A

Project Report

Entitled

Disaster Prediction based on Satellite Imagery using Deep Learning

Submitted to the Department of Electronics Engineering in Partial Fulfilment for the Requirements for the Degree of

Bachelor of Technology

(Electronics and Communication)

: Presented & Submitted By :

SANJIT ANAND, SIDDHARTH GAUTAM, SOMYA GUPTA

Roll No. (U19EC008, U19EC023, U19EC032)

B. TECH. IV(EC), 8th Semester

: Guided By :

Dr. Prashant K. Shah

Associate Professor, DoECE

(Year: 2022-23)

DEPARTMENT OF ELECTRONICS ENGINEERING

SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY

Surat-395007, Gujarat, INDIA.

Sardar Vallabhbhai National Institute Of Technology

Surat - 395 007, Gujarat, India

DEPARTMENT OF ELECTRONICS ENGINEERING

CERTIFICATE

This is to certify that the Project Report entitled "**Disaster Prediction based on Satellite Imagery using Deep Learning**" is presented & submitted by SANJIT ANAND, SIDDHARTH GAUTAM, SOMYA GUPTA, bearing Roll No. U19EC008, U19EC023, U19EC032, of B.Tech. IV, 8^{th} Semester in the partial fulfillment of the requirement for the award of B.Tech. Degree in Electronics & Communication Engineering for academic year 2022-23.

They have successfully and satisfactorily completed their **Project Exam** in all respects. We, certify that the work is comprehensive, complete and fit for evaluation.

Dr. Prashant K. Shah
Associate Professor& Project Guide

PROJECT EXAMINERS:

Name of Examiners	Signature with Date
1. _____	_____
2. _____	_____
3. _____	_____
4. _____	_____
5. _____	_____

Dr. (Mrs.) Rasika Dhavse
Head, DoECE, SVNIT

Seal of The Department
(May 2023)

Acknowledgements

We would like to express our profound gratitude and deep regards to our guide Dr. PRASHANT K SHAH for his guidance. We are heartily thankful for suggestion and the clarity of the concepts of the topic that helped me a lot for this work. We would also like to thank Prof. , Head of the Electronics Engineering Department, SVNIT and all the faculties of ECED for their co-operation and suggestions. We are very much grateful to all our classmates for their support.

Sanjit Anand,Siddharth Gautam,Somya Gupta
Sardar Vallabhbhai National Institute of Technology
Surat

21 Dec. 2022

Abstract

Natural disasters are events or phenomena caused by natural processes that result in widespread harm to people, animals, and the environment. These events can include earthquakes, hurricanes, typhoons, floods, droughts, wildfires, and volcanic eruptions. They can have serious consequences for human health, including injury, illness, and loss of life. Natural disasters are often unpredictable and can occur with little warning, making it difficult for people to prepare and respond to them effectively. In this report, we have mainly focussed on cyclone detection and prediction as a part of disaster management. Cyclones, also known as hurricanes or typhoons, are large-scale atmospheric systems that can cause significant damage and loss of life. Early detection and warning of cyclones is essential for disaster management and mitigation efforts. Cyclone detection is a crucial aspect of disaster management and early warning systems. Traditional methods of cyclone detection rely on ground-based observations and measurements, such as weather stations, radar, and aircraft observations. These methods are limited by the availability and accessibility of ground stations, and may not provide timely and accurate information in all cases. Satellite imagery offers a promising alternative for cyclone detection as it provides a wide coverage and high spatial resolution. In order to efficiently implement our project, we have studied various research papers that employed different architectures and techniques to predict cyclones in images. In this report, we have implemented a deep learning approach for detecting cyclones in satellite imagery. Our approach involves the use of YOLOv7 architecture model to analyze satellite images and identify the presence of cyclones. We evaluate the performance of our approach on a dataset of satellite images in terms of accuracy, precision, and recall for YOLOv7 architecture and demonstrate that it outperforms traditional methods of cyclone detection. Our approach has the potential to improve the efficiency and accuracy of cyclone detection, enabling better preparedness and response to cyclones and other natural disasters. It can also contribute to the development of more advanced and robust deep learning approaches for satellite image analysis and other geospatial applications.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artifical Intelligence
ANN	Artificial Neural Network
AP	Average Precision
COCO	Common Objects in Context
CNN	Convolutional Nerual Network
CV	Computer Vision
DL	Deep Learning
E-ELAN	Extended efficient layer aggregation networks
FPN	Feature Pyramid Network
FCNN	Fully Connected Neural Network
GAN	Generative Adversial Networks
GPU	Graphics Processing Unit
I/O	Input Output
IoU	Intersection Over Union
KNN	K-nearest Neighbour
LFLB	Local Feature Learning Block
ML	Machine Learning
mAP	Mean Average Precision
NAS	Network Architecture Search
OS	Operating System
R-CNN	Regional Convolutional Neural Network
RF	Random Forest
SVM	Support Vector Machines
TPU	Tensor Processing Units
TC	Tropical Cyclone
VGG16	Visual Geometry Group 16
YOLO	You Only Look Once

Chapter 1
Introduction

A disaster is an event, such as a natural or man-made disaster, that causes significant damage or destruction, and often results in loss of life. Disasters can include events such as earthquakes, hurricanes, floods, and fires, and can have long-lasting effects on individuals, communities, and societies. It is important for individuals and communities to be prepared for disasters and to have plans in place for responding to and recovering from them. India, possessing a large coastline of 7516.6 km is highly prone to cyclones. Every year around five to six tropical cyclones are formed, of which two or three could be severe and cause huge damage to life and property.

1.1 Background

Cyclone prediction is a crucial aspect of meteorology and natural disaster management, as it allows governments, emergency services, and the public to take appropriate actions to protect lives and property in the event of a cyclone. Cyclones, also known as tropical storms or hurricanes depending on their location, can cause significant damage through strong winds, heavy rainfall, and storm surge. Accurate prediction of cyclones allows for the implementation of effective emergency response plans and the evacuation of affected areas. It also enables businesses and other organizations to make informed decisions about their operations and the safety of their employees during a cyclone event.

Effective prediction of cyclones is also important for the insurance industry, as it allows for the proper assessment of risk and the setting of premiums. This, in turn, helps to ensure that the financial burden of cyclone damage is distributed fairly among policyholders. Overall, the importance of cyclone prediction lies in its ability to reduce the impact of these storms on people and society. By providing advance warning of an impending cyclone and enabling the implementation of appropriate protective measures, the harm caused by these storms can be minimized. In addition to the practical benefits of cyclone prediction, it is also a vital field of research for meteorologists and climate scientists. The study of cyclones can provide valuable insights into the factors that influence their formation and behavior, as well as the larger-scale atmospheric and oceanic processes that they are a part of. This knowledge can help to improve our understanding of the Earth's climate and weather systems, and may ultimately lead to more accurate and reliable cyclone prediction.

1.2 Objective

The objective of this topic is to develop a deep learning model that can accurately predict the formation and movement of cyclones using satellite imagery as input. This model should be able to analyze the features of the satellite data and use this information to make accurate forecasts of cyclone formation and movement. The ultimate goal is to improve the accuracy of cyclone prediction and forecasting, which can help to save lives and minimize the adverse impact of cyclones on communities.

1.3 Why use Deep Learning?

Deep learning could be used to analyze large amounts of data, such as weather patterns, ocean temperatures, and atmospheric conditions, to improve the accuracy and reliability of cyclone forecasts. Deep learning algorithms can learn to identify patterns and relationships in the data that may not be immediately obvious to humans, and can continue to improve their performance over time as they are exposed to more data.

One potential application of deep learning in cyclone prediction is the development of more accurate numerical weather prediction models. These models use complex equations to simulate the behavior of the Earth's atmosphere and oceans, and can be used to generate forecasts of weather and climate. However, these models often rely on a large number of assumptions and simplifications, and can be limited in their accuracy and resolution. Deep learning techniques could potentially be used to improve the performance of these models by providing a more flexible and adaptive approach to modeling the Earth's climate and weather systems. Another potential application of deep learning in cyclone prediction is the analysis of satellite and other remote sensing data to identify and track cyclones. Deep learning algorithms could be trained to recognize patterns in the data that are indicative of cyclone formation and development, and could be used to provide real-time monitoring and tracking of these storms.

1.4 Motivation

Cyclones are natural disasters that can have devastating impacts on communities, causing loss of life, damage to property, and disruption to essential services. Accurate and reliable predictions of cyclone formation and development are therefore critical for helping people in affected areas to evacuate or take other necessary precautions to protect themselves and their property.

Satellite images can provide valuable information about cyclones, such as their size, shape, and intensity. By using deep learning techniques to analyze satellite images, it is possible to make more accurate predictions about the path and intensity of cyclones.

This can help governments and disaster management agencies to mobilize resources and aid in advance, so that they can respond more effectively when the cyclone strikes. It can also help businesses, such as insurers and commodity traders, to assess and manage the potential risks and impacts of the cyclone on their operations.

A final year project on the topic of disaster prediction based on satellite images using deep learning would be a timely and relevant contribution to the field of disaster management. It would have the potential to help save lives and minimize the impact of cyclones on communities, and would provide an opportunity for the student to develop their skills in the areas of machine learning and data analysis, which are in high demand in many industries. In addition, the project could also contribute to the broader goal of improving prediction models and techniques for cyclones and other natural disasters, which would ultimately lead to more accurate and reliable predictions in the future.

1.5 Flow of Report

In chapter 2, literature review is presented. Literature review discussed the basic essence of the research papers and journals that have been studied so far.

In chapter 3, theoretical background is mentioned which basically presents the problem statement, technical stack, tools and libraries used and some brief overview of the algorithms that are implemented in this project.

In chapter 4, methodology and implementation of the work discussed. Methodolgy involves dataset description, data preprocessing, workflow along with challenges faced.

In chapter 5, the results are shown based on both the algorithms that are implemented. And Lastly, the conclusion and future scope is mentioned.

Chapter 2
Literature Review

Recently, a number of satellite image processing approaches have been introduced. Some researchers introduced traditional methodology, but Deep Learning and Machine Learning techniques were very recently launched. We will go over a few of the earlier studies that have been conducted in this area for a better understanding of the previous studies conducted.

Md.Nazmul Haque *et al.* [7] worked on various Deep Learning models, to accurately identify cyclones. They put in place an object-detecting system to deal with the problem of detecting cyclone centres. With a bounding box, this stage predicts the core revolving zone. The object localization result was provided by a You Only Look Once (YOLO) model that was trained using an Nvidia Tesla GPU in collaboration with Google. To efficiently detect a cyclone and its core cloud pattern, they have developed a two-step methodology. As a part of pre-processing pictures have been grayscaled. To differentiate the object from the backdrop, or to work with the areas of an image binary thresholding was utilised. The cyclone's eye and the surrounding clouds were made more visible for models to train due to thresholding. The output pixel's value is the lowest of all the pixels nearby because they employed a size 2 erosion, which diminished the size of objects and eradicated small abnormalities.By using different preprocessing techniques on images before training the models they overcome the problem of distinguishing cyclones from the neighbor clouds. They also used cross-validation techniques to prevent our models from bias and overfitting. In the first step, they detected the cyclone using various models like CNN, SVM, KNN, VGG16, RF, and ANN, and in the second step, they worked on detecting the eye of the cyclone using the YOLO algorithm. They reached over 95% accuracy for CNN,94% for SVM, 93% for RF, 93% for VGG16, 88% for ANN, and 86% with the KNN model. They obtained 87.86% mean average precision for object detection using YOLO. Their model was able to locate multiple cyclones on the image.

Aravind *et al.* [8] proposed a unique deep learning method-based automated Tropical Cyclone detection from satellite pictures. They presented a multistage deep learning framework for the detection of Tropical Cyclones, which consists of three components: a wind speed filter, a mask region convolutional neural network (R-CNN), and a convolutional neural network classifier (CNN). They first passed the input satellite image to the detector (Mask R-CNN R50 FPN model). The detections are acquired for the input image and saved in the output format of detectron2; The wind speed of the associated

date is examined if one or more bounding boxes are found. The predictions for the image are ignored if the wind speed is less than 34 knots. Images cropped from the input satellite image using the bounding box coordinates are then given to the classifier (DenseNet169) if more than one prediction is made for an image. If more than one of the cropped images is classified as a cyclone, the one with the highest confidence score from the classifier is chosen as the correct prediction. They found that Mask R-CNN models had a significant false-positive rate but were optimized towards the F1 score. The higher success was obtained by optimizing the Mask R-CNN and CNN models for accuracy. It yielded a model with higher true positives and true-negatives. The use of wind speed filters helped reduce the number of false positives and increasing true-negatives. Hence, a combination of both of these approaches as per the proposed model gave them most accurate predictions using wind speed which reduced the false predictions. They obtained a precision of 97.10 using the above proposed model.

In this research paper [9] Shakya S. *et al.* have used a deep learning approach with satellite images as the primary data for cyclone detection. The outcome of the presented work depicts the involved mathematical nature of issues and highlights an approach to find an optimal classical pre-processing candidate before employing a neural network, in the form of a complete DL algorithm. They have pre-processed the data using interpolation and data augmentation techniques. They have used the deep learning techniques for two purposes, viz. classifying an image under the category of storm or no storm and predicting the storm location in the near future. For classifying storm or no storm, an accuracy of 97% was obtained, and for the detection of the cyclone, a confidence of more than 84% was achieved. They have concluded that the basic CNN model outperformed all the standard models for classifying remote sensing images. The YOLO model is suggested for detecting and locating the eye of a cyclone along with R-CNN model for predicting the path and location of the storm. They have concluded that the Performance of DL algorithm improves with densified datasets using interpolation and data augmentation. Also, tracking and predicting the eye of cyclones is much more accurate by the DL algorithm as compared to the manual process, if images do not contain more than one cyclone.

Yuqiao Wu [10] *et al.* proposed a multitask machine learning framework to forecast tropical cyclone path and intensity, which possesses two modules: one is the prediction module and the other is the estimate module. They utilized an enhanced generative adversarial network as the prediction module to forecast the spatial data of a tropical storm at a specific future time. Then, from the obtained prediction data, we extract the position and intensity using two separate deep neural networks as the estimation module. Utilizing the Wasserstein loss in the prediction module, they set a retrospective

CycleGAN. Then, using TCLNet to predict position and a new model called TIENet to predict intensity, they developed in the estimation module. Their research yielded average path forecast errors of 61 km over the course of 6 hours and 116 km over the course of 24 hours, while their intensity forecast errors of 14.20 kt and 13.06 kt, respectively, over the course of 6 hours and 24 hours. Furthermore, they proved that WCycleGAN is a more effective new model that handles unpaired images, and the TIENet is a specialized model for predicted tropical cyclone image interpretation.

Aryan Khandelwal *et al.* [11] presents a research which focuses on cyclone detection and prediction using INSAT-3D satellite pictures using the proposed model and techniques. The solution put forward in this paper elaborates on the neural network based method that involves a combination of CNN (Convolutional Neural Network) and Bi-GRU. The model includes three layers of LFLB (Local feature learning block) followed by two layers of the CNN Bi-GRU model. Furthermore, the cyclone is detected using K-means that is compared with other segmentation techniques like Detecron2 and Mask-RCNN(Recurrent Convolutional Neural Network). Applications of the proposed approach include the ability to identify cyclones and determine their paths using satellite imagery. The model underwent 150 iterations of training, yielding training losses of 58.6879 and validation losses of 1179.

In this paper by Minsang *et al.* [12] models based on three different machine learning (ML) algorithms—decision trees (DT), random forest (RF), and support vector machines (SVM), as well as a model based on linear discriminant analysis—were examined for their detection abilities for tropical cyclone (TC) development (LDA).The performance of the trained models is compared using HR, FAR, and PSS from the 2 × 2 contingency table of observed versus the model-classified number of DEVs and non-DEVs. HR is the number of model-classified DEVs divided by the number of observed DEVs. FAR is defined as the number of the model-classified DEVs divided by the number of observed non-DEVsOverall, the ML algorithms showed better performance with significantly higher hit rates (95%) compared with that from the LDA-based model (77%), although false alarm rates were slightly higher in MLs (21–28%) than that by LDA (13%).The comparison among the three ML approaches revealed a slight difference in the performance. SVM exhibited less FAR compared with the other ML approaches, although all showed comparable skill in terms of HR. MLs were able to identify TC development at the time up to 26 to 30 hours before the JTWC best track first identified it as a tropical depression, which was also 5 to 9 hours earlier than that by LDAThis study shows that when compared to traditional linear approaches, ML approaches have a better ability to detect Tropical Cyclones generation.

Chandan Roy *et al.* [13] presents a paper on automatic cyclone forecasting using artificial neural network techniques to interpret NOAA-AVHRR satellite images. Researchers found that the shape and relative positioning of the cumulonimbus clouds around a cyclone reflect the cyclone's trajectory. These traits may be seen in satellite photographs, thus utilising a neural network to detect and classify them would be useful for automated cyclone forecasting.A multilayered neural network, resembling the human visual system, was trained to forecast the movement direction of cyclones. They devloped a neural network based on stepwise feature integration is similar to the what-pathway.The network was implemented and trained using the artificial neural network simulation tool Leabra++, which is specialized for modeling biologically-based networks.95% of the training photos were correctly predicted by the network after training. Additionally, the network handled 95% of brand-new test images, demonstrating high generalisation ability.The results showed that multi-layered neural networks might be improved into a useful tool for forecasting cyclone tracks using data from remote sensing.The paper concluded that processing of high-resolution images in an artificial neural network most probably requires parallel computing resources, such as a cluster of PCs using distributed memory.

Yanfei Zhang *et al.* [14] used matrix neutral networks for cyclone track prediction for South Indian Ocean. They employed prominent RNN architectures as baseline models which includes Elman RNNs, Long Short-Term Memory and Gated Recurrent Unit Networks. The proposed approach involves using MNNs to predict the future location of a cyclone based on its current position and various atmospheric variables such as temperature, humidity, and wind speed. MNNs are found flexible in predicting cyclone path since they can handle matrix inputs and outputs, making them well-suited for processing atmospheric data. MNNs enables us to adjust the input feature to any size and generate the output that is consistent or lower in dimension. In other words, the input can be of 2D for each input unit, and we can have 2D or 1D outputs. MNNs can more easily enable auxiliary features in the input while preserving the context in terms of spatial andtemporal dependencies. It is believed that auxiliary featureswould improve the prediction. They also removed the spatial bases in original dataset by normalising all cyclones with their starting positions. The result showed that the proposed method achieved better performance when compared to RNNs. Matrix Neural Network can preserve the spatial information from the cyclone track.

Sheila Alemany *et al.* [15] presented a novel approach for hurricane trajectory prediction using a recurrent neural network (RNN). The authors address the challenge of predicting the path of a hurricane, which is a complex problem due to the large num-

ber of variables involved, including the position, speed, and intensity of the hurricane, as well as the topography and weather conditions of the surrounding area. They have proposed a recurrent neural network employed over a grid system with the intention to encapsulate the non-linearity and complexity behind forecasting hurricane trajectories and potentially increase the accuracy compared to the operating hurricane track forecasting models. The authors begin by discussing the current state-of-the-art in hurricane trajectory prediction, which typically involves a combination of physical models, statistical models, and machine learning techniques. They note that while these approaches can be effective, they often require a significant amount of computational resources and can be difficult to implement in real-time scenarios. They have performed the hurricane prediction using a fully-connected RNN employed over a grid system. This network has the capability to accumulate the historical information about the nonlinear dynamics of the atmospheric system by updating the weights. LSTM was used as the underlying building units. The mean-squared error and root-mean squared error were 0.01 and 0.11 respectively. Although the idea of grid based models and RNN are not new, however the combination of both to model complex nonlinear temporal relationships was a novel contribution.

Selim Furkan Tekin *et al.* [**?**] studied tropical cyclone trajectories using deep learning techniques. They have employed a recurrent neural network (RNN)-baed deep learning architecture called TrajGRU to capture complex temporal patterns and perform forecasts. The performance was analysed in terms of kilometers at various hourly resolutions and results were with two baseline methods including naive linear predictor and long short-term memory (LSTM) networks. Various contributions are made by the authors through the paper. As the first time in the literature, they employed TrajGRU model for hurricane trajectory forecasting. This model used atmospheric feature images at different pressure levels and handles both spatial and temporal complexity of the hurricane phenomena. They also utilized a early fusion module to incorporate external features including distane to land, storm speed, storm direction, sustained wind speed and center pressure into the architecture.We use IBTrACS Dataset for hurricane trajectories. This dataset contains the hurricane trajectory data since 1842 from multiple meteorology agencies. Every hurricane trajectory has 3-hourly center coordinates, with additional features such as land distance, hurricane type and wind speed. The dataset contains more than 13000 hurricane tracks. The dataset includes samples from different basins, e.g North Atlantic (NA), Eastern North Pacific (EP). Except the 3-hour case, TrajGRU performs better that other models. When the forecasting duration increases, the performance improvement becomes more significant. The encoder decoder structure provides higher accuracy for multi-step long duration predictions.

Chapter 3
Theoretical Background

In this chapter we will discuss about all underlying algorithms for cyclone detection. We have also discussed about the various tools and libraries used for creating the model for cyclone detection in the satellite images such as Google Colaboratory and PyTorch etc.

3.1 Python

Python is a high-level, interpreted programming language that is widely used for web development, scientific computing, data analysis, and artificial intelligence. Some of the key features of Python include:

1. Easy to learn and use: Python has a simple and easy-to-learn syntax, which makes it an excellent choice for beginners. Its code is easy to read and understand, which makes it easy to maintain and modify.

2. Dynamically-typed: Python is dynamically-typed, which means that you don't need to specify the type of a variable when declaring it. This makes it easier to write code quickly, as you don't need to spend time figuring out the exact type of each variable.

3. Data Preprocessing: Python's powerful data manipulation libraries such as NumPy and Pandas can be used to import, clean, and transform raw data into a form that is suitable for machine learning.

4. Training and Testing Machine Learning Models: Python provides a wide range of libraries and frameworks for training machine learning models, such as scikit-learn, TensorFlow, and PyTorch. These libraries provide a range of algorithms and tools for tasks such as classification, regression, clustering, and deep learning.

3.2 Object Detection

Object detection is a subfield of machine learning and computer vision that involves identifying and locating objects in images or videos. It is a crucial task in various applications such as autonomous vehicles, surveillance systems, and image/video search engines.

In traditional machine learning (ML), object detection algorithms rely on handcrafted features, such as edge detection and color histograms, to detect objects in an image. These features are then fed into a classifier, such as a support vector machine (SVM), to classify the object and locate it within the image.

However, with the advancement of deep learning (DL), object detection has significantly improved. DL algorithms, such as convolutional neural networks (CNNs), can learn high-level features from raw data automatically, without the need for manual feature engineering. This has led to the development of successful object detection models, such as R-CNN and YOLO.

- R-CNN (Regional CNN) is a pioneering DL object detection model that uses a CNN to classify and locate objects in an image. It first generates a set of region proposals, which are potential locations of objects in the image. The CNN then processes these region proposals and outputs class probabilities and bounding box coordinates for each proposal. R-CNN and its variants have achieved impressive results on various object detection benchmarks. However, they are computationally expensive, as they require multiple forward and backward passes through the CNN for each image.

- YOLO (You Only Look Once) is a more recent object detection model that aims to address the computational efficiency issue of R-CNN. YOLO processes the entire image in a single forward pass and predicts class probabilities and bounding box coordinates for multiple objects in the image. It has a simpler architecture compared to R-CNN and can achieve real-time object detection performance on a single GPU. However, it sacrifices some accuracy for speed.

3.3 YOLO

YOLO (You Only Look Once) is a popular algorithm for object detection in images and videos. It was developed by Joseph Redmon and Ali Farhadi in 2015 and has been improved upon several times since then. YOLO v7 is a deep learning-based object detection algorithm that uses a single convolutional neural network (CNN) to predict the bounding boxes and class probabilities for objects in an image. It is designed to be fast and accurate, and is able to process images in real-time on a standard computer.

FCNN(Fully Connected Neural Network) is the foundation of the YOLO architecture. The YOLO family has also recently included variations based on Transformer.

The YOLO framework has three main components.

- Backbone

- Head

- Neck

The Backbone primarily pulls out the most important aspects of an image and transmits them through the Neck to the Head. The Neck compiles feature maps that the Backbone has retrieved and builds feature pyramids. Finally, the head consists of output layers that have final detections.

3.3.1 YOLOv3

The architecture that is used in YOLO v3 is called DarkNet-53 [16] is shown in Figure 3.1. Its main responsibility is to extract features. There are 53 convolutional layers in it. Here, there is no maximum pooling. For each convolution operation, there is convolution followed by Batch Normalization and leaky RELU. The YOLO-V3 feature extractor, Darknet-53 which was inspired by ResNet and FPN (Feature-Pyramid Network) architectures, contains skip connections (like ResNet) and 3 prediction heads (like FPN), each of which processes the image at a different spatial compression. Like its predecessor, Yolo-V3 boasts good performance over a wide range of input resolutions. It can process input images of any size. But, since different input resolutions give rise to different network parameters, the network is trained with resolution augmentation.

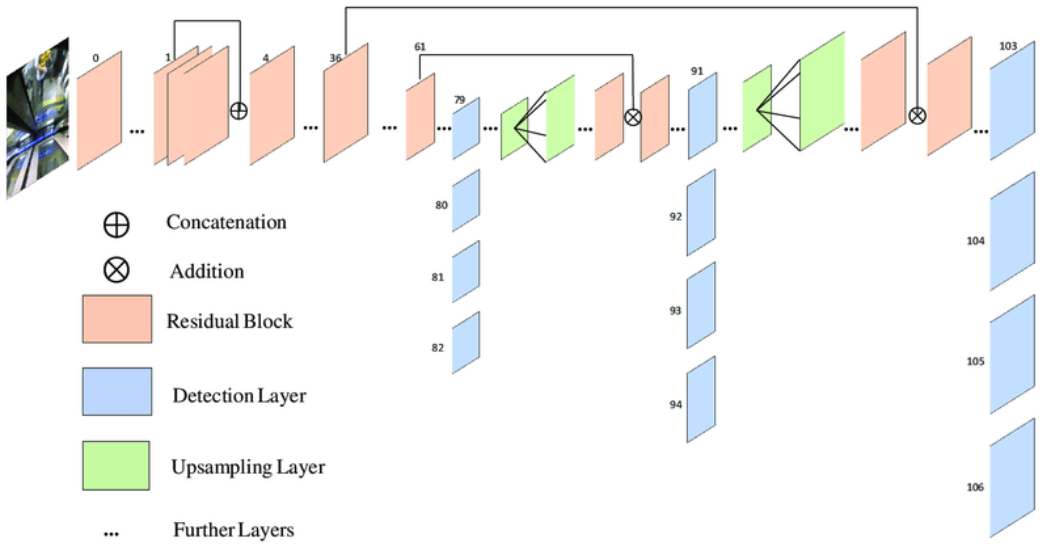


Figure 3.1: Network-architecture-of-YOLOv3 [1]

Initially, the YOLOv3 algorithm divides an image into a grid. Each grid cell predicts the presence of a specific number of boundary boxes (also known as anchor boxes) around items that perform well in the aforementioned predetermined classes. Only one object is detected by each boundary box, which has a corresponding confidence score

indicating how correct it expects that prediction to be. The ground truth boxes' dimensions from the original dataset are clustered to identify the most typical sizes and shapes before being used to create the border boxes [17].

3.3.2 YOLOv7

YOLOv7 introduces a number of architectural improvements that increase efficiency and accuracy. YOLOv7 backbones do not use ImageNet pre-trained backbones, similar to Scaled YOLOv4. Instead, the complete COCO dataset is used to train the models. As Scaled YOLOv4, an extension of YOLOv4, was produced by the same authors as YOLOv7, the similarity is to be expected. The YOLOv7 document has undergone the significant revisions listed below. We'll go over each one separately.Figure 3.2 shows the supermacy of YOLOv7 structure over other architectures in terms of speed.

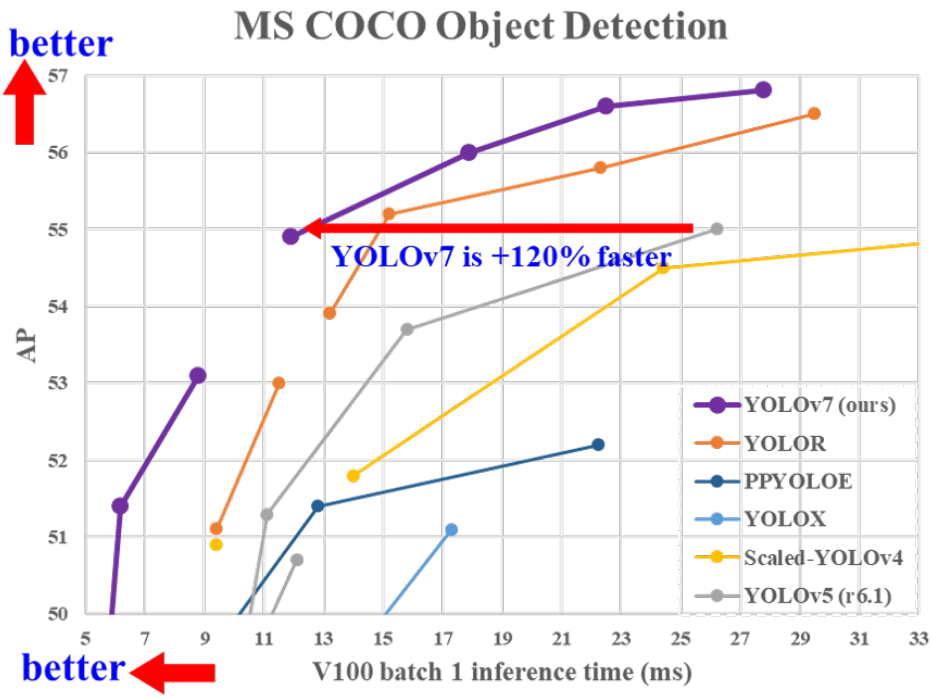


Figure 3.2: Comparison of YOLOv7 with other frameworks [2]

Architectural Reforms

- E-ELAN (Extended Efficient Layer Aggregation Network)
- Model Scaling for Concatenation-based Models

Trainable BoF (Bag of Freebies)

- Planned re-parameterized convolution

- Coarse for auxiliary and Fine for lead loss

The architecture of YOLOv7 is derived from YOLOv4, Scaled YOLOv4, and YOLO-R. The key aspects of architecture of YOLOv7 are:

- **Extended Efficient Layer Aggregation Network (E-ELAN) in YOLOv7**: The computational building piece of the YOLOv7 is E-ELAN. It draws influence from earlier studies on network effectiveness. It was created by looking at the elements that affect speed and accuracy. Theses elements include Memory access cost, I/O channel ratio , Element wise operation, Activations, Gradient path. The suggested E-ELAN employs expand, shuffle, and merge cardinality to continuously improve the network's capacity for learning while preserving the original gradient path [18] as shown in Figure 3.3.

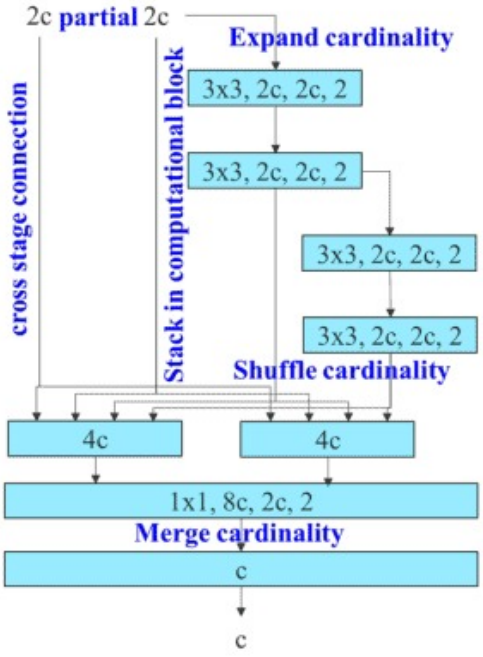


Figure 3.3: E-ELAN Architecture [2]

- **Compound Model Scaling in YOLOv7**: While scaling a model size, resolution ,width, depth, stage are considered. Model scaling techniques like NAS (Network Architecture Search) are frequently employed. Researchers employ it to iteratively search through the parameters in pursuit of the ideal scaling factors. Methods like NAS, however, scale parameters specifically. In this instance, the scaling factors are independent.

  The authors of the YOLOv7 study demonstrate that a compound model scaling strategy can further optimise it as shown in Figure 3.4. For concatenation-based models, width and depth are scaled in this case coherently [18].

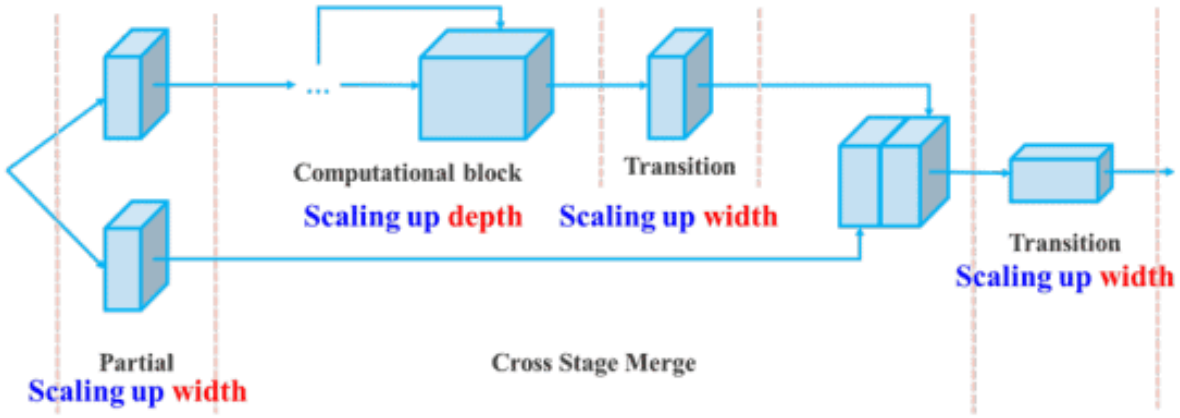


Figure 3.4: Compound Scaling in YOLOv7 [2]

- **Trainable Bag of Freebies in YOLOv7**: BoF or Bag of Freebies are methods that increase the performance of a model without increasing the training cost. YOLOv7 has introduced the Planned Re-parameterized Convolution to improve inference result at the cost of training time. Model level and Module level ensemble are two types of re-parametrization [18].

3.3.3 Comparison between YOLOv3 and YOLOv7

One key difference between YOLOv3 and YOLOv7 is the size of the model. YOLOv3 is larger and more accurate than YOLOv7, but it is also slower and requires more computational resources to run. On the other hand, YOLOv7 is smaller and faster, but it is also less accurate than YOLOv3.

Another difference between the two algorithms is their ability to detect small objects. YOLOv3 has been designed to be more sensitive to small objects, while YOLOv7 is better at detecting larger objects.

In terms of performance, YOLOv3 has achieved state-of-the-art results on several object detection benchmarks and is widely used in a variety of applications. YOLOv7 is still a newer algorithm and has not yet been as widely tested, but it has shown promising results in early evaluations.

Overall, the choice between YOLOv3 and YOLOv7 will depend on the specific needs and constraints of the application. YOLOv3 may be a better choice for applications that require high accuracy and are able to tolerate slower processing times, while YOLOv7 may be more suitable for applications that require faster processing times and can tolerate lower accuracy.

3.4 Time Series Forecasting

Time series forecasting in deep learning is an application of deep learning techniques to predict the future values of a time series. A time series is a sequence of data points collected at regular intervals over time. Examples of time series data include stock prices, weather data, and sensor data from industrial equipment or latitudes, longitudes, and other features of a cyclone's eye at regular intervals, which is being used in this project. Cyclone prediction is a complex task that involves analyzing a wide range of meteorological and oceanographic data to determine the path, intensity, and duration of a cyclone. Multivariate time-series forecast analysis means that we have multiple variables (more than one) based upon which we need to forecast the target variable.

Time series forecasting using deep learning involves training a deep learning model on historical time series data and using the model to predict future values of the time series. This requires the use of specialized deep learning architectures, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs), that can capture the temporal dependencies and patterns in the data. RNNs are particularly well-suited for time series forecasting as they are designed to handle sequential data. RNNs use a feedback loop to feed the output of one time step back into the network as input for the next time step, allowing them to capture the temporal dependencies in the data. The long short-term memory (LSTM) network is a type of RNN that is particularly effective for time series forecasting as it can handle long-term dependencies in the data.

3.5 Recurrent Neural Network

In traditional neural networks, all the inputs, and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words(as the next word will depend on your previous input). Example you watch a movie and in between you stop to predict the ending, it will depend on how much you have seen it already and what context has come up yet. Similarly RNN remembers everything. It solves this problem of traditional neural network with a hidden layer. An RNN remembers each and every information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory. A recurrent neural network (RNN) is a type of neural network that is designed to handle sequential data. Unlike feedforward neural networks, which process inputs in a single forward pass, RNNs can maintain a memory of previous inputs and use it to make predictions for the current input.

RNNs have a loop in their architecture, which allows them to process sequential data one step at a time. At each time step, the RNN takes an input and updates its

internal state based on the input and the previous state. The updated state is then used to make a prediction for the current output. This process is repeated for each time step, and the RNN can process inputs of variable lengths.

One of the key advantages of RNNs is their ability to model temporal dependencies in data. For example, in natural language processing, RNNs can be used to predict the next word in a sentence based on the previous words. Similarly, in time series analysis, RNNs can be used to forecast future values based on past values. RNN architecture is shown in Figure 3.5.

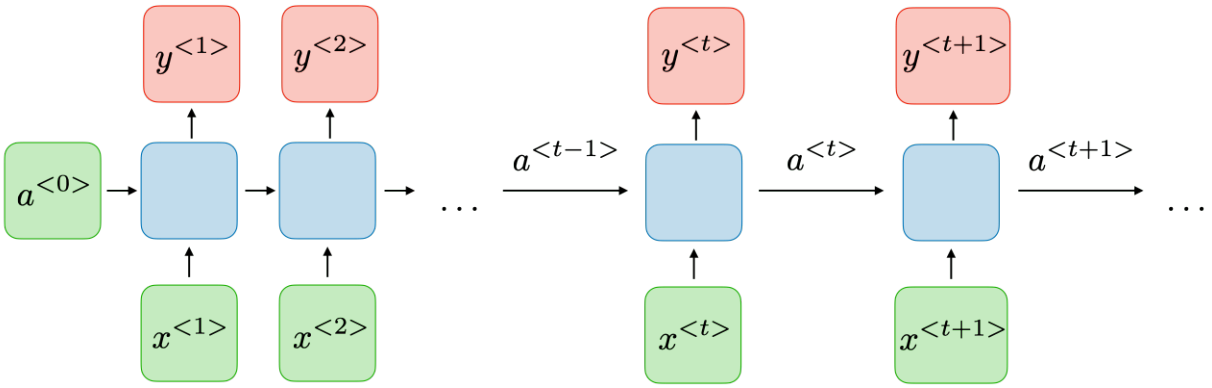


Figure 3.5: Recurrent Neural Network Architecture [3]

3.5.1 Long Short-Term Memory

LSTM stands for Long Short-Term Memory which is a Recurrent Neural Network that are capable of learning long-term dependencies, especially in sequence prediction problems. LSTM networks are designed to handle long-term dependencies in sequential data by using a memory cell that can selectively forget or retain information from previous time steps. The memory cell is controlled by three gates: the input gate, the forget gate, and the output gate. Bidirectional recurrent neural networks(RNN) are really just putting two independent LSTM RNNs together hown in Figure 3.6. This structure allows the networks to have both backward and forward information about the sequence at every time step.

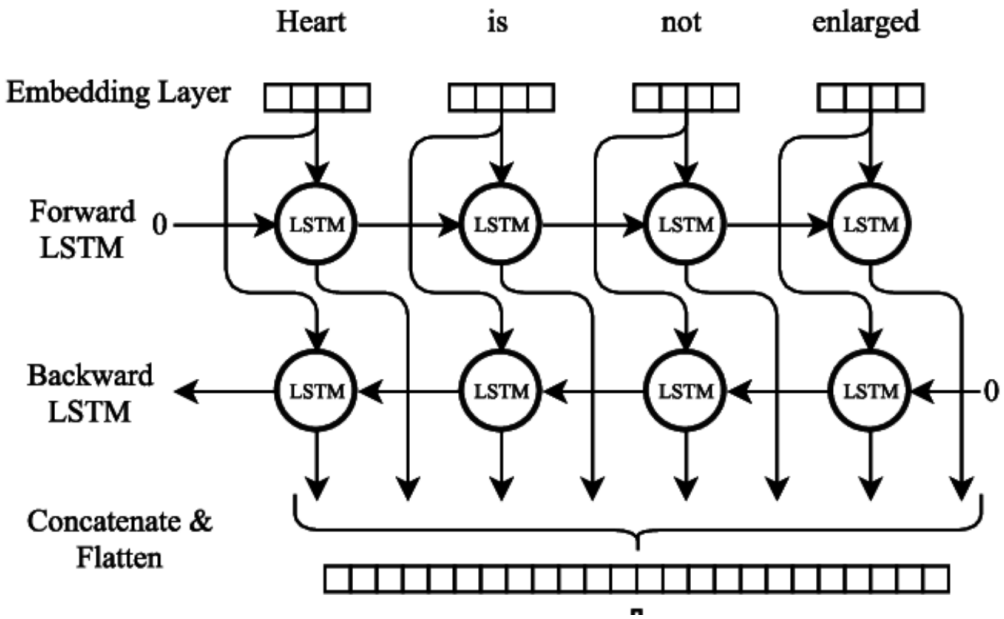


Figure 3.6: Bidirectional LSTM [4]

Using bidirectional will run your inputs in two ways, one from past to future and one from future to past and what differs this approach from unidirectional is that in the LSTM that runs backward, you preserve information from the future and using the two hidden states combined you are able in any point in time to preserve information from both past and future.

3.5.2 Gated Recurrent Unit

Gated Recurrent Unit (GRU) is a variant of Recurrent Neural Networks (RNNs) that was introduced by Kyunghyun Cho in 2014. GRUs are designed to address the vanishing gradients problem that can occur in traditional RNNs when processing long sequences.Like other RNNs, a GRU processes sequential data one element at a time, and has a hidden state that is updated at each time step. However, GRUs have a gating mechanism that allows them to selectively update the hidden state based on the input and the previous state. This gating mechanism consists of two gates: the update gate and the reset gate.

The update gate controls how much of the previous hidden state is retained and how much of the new input is added to the current state. The reset gate determines how much of the previous hidden state is forgotten and how much of the new input is used to create a new hidden state. By using these gating mechanisms, GRUs can selectively choose which information to retain and which to discard, making them particularly effective for processing long sequences. Additionally, GRUs have fewer parameters than Long

Short-Term Memory (LSTM) networks, another popular variant of RNNs, which makes them faster to train and more computationally efficient. GRU architecture is shown in Figure 3.7.

GRUs have been shown to be effective in a variety of applications, including natural language processing, speech recognition, and image captioning. They have achieved state-of-the-art results in tasks such as language modeling, machine translation, and speech recognition.

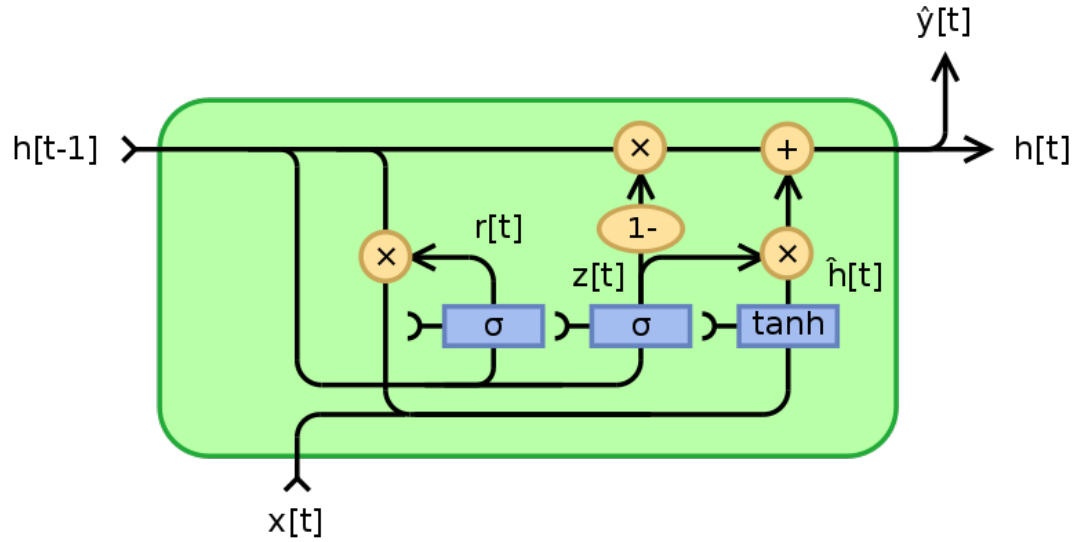


Figure 3.7: Gated Recurrent Unit Architecture [5]

3.6 Tools and Libraries Used

In this section, we will be discussing the various tools and libraries used during our project work which enabled us to implement and produce the idea of cyclone detection and forecasting into the desired output.

3.6.1 Google Colaboratory

Google Colab is a free online platform that allows users to run and execute Python code in the cloud. It is designed for data science and machine learning workflows, and provides access to a range of powerful computing resources, including GPUs and TPUs. Colab is a great resource for developers and data scientists who want to work with Python, as it allows them to write and execute code without the need to install any software on their local machines. It is also useful for teams who want to collaborate on code and data projects, as Colab provides a shared workspace that can be accessed by multiple users.

3.6.2 LabelImg

LabelImg is a graphical annotation tool for labeling images. It is written in Python and uses Qt for its graphical interface. It allows users to draw bounding boxes around objects in images and label them with class names. The tool also provides editing and visualization capabilities, including the ability to zoom in and out, navigate between images, and view annotation data. LabelImg is useful for creating datasets for training machine learning models, particularly in the field of computer vision. It is an open source project and can be downloaded and used for free.

3.6.3 Kaggle

Kaggle is a platform for data science and machine learning. It is owned by Google and was founded in 2010. The platform offers a variety of resources for data scientists, including a cloud-based workbench for developing and running code, a public data library, and a number of machine learning competitions. Kaggle is particularly well-known for hosting machine learning competitions, which often involve developing a model to solve a real-world problem using a provided dataset. These competitions are often sponsored by organizations looking to advance the state of the art in a particular field or to solve a specific problem. Kaggle also offers a number of educational resources, including tutorials and courses on data science and machine learning topics.

3.6.4 PyTorch

PyTorch is an open-source machine learning library for Python, used for developing and training deep learning models. It is developed by Facebook's AI research group and is used by researchers and developers at companies and organizations around the world.One of the main features of PyTorch is its ability to perform computations on tensors, which are multi-dimensional arrays. This makes it easy to perform operations on large datasets, such as those used in image or natural language processing tasks.We used pyTorch to train our cyclone detection model with YOLOv3 and YOLOv7.

3.6.5 Libraries Used

We used a variety of python libraries available for preprocessing the data and making the model for the detection of cyclones.Some of them are:

- OpenCV: OpenCV (Open Source Computer Vision) is a free and open-source library of computer vision and machine learning algorithms. It was developed by Intel in 1999 and is now maintained by a community of developers. It is used for a wide range of applications, including object detection and recognition,

image classification, tracking and motion analysis, and 3D reconstruction. We used OpenCV to preprocess out dataset and augument it. It was also used to draw the bonding boxes on the images [19].

- Keras: Keras is a open-source deep learning library that provides a user-friendly API for building and training deep learning models. It is designed to be modular, flexible, and easy to use, making it accessible to both beginners and experienced researchers. Keras was initially developed as a user-friendly interface to build deep learning models on top of other lower-level frameworks such as TensorFlow, Theano, and CNTK. However, with the release of TensorFlow 2.0, Keras has been integrated as a part of TensorFlow and is now the official high-level API for building and training models in TensorFlow. We have used Keras library to implement various models for path prediction of cyclones.

- Pandas: Pandas is a popular open-source data manipulation and analysis library which provides data structures for efficiently storing and manipulating large datasets, as well as a range of tools for cleaning, transforming, and analyzing data.Pandas is widely used in data science and machine learning for tasks such as data cleaning, data preprocessing, exploratory data analysis, and feature engineering. It is also used for data analysis and manipulation in a wide range of industries, including finance, healthcare, and retail. We have used Pandas library for the analysis and modification of the dataset for path prediction model.

- Scikit-learn: Scikit-learn (also known as sklearn) is a open-source machine learning library in Python that provides a range of tools for building and training machine learning models. It is built on top of NumPy, SciPy, and matplotlib, and provides a user-friendly API for a variety of machine learning tasks. Scikit-learn is widely used in industry and academia for a wide range of applications, including image and speech recognition, natural language processing, and finance. It is a versatile library that is easy to use and can help you quickly get started with machine learning. We have used several functions of sklearn for evaluating the error matrices.

- Matplotlib: Matplotlib is a popular data visualization library in Python. It provides a high-level interface for drawing attractive and informative statistical graphics. With Matplotlib, we can create line plots, scatter plots, bar plots, error bars, histograms, bar charts, pie charts, box plots, and many other types of visualizations. It is also possible to customize the appearance of the plots and add text, labels, and annotations. We used this library to make graphs of vrious outputs.

- GeoPandas: GeoPandas is an open-source library for working with geospatial data. It is built on top of the pandas library and provides a range of tools for

working with geospatial data, including support for common geospatial file formats, spatial operations, and visualization. It provides the GeoDataFrame data structure, which is similar to the pandas DataFrame but with support for storing and manipulating geospatial data. GeoPandas provides tools for visualizing geospatial data, including plotting and mapping. We have used this library for the visualization of actual and predicted path of the cyclone.

- OS: The OS module in python provides functions for interacting with the operating system. OS, comes under Python's standard utility modules. We used this module for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.

3.7 Summary

In this chapter we discussed about the various deep learning algorithms that can be used for cyclone detection in detail. YOLOv3 is widely used for object detection whereas YOLOv7 is comparatively new architecture which have shown promising results in object detection. We discussed about the various libraries, tools and frameworks which are used for training the model for cyclone detection and creation of dataset.

Chapter 4
Cyclone Detection Methodology

In this chapter we will discuss in detail about the methadology and the workflow for cyclone detection. The workflow begins with details about creation of the dataset and data preprocessings like filtering, grayscaling and binary thresholding as summarized in Figure 4.1. Further, the training which consists of cloning the model, hyperparameter tuning and training; and testing of the model for cyclone detection is presented. At last, the challenges faced during the dataset creation and training the model to create accurate results are discussed.

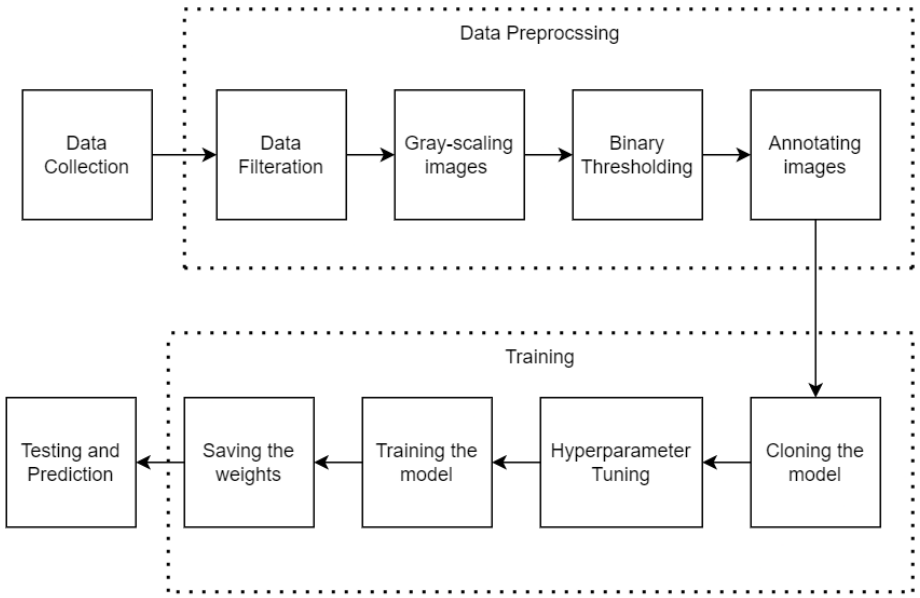


Figure 4.1: Workflow

4.1 Dataset Description

The satellite images of the cyclone have been collected from Kaggle. There are two parts to our dataset: training and testing for detection of the cyclone. We double-checked that no images were duplicated. For the detection purpose, the dataset contains overall 1005 images overall. The train set has 804 images, accounting for about 80% of the whole dataset, whereas the test set has 201 images, accounting for slightly over 20% of the total data.

4.2 Data Preprocessing

Image data has been cleaned using preprocessing methods for the model's input. The likelihood of smaller cloud formations, shear, centre density overcast, as well as noise, necessitated preprocessing of the photos we collected. Additionally, it has speed up model execution and cut down on model training time.

We have converted the images of our dataset into grayscale.The original and grayscale images are shown in Figure 4.2 and 4.3 respectively. Our model will be simpler and more performant when working with grayscale photos because it only needs to maintain track of one matrix per image. Binary thresholding, or working with the portions of an image we are interested in, has been used to separate an object from the background. Thresholding made the cyclone's eye and surrounding clouds more discernible for our models to train on.

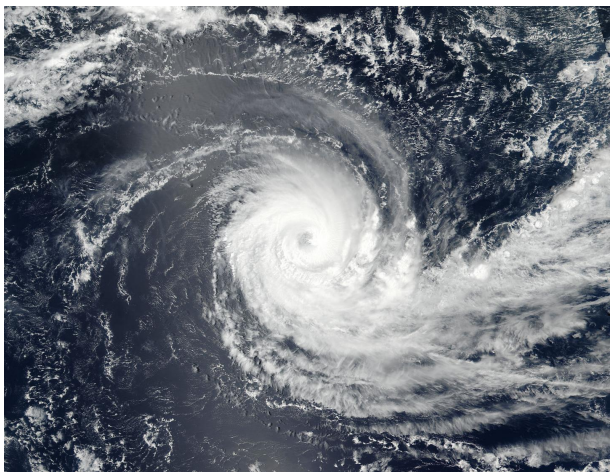
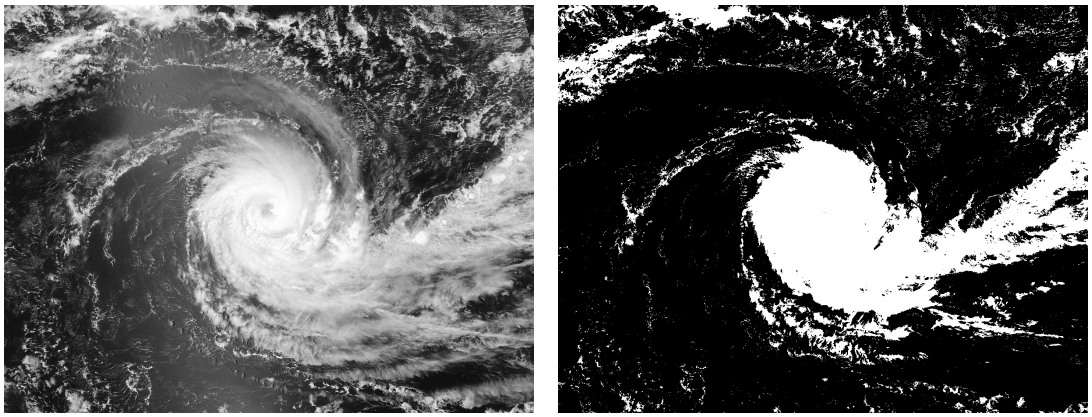


Figure 4.2: Orignal Image [6]



(a) Grayscale Image (b) Final Image

Figure 4.3: Processed Images

The output pixel's value is the lowest of all the pixels nearby since we applied a size 2 erosion to reduce the size of objects and remove small imperfections. Images in our dataset come in various sizes. As a result, all of the photographs have been reduced in size to 300*300.

4.2.1 Annotations

Annotations are used in object detection to label the objects in an image or video that the object detection algorithm should identify. These labels are used to train the object detection model, and they help the model learn to recognize the objects of interest in an image or video.

There are typically two types of annotations used in object detection: bounding box annotations and segmentation annotations. Bounding box annotations involve drawing a box around the object of interest in an image, while segmentation annotations involve outlining the object of interest and labeling each pixel within the outline as belonging to the object.We have used bounding box annotations for our images. We have used the tool named 'LabelImg' for annotating images. The original image and the annotated image is shown in Figure 4.4 and 4.5 respectively.

Annotations are important in object detection because they provide the model with the necessary information to accurately identify and locate objects in an image or video. Without annotations, the model would not have any way of knowing what it should be looking for, and as a result, it would not be able to perform object detection effectively.

Overall, annotations play a crucial role in the training and performance of object detection models, and they are an essential part of the object detection process.

Figure 4.4: Orignal Image [6]

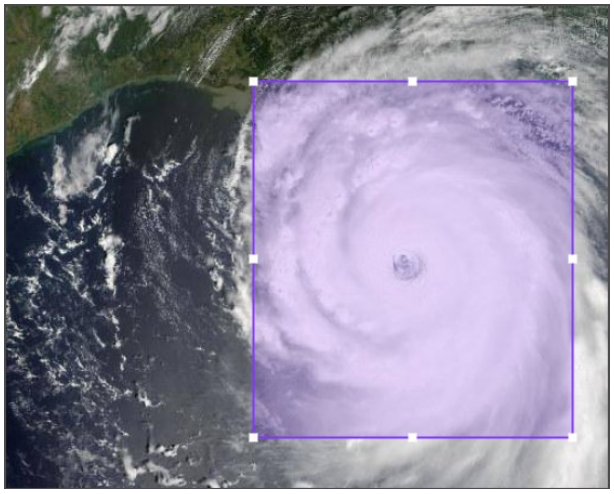


Figure 4.5: Annotated Image

4.2.2 Data Augumentation

Image augmentation is a common technique used in deep learning for image classification and object detection tasks. It involves creating modified versions of existing images in the training dataset by applying random transformations such as rotation, scaling, translation, and flipping. These transformed images are then added to the training dataset, increasing its size and diversity.

The goal of image augmentation is to improve the generalization ability of the deep learning model, by reducing overfitting and increasing the variability of the training data. By training the model on a larger and more diverse dataset, it can learn more robust features that are not specific to the training data, and perform better on unseen data.

There are many different techniques that can be used for image augmentation, and the choice of which techniques to use will depend on the specific application and the nature of the data. Some common techniques include:

- Random cropping: Cropping a random portion of the image and resizing it to the original size.
- Random rotation: Rotating the image by a random angle.
- Random flipping: Flipping the image horizontally or vertically.
- Color jittering: Adding random noise to the color channels of the image.

As our dataset was not large enough we used the data augmenatation techniques to increases its size by 3 times. We used several python libraries like 'os', 'NumPy' 'OpenCV' etc. for data augmenation.

4.3 Training the Model

4.3.1 Setting up the Model

We have used YOLOv7 for the detection of cyclones in satellite image. We cloned the model with pretrained weights from [20]. We put the train , test and validation images in different folders along with their annotations and specified their path in the model by adding data.yaml file in data folder of YOLOv7. In order to detect only one class we finetuned the model since YOLO is originally trained on COCO dataset with 80 classes. We need to add parameter 'nc: 1' and 'names:['cyclone]' as shown in Figure 4.6, where nc represents number of classes, to detect only one class with label of cyclone.

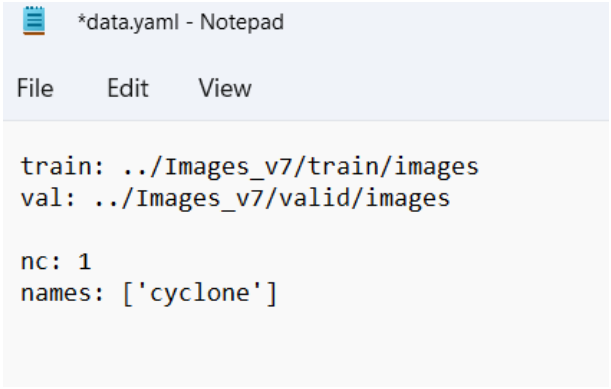


Figure 4.6: Content of data.yaml file

4.3.2 Hyperparameter

For training purpose, we have used batch size of 4. Keeping the batch size larger causes memory problems in Google Colab and stops the training process. We have trained the model for 100 epochs. Initial learning rate in YOLOv7 is 0.01. Momentum in YOLOv7 is 0.937 and weight decay is 0.0005 as shown in Table. 4.1.

Sr. No.	Hyperparameter	Value
1	Batch Size	4
2	Epochs	100
3	Initial Learning Rate	0.01
4	Momentum	0.937
5	Weight Decay	0.0005

Table 4.1: Hyperparameters

4.3.3 Training

To train the model we need to execute the train.py file present in YOLOv7 folder. We need to specify data.yaml file in the command along with input size of image, batch size and epochs. After every epoch we see the value of Mean Average Precision (MAP). Once the training is completed all the weights are saved in the weights folder inside run folder. All the graphs of precision, recall , F1-Score and confusion matrix are also saved in the run folder once training is completed. Training metrices are shown in Figure 4.7

```
      Epoch   gpu_mem       box       obj       cls     total    labels  img_size
      96/99     11.5G   0.03899  0.007549         0   0.04654         4       640: 100% 70/70 [00:54<00:00,  1.28it/s]
                Class      Images      Labels          P          R      mAP@.5  mAP@.5:.95: 100% 7/7 [00:03<00:00,  2.22it/s]
                  all          56          57      0.721      0.456       0.551       0.208

      Epoch   gpu_mem       box       obj       cls     total    labels  img_size
      97/99     11.5G    0.0382  0.007572         0   0.04577         9       640: 100% 70/70 [00:58<00:00,  1.19it/s]
                Class      Images      Labels          P          R      mAP@.5  mAP@.5:.95: 100% 7/7 [00:03<00:00,  2.16it/s]
                  all          56          57      0.729      0.439       0.546       0.206

      Epoch   gpu_mem       box       obj       cls     total    labels  img_size
      98/99     11.5G   0.04185  0.008438         0   0.05028         7       640: 100% 70/70 [01:09<00:00,  1.01it/s]
                Class      Images      Labels          P          R      mAP@.5  mAP@.5:.95: 100% 7/7 [00:03<00:00,  2.10it/s]
                  all          56          57      0.537      0.632       0.562       0.199

      Epoch   gpu_mem       box       obj       cls     total    labels  img_size
      99/99     11.5G   0.04091  0.007799         0   0.04871         7       640: 100% 70/70 [01:05<00:00,  1.08it/s]
                Class      Images      Labels          P          R      mAP@.5  mAP@.5:.95: 100% 7/7 [00:04<00:00,  1.74it/s]
                  all          56          57      0.538      0.614       0.552       0.188
  100 epochs completed in 1.885 hours.

  Optimizer stripped from runs/train/yolov73/weights/last.pt, 74.8MB
  Optimizer stripped from runs/train/yolov73/weights/best.pt, 74.8MB
```

Figure 4.7: Metrics shown during training

4.3.4 Testing the Model

Once the training is completed weights best.pt which signifies the best weights are saved in the weights folder. Now we use those weights to detect the cyclone in the image and to measure our accuracy . In order to test the accuracy we need to run the test.py and provide the value of weights we want to use , batch size and confidence score. To detect cyclone in any image we need to run detect.py and give the weights and location of image or video.

4.4 Challenges Faced

Deep learning is one of the primary research areas in developing intelligent machines. But, with great technological advances, comes complex difficulties and hurdles. The following challenges were faced during our implementation phase of the project:

4.4.1 Dataset Collection and Creation

Dataset collection was one of the first and major challenge that we faced during the implementation. No dataset of images of cyclones was present with annotations. Thus, we had to manually select images and annotate them using the tool LabelImg which was time consuming.

4.4.2 Threshold Parameter tuning

Tuning the value of the threshold parameter was also a challenge that we faced. A higher value of threshold resulted in removing the white cloud parts of the image and thus cyclone pattern were not visible in this case. On the other hand, keeping a low value of threshold parameter made the white pattern more dense and thus the features were not visible properly. Hence, an optimal and appropriate value of threshold parameter was chosen which lied in the middle of the two cases.

4.4.3 Batch size and Memory Limit

Batch size was one of the hyperparameter that was tuned. Batch size refers to the number of images utilized in one iteration. If we increased the batch size, we faced the problem of memory limit since increasing the batch size directly results in increasing the required GPU memory. Thus, due to the constraints on the GPU power and computations, we had to reduce the value of batch size.

4.4.4 Problem of overfitting

Problem of overfitting was faced when we were training the model as the size of the data was not very large. If we don't apply for early stopping of the model, then it may result in overfitting and thus it learns the data. The bias is low and variance is high as the model has less training error but very high testing error. Thus finding a sweet spot for number of iterations is essential and we can identify it by finding the state where the validation error increases while the training error still decreasing.

4.4.5 Limited Support of GPU

We have used Google Colab platform for training the model. The free tier in Google Colab restricts a continuous usage after 12 hours. Hence this was also a challenge that we faced training the model.

4.5 Summary

The satellite images of the cyclone have been collected from Kaggle. The dataset contains 1005 images overall with 80:20 train-test split. We have converted the images of our dataset into grayscale and then used thresholding technique. Annotations labels are made for YOLO format along with the image augmentation technique. After that training of the model is done by using hyperparameter training. The major challenges faced during the implementation are dataset collection, threshold parameter tuning, batch size, memory limit, overfitting and limited GPU support.

Chapter 5
Cyclone Path Forcasting Methodology

In this chapter we will discuss in detail about the methadology and the workflow for cyclone path detection. The workflow begins with details about data preprocessing like cleaning, filtering, feature extraction and padding as summarized in Figure 4.1. Further, the model creation which consists of deciding model architecture, activation function and hyperparameter tuning. Then the model is trained over the train dataset ad tested on the test split. The results of predicted path are plotted on India's map. At last, the challenges faced during the dataset creation and training the model to create accurate results are discussed.

5.1 Data Description

The cyclone data that we are using for our project is

5.2 Data Preprocessing

5.2.1 Feature Extraction

5.2.2 Data Cleaning

5.2.3 Data Normalization

Data normalization is the process of transforming data into a standardized format, allowing for easier comparison and analysis. Data normalization is a crucial step in machine learning as it can significantly affect the performance of the model. Machine learning algorithms typically work better when all input features are on a similar scale. If the input features have vastly different scales, then some features may dominate the learning process, leading to suboptimal results.variables with larger scales can dominate the results.

There are several methods for normalizing data in machine learning. We have used Min-Max normalization.This method scales the data so that it falls within a specified range (usually between 0 and 1).

5.2.4 Train Test Split

The dataset has been divided into train and test data. In train data, 75% of the orignal dataset is used and the other 25% is used for test split. Scikit learn library is used for train test split.

For every cyclone 23 datapoints are used as input and 12 datapoints are predicted by the forcasting model.Datapoints are recorded at the intervals of 3 hours. This implies almost 3 days of data is used to predict the path of cyclone for next two days.

5.3 Model Preparation

5.3.1 Setting up the Model

Simple Recurrent Neural Network

Simple RNN model is implemented using Sequential model in Keras. The model contains 4 layers where first layer has 256 nodes and activation is relu. It is followed by Dropout layer which randomly sets input units to 0 with a rate of 0.3 at each step during training time. It is followed by Dense layer. Second layer contains 356 nodes with relu activation followed by dropout layer of rate 0.5. Third layer contains 400 nodes with relu activation followed by dropout layer of rate 0.4. Fourth layer has 280 nodes and relu activation function. Model summary is shown in Figure 5.1

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 256)               66304

 dropout (Dropout)           (None, 256)               0

 repeat_vector (RepeatVector  (None, 12, 256)          0
 )

 dense (Dense)               (None, 12, 2)             514

 simple_rnn_1 (SimpleRNN)    (None, 12, 356)           127804

 dropout_1 (Dropout)         (None, 12, 356)           0

 time_distributed (TimeDistr  (None, 12, 2)            714
 ibuted)

 simple_rnn_2 (SimpleRNN)    (None, 12, 400)           161200

 dropout_2 (Dropout)         (None, 12, 400)           0

 simple_rnn_3 (SimpleRNN)    (None, 12, 280)           190680

 dense_2 (Dense)             (None, 12, 2)             562

=================================================================
Total params: 547,778
Trainable params: 547,778
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.1: Simple RNN Model Summary

Bidirectional Long Short-Term Memory

Bidirectional Long Short-Term Memory is also implemented using Sequential model in Keras. The model contains 3 layers where first layer has 256 nodes and activation is tanh. It is followed by Dense layer. Second layer contains 356 nodes with tanh activation followed by dense layer. Third layer contains 64 nodes with relu activation followed by dense layer.Model summary is shown in Figure 5.2

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional (Bidirectiona  (None, 512)              530432
 l)

 repeat_vector_2 (RepeatVect  (None, 12, 512)          0
 or)

 dense_6 (Dense)             (None, 12, 2)             1026

 bidirectional_1 (Bidirectio  (None, 12, 712)          1022432
 nal)

 time_distributed_2 (TimeDis  (None, 12, 2)            1426
 tributed)

 bidirectional_2 (Bidirectio  (None, 12, 128)          34304
 nal)

 dense_8 (Dense)             (None, 12, 2)             258

=================================================================
Total params: 1,589,878
Trainable params: 1,589,878
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.2: Bi-LSTM Model Summary

Gated Recurrent Unit

Gated Recurrent Unit is also implemented using Sequential model in Keras. The model contains 3 layers where first layer has 256 nodes and activation is tanh. It is followed by Dense layer. Second layer contains 356 nodes with tanh activation followed by dense layer. Third layer contains 64 nodes with relu activation followed by dense layer.Model summary is shown in Figure 5.2

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 256)               199680

 repeat_vector (RepeatVector  (None, 12, 256)          0
 )

 dense (Dense)               (None, 12, 2)             514

 gru_1 (GRU)                 (None, 12, 356)           384480

 time_distributed (TimeDistr  (None, 12, 2)            714
 ibuted)

 gru_2 (GRU)                 (None, 12, 64)            13056

 dense_2 (Dense)             (None, 12, 2)             130

=================================================================
Total params: 598,574
Trainable params: 598,574
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.3: GRU Model Summary

5.3.2 Hyperparameter

For training purpose, we have used batch size of 16. Keeping the batch size larger causes memory problems in Google Colab and stops the training process. We have trained the model for 500 epochs. Initial learning rate is is 0.001. Model parameters are shown in Table. 5.1.

Sr. No.	Hyperparameter	Value
1	Batch Size	16
2	Epochs	500
3	Initial Learning Rate	0.001
4	Loss Function	Mean Square Error
5	Optimizer	Adam
6	Activation	Relu and Tanh

Table 5.1: Hyperparameters

5.3.3 Training

Once the models are defined we train all the 3 model on the train data set for 500 epochs. After every epoch we notice the value of loss, Mean Average Precision (MAP) and Mean Absolute Percentage Error (MAPE). Validation loss is plotted once the training is completed to check how loss varies with increasing epochs. Training metrices

for Simple RNN , Bi-LSTM and GRU are shown in Figure 5.4 Figure 5.5 Figure 5.6 respectively.

```
Epoch 495/500
90/90 [==============================] - 4s 48ms/step - loss: 5.8643e-04 - mae: 0.0166 - mape: 4.897
0 - acc: 1.0000
Epoch 496/500
90/90 [==============================] - 4s 48ms/step - loss: 6.2980e-04 - mae: 0.0172 - mape: 4.886
3 - acc: 1.0000
Epoch 497/500
90/90 [==============================] - 4s 47ms/step - loss: 5.6888e-04 - mae: 0.0170 - mape: 4.899
1 - acc: 1.0000
Epoch 498/500
90/90 [==============================] - 4s 47ms/step - loss: 5.3920e-04 - mae: 0.0166 - mape: 4.752
2 - acc: 1.0000
Epoch 499/500
90/90 [==============================] - 4s 49ms/step - loss: 5.6034e-04 - mae: 0.0170 - mape: 4.926
6 - acc: 1.0000
Epoch 500/500
90/90 [==============================] - 5s 54ms/step - loss: 5.0778e-04 - mae: 0.0159 - mape: 4.660
2 - acc: 1.0000
```

Figure 5.4: Simple RNN Metrics shown during training

```
Epoch 495/500
90/90 [==============================] - 1s 16ms/step - loss: 1.7863e-04 - mae: 0.0094 - mape: 2.5660
Epoch 496/500
90/90 [==============================] - 1s 15ms/step - loss: 1.7817e-04 - mae: 0.0092 - mape: 2.4373
Epoch 497/500
90/90 [==============================] - 1s 13ms/step - loss: 2.0410e-04 - mae: 0.0097 - mape: 2.3958
Epoch 498/500
90/90 [==============================] - 1s 13ms/step - loss: 1.8411e-04 - mae: 0.0092 - mape: 2.4183
Epoch 499/500
90/90 [==============================] - 1s 14ms/step - loss: 1.9390e-04 - mae: 0.0097 - mape: 2.4802
Epoch 500/500
90/90 [==============================] - 1s 13ms/step - loss: 1.8442e-04 - mae: 0.0095 - mape: 2.4103
```

Figure 5.5: Bi-LSTM Metrics shown during training

```
Epoch 495/500
90/90 [==============================] - 1s 10ms/step - loss: 1.7271e-04 - mae: 0.0088 - mape: 2.3078 - acc: 1.0000
Epoch 496/500
90/90 [==============================] - 1s 9ms/step - loss: 1.8287e-04 - mae: 0.0094 - mape: 2.5734 - acc: 1.0000
Epoch 497/500
90/90 [==============================] - 1s 8ms/step - loss: 1.5333e-04 - mae: 0.0083 - mape: 2.1679 - acc: 1.0000
Epoch 498/500
90/90 [==============================] - 1s 8ms/step - loss: 1.6673e-04 - mae: 0.0088 - mape: 2.2615 - acc: 1.0000
Epoch 499/500
90/90 [==============================] - 1s 8ms/step - loss: 1.6195e-04 - mae: 0.0087 - mape: 2.2249 - acc: 1.0000
Epoch 500/500
90/90 [==============================] - 1s 8ms/step - loss: 1.5797e-04 - mae: 0.0084 - mape: 2.1941 - acc: 1.0000
```

Figure 5.6: GRU Metrics shown during training

5.3.4 Testing the Model

Once the training is completed we check the accuracy of model. For finding accuracy we find the mean squared error and R-squared error values. We compare the actual path of cyclone and the path predicted by our model by plotting it on the graphs and on

world map. We also find the difference in km of actual paths of cyclones and the paths predicted by our model.

5.4 Plotting on Map

5.5 Challenges Faced

Cloud path forecasting is a specific application of time series forecasting using deep learning, which involves predicting the path of a cloud system over time based on historical data. This application presents several challenges that must be addressed to achieve accurate predictions.

5.5.1 Uneven Length of Cyclone Data

We are considering 35 timestamps of each cyclone. But there is a huge disparity in the data available in the dataset. There are various cyclone where only as few as 1-10 timestamps are available albeit there are some with extensive data of upto 180 timestamps. We require data with equal lengths to feed into our deep learning model. We take the mean value of 35 as the optimum value of number of timestamps of each data with 23 as input and 12 as output split. Hence we reject all cyclone which have less than 18 timestamps, extrapolate the data to pad them upto 35 samples by insertng their means in between for cyclones which have more than 18 and less than 35 timestamps and finally taking only 35 timestamps for cyclones where timestamps exceeds the baseline 35 limit.

5.5.2 Batch size and Memory Limit

Batch size was one of the hyperparameter that was tuned. Batch size refers to the number of images utilized in one iteration. If we increased the batch size, we faced the problem of memory limit since increasing the batch size directly results in increasing the required GPU memory. Thus, due to the constraints on the GPU power and computations, we had to reduce the value of batch size.

5.5.3 Hyperparameter Tuning

This was the most difficult challenge we have faced. We had to set and tune the hyperparameter in such a way so as we get the minimum value of error. We have used 3 different deep learning RNN architectures viz. RNN, Bi-LSTM and GRU. Different

hyperparameters like number of layers, activation functions like ReLu and TanH, optimizers like Adam and Adagrad, number of neurons, learning rate, number of epochs ranging from 50 to 500, batch sizes like 8, 16 and 32 were altered and suitable hyperparameters were chosen accordingly.

5.5.4 Problem of Overfitting

Problem of overfitting was faced when we were training the model as the size of the data was not very large. If we don't apply for early stopping of the model, then it may result in overfitting and thus it learns the data. The bias is low and variance is high as the model has less training error but very high testing error. Thus finding a sweet spot for number of iterations is essential. We can reduce this problem by the use of introducing dropout into the model architecture so that it can introduce some amount of randomness and by increasing the amount of data.

Chapter 6

Results

This presents presents final prediction images with the predicted bounded boxes along with the results and graphs that we have obtained. We have briefly discussed about some of the commonly used metrics like precision, recall, f1 score and mAP in object detection.

6.1 Prediction Output Images

Here are some images showing the bounding boxes on the detected cyclone by the YOLOv7 model:

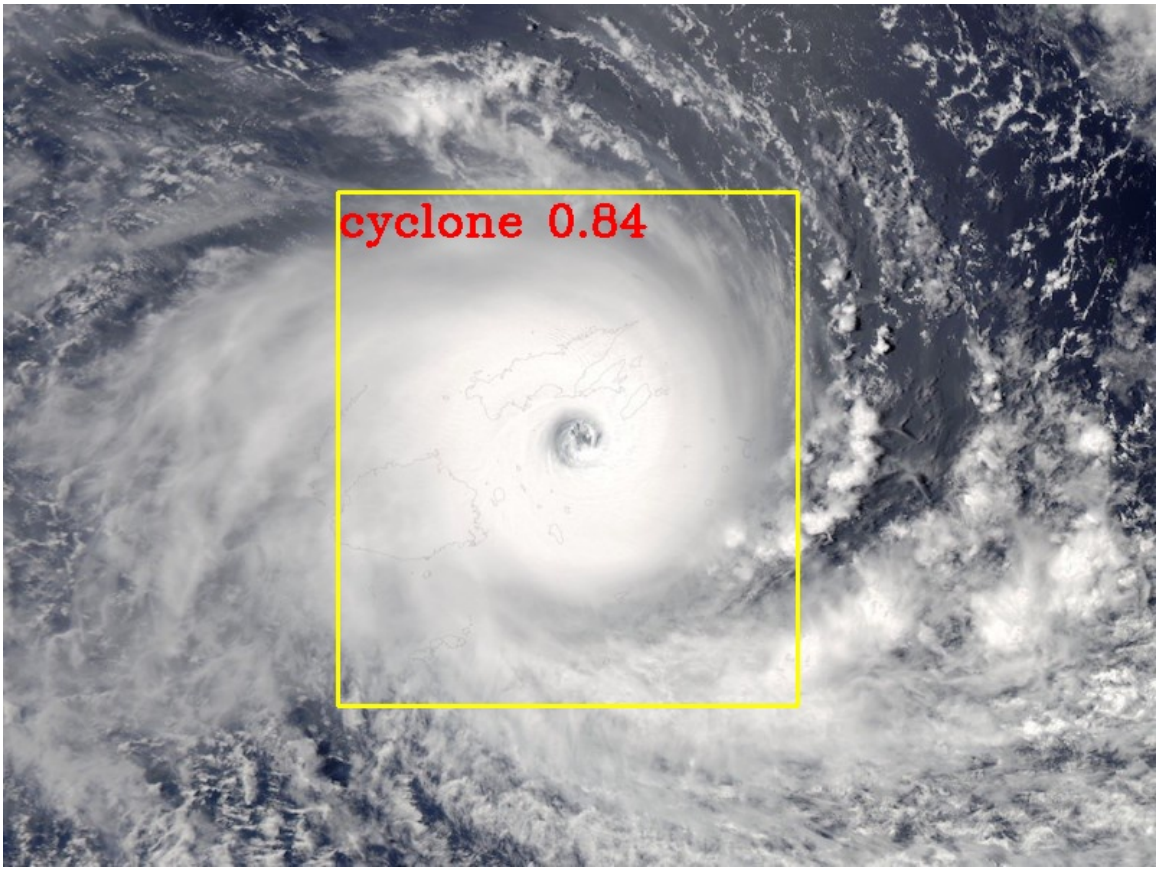


Figure 6.1: Output Image 1

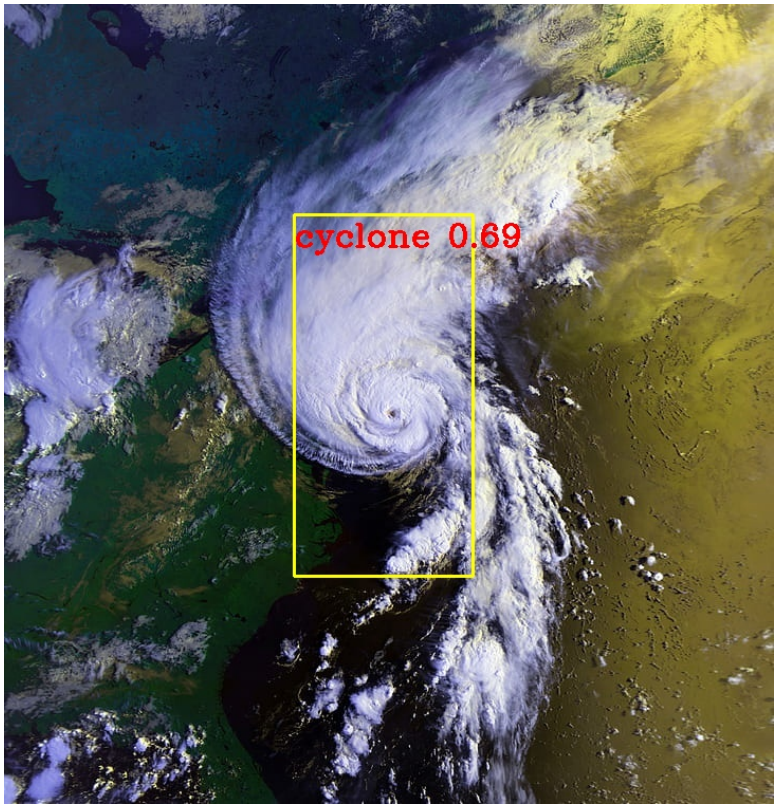


Figure 6.2: Output Image 2

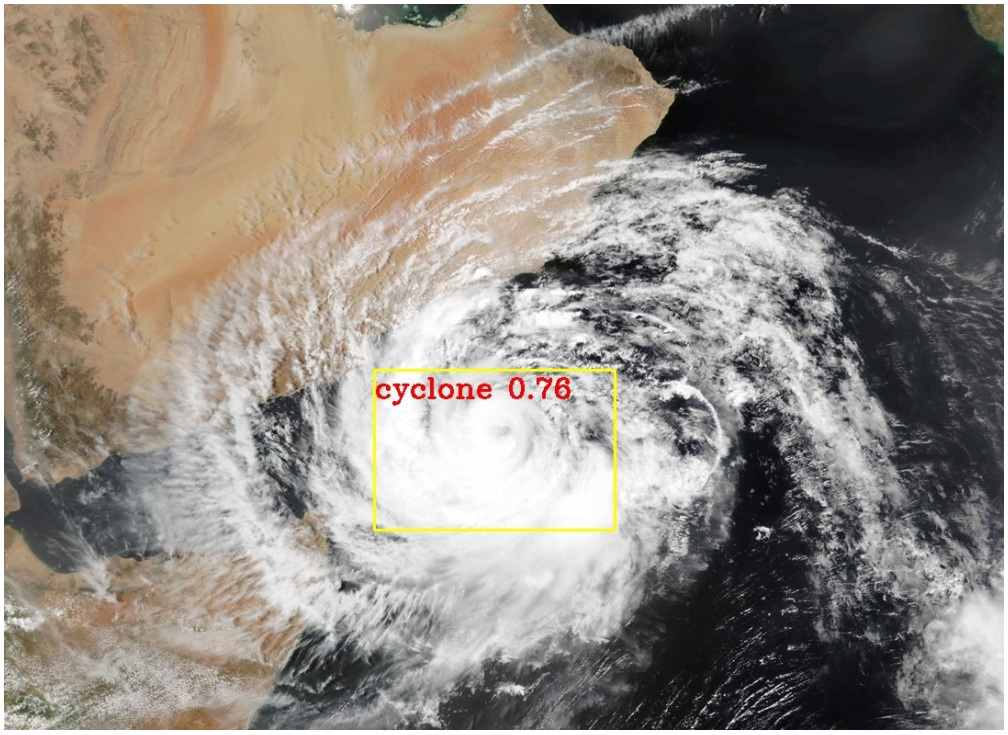


Figure 6.3: Output Image 3

Figure 6.1,6.2,6.3 shows that the cyclones were correctly detected with decent confidence level by the model.

6.2 Metrics in Object Detection

There are several metrics that can be used to evaluate the performance of a YOLO object detection model. Some of these have been used in this project and are discussed as follows:

In object detection in images using deep learning, precision is a measure of the accuracy of the object detector in identifying the location of an object in an image. It is defined as the number of true positive detections (i.e., the number of times the object detector correctly identifies the presence of an object) divided by the total number of detections made by the object detector (true positive detections plus false positive detections) [21].

Precision is usually expressed as a percentage or as a decimal value between 0 and 1, with higher values indicating better performance. A perfect precision score would be 1.0, which would mean that the model made no false positive predictions. The equation for precision is as follows:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{6.1}$$

6.2.1 Recall

In object detection in images, the recall score is a measure of how well the object detector is able to find all instances of the objects of interest in the image. It is calculated as the ratio of the number of instances of the objects that were correctly detected by the object detector to the total number of instances of the objects present in the image. The recall score is an important metric for evaluating the performance of an object detector, especially in situations where it is important to detect all instances of the objects of interest. The formula for precision is given as:

$$Precision = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{6.2}$$

6.2.2 F1 score

F1 score is calculated as the harmonic mean of precision and recall, where precision is the number of true positive detections divided by the total number of positive detections made by the model, and recall is the number of true positive detections divided by the total number of objects in the ground truth [22].

$$Precision = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{6.3}$$

6.2.3 Confidence

Confidence is a measure of how certain the model is that it has correctly detected an object in an image. It is calculated as the product of the probability that the model predicts an object is present in the image and the intersection over union (IoU) of the predicted bounding box and the ground truth bounding box. A higher confidence score indicates that the model is more confident in its detection, while a lower confidence score indicates that the model is less confident.

6.2.4 Mean Average Precision (mAP)

Mean Average Precision (mAP) is a common metric used to measure the accuracy of the model in detecting objects within a set of images. The mAP is calculated by averaging the average precision (AP) of the model across all classes of objects being detected. The AP for each class is calculated by comparing the model's predicted bounding boxes with the ground truth bounding boxes for that class. To calculate the AP for a class, the model's predictions are first ranked by their confidence scores. The AP is then calculated by considering the predicted bounding boxes in order of their confidence scores and computing the precision and recall at different points.

Intersection over Union (IoU) is a metric measures the overlap between the predicted bounding box and the ground truth bounding box. It is calculated as the area of overlap divided by the area of union. A high IoU indicates that the predicted bounding box is a good match for the ground truth bounding box.

6.2.5 Confusion Matrix

A confusion matrix is a table that compares the predicted labels for a set of data with the true labels for that data. The rows of the matrix represent the true labels and the columns represent the predicted labels. Each cell in the matrix represents the number of times a particular true label was predicted as a particular predicted label. The diagonal elements of the matrix represent the number of times the model correctly predicted each category, while off-diagonal elements represent incorrect predictions. The confusion matrix can be used to calculate various evaluation metrics such as precision, recall, and F1 score, which provide a more detailed understanding of the model's performance.

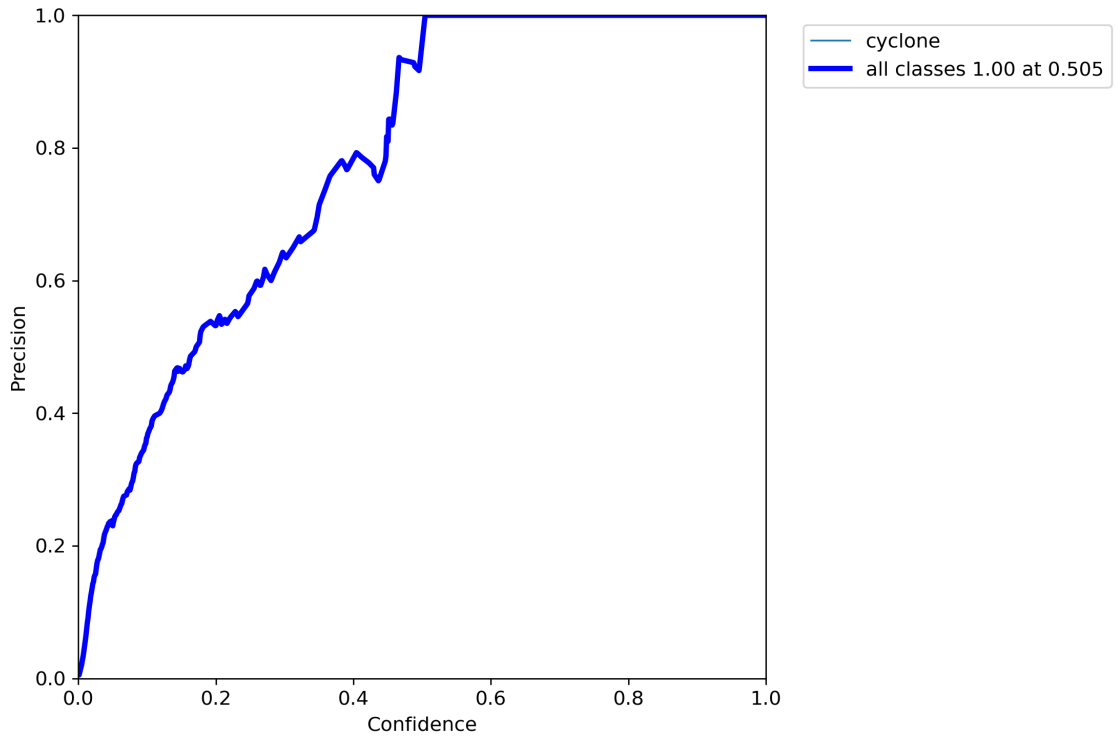


Figure 6.4: Precision Curve

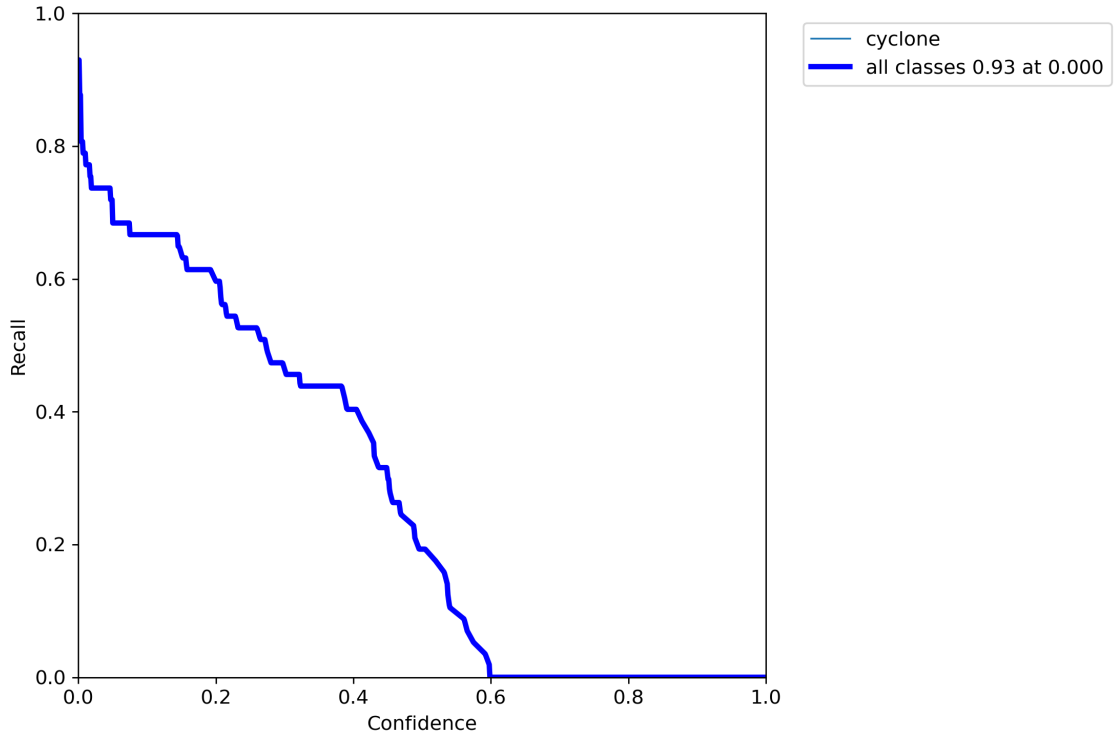


Figure 6.5: Recall Curve

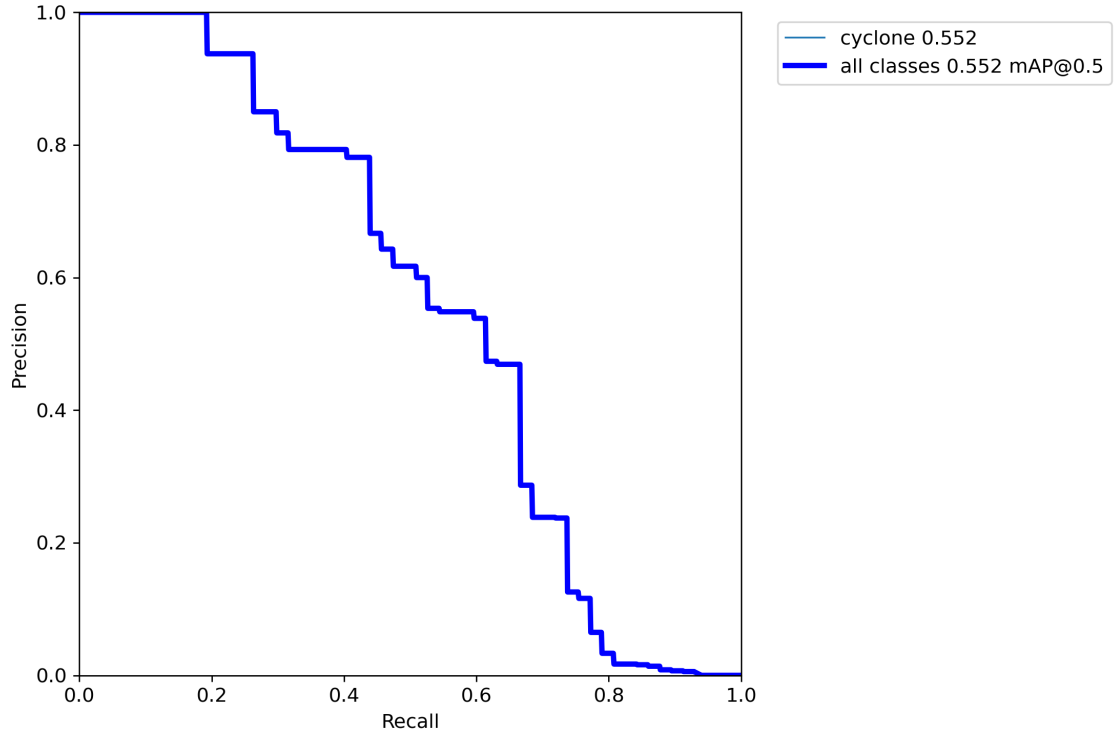


Figure 6.6: Precision-Recall Curve

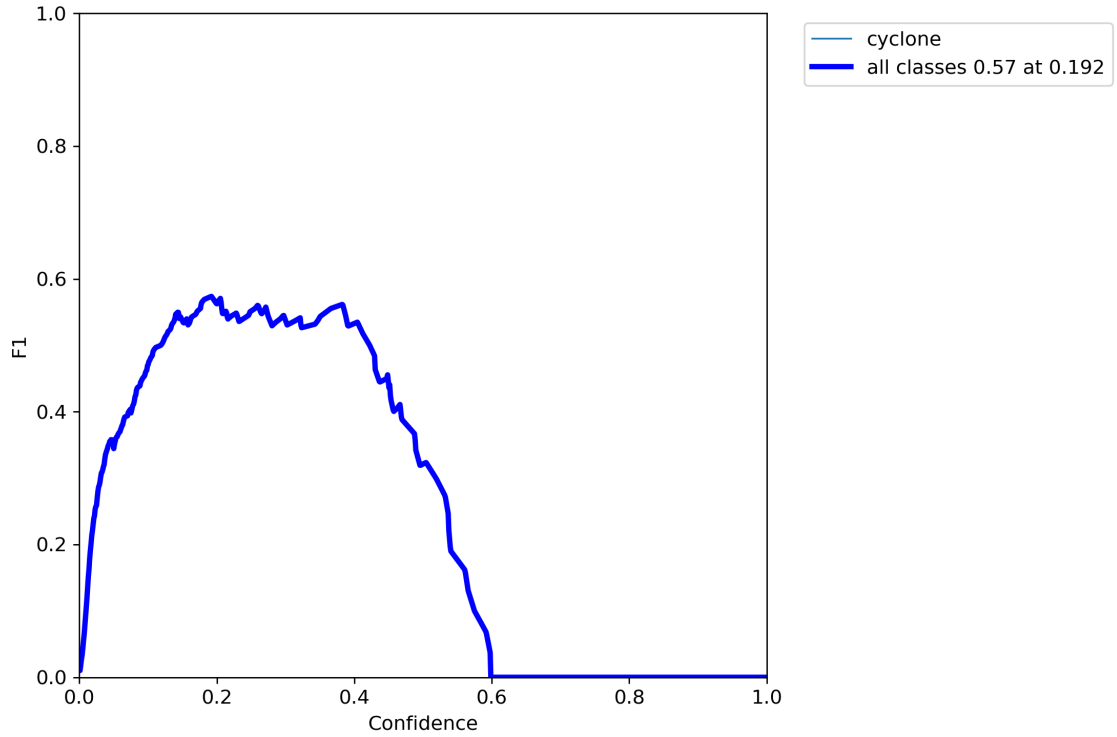


Figure 6.7: F1 Curve

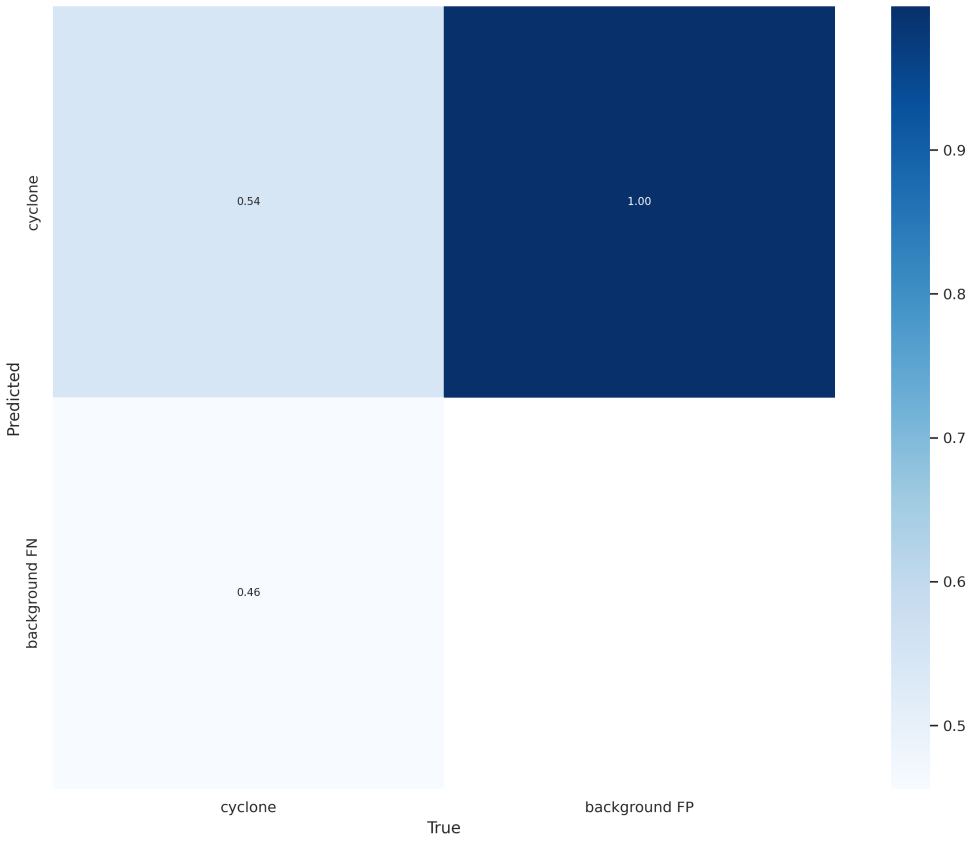


Figure 6.8: Confusion Matrix

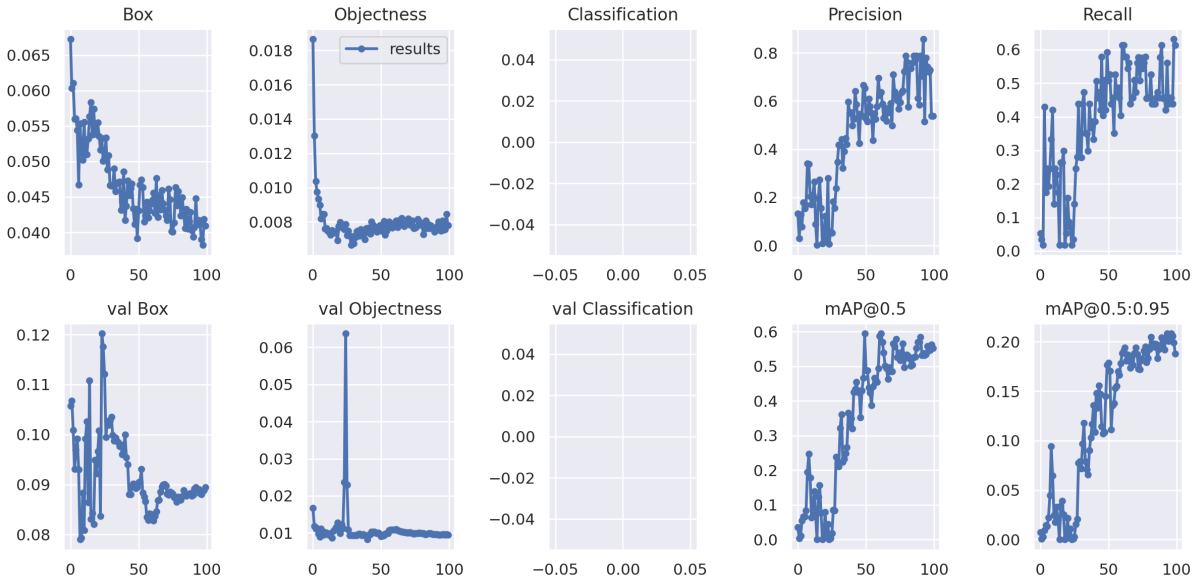


Figure 6.9: Results

We have observed that the precision of our model increases with increase in confidence which the recall decreases as shown in figures 6.4 and 6.5. This signifies that most of the predicted labels are correct when compared to the training labels. The area under the precision recall curve is high as shown in figure 6.6 signifying both high recall and precision. Confusion matrix (figure 6.8) and variation of some matrices per epoch are observed in Figure 6.9.

Sr. No.	Metrics	Training	Testing
1	Class	cylone	cyclone
2	Images	804	201
3	Precision	0.595	0.699
4	Recall	0.614	0.425
5	mAP	0.612	0.584

Table 6.1: Training and Testing Results

6.3 Summary

We only have 1 class to identify i.e, cyclone. As mentioned in the Table. 6.1, for the training process, we have 804 images along with the annotations. We have obtained training precision of 0.595, recall of 0.614 and mAP of 0.612. As far as testing is concerned, we have 201 annotated images along with their label. We have obtained the testing precision as 0.699, recall of 0.425 and mAP of 0.584.

Conclusion & Future Scope

In this project we have discussed the accuracy of YOLOv7 in detecting the cyclone on a satellite image as well as video. We studied and analysed different deep learning models on detecting the cyclone on satellite imagery. We found the Mean Average Precision of our model to be 0.584 in detetcing the cyclone.

An accurate track detection of cyclone is important, because if the track forecast is incorrect, forecasts for intensity, rainfall, storm surge, and tornado threat will also be incorrect. We have witnessed that cyclone can be highly destructive if we do not take actions before it hitting the cities. Hence it is important to detect the cyclone accurately. There have been several research over the past 100 years on the axisymmetric structures, dynamic dynamics, and forecasting methods of tropical cyclones, which have long been a source of concern for meteorologists. Our projects highlights both the numerous difficulties still present and the steady progress. Numerous researchers have confirmed that machine learning, a form of artificial intelligence, can offer a fresh approach to resolving tropical cyclone forecasting challenges, whether employing a pure data-driven model or enhancing numerical models by including machine learning.

In the future we have decided to collect more data of cyclone satellite imagery and further refine the dataset for better accuracy. As we increase the dataset, the accuracy is expected to increase. We will also try to use newer models to see if they can surpass YOLOv7 in accurately predicting the position of cyclone.

We would work in predicting the intensity and direction of cyclone so that we can know in which cities cyclone is headed and we can alert the people in advance. Since forecasting the future location and intensity of tropical cyclones is considered to be the most important function of tropical cyclone warning centres. One reason for the importance is the potential damage and loss of life which can occur with the passage of an intense tropical cyclone.

References

[1] Y. Dai, W. Liu, H. Li, and L. Liu, "Efficient foreign object detection between psds and metro doors via deep neural networks," *IEEE Access*, vol. PP, pp. 1–1, 03 2020.

[2] "Yolov7 object detection paper explanation inference." [Online]. Available: https://learnopencv.com/yolov7-object-detection-paper-explanation-and-inference/

[3] "Recurrent neural networks cheatsheet." [Online]. Available: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

[4] "Bi-lstm explained, papers with code." [Online]. Available: https://paperswithcode.com/method/bilstm

[5] "Gated recurrent unit." [Online]. Available: https://en.wikipedia.org/wiki/Gated_recurrent_unit

[6] "Satellite image dataset." [Online]. Available: https://www.kaggle.com/datasets/mikolajbabula/disaster-images-dataset-cnn-model

[7] M. Haque, A. Adel, and K. Alam, "Deep learning techniques in cyclone detection with cyclone eye localization based on satellite images," 12 2021.

[8] A. Nair, K. S. S. S. Srujan, S. R. Kulkarni, K. Alwadhi, N. Jain, H. Kodamana, S. Sandeep, and V. O. John, "A deep learning framework for the detection of tropical cyclones from satellite images," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

[9] S. Shakya, S. Kumar, and M. Goswami, "Deep learning algorithm for satellite imaging based cyclone detection," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 827–839, 2020.

[10] Y. Wu, X. Geng, Z. Liu, and Z. Shi, "Tropical cyclone forecast using multitask deep learning framework," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

[11] A. Khandelwal, R. S, A. S, B. R, A. P, K. Jhawar, A. Shah, and V. R, "Tropical cyclone tracking and forecasting using bigru [tctfb]," 08 2022.

[12] M. Kim, M.-S. Park, J. Im, S. Park, and M.-I. Lee, "Machine learning approaches for detecting tropical cyclone formation using satellite data," *Remote Sensing*, vol. 11, no. 10, 2019. [Online]. Available: https://www.mdpi.com/2072-4292/11/10/1195

[13] R. Kovordányi and C. Roy, "Cyclone track forecasting based on satellite images using artificial neural networks," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 64, no. 6, pp. 513–521, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271609000434

[14] Y. Zhang, R. Chandra, and J. Gao, "Cyclone track prediction with matrix neural networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[15] S. Alemany, J. Beltran, A. Perez, and S. Ganzfried, "Predicting hurricane trajectories using a recurrent neural network," 2018.

[16] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[17] "Train yolov3 on custom data." [Online]. Available: https://github.com/ultralytics/yolov3/wiki/Train-Custom-Data

[18] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022. [Online]. Available: https://arxiv.org/abs/2207.02696

[19] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[20] "Train yolov7 on custom data." [Online]. Available: https://github.com/WongKinYiu/yolov7

[21] "Accuracy, precision, recall or f1?" [Online]. Available: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

[22] "Confusion matrix, accuracy, precision, recall, f1 score." [Online]. Available: https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd