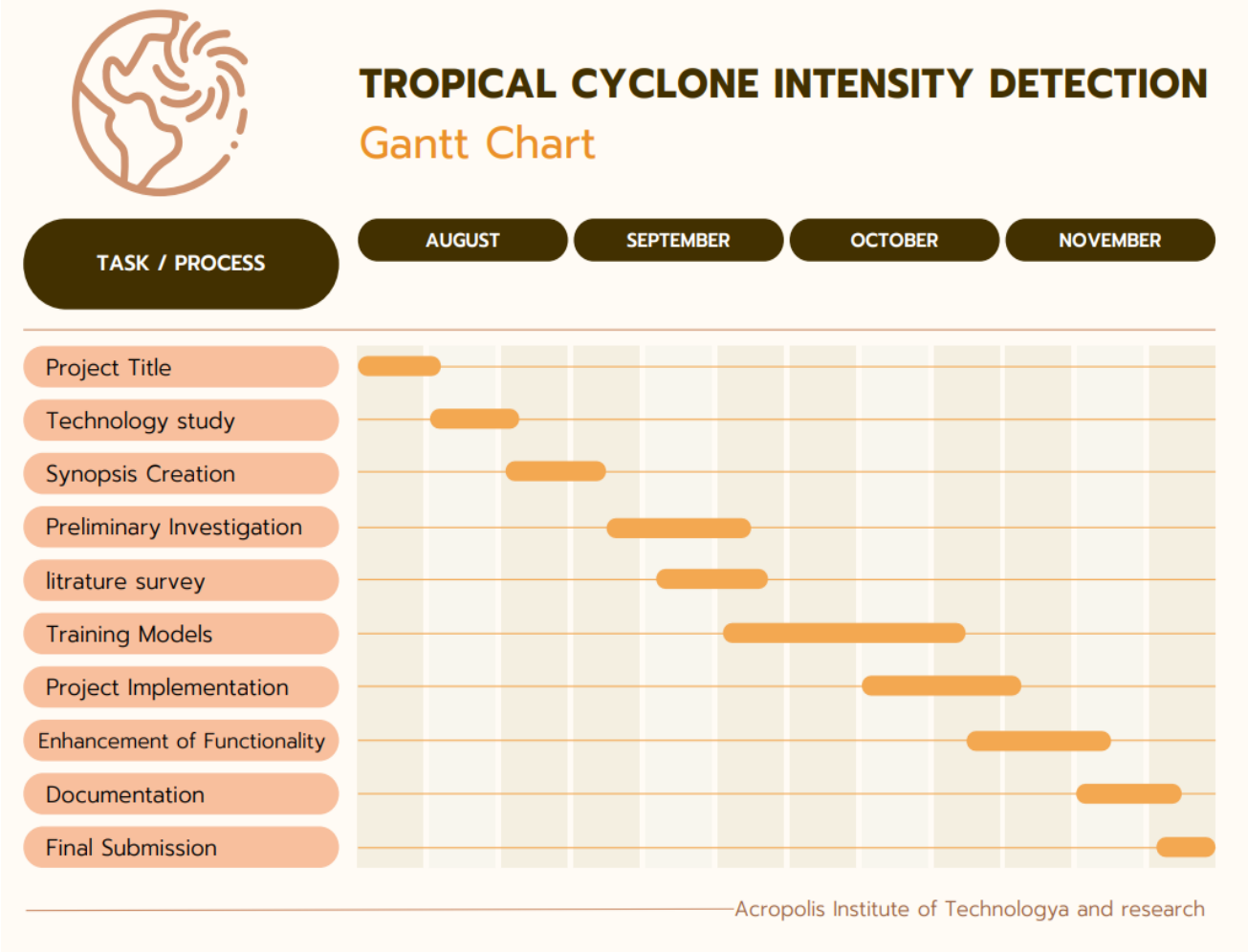
- The Model would be trained for detecting more number of objects.
- We aim to build a user friendly and interactive dashboard which will directly procure the real time data from the satellite and do real time prediction of the intensity level of the upcoming cyclones in future. The future work of the project includes improving the accuracy and

introducing more features to extend the model to an extensive one

Bibliography

1. Regional Specialized Meteorological Centre for Tropical Cyclones over North Indian Ocean, IMD Frequently asked Question. http://www.rsmcnewdelhi.imd.gov.in/images-/pdf/cyclone-awareness/terminology/faq.pdf.
2. Hurricane Research Division, NOAA. https://www.aoml.noaa.gov/hrd/tcfaq/A1.html. Accessed 25 July 2019
3. Dube SK, Rao AD, Sinha PC, Murty TS, Bahulayan N (1997) Storm surge in the Bay of Bengal and Arabian Sea: the problem and its prediction. Mausam 48:283–304
4. Dube SK, Jain I, Rao A, Murty T (2009) Storm surge modelling for the Bay of Bengal and Arabian Sea. Nat Hazards 51:3–27. https://doi.org/10.1007/s11069-009-9397-9
5. Worldwide Tropical Cyclone Centers. https://www.nhc.noaa.gov/aboutrsmc.shtml. Accessed 1 Aug 2019
6. WMO/ESCAP Panel on Tropical Cyclones (2015) Tropical cyclone operational plan for the Bay of Bengal and the Arabian Sea 2015 (report no. TCP-21). World Meteorological Organization, March 2015, pp 11–12

Project Plan



Guide Interaction Sheet

Acropolis Institute of Technology and Research, Indore

Department of Information Technology / CSE-DS/CSE-IoT

PROJECT - LOG BOOK

Project Title:					
Team Name:		**Team Id:**			
Guide Name:		**Semester:**			
Coordinator Name:		**Section:**			
Technology:		**Domain:**			

S No	Enrollment	Team Member Name	Mobile Number	Email Id	Role
1					
2					
3					
4					
5					

S No	Meeting Date	Summary of Work & Discussion	Member Present	Remarks/ Suggestions Given	Guide Sign
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					

12					
S No	**Due Date**	**Particular**	**Submission Date**	**Observations**	**Guide Sign**
1		Team Formation			
2		Project Title			
3		Synopsis			
4		Synopsis Presentation			
5		Design Diagrams			
6		Paper Publication			
7		Presentation-II			
8		Video			
9		Technical Poster			
10		Report			
		Coordinator Signature		***HOD Signature***	

Source Code

1. **Preprocessing.ipynb**

```
import numpy as np
import pandas as pd

import h5py

import matplotlib.pyplot as plt

import tensorflow as tf

## data loading

data_path = "TCIR-ALL_2017.h5"
data_info = pd.read_hdf(data_path, key="info", mode='r')
with h5py.File(data_path, 'r') as hf:
   data_matrix = hf['matrix'][:]



data_path2 = "TCIR-ALL_2017.h5"
data_info2 = pd.read_hdf(data_path2, key="info", mode='r')
with h5py.File(data_path2, 'r') as hf2:
   data_matrix2 = hf2['matrix'][:]

print(np.shape(data_matrix),np.shape(data_matrix2))

data=np.concatenate((data_matrix, data_matrix2))

np.shape(data)

tmp= [data_info,data_info2]

data_label=pd.concat(tmp)
```

```
# # reshape and flat the data
# flat_arr = [data[i].ravel() for i in range(len(data[:,0,0,0]))]
# np.shape(flat_arr)
# vector = np.matrix(flat_arr[0])
# np.shape(vector)
# shape = data[0].shape
# arr2 = np.asarray(vector).reshape(shape)
# np.shape(arr2)
# np.shape(flat_arr[0])

# # concatenate the image into the label dataframe
# data2=pd.DataFrame(pd.np.column_stack([data_label, flat_arr]))

index=-1
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(15,15))
img = data_matrix[index,:,:,0].copy()
pos = ax1.imshow(img, plt.cm.gray)
cbar = ax1.figure.colorbar(pos, ax=ax1)

img1 = data_matrix[index,:,:,1].copy()
pos1 = ax2.imshow(img1, plt.cm.gray)
cbar = ax2.figure.colorbar(pos1, ax=ax2)

img2 = data_matrix[index,:,:,2].copy()
pos2 = ax3.imshow(img2, plt.cm.gray)
cbar = ax3.figure.colorbar(pos2, ax=ax3)

img3 = data_matrix[index,:,:,3].copy()
pos3 = ax4.imshow(img3, plt.cm.gray)
cbar = ax4.figure.colorbar(pos3, ax=ax4)

## Data preprocessing

train_label,  validate_label,  test_label is the label

train,  validate,  test is the data

# 1. subtract the index and shuffle it.
tc_id=data_label['ID'].drop_duplicates()
```

```
# 2. create a seed, and shuffle the tc_id
seed=100
np.random.seed(seed)
perm = np.random.permutation(tc_id)

# 3. split the training set
train_percent,validate_percent=0.6,0.2

m = len(tc_id.index)
train_end = int(train_percent * m)
validate_end = int(validate_percent * m) + train_end

#  the labels
tmp=[]
for i in range(train_end):
   tmp.append(data_label[data_label['ID']==perm[i]])
train_label=pd.concat(tmp)

tmp=[]
for i in range(train_end,validate_end):
   tmp.append(data_label[data_label['ID']==perm[i]])
validate_label =pd.concat(tmp)

tmp=[]
for i in range(validate_end,len(perm)):
   tmp.append(data_label[data_label['ID']==perm[i]])
test_label =pd.concat(tmp)

# split the data
length=len(test_label.index)
tmp=np.empty(shape=[length,201,201,4])
for i in range(length):
   tmp[i,:,:,:]=data[test_label.index[i]]

length=len(train_label.index)
train=np.empty(shape=[length,201,201,4])
for i in range(length):
   train[i,:,:,:]=data[train_label.index[i]]
```

```
length=len(validate_label.index)
validate=np.empty(shape=[length,201,201,4])
for i in range(length):
  validate[i,:,:,:]=data[validate_label.index[i]]

# def train_validate_test_split(df, train_percent=.6, validate_percent=.2,
seed=None):
#    np.random.seed(seed)
#    perm = np.random.permutation(df.index)
#    m = len(df.index)
#    train_end = int(train_percent * m)
#    validate_end = int(validate_percent * m) + train_end
#    train = df.iloc[perm[:train_end]]
#    validate = df.iloc[perm[train_end:validate_end]]
#    test = df.iloc[perm[validate_end:]]
#    return train, validate, test
```

2. Train_program.ipynb

```
import numpy as np
import pandas as pd
import h5py
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split, KFold
print(tf._version_)

data_path = "TCIR-ALL_2017.h5"
data_info = pd.read_hdf(data_path, key="info", mode='r')
with h5py.File(data_path, 'r') as hf:
  data_matrix = hf['matrix'][:]

print("Min Intensity ",data_info.Vmax.min())
print("Min Intensity ",data_info.Vmax.max())
print("Min Intensity ",round(data_info.Vmax.mean(),2))

np.shape(data_matrix)

img = data_matrix[4000,:,:,0].copy()
```

```
fig, ax = plt.subplots()
pos = ax.imshow(img)

img = data_matrix[4000,:,:,0].copy()
fig, ax = plt.subplots()
pos = ax.imshow(img,plt.cm.brg)

img = data_matrix[4000,:,:,0].copy()
fig, ax = plt.subplots()
pos = ax.imshow(img,plt.cm.BrBG)

img = data_matrix[4000,:,:,0].copy()
fig, ax = plt.subplots()
pos = ax.imshow(img,plt.cm.binary)

img = data_matrix[4000,:,:,1].copy()
fig, ax = plt.subplots()
pos = ax.imshow(img, plt.cm.gray)

data_info = data_info.assign(time=pd.to_datetime(data_info.time,
format=r'%Y%m%d%H'))

## keep only IR and PMW
X_irpmw = data_matrix[:,:,:,0::3]
y = data_info['Vmax'].values[:,np.newaxis]

X_irpmw[np.isnan(X_irpmw)] = 0
X_irpmw[X_irpmw > 1000] = 0

# X_std = tf.image.per_image_standardization(X_irpmw)
img = data_matrix[4000,:,:,0].copy()
fig, ax = plt.subplots()
pos = ax.imshow(img, plt.cm.gray)

class Preprocessing(keras.layers.Layer):
  def _init_(self):
    super(Preprocessing, self)._init_()
  def call(self, inputs, training=None):
    if training:
      inputs = tf.image.rot90(inputs, k=np.random.randint(4))
```

```
        return tf.image.central_crop(inputs, 0.5)

## Alexnet CNN


def train_val_model(train_x,train_y,val_x,val_y, n_epochs, batch_size):
    reg_param = 1e-5

    train_X = tf.convert_to_tensor(train_x)
    train_Y = tf.convert_to_tensor(train_y)

    val_X = tf.convert_to_tensor(val_x)
    val_Y = tf.convert_to_tensor(val_y)

    weights_initializer = keras.initializers.GlorotUniform()


    model = keras.models.Sequential([
        Preprocessing(),
        keras.layers.Conv2D(filters=96, kernel_size=(11,11),
strides=4,padding='valid', activation='relu', input_shape=(224,224,3)),
        keras.layers.MaxPool2D(pool_size=(3, 3),strides=2),
        keras.layers.Conv2D(filters=256, kernel_size=(5,5), padding='same',
activation='relu'),
        keras.layers.MaxPool2D(pool_size=(3, 3),strides=2),
        keras.layers.Conv2D(filters=384, kernel_size=(3,3), padding='same',
activation='relu'),
        keras.layers.Conv2D(filters=384, kernel_size=(3,3), padding='same',
activation='relu'),
        keras.layers.Conv2D(filters=256, kernel_size=(3,3), padding='same',
activation='relu'),
        keras.layers.MaxPool2D(pool_size=(3, 3),strides=2),
        keras.layers.Flatten(),
        keras.layers.Dense(4096, activation='relu'),
        keras.layers.Dropout(0.4),
        keras.layers.Dense(4096, activation='relu'),
        keras.layers.Dropout(0.4),
        keras.layers.Dense(1, activation='relu'),
    ])

    #Compiling the model
```

```
   model.compile(optimizer=keras.optimizers.Adam(lr=5e-4,
beta_1=0.99, beta_2=0.9999),
              loss='mean_squared_error',
              metrics=['mean_squared_error'],
             )

   #Training the network
   history = model.fit(train_X,train_Y,
       epochs=n_epochs,
       batch_size=batch_size,
       verbose=1
      )

   val_score = model.evaluate(val_X, val_Y)
   print("Val Score: ",val_score)
   return history,val_score,model

model_history=[]
val_scores=[]
n_epochs=20
batch_size=256
train_x, val_x, train_y, val_y = train_test_split(X_irpmw, y, random_state =
101, test_size=0.1)
train_x, test_x, train_y, test_y = train_test_split(train_x, train_y,
random_state = 101, test_size=0.1)
history,val_score,model = train_val_model(train_x,train_y,val_x,val_y,
n_epochs, batch_size)
model_history.append(history)
val_scores.append(val_score)

y_pred = model.predict(test_x)
print('Testing...')
score = model.evaluate(test_x,test_y,
                batch_size=16, verbose=1)

print('Test accuracy:', score[1])

abcd = []
for x in y_pred:
   abcd.append(int(x))
```

```
cate = []
for x in abcd:
  if x <=33:
    cate.append('Tropical Depression')
  elif x>33 and x<=63:
    cate.append('Tropical Storm')
  elif x>63 and x<=129:
    cate.append('Typhoon')
  elif x>129:
    cate.append('Super Typhoon')

cate_dataset = list(zip(abcd,cate))

df = pd.DataFrame(cate_dataset,columns=['Intensity','Category',])

df



## Deep CNN


def train_val_model(train_x,train_y,val_x,val_y, n_epochs, batch_size):
  reg_param = 1e-5

  train_X = tf.convert_to_tensor(train_x)
  train_Y = tf.convert_to_tensor(train_y)

  val_X = tf.convert_to_tensor(val_x)
  val_Y = tf.convert_to_tensor(val_y)

  weights_initializer = keras.initializers.GlorotUniform()


  model = keras.models.Sequential([
    Preprocessing(),
    keras.layers.Conv2D(filters=16, kernel_size=4, strides=2,
padding='valid', activation='relu', kernel_initializer =
weights_initializer,
kernel_regularizer=keras.regularizers.l2(reg_param)),
    keras.layers.Conv2D(filters=32, kernel_size=3, strides=2,
```

```
padding='valid', activation='relu', kernel_initializer =
weights_initializer,
kernel_regularizer=keras.regularizers.l2(reg_param)),
    keras.layers.Conv2D(filters=64, kernel_size=3, strides=2,
padding='valid', activation='relu', kernel_initializer =
weights_initializer,
kernel_regularizer=keras.regularizers.l2(reg_param)),
    keras.layers.Conv2D(filters=128, kernel_size=3, strides=2,
padding='valid', activation='relu', kernel_initializer =
weights_initializer,
kernel_regularizer=keras.regularizers.l2(reg_param)),
    keras.layers.Flatten(),
    keras.layers.Dense(256, activation='relu', kernel_initializer =
weights_initializer,
kernel_regularizer=keras.regularizers.l2(reg_param)),
    keras.layers.Dense(128, activation='relu', kernel_initializer =
weights_initializer,
kernel_regularizer=keras.regularizers.l2(reg_param)),
    keras.layers.Dense(1, activation='relu', kernel_initializer =
weights_initializer,
kernel_regularizer=keras.regularizers.l2(reg_param)),
  ])
  #Compiling the model
  model.compile(optimizer=keras.optimizers.Adam(lr=5e-4,
beta_1=0.99, beta_2=0.9999),
          loss='mean_squared_error',
          metrics=['mean_squared_error'],
         )

  #Training the network
  history = model.fit(train_X,train_Y,
    epochs=n_epochs,
    batch_size=batch_size,
    verbose=1
   )

  val_score = model.evaluate(val_X, val_Y)
  print("Val Score: ",val_score)
  return history,val_score,model


model_history=[]
val_scores=[]
```

```
n_epochs=10
batch_size=64
train_x, val_x, train_y, val_y = train_test_split(X_irpmw, y, random_state =
101, test_size=0.1)
train_x, test_x, train_y, test_y = train_test_split(train_x, train_y,
random_state = 101, test_size=0.1)
history,val_score,model = train_val_model(train_x,train_y,val_x,val_y,
n_epochs, batch_size)
model_history.append(history)
val_scores.append(val_score)

y_pred = model.predict(test_x)
print('Testing...')
score = model.evaluate(test_x,test_y,
                batch_size=16, verbose=1)
print('Test accuracy:', score[1])

abcd = []
for x in y_pred:
  abcd.append(int(x))

cate = []
for x in abcd:
  if x <=33:
    cate.append('Tropical Depression')
  elif x>33 and x<=63:
    cate.append('Tropical Storm')
  elif x>63 and x<=129:
    cate.append('Typhoon')
  elif x>129:
    cate.append('Super Typhoon')

cate_dataset = list(zip(abcd,cate))

df = pd.DataFrame(cate_dataset,columns=['Intensity','Category',])

df

## Kfold

n_epochs=5
```

```
batch_size=32
model_history = []  #save the model history in a list after fitting so that we can plot later
val_scores=[]
kf = KFold(n_splits=3)

i=0
for train_index, test_index in kf.split(X_irpmw):
   print("Training on Fold: ",i+1)
   i+=1
   train_x, val_x = X_irpmw[train_index], X_irpmw[test_index]
   train_y, val_y = y[train_index], y[test_index]
   history,val_score,model = train_val_model(train_x,train_y,val_x,val_y, n_epochs, batch_size)
   model_history.append(history)
   val_scores.append(val_score)
   print("=======*12, end="\n\n\n")

plt.figure(figsize=(5,5))
plt.title('Train Accuracy vs Val Accuracy, batch size is 64')
colors=['black','red','green']
for i in range(3):
   plt.plot(model_history[i].history['mean_squared_error'], label='Train MSE Fold '+str(i+1), color=colors[i])
#    plt.plot(model_history[i].history['val_mean_squared_error'], label='Val MSE Fold '+str(i+1), color=colors[i], linestyle = "dashdot")

plt.legend(loc='upper right')
plt.xlabel('epochs')
plt.ylim(0,1000)
plt.text(20,800,"MSE on validation sets: "+str([int(v) for v,v2 in val_scores]))
plt.text(20,700,"mean MSE on validation set: "+str(int(np.mean(val_scores,axis=0)[0])))
plt.show()
```

3. Eda.ipynb

```
import numpy as np
import pandas as pd
import h5py
import matplotlib.pyplot as plt
```

```
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split
print(tf._version_)

## data loading

data_path = "TCIR-ALL_2017.h5"
data_info = pd.read_hdf(data_path, key="info", mode='r')
with h5py.File(data_path, 'r') as hf:
   data_matrix = hf['matrix'][:]

data_info.head()

data_info.info()

np.shape(data_matrix)

img = data_matrix[4,:,:,0].copy()
fig, ax = plt.subplots()
pos = ax.imshow(img, plt.cm.gray)

img = data_matrix[4,:,:,1].copy()
fig, ax = plt.subplots()
pos = ax.imshow(img, plt.cm.gray)

data_path_1 = "/home/ec2-
user/SageMaker/DeHurricane/data_downloaded/TCIR-
ATLN_EPAC_WPAC.h5"
data_info_1 = pd.read_hdf(data_path_1, key="info", mode='r')
data_info_1.head()

data_path_2 = "/home/ec2-
user/SageMaker/DeHurricane/data_downloaded/TCIR-CPAC_IO_SH.h5"
data_info_2 = pd.read_hdf(data_path_2, key="info", mode='r')
data_info_2.head()
print(len(data_info), len(data_info_1), len(data_info_2))

Is there missing value?

data_info.isna().sum(axis=0)
```

```
data_info_1.isna().sum(axis=0)
data_info_2.isna().sum(axis=0)

data_info.data_set.unique()
data_info_1.data_set.unique()
data_info_2.data_set.unique()

data_info.groupby('ID').count()
data_info_1.groupby('ID').count().head()
data_info_2.groupby('ID').count().head()

data_info_1.iloc[-1]
data_info_2.iloc[-1]

data_info = data_info.assign(time=pd.to_datetime(data_info.time,
format=r'%Y%m%d%H'))

Is it true that for every ID the increase in time is 3 hours?

data_info[['ID', 'time']].groupby('ID').diff().nunique()

data_info_1 = data_info.assign(time=pd.to_datetime(data_info_1.time,
format=r'%Y%m%d%H'))
data_info_2 = data_info.assign(time=pd.to_datetime(data_info_2.time,
format=r'%Y%m%d%H'))

data_info_1[['ID', 'time']].groupby('ID').diff().nunique()
data_info_2[['ID', 'time']].groupby('ID').diff().nunique()

## Data preprocessing

## keep only IR and PMW
X_irpmw = data_matrix[:,:,:,0::3]
y = data_info['Vmax'].values[:,np.newaxis]

X_irpmw[np.isnan(X_irpmw)] = 0
X_irpmw[X_irpmw > 1000] = 0

train_x, test_x, train_y, test_y = train_test_split(X_irpmw, y, random_state
= 101, test_size=0.2)
```

```
# kf = KFold(n_splits=2)
# for train_index, test_index in kf.split(X_irpmw):
#     print("TEST:", len(list(train_index)), type(list(test_index)))
#     X_train, X_test = X_irpmw[train_index], X_irpmw[test_index]
#     y_train, y_test = y[train_index], y[test_index]

X_tensor = tf.convert_to_tensor(X_irpmw)
y = tf.convert_to_tensor(y)

### Standardization

X_std = tf.image.per_image_standardization(X_tensor)

img = X_std[0,:,:,:]
print(np.mean(img), np.std(img))

## Preprocessing layers

class Preprocessing(keras.layers.Layer):
   def _init_(self):
      super(Preprocessing, self)._init_()
   def call(self, inputs, training=None):
      if training:
         inputs = tf.image.rot90(inputs, k=np.random.randint(4))
      return tf.image.central_crop(inputs, 0.5)

pp = Preprocessing()

rotated = pp(X_std[:5,:,:,:], training=True)

fig, ax = plt.subplots()
pos = ax.imshow(rotated[4,:,:,0], plt.cm.gray)

fig, ax = plt.subplots()
pos = ax.imshow(X_std[4,:,:,0], plt.cm.gray)

## Building a model

X_std[list(np.array([0,1,2]))]
```

```
# input_size = len(X_std)
# output_size = 10

"""
references:
layers API: https://keras.io/api/layers/

1. some parameter tuning:
   batch size: BATCH_SIZE = 128 #@param ["64", "128", "256", "512"]
   regularizer: l1,l2
   how to set the initial weight: weights_initializer =
keras.initializers.GlorotUniform()
   batch size: 32
   how to choose metric?
https://www.tensorflow.org/guide/keras/train_and_evaluate
      right now, I am using mse.
   what is callback?  https://keras.io/api/callbacks/
3. 3 fold cross-validation
4. hold out data for testing
5. all years data
"""



#A function that trains and validates the model and returns the MSE
def train_val_model(train_x,train_y,val_x,val_y, n_epochs, batch_size):
   reg_param = 1e-5

   train_X = tf.convert_to_tensor(train_x)
   train_Y = tf.convert_to_tensor(train_y)
   train_X = tf.image.per_image_standardization(train_X)

   val_X = tf.convert_to_tensor(val_x)
   val_Y = tf.convert_to_tensor(val_y)
   val_X = tf.image.per_image_standardization(val_X)

   weights_initializer = keras.initializers.GlorotUniform()



   model = keras.models.Sequential([
      Preprocessing(),
```

```
    keras.layers.Conv2D(filters=16, kernel_size=4, strides=2,
padding='valid', activation='relu', kernel_initializer =
weights_initializer,
kernel_regularizer=keras.regularizers.l2(reg_param)),
    keras.layers.Conv2D(filters=32, kernel_size=3, strides=2,
padding='valid', activation='relu', kernel_initializer =
weights_initializer,
kernel_regularizer=keras.regularizers.l2(reg_param)),
    keras.layers.Conv2D(filters=64, kernel_size=3, strides=2,
padding='valid', activation='relu', kernel_initializer =
weights_initializer,
kernel_regularizer=keras.regularizers.l2(reg_param)),
    keras.layers.Conv2D(filters=128, kernel_size=3, strides=2,
padding='valid', activation='relu', kernel_initializer =
weights_initializer,
kernel_regularizer=keras.regularizers.l2(reg_param)),
    keras.layers.Flatten(),
    keras.layers.Dense(256, activation='relu', kernel_initializer =
weights_initializer,
kernel_regularizer=keras.regularizers.l2(reg_param)),
    keras.layers.Dense(128, activation='relu', kernel_initializer =
weights_initializer,
kernel_regularizer=keras.regularizers.l2(reg_param)),
    keras.layers.Dense(1, activation='relu', kernel_initializer =
weights_initializer,
kernel_regularizer=keras.regularizers.l2(reg_param)),
  ])

  #Compiling the model
  model.compile(optimizer=keras.optimizers.Adam(lr=5e-4,
beta_1=0.99, beta_2=0.9999),
          loss='mean_squared_error', #Computes the mean of squares of
errors between labels and predictions
          metrics=['mean_squared_error'], #Computes the mean squared
error between y_true and y_pred
         )
  # initialize TimeStopping callback
#   time_stopping_callback = tfa.callbacks.TimeStopping(seconds=5*60,
verbose=1)

  #Training the network
  history = model.fit(train_X,train_Y,
    epochs=n_epochs,
    batch_size=batch_size,
```

```
        verbose=1,
        validation_split=0.1,
        #callbacks=[tf.keras.callbacks.TensorBoard(run_dir + "/Keras"),
time_stopping_callback]
       )

    val_score = model.evaluate(val_X, val_Y)
    print("Val Score: ",val_score)
    return history,val_score

n_epochs=200
batch_size=64
model_history = []  #save the model history in a list after fitting so that
we can plot later
val_scores=[]
kf = KFold(n_splits=5)

i=0
for train_index, test_index in kf.split(X_irpmw):
    print("Training on Fold: ",i+1)
    i+=1
    train_x, val_x = X_irpmw[train_index], X_irpmw[test_index]
    train_y, val_y = y[train_index], y[test_index]
    history,val_score = train_val_model(train_x,train_y,val_x,val_y,
n_epochs, batch_size)
    model_history.append(history)
    val_scores.append(val_score)
    print("=======*12, end="\n\n\n")

def plot_xy(history,title):
    fig = plt.figure()
    x=range(n_epochs)
    train_mse=history.history['loss']
    val_mse=history.history['val_loss']
    plt.plot(x,train_mse,label="train_mse")
    plt.plot(x,val_mse,label="val_mse")
    plt.xlabel('epochs')
    plt.ylabel('mse (kt^2)')
    plt.title(title)
    plt.legend()
```

```
plt.figure(figsize=(9,6))
plt.title('Train Accuracy vs Val Accuracy, batch size is 64')
colors=['black','red','green','purple','orange']
for i in range(5):
   plt.plot(model_history[i].history['mean_squared_error'], label='Train MSE Fold '+str(i+1), color=colors[i])
   plt.plot(model_history[i].history['val_mean_squared_error'], label='Val MSE Fold '+str(i+1), color=colors[i], linestyle = "dashdot")

plt.legend(loc='upper right')
plt.xlabel('epochs')
plt.ylim(0,1000)
plt.text(20,800,"MSE on validation sets: "+str([int(v) for v,v2 in val_scores]))
plt.text(20,700,"mean MSE on validation set: "+str(int(np.mean(val_scores,axis=0)[0])))
plt.show()

plot_xy(history,"L1 regularization, batch_size=128")



plot_xy(history,"L1 regularization, batch_size=32")



print("Fit model on training data")
history = model.fit(
   x_train,
   y_train,
   batch_size=64,
   epochs=2,
   # We pass some validation for
   # monitoring validation loss and metrics
   # at the end of each epoch
   validation_data=(x_val, y_val),
)
```

```
# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=128)
print("test loss, test acc:", results)

# Generate predictions (probabilities -- the output of the last layer)
# on new data using `predict`
print("Generate predictions for 3 samples")
predictions = model.predict(x_test[:3])
print("predictions shape:", predictions.shape)



model.summary()

print("GPU Available: ", tf.test.is_gpu_available())
```