Detection of Shipping Vessels from Satellite Imagery Using Deep Learning

By
Gadde MuraliChowdary

This document is submitted to the Gannon University graduate faculty in partial fulfillment for the degree Master of Sciencein the Department of Computer and Information Science
Option: IT

Approved By:

Dr.Rahman,Md TajmIlur	Dr.Mariam manakkadu sheheeda
Dr.Nwokeji, joshua	Dr.mei-huei tang **Chair, CIS Department**

Gannon University
Erie, Pennsylvania 16541

December, 2022

ACKNOWLEDGMENTS

I wish to express my sincere thanks to Dr.Rahman,Md TajmIlur for continuous guidance during the project. I am extremely grateful and indebted for his expert, sincere and valuable guidance and encouragement extended to me. I take this opportunity to record my sincere thanks to all the faculty members of the Ganno University for their help and encouragement. I am also thankful to my parents for their unceasing encouragement and support. I also place on record, directly or indirectly, have lent their helping hand in this venture.

ABSTRACT

Vessel detection and tracking is an important process in maritime traffic surveillance areas, whether at sea or on land. It can be used in ocean and coastal surveillance and monitoring applications. This has a significant impact on the safety of navigation, so different systems and technologies are used to determine the best way to track and identify vessels. SAR (Synthetic Aperture Radar) is a top technology for marine surveillance. Further, automatic identification of shipping vessels using deep learning technology can extract the vessel position and categories information from satellite images. Machine learning a subset of Data Science has recently gained popularity on account of improvements in modern technology and the scope of data. Deep Learning demonstrates aptitude and efficiency in completing challenging learning problems that were previously intractable. In this project image segmentation techniques applied for detection, localization, and tracking shipping vessels from satellite images. Although In this project standard U-net model is applied using the Keras library in Python. Different augmentation strategies, model parameterization, and training schedules were tested. Google Colab was used for model development and computation.

Keywords: Image Classification, Deep Learning, Ship Detection, Satellite Images, surveillance

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
Introduction

1.1 Overview

Since ships are used to move freight and promote trade, marine-based transport is crucial because more than 70% of the world is covered by water. An increase in ship traffic is a result of the trend toward globalisation. Ships account for about 80% of world trade by volume and 70% by weight. As trade expands, In order to manage or control ship/vassel traffic, lessen ecological harm and identify illegal ship/vassel movements, it is necessary to track and monitor ship movements.

Ship detection from satellite images is a useful tool for ship monitoring because the quantity of satellites and image quality have significantly improved in the last ten years. For a number of marine uses, like ship traffic services and oceangoing handling, ship detection utilising remote sensing photos is thought to be important Zhu et al. (2010). Automatic ship detection from satellite photos presents a number of issues for academics due to the growth in the volume of optical satellite-collected picture data.

The majority of detection techniques routinely analyze enormous chunk of data beyond sacrificing the efficiency and pace of their calculations Liu et al. (2014). The research look at the most effective methods to achieve respectable results, and similar procedures will be employed.

The two image types that can be used to detect ships are using synthetic aperture radar generally known as SAR process as well as optical remote sensing images. Detailed information about ocean is provided by the SAR images. SAR images make it simple to distinguish between ships and the sea by clearly displaying

the differences between the two. Small images are tough to bundle due to their poor quality Liu et al. (2014).

improvements in optical remote sensing now a days technologies have made it simpler to take high-resolution pictures. This improved resolution aids in giving the detecting system's decision-making process additional spatial and optical detail. It is challenging to identify the ships manually in very detailed satellite pictures due to processing enormous amounts of data.

Therefore, utilising computer vision techniques, one can propose an automated strategy to automate the detecting procedure. Understanding and automating tasks that human visual systems can complete are the main goals of computer vision. The application of tools and techniques from machine learning, signal processing and deep learning for object detection in images and videos is the core of the object detection subfield of computer vision. Convolutional neural networks (CNN) and other deep learning models have been effectively used to solve image identification and object detection issues. Many different CNN variations have been employed in published research projects in the past to recognise ships from satellite photos.

When trying to find ships in optical satellite photography, clouds, landmasses, and artificial structures can lead to erroneous positives. The community of remote sensing experts has highlighted this as an important topic of interest. This effort target is to improve performance in ship/vassel detection and identification by including optical satellite imagery for decision making over relevant strategy. Implementing standard machine learning supervised learning process of training, testing and validating the

given satellite images handling to the matches the final results, a UNet Deep Learning model will be trained to forecast segmentation masks of ships of various sizes.

Gallego et al. (2018) Zhang et al. (2016). The sample optical remote sensing image in 1.1 captured for detection purposes.

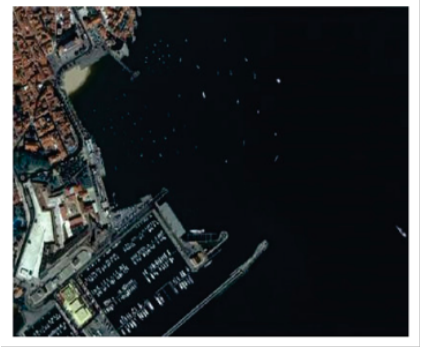

Figure 1.1: Optical remote sensing image

This particular UNet Deep Learning model uses extremely deeper layers and shallow to produce a class segmentation mask that is more precise than those from other studies and a much improved extracted feature representation while processing in the encoding developement. This claim will be supported visually by an interpretive examination of segmentation masks on the target and validation data, as well as practically by class accuracy scores. Two distinct systematic methods will be used to train the customized U-Net: a parameter server version and a single node design that is quite similar to conventional training. This variation uses gradient accumulation to implement the workers as weak learner boosters in order to train over the dataset. A more generalizable model is intended as a result of the parameter server variant's

multiple worker exploration of various gradient paths. In the fields of economic, ecological, and security sectors, the outcome of improved class accuracy recognising ships in satellite optical imaging can be useful for business and governmental organisations.

1.2 Project Description

Shipping traffic is growing fast. More ships increase the chances of infractions at sea like environmentally devastating ship accidents, piracy, illegal fishing, drug trafficking, and illegal cargo movement. This has compelled many organizations, from environmental protection agencies to insurance companies and national government authorities, to have a closer watch over the open seas.

Aim of this project is to derive meaningful solution for wide coverage, fine details, intensive monitoring, premium reactivity and interpretation response. Combining its proprietary-data with highly-trained analysts, they help to support the maritime industry to increase knowledge, anticipate threats, trigger alerts, and improve efficiency at sea.

The major focus of this project is to develop an AI application for detection of Shipping Vessels from Satellite Imagery using deep learning. Ship detection from remote sensing imagery is a crucial application for maritime security which includes among others traffic surveillance, protection against illegal fisheries, oil discharge control and sea pollution monitoring. This is typically done through the use of an Automated Identification System (AIS), which uses VHF radio frequencies to wirelessly broadcast the ships location, destination and identity to nearby receiver devices on other ships and land-based systems.

CHAPTER 2
Literature Review

2.1 Detection and Tracking of a ship based on Deep Learning

It is essential to be aware of the area around ships in order to prevent marine mishaps. A ship may be made aware of other ships nearby using the images captured by a camera installed on it. Three techniques were used in this work to accomplish ship awareness: detection, localisation, and tracking. Ship recognition from camera images was the first application for the You Only Look Once, or YOLO, Version 3 which is well known technique for detection and identification. Unity was used to create a virtual image collection in order to get past a challenging situation acquiring onboard standard images captured by camera of ships in various sizes and in the process to get as much possible better identification and detection.

The word "detection" is used broadly, and other times it is used more specifically. The three consecutive main processes that make up the whole vessel detection procedure are detection in the broadest sense, and they are as follows:

- Finding potential vessels in the image and locating them is known as Ship and Vessel detection.
- Vessel classification: determining whether detected targets are vessels or not before determining the class of the ship and vessel
- Vessel identification is the process of determining the ship's distinctive identity.

The localization process involved using the ship's orientation data to determine where the horizon was located on the image. After that, the identified vessel/ship's location there existed in a spatial coordinate system was ascertained using the horizon data.

The expanded Kalman filter was then used to track the target ships' location, course over land, and speed over ground in time. The deep learning algorithm-based network was created to determine the picture's ship's heading and use that information to determine the starting parameters of the Kalman filter using the wealth of camera data.

The suggested technique for ship awareness was approved using a real film collected from a camera mounted on a real ship with a range of encountering circumstances. Actual data were contrasted with the tracking findings collected by automated ship identification systems from other ships. The entire identification, localization, and tracing procedures worked well, indicating the suggested way for a ship/vessel's awareness of its nearby, from the camera images.

In order to learn more about nearby maritime or inland traffic, vessels frequently employ data from sensors like Automatic Identification Systems (AIS), Radio Detection and Ranging (RADAR), and Light Detection and Ranging (LIDAR) Lee et al. (2020). All of these sensors, however, have some limits. Without AIS technology, it is impossible to get information on items like buoys and small boats. Radar has the benefit of having a long range of object detection. However, there is little object information available, and updates happen very slowly. Although LIDAR is capable of precise detection, light reflects off the water's surface and the detection distance is short. Therefore, to overcome these shortcomings, it would be very beneficial to design a ship-detecting system based on satellite images.

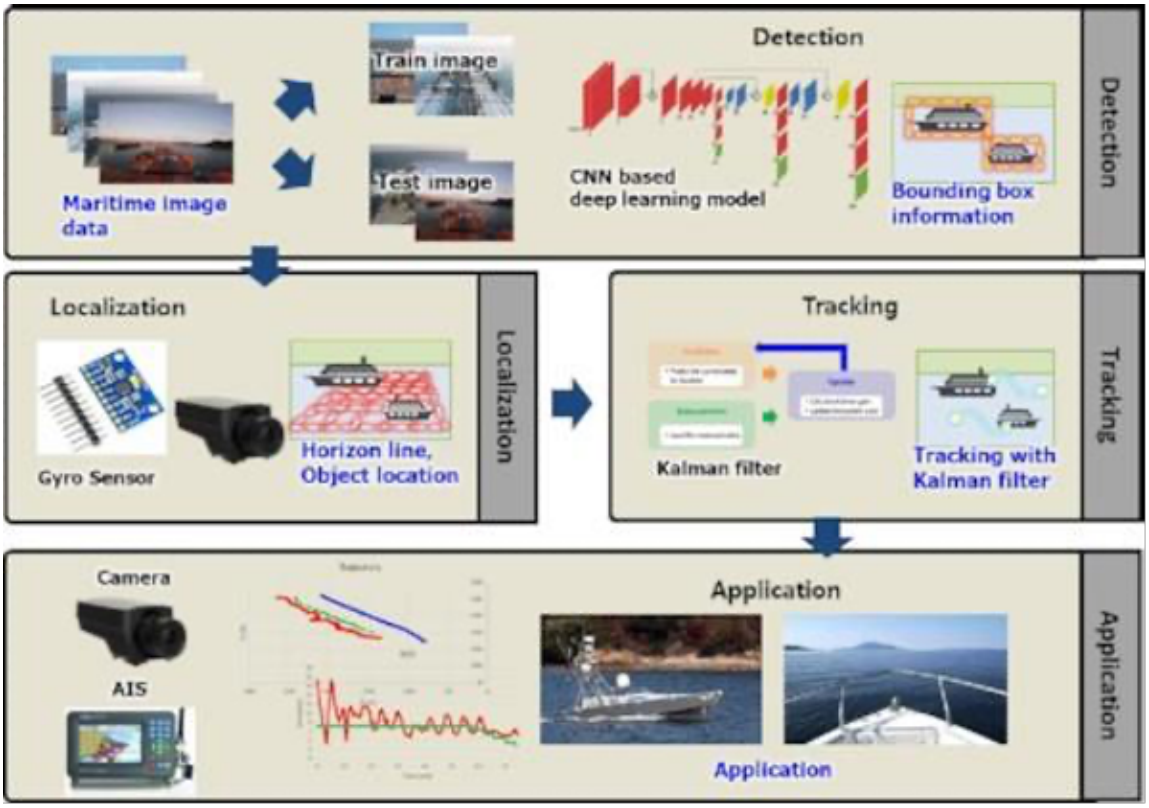

Figure 2.1: Ship Detection

2.2 Ship Detection using Satellite Images

Monitoring maritime environments with remote sensing presents a significant challenge for both environmental and security reasons. The field of remote sensing has recently shown a specialized interest in learning methods from the study of machine learning. It is a particularly challenging challenge to automatically categorise ships from satellite imagery, which is required for traffic surveillance systems, the defence of illicit fisheries, oil discharge control systems, and the monitoring of maritime pollution. The abundance of data and recent developments in digital technology have stoked interest in deep learning, a subfield of machine learning.

Recent years have seen a rise in the popularity of machine learning, and numerous applications of artificial intelligence now play a significant role in our daily lives. Changes in our approach to algorithm design are the driving force behind these new advancements in artificial intelligence. In particular, human preprocessing and feature engineering were essential for data-driven machine learning. Deep learning is the study of these data-driven methods. Consider deep learning as a solution for your object detection use case. As a result, we employ more sophisticated techniques like transfer learning, which involves applying a model that has been trained for one job to another Swamidason et al. (2020).

Developing a deep learning-based automatic ship detection system to analyze the Airbus ship Detection dataset, which contains nearly 40K images captured by satellite. This study examines and evaluates the YOLO technique for the identification of ships/vessels from images captured by satellite. Following training to be done on a desktop or laptop computer using a sizable dataset of images captured by satellite from the Airbus Ship Detection Challenge and Shipsnet, various iterations of the YOLO model for ship recognition, including YOLOv3, YOLOv4, and YOLOv5, are compared. The study's findings demonstrate that the YOLOv5 object detection algorithm has given better outcomes than the YOLOv4 and YOLOv3 algorithms.

A supervised algorithm (SVM) is used in the paper Morillas et al. (2015) to solve the problems with ship detection. In earlier applications, a variety of images were used; for instance, when using high-resolution images, it is very challenging to identify ships from the images. Block division is therefore used in the proposed method to address or circumvent these kinds of problems. The optical satellite images are split

up into tiny pixels by this block division. In the images, this either symbolizes little ships or nothing at all. With the help of the images' texture and colour features, the proposed supervised algorithm (SVM) will be trained. The colour depicts the chromatic properties of the area, while texture features reveal information about the dispersion of images. Therefore, using a reformation algorithm will enhance ship detection after classification. This reforming algorithm removes the detected ships from the images and corrects any incorrectly classified parts or blocks. This colour and texture combination achieves an accuracy value of 96.98%. Additionally, in the last phase of ship detection, the classification of ship blocks and no ship blocks reaches 98.14%.

In this project, the general architecture of Standard U-net Model Ronneberger et al. (2015) is used. The model implemented in Keras uses a series of convolutional with a ReLU activation function and pooling layers. An important part of model building is specifying the loss function. It is used to quantify the error between actual and predicted outputs at each step. The loss function plays a large role in how the model behaves and converges.

The term "optical remote sensing images" refers to multispectral (MS) and panchromatic (PAN) remote sensing images that are primarily composed of blue colour, green colour, red colour, and near-infrared bands. The four additional colour sensors on the WorldView-2 satellite are the Coastal, Second Near-Infrared, Red Edge, Yellow, and bands. There were also a few multispectral remote sensing images from thermal infrared sensors (TIRS), mid-wavelength infrared sensors (MWIS), and short-wavelength infrared sensors (SWIRS). In terms of concepts, "ship" refers

to manufactured items that ply the seas. As for "object detection," it refers to observable man-made items in remote sensing photographs that include a range of targets, including ships, aeroplanes, vehicles, etc.

2.3 Workflow for vessel detection on optical images

The vessel detection problem can be conceptualized on optical images as a relatively straightforward detection of bright objects against a darker background. Truth, however, is far more nuanced; a ship might actually be darker compared to the ocean surface around it, and it might also be a number of other bright objects in the scene that are mistaken for vessels. The majority of the collected material on vessel detection breaks down the workflow of their processing into a few fundamental components. The first step in the main workflow is the sea-land separation (Fig. 4), where land portions are removed from the image and not further processed. The algorithm's following stage, the elimination or moderation of environmental influences (such as sunglint, clouds, and waves), was only partially implemented by the authors. The most crucial phase of the procedure that comes next is vessel element detection when the vessel element objects are gathered. Following that, objects are separated from the remaining detected, non-vessel objects. Classification, in which vessels are further classified into more specific vessel classes based on particular qualities, may upgrade discrimination. The final step in the vessel detection method is evaluating the acquired results. Below are detailed explanations for each of the main sections. Kanjir et al. (2018)

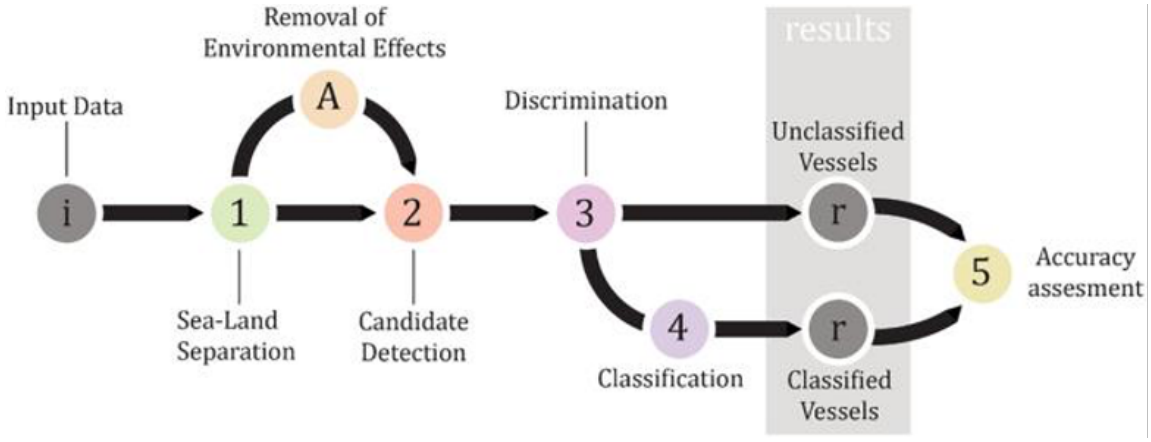

Figure 2.2: Sea-land separation Workflow

CHAPTER 3
Data Processing

3.1 Dataset

The Airbus Ship Detection challenge dataset was used in this project. Remote sensing images of ships in ports and the ocean make up this dataset. The ship detection dataset includes more than 100k satellite images of resolution 768 x 768 with the total size about 30 Gb and it is little bit imbalanced it means that only approximately $\frac{1}{4}$ of the data images have ships and many contains more than one ships. Ships within and between images might be in the open ocean, at docks, marinas, etc., and their sizes might vary (sometimes noticeably). The dataset included a csv formatted file with dual parameters: the first independent parameter is image identification number, and the second independent parameter is encoded pixel coordinates for the given image identification number in first parameter, which represent the segmented for training and testing.

Segments of an item cannot overlap for this metric. When two ships were adjacent to one another, there were a small number of photos in both the Train and Test set that had a slight overlap of object segments. By switching the overlapping portions to background (or "non-ship") encoding, any overlaps were eliminated. As a result, some images have a ground truth that could be a bounding box that is aligned but has some pixels removed from one of the segment's edges.

3.2 Data Pre-processing

Data pre-processing includes following process followed:

3.2.1 RLE-Encoding and Decoding Function

First of all we will read the given csv file which contains two attribute one is image id and the other is RLE run length encoding value. The csv file will be loaded from drive using pandas in the for of dataframe using read˙csv method. After reading the file from drive the given RLE run length encoding will be decoded or encoded using the functions mentioned below. As we have already discussed in previous section the given location of ship is in the RLE run-length encoding. To decode the given values in this project rle˙encoding and rle˙decoding function is used. The encode function takes input for image as a numpy array where 1 stands for mask and 0 stands for background, after processing function returns run length as a string formatted which is RLE run-length-encoding. The decode function takes run-length as string formatted as a input and this function returns a numpy array where 1 stands for mask and 0 stands for background. Figure show preview for the encoded mask from the given RLE run-length-encoding in csv file.

3.2.2 Process Multiple Ships

Next pre-processing step is to merge image layers with multiple ships which means that it might be possible that there are more than one ships in single images. For these images decoding of RLE run length encoding will be done as mentioned in previous section after processing the same preview of the image can be verified form the Figure. Once the multiple ships are detected in images next step is to remove corrupted or very small images in this project I have removed image files less then size 50kb.

Figure 3.1: Multiple Ships Mask

3.2.3 Train Test Split

An evaluation technique for machine learning algorithms is the train-test split. As part of the method, a dataset is divided into two subsets. For training our model it is the process to fit the given data to the model the data that is being given to model for learning refeared as training data generally the training data is the 80 percentage of the entire given data. Then the model will be trained without using the remaining test subset which is remaining 20 percentage of entire given data, and predictions are made before being compared to expected values. The official name of this second dataset is the test dataset. Make use of the train dataset when fitting the machine learning model. To evaluate the learning made by the machine learning model using training data or the model trained properly using training data or not, the testing subset will be used. The main goal of doing this process is when new

unseen data will be provided to model how the model will perform towards that data as the testing data was not provided to the model when training the model. This the the standard process used for training and testing model. In order to make predictions about future examples where we won't have the target values or expected outputs so it is important to validate our model for the data points for which we have both input and output both values. Train and Test Split done in this project using sklearn library's train˙test˙split() function. Another step is to take sample of 10000 from the given dataset using pandas dataframe's sample function. For training and testing our model. A balanced sampling ship count can be visualized in the Figure. Balanced sampling is used for better training of model.

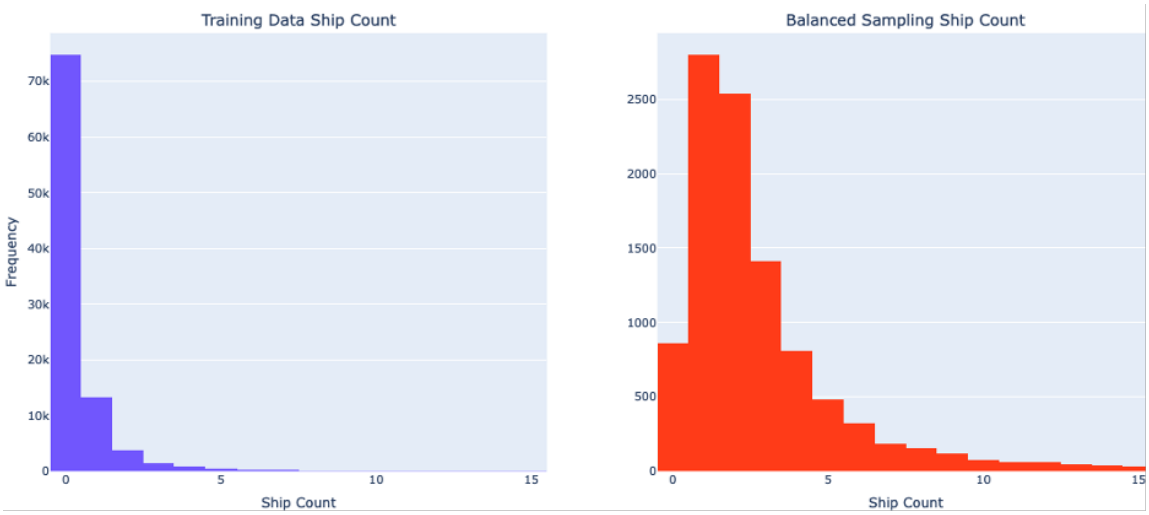

Figure 3.2: Balanced Data Sampeling

3.2.4 Image Augmentation & Data Generator

Image augmentation is the process of altering original images in various ways to create numerous altered versions of the given image. Each image is different copy of

the image given in input the only variation will be depending upon the parameters used in augmentation techniques, different techniques are shifting, rotating, flipping, etc. The target class is unaffected by due to this minor effect given to the input image; they merely provide a different perspective for getting the sample of situation from different angles in real life situation. The augmentation technique is very useful for making the deep learning model robust towards real world situations. These methods of image augmentation not only expand the size of your dataset but also add some variance, making it easier for your model to generalize to new data. Additionally, the model gets stronger as It is honed using recent, minimally altered images. This means that you can quickly create a large corpus of comparable images with only a few lines of code without having to worry about gathering fresh images, which is impractical in a real-world scenario.

One quick and simple way to improve your photos is to use the Keras ImageDataGenerator class. Entire range of different augmentation techniques are included. The Keras ImageDataGenerator's primary benefit is that real-time data augmentation is one of its main goals. The ImageDataGenerator class provides new versions of the images to the model every epoch. It only provides the edited images; it does not include it in the corpus of original photographs. If this were the case, our model would be overfit because the model would be viewing the original images more than once. Figure shows the augmented sample image.

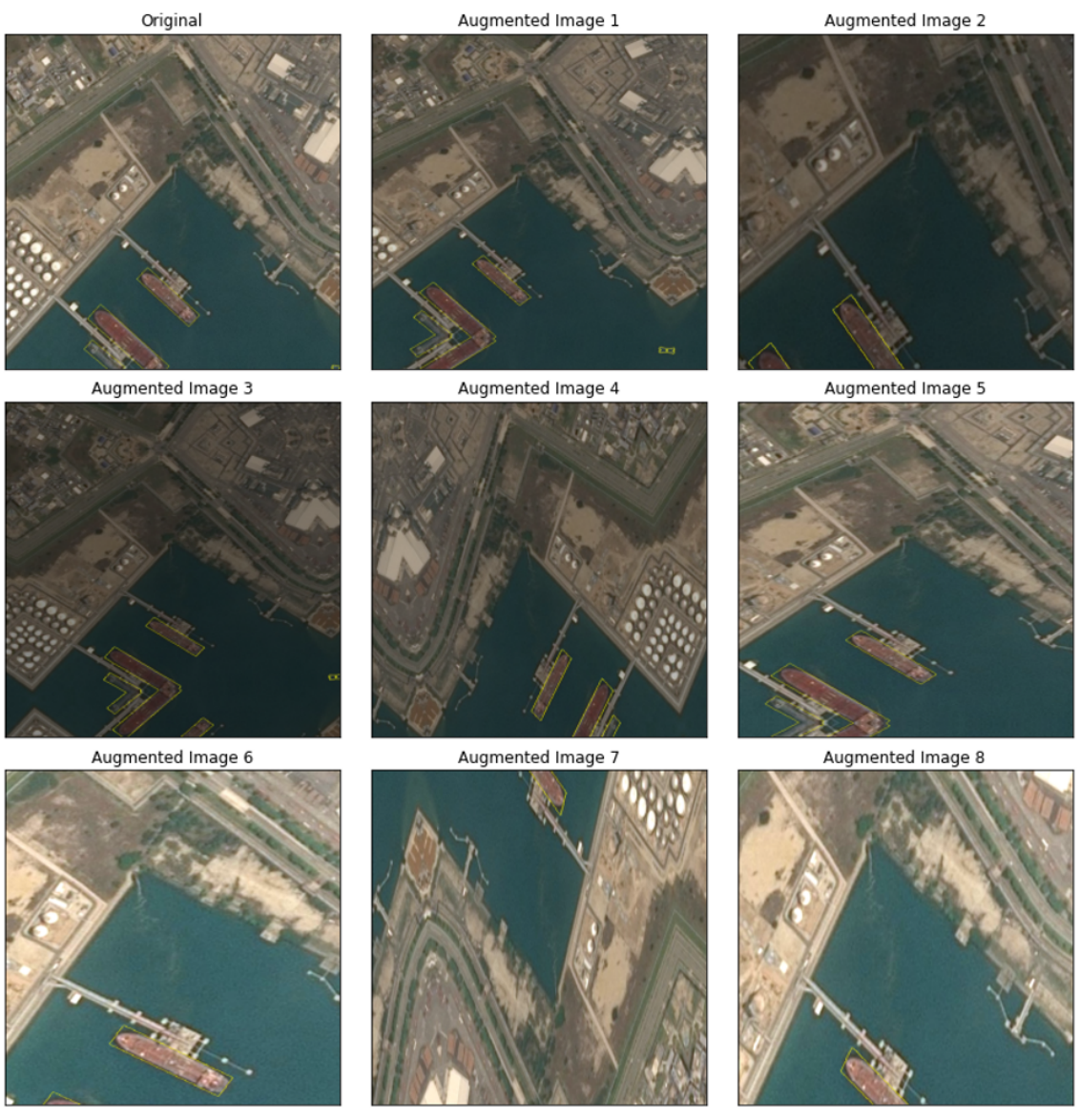

Figure 3.3: Data Augmentation

CHAPTER 4
Chapter 4 Methodology

4.1 Deep Learning

Deep learning is a type of machine learning and artificial intelligence (AI) approach that mimics how individuals study specific subjects. In addition to statistics and predictive modeling, deep learning is a crucial part of data science. One way to think of deep learning is as an automated type of predictive analytics. Deep learning algorithms are layered in a hierarchy with more layers of complexity and abstraction than conventional linear machine learning algorithms.

The main key point of deep learning model is that first of all it will consider a problem as a nonlinear problem then train the model based on that. There are several number of iterations until the desired outcome is solid enough so that decision can be made based on the model's prediction. Deep was given its name because of the quantity of processing layers that data must go through. A programmer must be very explicit when instructing the computer what to search for in an image to evaluate whether or not it includes a desired item when utilising machine learning, where the learning process is often supervised. The programmer's ability to precisely define a feature set for dogs determines the computer's success rate during this laborious process known as feature extraction. The benefit of deep learning is that the feature set is developed by the software independently and without supervision. Unsupervised learning typically improves accuracy while also moving more quickly.

A large portion of deep learning models are supported by an artificial neural network, which is a kind of highly developed machine learning algorithm. Deep

learning is sometimes known as deep neural learning or deep neural networking as a result. There are benefits for specific use cases for each type of neural network, including

- Feedforward neural networks formally known as FNN
- Recurrent neural networks formally known as RNN
- Convolutional neural networks formally known as CNN
- Artificial neural networks formally known as ANN

But they all function somewhat similarly in that data is fed into the model, which then determines whether or not it has correctly interpreted something for a the givne input of independent parameters.

4.1.1 Artificial Neural Network

Computer programs called artificial neural networks are modeled after the biological neural systems in animal brains. The majority of the time, these systems lack task-specific rules and instead "learn" how to do tasks by considering instances.

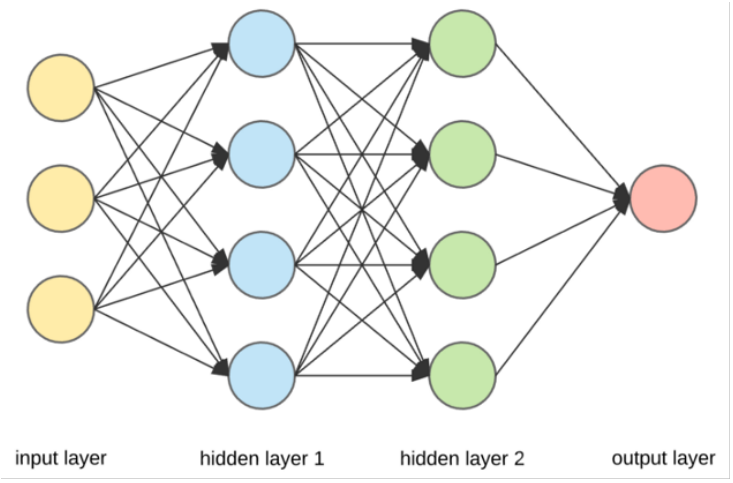

Figure 4.1: Structure of Artificial Neural Network

The three types of layers that make up the neural network are:

- Initial data for the neural network is in the input layer.
- Between the input and output layers are hidden layers, which serve as the hub for all computing.
- Produce the desired outcome for the inputs in the output layer.

On the image 4.1, On the left, there are yellow circles.They often go by the name "vector X" and stand in for the input layer. The next blue and green circles signify known as the hidden layers. These circles, which stand in for the activation nodes, are frequently abbreviated as W. The red circle is a representation of the output layer or predicted value.

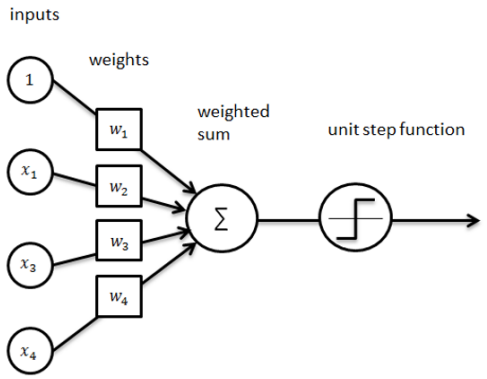

Figure 4.2: Artificial Neural Network Process

There is a particular weight assigned to each connection (black arrow) between a node and a node in the layer beneath it. Weight can be thought of as the influence a node has on a node in the following tier. As a result, if we were to focus on just single node that would be looing as it is. See 4.1 and focus on the blue node in the

top position. It is connected to every node from the preceding layer (yellow). Each of these connections has a weight component (impact). When all of the yellow layer's node values are multiplied by their weights and added together the top most blue colored nodes has a calculated values. The blue colored node's predefined "activation" function unit step function on 4.3 determines whether or how "active" it. The extra node is called the bias node which will have the value 1. A more straightforward Neural Network model is employed to make the mathematical equations easier to understand. There will be 4 input nodes in this model. There is a hidden layer in the network with four nodes those are three plus one for "bias" and a single output node.

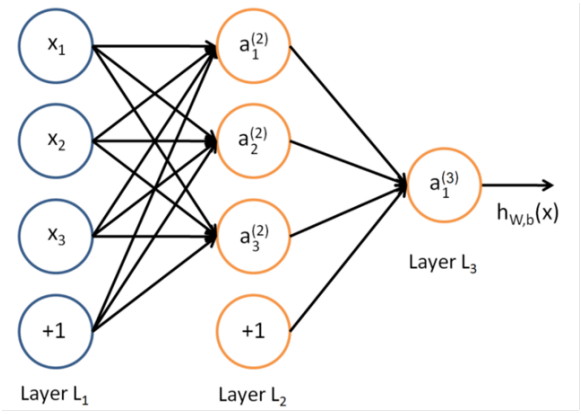

Figure 4.3: Artificial Neural Network Process

Place the "bias" nodes, x_0, and a_0, in that order. So, it is possible to place the nodes in the particular layer from the input layer in the form of vector A and the

nodes in particular layer from the hidden layer in the vector X.

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, A = \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \end{bmatrix},$$

Typically, the weights (arrows) are denoted by W or. On writing them down as in this instance. The representation of the weights between the given input layer in the network and the given hidden layer in the network as a 3x4 matrix. Weights among the given hidden layer in the network and the output layer in the network will also be represented as a 1x4 matrix. When a given neural network has a unit in layer named j and another unit in layer named as j+1, where layer j's dimension is b (a+1).

$$\Theta^{(1)} = \begin{bmatrix} \Theta_{10} & \Theta_{11} & \Theta_{12} & \Theta_{13} \\ \Theta_{20} & \Theta_{21} & \Theta_{22} & \Theta_{23} \\ \Theta_{30} & \Theta_{31} & \Theta_{32} & \Theta_{33} \end{bmatrix},$$

The hidden layer's "activation" nodes are then calculated.Prior to using the activation function g, the input vector X and weights matrix 1 for the top layer (X*1) must first be multiplied. The outcome is:

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

The output for the hypothesis function is derived by dividing the hidden layer vector by the second layer's weights matrix:

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

In the given example of neural network the is only one hidden layer included with four nodes. In general situation for neural network with n number of hidden layers then ther would be very large number of nodes in the particular layer.

4.1.2 Activation Functions

Based on the weighted total, the activation function in a neural network determines whether a specific node should be "activated" or not. This weighted total value will be defined as z. In this section the "Sigmoid Function," generally used activation functions,

Step Function

The so-called "Step Function" (discrete output values) is one of the first concepts that comes to mind. We specify the threshold value and:

In case when (threshold ¡ z) : activate the node it means value will be 1 In case when (threshold ¿ z) : don't activate the node it means value will be 0

Although it has a lovely appearance, this has the disadvantage that the node can only output a value of 1 or 0. We run into trouble if we try to map numerous output classes (nodes). The issue is that numerous output classes/nodes could be turned on . Therefore, we are unable to accurately categorise or determine.

Linear Function

A range of output values could also be obtained by defining "Linear Function."

$$y = ax$$

Any non-linear input cannot be mapped because the neural network's output layer will always have a linear function if only linear functions are used in the neural network. There is proof to back this up that

$$f(x) = x + 3$$

$$g(x) = 2x + 5$$

By funciton composition

$$g(f(x)) = 2(x + 3) + 5 = 2x + 11$$

The result is linear funciton.

Sigmoid Function

It is currently actively used activation function, With the formula below, its equation is provided.

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

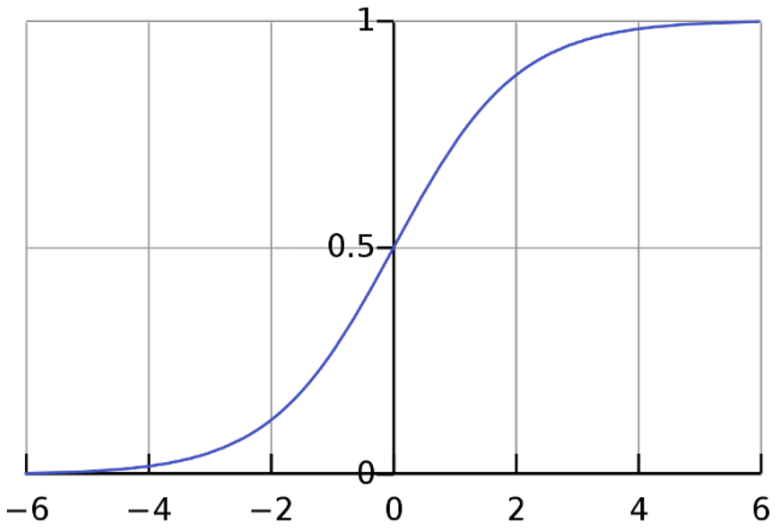

Figure 4.4: Sigmoid Function

It is so well-liked because of its many positive traits: Its function is non-linear, Range values range from to (0,1) and the function is quite steps between (-2,2) on the x-axis, that result it to categorise values as either 1 or 0. These characteristics enable the nodes to accept any values between 0 and 1. Multiple output classes would ultimately lead to various probability of "activation" for each output class. The option with the highest "activation" value will be picked.

Bias Node

Making an effective learning model typically requires the use of the "bias" node. In essence, a bias value enables the activation function to be moved to the left or right, improving data fit (better prediction function as output).

The three sigmoid functions 4.5 show how multiplying, adding, or subtracting the x variable by a certain amount can affect the function.

When x is multiplied, the function becomes steeper. Add/Subtract x, then left- or right-shift the function

Figure 4.5: Different Examples of Sigmoid Function

Cost Function

Let's begin by defining the cost function's general equation. The difference among the anticipated value and actual value is represented by this function as the sum of errors.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}$$

The only discrete values that y can have in this classification problem are 0 or 1. There can only be one kind of class for it. Because of this, we need our hypothesis to be true.

$$0 \leq h_\theta(x) \leq 1$$

Therefore, we shall describe our theory as follows:

$$h_\theta(x) = g(\theta^T x)$$

where g will be the Sigmoid function in this instance because its range of values is between (0,1). Finding min $J(\theta)$ is necessary because our objective is to optimise

the cost function. However, because the sigmoid function is "non-convex", there are numerous local minimums. As a result, it is not assured that it will reach the worldwide minimum. In order for the gradient descent approach to obtain the global minimum (minimise $J(\theta)$), we require a "convex" function. Using the log function, we can accomplish that.

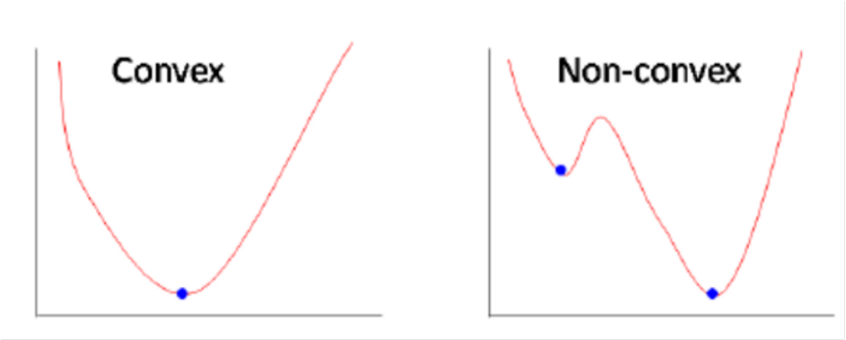

Figure 4.6: Convex - Non-Convex

So, for neural networks, it utilise the following cost function.

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)), & \text{if } y = 1 \\ -\log(1 - h_\theta(x)), & \text{if } y = 0 \end{cases}$$

Looking at the function graphs will reveal how straightforward the idea is. Let's first consider the scenario in which y=1. The graph below would be the result of $-log(h(x))$.By considering the $(0, 1)$ since only values within that range are allowed for the x-axis interval 4.7.

Figure 4.7: Cost Function

The graph shows that, as it is the correct prediction, on considering $y = 1$ and $h(x)$ approaches towards the value of 1 which is x-axis and the output cost value approaches to the value of 0 such that $(h(x) - y$ would be 0. Otherwise, the cost function increases infinite as $h(x)$ gets closer to 0.

The cost function in the alternative scenario with $y = 0$ is $-log(1 - h(x))$.

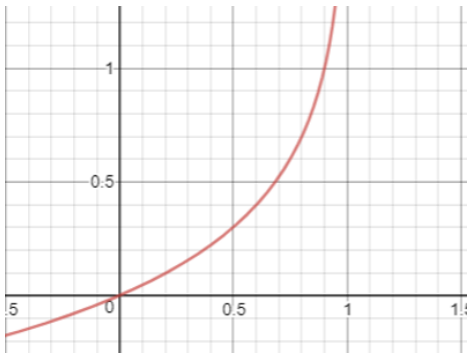

Figure 4.8: Cost Function

Since this is the right prediction in this circumstance, from the 4.8 that if $h(x)$ approaches towards the value of 0, the final cost would also approach towards 0.

Since y is binary function the output must be either 0 or 1, the cost function can be written as a single equation.

$$Cost(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x)))$$

The cost function "regularisation" is represented by the right parts of the equations. By lowering the magnitude/values of θ, this regularisation prevents the data from "overfitting."

4.1.3 Layer Types

Convolutional neural networks can be constructed with a variety of layers, however, most probably below mentioned layers are included:

- Convolution Layer
- Activation Function
- Pooling Layer
- Fully connected
- Batch normalization
- Dropout

A Convolution Neural Network's basic components are mentioned above. Frequently used short text diagrams to represent a Convolution Neural Network

$$\text{INPUT} => \text{CONV} => \text{RELU} => FC => \text{SOFTMAX}$$

The only layer types which have parameters learned through training are the convolution layer and the Fully Connected Layer. Although activation and dropout layers are frequently shown in network diagrams to make the structural plain, they are not

really considered to be "layers" in and of themselves. Network diagrams also feature pooling layers, which are equally important to Convolution and Fully Connected and significantly affect an image's spatial dimensions as it passes through a CNN. The most crucial components for defining your actual network architecture are the convolution layer, pooling layer RELU Activation Function, and Fully Connected Layer. The other layers are still important, but this key group of four determine the architecture itself, thus they take a backseat.

Convolution Layers

The Convolution Neural Network must have a convolution layer as it is the basic need of the Convolution Neural Network. kernels that each have a width and a height and are almost always square, make up the parameters of the convolution layer. These filters, despite being small, extend to the volume's full depth. The number of channels in the image determines the depth for inputs to CNN. The depth for volumes further down the network will depend on how many filters were used in the layer before. Consider the CNN's forward pass, which combined each of the K filters across the width and height of the input volume. Think of our K kernels moving over the input region, multiplying elements one by one, adding the results, and then storing the results. The result is a 2-D activation map exactly as the one in **??**.

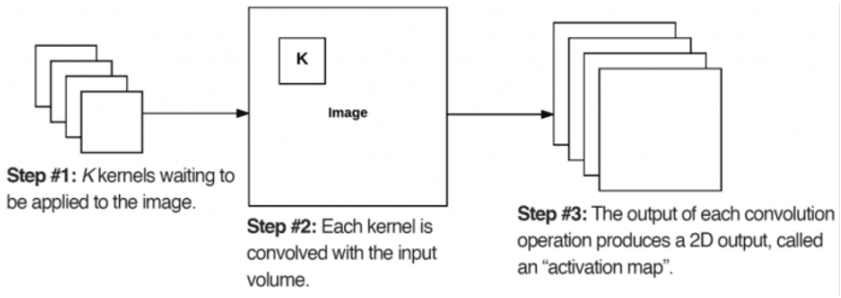

Figure 4.9: Convolution layer Structure

After applying each of the K filters—K denotes the two-dimensional activation maps—to the input volume. After that, we stack our K activation maps along the array's depth dimension to create the final output volume. **??**.

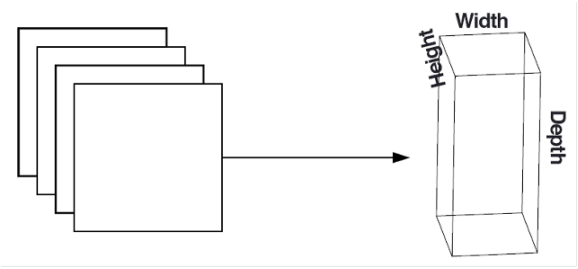

Figure 4.10: Convolution layer Output

Since each neuron, only "looks" at a portion of the input, each entry in the output volume represents the output of a neuron. This allows the network to "learn" filters that turn on when particular features are present in a particular spatial area of the input volume. Filters in the network's lower layers may activate when they find regions that resemble an edge or a corner.

When high-level features like a face's position, whether it is a dog or a car, a car or a bus, a car's licence plate, a road lane, etc. are present, filters in the network's

deeper layers may activate. This theory of activation proposes that specific patterns in an input image "excite" and "activate" these neurons.

The principle of convolving a small filter with a large(r) input volume is particularly important to convolutional neural networks since it corresponds to the local connections and receptive field of a neuron. When working with photographs, it is typically impossible to connect all of the neurons in the current volume to all of the neurons in the previous volume since there are just too many connections and weights, making the training of deep networks on images with vast spatial dimensions difficult. Each CNN neuron should ideally only be connected to a very small piece of the input volume, known as the neuron's receptive field. The three parameters depth, stride, and zero-padding size all have an impact on the output volume size.

Stride

The size of the image pixels should be surrounded by a new depth column in the context of CNNs for each step, then Convolute every one of the K filters with the area, then save the results the volume in 3D. The convolution layers are typically created with a stride with step size S would be either S = 1 or S = 2. Less steps size will result in greater output volumes and overlapping receptive fields. On the other hand, bigger strides will lead to lower output volumes and less overlapping receptive fields. Consider 4.11 for a more realistic example of convolutional stride, where we have a 33 Laplacian kernel and a 55 input picture.

95	242	186	152	39
39	14	220	153	180
5	247	212	54	46
46	77	133	110	74
156	35	74	93	116

0	1	0
1	-4	1
0	1	0

Figure 4.11: Convolve Laplacian Kernal

Our kernel moves one pixel at a time from the left to the right and from the top to the bottom when S = 1, resulting in the display seen 4.12 . The output volume would be smaller if the same operation were applied, but by considering the stride having value S = 2, which skips two pixels.

692	-315	-6
-680	-194	305
153	-59	-86

692	-6
153	-86

Figure 4.12: The convolution's output with a 22 stride

Thus, by adjusting the kernel's stride, convolution layers can be utilised to minimise input volumes' spatialdimensions. The two main techniques for reducing the amount of the spatial input are convolutional layers and pooling layers.

Zero Padding

For internal filters, the same is true for a CNN when adopting a convolution; it is necessary to "pad" the image boundaries to maintain the real image size. Output volume size can be made to match the input volume size by using the consept of zero-padding to "pad" the input around given borders of image. The parameter P controls how much padding should be used. This method is very important at the beginning of deep convolutional neural network systems that layer many convolution

filters. Referring once more to Table 1, where it applied a 3 X 3 the Laplacian kernel to a 5 X 5 of given picture along a stride of size S = 1, that is see zero-padding.

To demonstrate how the properties of the convolution operation cause the output volume (33) to be lower than the input volume (55), see 4.13 (left). Alternatively, if value of P is 1 then it can pad the initial volume with zeros to create a desired volume, then use the convolution technique to produce an output volume that is the same size as the 55 original input volume.

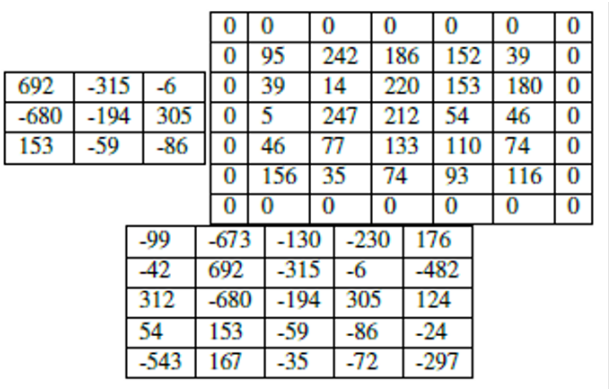

Figure 4.13: Zero Padding

Without the zero padding, the input volume's spatial parameters would shrink too quickly, making it impossible to train deep networks.

4.1.4 Activation Function

For each convolution layer in CNN, a nonlinear activation function is applied. The RELU activation is used the most or data scientists or machine learning engineers, activation layers are frequently abbreviated as RELU in network diagrams. To demonstrate that the network architecture incorporates an activation function.

Since it is expected that an activation comes right after a convolution, activation layers are commonly left off of network architecture diagrams even though they aren't technically "layers".

The specified activation function is applied when an activation layer receives an initial volume of size $W_{input} \times H_{input} \times D_{input}$ 4.14. The activation function is processed element-by-element, $W_{input} = W_{output}, H_{input} = H_{output}$, and $D_{output} = D_{input}$, an activation layer's output is always equal to its input dimension.

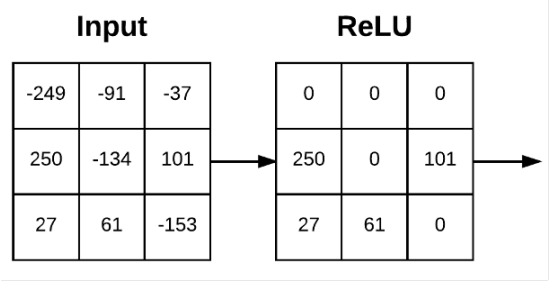

Figure 4.14: RELU Activation

Deep neural networks or multi-layer neural networks both employ the non-linear activation function known as ReLu. That function can be represented as:

$$f(x) = max(0, x)$$

where $x =$ an input value RELY Activation Function gives the highest value between zero and the given initial value depending upon above equation. If the input value is non-zero, the output value will be similar to either value. Thus, we can rewrite equation 1 as follows:

$$f(x) = \begin{cases} 0, & \text{if} x < 0 \\ x, & \text{if } x \geq 0 \end{cases}$$

Other Activation Functions

The activation function of neural networks is a node that is positioned at their core or in the middle. They influence whether or not the neuron would fire.

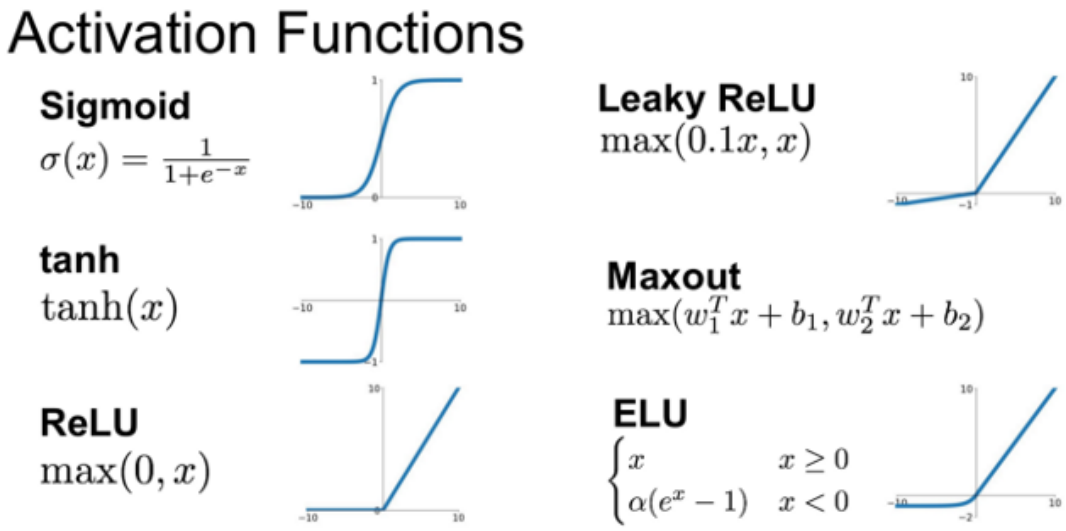

Figure 4.15: Other Activation Functions

4.1.5 Pooling Layers

For Compressing the size of image convolution neural network uses Pooling layers or Convolution layers with a stride's minimum value 1 Pooling layers are typically added between neighbouring Convolution layers in Convolution Neural Network structure:

INPUT => CONV => RELU => POOL => CONV => RELU => POOL => FC

The basic function of the Pooling Layer layer is to gradually reduce the dimensions of the input volume. We can reduce the network's compute and parameter requirements in this way, and pooling helps to manage overfitting.

Pooling layers perform independent operations on each of the input's depth slices using the maximum or average/mean function. While average pooling is frequently used as the network's final layer, where we want to completely avoid using Fully Connected layers,generally pooling done in center of the Convolution Neural Network design to reduce spatial size. Most of the times Maximum pooling layer is used but as more unusual micro-architectures are developed, this tendency is changing.

Although earlier designs of deeper CNNs with larger given images may process with a 3 X 3 pool size, the most typical pool size is 2 X 2. Additionally, S = 1 or S = 2 have been used frequently as the stride settings. With a pooling window size of 2 X 2 and a stride of S = 1, 4.16 demonstrates how to use maximum pooling. Take one step (like sliding a window) and repeat the process until you have a 3 X 3 output volume size. Note that only the greatest value is kept for each 2 X 2 block.

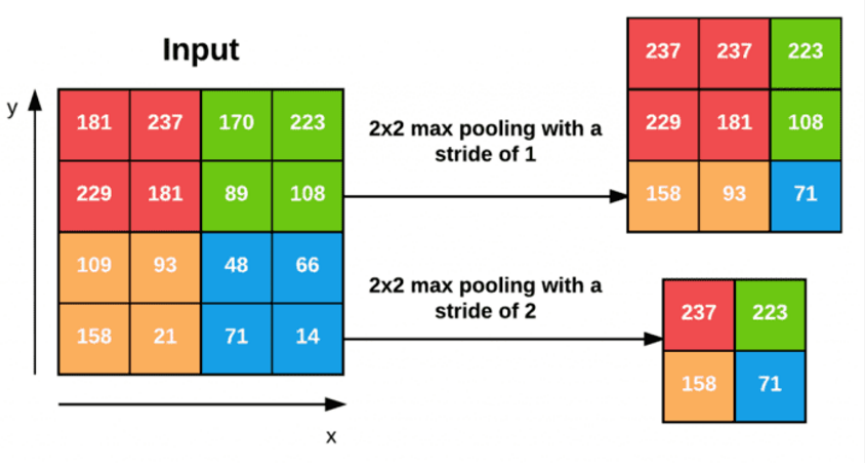

Figure 4.16: Convex - Non-Convex

By extending the stride and using S = 2 to the same input, the output volume can be further reduced.(4.16, bottom). We only save the greatest value for each 22-block input, advance two pixels, and then repeat the procedure. We effectively remove 75% of the activations from the previous layer by reducing the breadth and height by a factor of two.

4.1.6 Fully Connected Layers

As is customary for feedforward neural networks, neurons in completely connected layers are fully linked to all activations in the layer above. Apply Fully Connected layers at the final destinaiton the end of the selected network; never apply a Convolution layer, a Fully Connected layer, and then another Convolution layer. Before using the softmax classifier, one or two FC layers are frequently used, as shown by the following design: INPUT => $textCONV$ => $textRELU$ => $textPOOL$ => $textCONV$ => $textRELU$ => $textPOOL$ => $textFC$ => $textFC$ Before the softmax classifier,

which will decide our final output probabilities for each class, two fully connected layers are used in this instance.

4.2 UNet Architecture

The first time UNet, which developed from the traditional convolutional neural network, was used to analyze images for biomedical applications was in 2015. With an input of an image and an output of a single label, a typical convolutional neural network focuses on classifying images. But in biomedical applications, it's crucial to pinpoint both the existence of a disease and the precise location of the abnormality.

UNet is committed to resolving this problem. Due to the classification of every pixel, which guarantees that the input and output are the same size, it can localize and identify borders. The following is an example for a 2x2 input image: [[255, 230], [128, 12]] The numbers represents pixel. the outcum will be exactly 2x2 in size. [[1, 0], [1, 1]] Where could be any number between [0, 1]

4.2.1 Overview

The model's second half features an increase in dimension, or the pooling layer, is well known to represent the application of the convolutional neural network's whole height and breadth by the dimension reduction method. By maintaining the same number of input matrix channels, the pooling reduces the height and breadth of the information.The calculation is a method used to make things simpler. Pooling is a pixel that demonstrates a collection of pixels, to sum up. These layers are designed to boost the output's resolution.The high-resolution features of the model are combined with the sampled output for localization. This data will be used by a sequential convolution layer to create a more accurate output.

The network's fundamental structure is as 4.17:

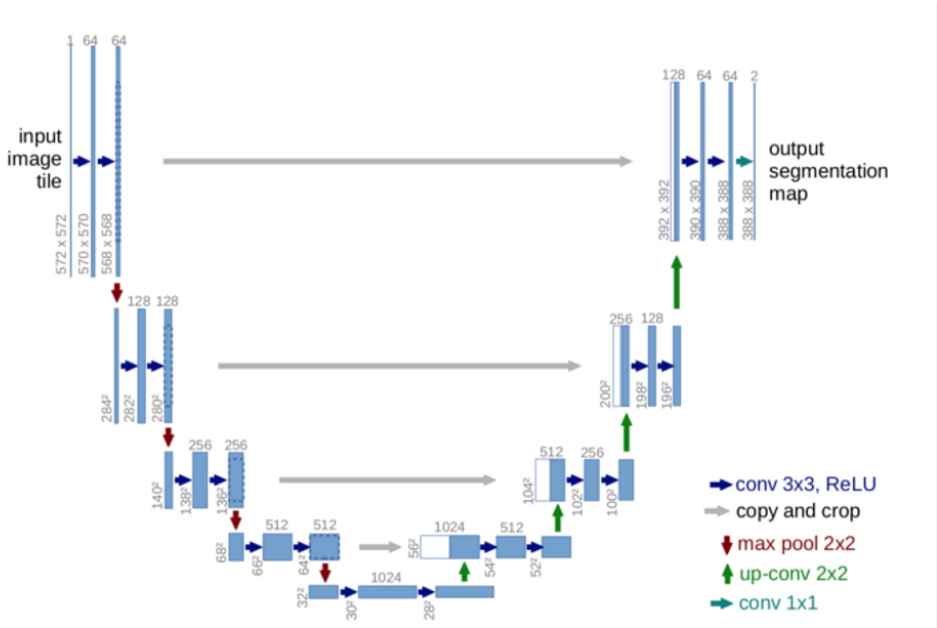

Figure 4.17: UNet

It resembles shape "U." The symmetrical design is separated into two major sections:

- the contracting route, the fundamental convolutional process in left
- the expanding path, transposed 2D convolutional layers in right

```
def build_model(input_layer, start_neurons):
      conv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu",
    padding="same")(input_layer)
     conv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu",
    padding="same")(conv1)
     pool1 = MaxPooling2D((2, 2))(conv1)
     pool1 = Dropout(0.25)(pool1)

```

```
    conv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu",
padding="same")(pool1)
    conv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu",
padding="same")(conv2)
    pool2 = MaxPooling2D((2, 2))(conv2)
    pool2 = Dropout(0.5)(pool2)

    conv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu",
padding="same")(pool2)
    conv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu",
padding="same")(conv3)
    pool3 = MaxPooling2D((2, 2))(conv3)
    pool3 = Dropout(0.5)(pool3)

    conv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu",
padding="same")(pool3)
    conv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu",
padding="same")(conv4)
    pool4 = MaxPooling2D((2, 2))(conv4)
    pool4 = Dropout(0.5)(pool4)

    # Middle
    convm = Conv2D(start_neurons * 16, (3, 3), activation="relu",
padding="same")(pool4)
    convm = Conv2D(start_neurons * 16, (3, 3), activation="relu",
padding="same")(convm)

```

```
26     deconv4 = Conv2DTranspose(start_neurons * 8, (3, 3), strides=(2,
       2), padding="same")(convm)
27     uconv4 = concatenate([deconv4, conv4])
28     uconv4 = Dropout(0.5)(uconv4)
29     uconv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu",
       padding="same")(uconv4)
30     uconv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu",
       padding="same")(uconv4)
31
32     deconv3 = Conv2DTranspose(start_neurons * 4, (3, 3), strides=(2,
       2), padding="same")(uconv4)
33     uconv3 = concatenate([deconv3, conv3])
34     uconv3 = Dropout(0.5)(uconv3)
35     uconv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu",
       padding="same")(uconv3)
36     uconv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu",
       padding="same")(uconv3)
37
38     deconv2 = Conv2DTranspose(start_neurons * 2, (3, 3), strides=(2,
       2), padding="same")(uconv3)
39     uconv2 = concatenate([deconv2, conv2])
40     uconv2 = Dropout(0.5)(uconv2)
41     uconv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu",
       padding="same")(uconv2)
42     uconv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu",
       padding="same")(uconv2)
43
```

```
    deconv1 = Conv2DTranspose(start_neurons * 1, (3, 3), strides=(2, 2), padding="same")(uconv2)
    uconv1 = concatenate([deconv1, conv1])
    uconv1 = Dropout(0.5)(uconv1)
    uconv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(uconv1)
    uconv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(uconv1)

    output_layer = Conv2D(1, (1,1), padding="same", activation="sigmoid")(uconv1)

    return output_layer

input_layer = Input((img_size_target, img_size_target, 1))
output_layer = build_model(input_layer, 16)


```

Contracting Path

The contracting process is as follows:

$$\text{conv˙layer1} -> \text{conv˙layer2} -> \text{max˙pooling} -> \text{dropout(optional)}$$

The first element of our code is thus:

```
    conv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(input_layer)
```

```
    conv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu",
padding="same")(conv1)
    pool1 = MaxPooling2D((2, 2))(conv1)
    pool1 = Dropout(0.25)(pool1)

```

It means that:

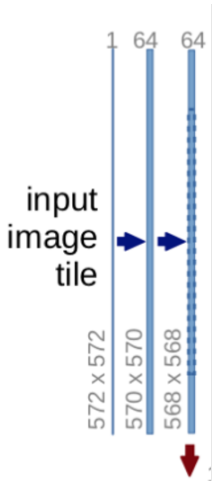

Figure 4.18: Contracting path UNet

The number of channels rises from 1 to 64 as the depth of the picture increases, and it is seen that each step in the convolution process consists of two convolutional layers. The max pooling process, which divides the picture in half, is indicated by the red arrow going downward.

The following steps will occur three more times:

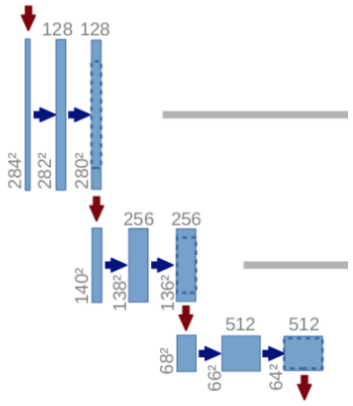

Figure 4.19: Contracting path UNet Process

After this bottommost:

Figure 4.20: Convex - Non-Convex

still two convolutional layers are constructed, but no maximum pooling is used:

```
1    convm = Conv2D(start_neurons * 16, (3, 3), activation="relu",
   padding="same")(pool4)
2    convm = Conv2D(start_neurons * 16, (3, 3), activation="relu",
   padding="same")(convm)
3
```

The image is currently 28x28x1024, and has been resized.

Expensive Path

The extended method will increase the image to its full size. This is the equation:

```
1    conv_2d_transpose -> concatenate -> conv_layer1 -> conv_layer2
2
```

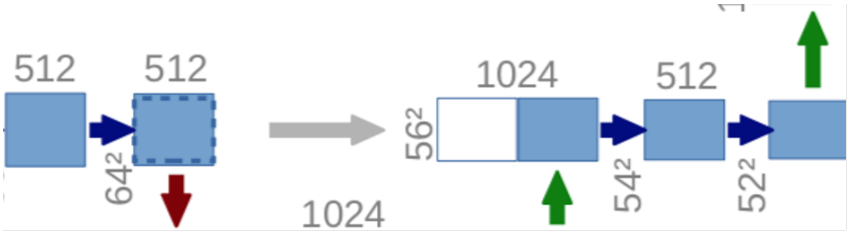

Figure 4.21: Expensive Path

```
1    deconv4 = Conv2DTranspose(start_neurons * 8, (3, 3), strides
    =(2, 2), padding="same")(convm)
2    uconv4 = concatenate([deconv4, conv4])
3    uconv4 = Dropout(0.5)(uconv4)
4    uconv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu",
    padding="same")(uconv4)
5    uconv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu",
    padding="same")(uconv4)
6
```

Transposed convolution is an upsampling method that enlarges images. The original image is essentially given some padding before a convolution is done. The transposed convolution increases the input image's size from $28x28x1024$ to $56x56x512$. The resulting image has the dimensions 56x56x1024, after being connected with the analogous image from the contracting path. In order to create a more precise prediction, the data from the earlier layers is combined here. Lines 4 and 5 add two additional convolution layers. This procedure is repeated three more times. The final step after reaching the peak of the architecture is to adjust the output image to match our forecast methods.

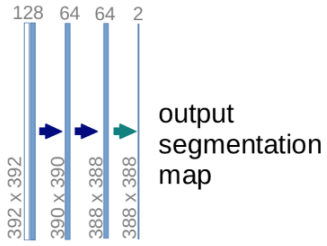

Figure 4.22: Segmentation map

```
output_layer = Conv2D(1, (1,1), padding="same", activation="
sigmoid")(uconv1)

```

The final layer is a convolution layer with a single 1x1 filter. For neural network training, the remaining information is the same. Image localization is accomplished by UNet by predicting the image pixel by pixel. By utilizing a variety of data augmentation techniques, the network is capable of producing precise predictions based on even small amounts of data. There are many applications for using UNet for image segmentation, and it is frequently employed in competitions.

4.2.2 Training Model

Defining the loss function is a crucial step in the model construction process. At each stage, the error between the predicted and actual outputs is measured using the loss function. The behaviour and convergence of the model are strongly influenced by the loss function. The dice˙p˙bce function, together with binary cross entropy, was employed in this instance.

The fit generator() method is used to perform model fitting. This function's ability to use callbacks to save model progress is a useful feature. If the validation

score for that epoch improved, the checkpoint callback will save the model's weights after each epoch. This was extremely useful because I could still utilise the model or keep training even though it crashed several times while being processed. Loss and validation score information will be saved by the csv logger callback. When a certain number of epochs have passed and the validation score has not increased, the earlystopping callback will automatically terminate the fitting process (the patience parameter).

The model trained on balanced sample of 10000 image after applying data augmentation. For training model Batch size of 1 and steps per epoch 100 due to computational limitations. Each epoch has taken more than 1 hour. Only 5 Epoch approached in this project.

4.3 How requirements are collected?

- For hardware/software requirements in the begining google colab and personal laptop with anaconda is used.
- For training the model data set downloaded from kaggle.
- For web application Flask is used.
- For Deployment AWS used.

CHAPTER 5
Result and Discussion

5.1 Features and Description

The for feature of this project is that it will detect the shipping vessels from satellite imaginary only. User only need to upload satellite image and our AI Application will detect the ship from that image. The detect images will be highlighted in the same images.

5.2 Results

It is challenging to find ships using optical satellite photography. Examples of problems and sources of false positives include clouds, landmasses, artificial objects, and highly reflective objects. The use of a deeper, bottleneck-implemented bespoke U-Net model in this study to get additional parameters shows an advancement over the previous related research work. By the implementation of U-Net architecture which achieves binary˙accuracy 0.9972, which is higher than the class accuracy of related works. Only 5 epoch completed due to computational limitation but further we can train up to 100 epoch.

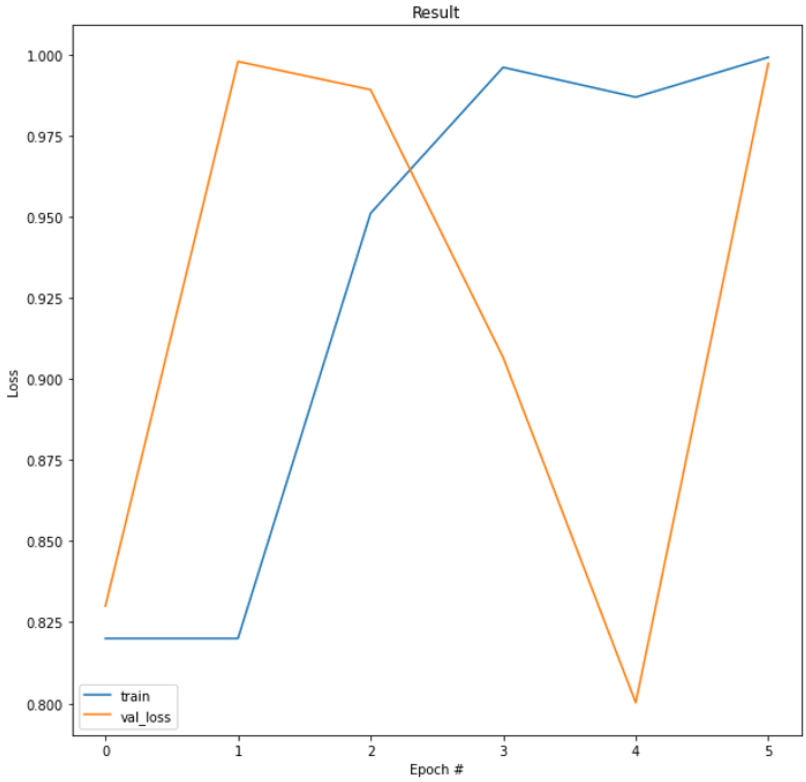

Figure 5.1: Final Result

5.3 Tools Techniques for URL

For GUI web application developed using FLASK and deployed using AWS. Below is the screenshot of the same:

Ship Detection

DETECTION OF SHIPPING VESSELS FROM SATELLITE IMAGERY USING DEEP LEARNING

Upload the image for Ship Detection

Browse... No file selected. Predict

Figure 5.2: User Interface

5.3.1 UML Diagram

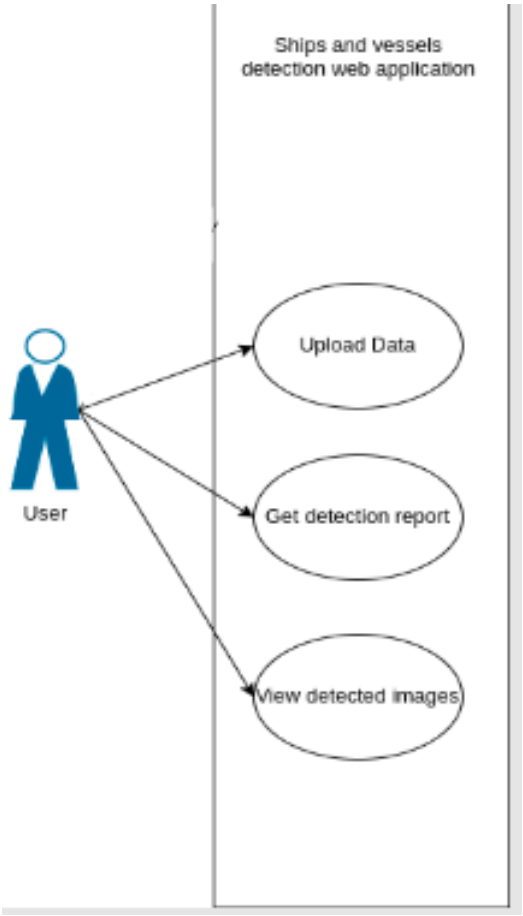

Figure 5.3: UML Diagram

5.3.2 Sequence Diagram

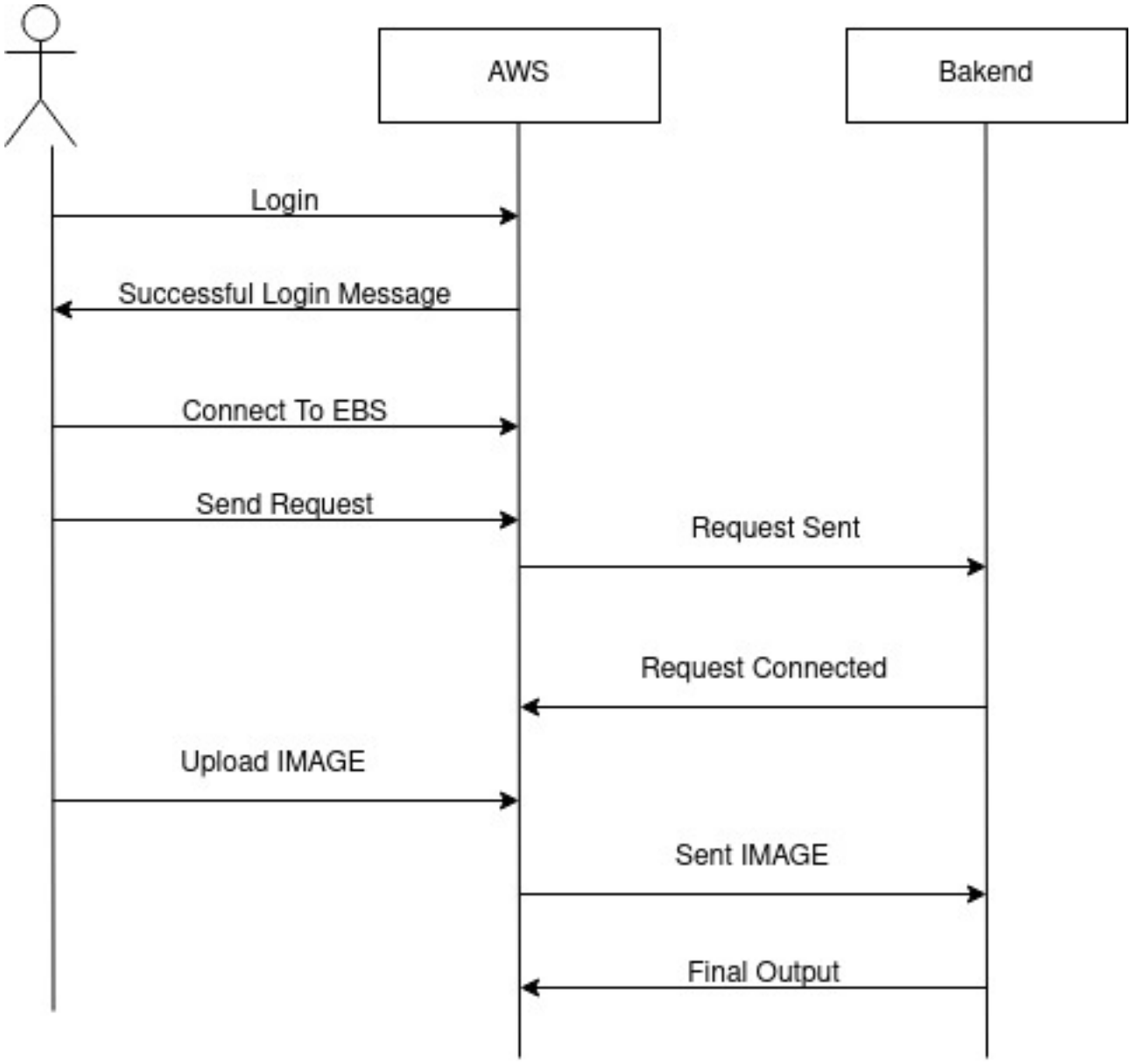

Figure 5.4: Sequence Diagram

5.3.3 Class Diagram

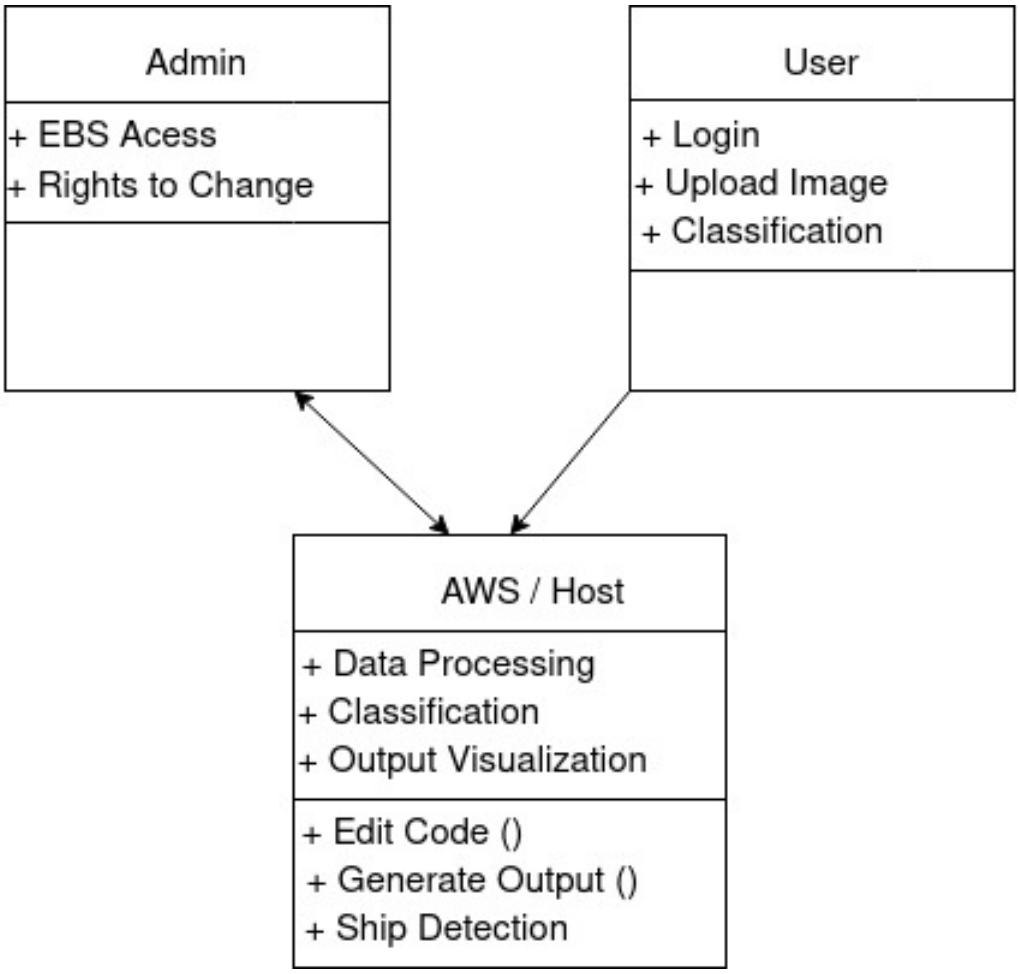

Figure 5.5: Class Diagram

CHAPTER 6
Conclusions and Future Work

6.1 Conclusion

There are certain patterns when one can look more closely at the integrated methods for marine vessel recognition and classification using optical satellite data provided. Traditionally, after beginning with the simple method of bright pixel thresholding against a busy the intricacy of the background was first raised by factoring in texture, shape, and segmentation as well as by using image transforms to match the image content to the expected shapes. Utilizing computer vision techniques has recently led to a rise in complexity, where the analysis may include numerous phases and is no longer at all simple. Deep learning, which includes training a multi-layer neural network with loads of data, is the most recent invention. This eliminates the need to define, which in some ways makes things easier once again.

6.2 Future Work

A continuation of the study's work can go in a number of different paths. The generalizability of the model can be enhanced by using a more recent dataset that includes a wider range of ship types operating in a wider range of weather conditions and in locations with a greater variation of topography. Additionally, a number of object detection models, including YOLO, EfficientDet might be utilised along with the CBAM to get more insight to explore the consequences of attention based modules on performance. Future directions for the study could include using transfer-learning to make use of previously trained models and using evolutionary deep

learning approaches to identify the optimal parameter settings for the greatest model performance.

References

Gallego, A. J., Pertusa, A., and Gil, P. (2018). Automatic ship classification from optical aerial images with convolutional neural networks. *Remote. Sens.*, 10:511.

Kanjir, U., Greidanus, H., and Oštir, K. (2018). Vessel detection and classification from spaceborne optical images: A literature survey. *Remote Sensing of Environment*, 207:1–26.

Lee, S.-J., Roh, M.-I., and Oh, M.-j. (2020). Image-based ship detection using deep learning. *Ocean Systems Engineering*, 10:415–434.

Liu, G., Zhang, Y., Zheng, X., Sun, X., Fu, K., and Wang, H. (2014). A new method on inshore ship detection in high-resolution satellite images using shape and context information. *IEEE Geoscience and Remote Sensing Letters*, 11(3):617–621.

Morillas, J. R. A., García, I. C., and Zölzer, U. (2015). Ship detection based on svm using color and texture features. In *2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 343–350.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation.

Swamidason, I. T. J., Sasikala, J., and Juliet, S. (2020). *Detection of Ship from Satellite Images Using Deep Convolutional Neural Networks with Improved Median Filter*, pages 69–82.

Zhang, R., Yao, J., Zhang, K., Feng, C., and Zhang, J. (2016). S-cnn-based ship detection from high-resolution remote sensing images. volume XLI-B7, pages 423–430.

Zhu, C., Zhou, H., Wang, R., and Guo, J. (2010). A novel hierarchical method of ship detection from spaceborne optical image based on shape and texture features. *IEEE Transactions on Geoscience and Remote Sensing*, 48(9):3446–3456.