DEEP LEARNING

(To Identify Ships In A Satellite Image)

Summer Internship Report Submitted in partial fulfillment

of the requirement for undergraduate degree of

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

By

Kasarla Sahith Reddy

221710308025

Under the Guidance of

Assistant Professor

Department Of Computer Science and Engineering

GITAM School of Technology

GITAM (Deemed to be University)

Hyderabad-502329

July 2020

DECLARATION

I submit this industrial training work entitled **"To Identify Ships In A Satellite Image**" to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of "**Bachelor of Technology**" in "**Computer Science and Engineering**". I declare that it was carried out independently by me under the guidance of , Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD Name: Kasarla Sahith Reddy

Date: Student Roll No: 221710308025

GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-502329, India

Dated: 22-07-2020

CERTIFICATE

This is to certify that the Industrial Training Report entitled **"To Identify Ships In A Satellite Image"** is being submitted by Kasarla Sahith Reddy (221710308025) in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science And Engineering** at GITAM (Deemed to Be University), Hyderabad during the academic year 2019-20

It is faithful record work carried out by him at the **Computer Science And Engineering Department**, GITAM University Hyderabad Campus under my guidance and supervision.

Dr.S.Phani Kumar

Assistant Professor
Department of CSE

Professor and HOD
Department of CSE

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank respected **Dr.N.Siva Prasad**, Pro Vice Chancellor, GITAM Hyderabad and **Dr.N.Seetharamaiah**, Principal, GITAM Hyderabad.

I would like to thank respected **S. Phani Kumar,** Head of the Computer Science and Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a internship report. It helped me a lot to realize of what we study for.

I would like to thank the respected faculties **Mr.** who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Kasarla Sahith Reddy

221710308025

ABSTRACT

Deep learning has achieved great success in many fields, such as computer vision and natural language processing. Compared to traditional machine learning methods, deep learning has a strong learning ability and can make better use of datasets for feature extraction. Image recognition is one of the most important fields of image processing and computer vision which can be achieved by deep learning.

To Identifying Ships in Satellite Images is the main motive of this project. Ship identification on satellite imagery can be used for fisheries management, monitoring of smuggling activities, ship traffic services, and naval warfare. Real Dataset is collected from website. I have adapted the view point of looking at features of the dataset, for deep understanding of the problem. My primary objective of this case study was to to classify whether the satellite image is consisting of ship or not. I have chosen Convolutional Neural Network (CNN) method, which has the advantage of being able to extract features automatically and produce reliable prediction for ship identification. Scaling is used for data as it is used to normalize and reduces computational complexity Conclusion is made on the base of achieve as higher accuracy as possible and correctly classifying unknown satellite images of ships which are not present in training dataset.

Table of Contents

List of Figures

CHAPTER 1: MACHINE LEARNING

1.1 INTRODUCTION:

Machine Learning(ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

1.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and "more items to consider" and "get yourself a little something" on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today's data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that's in part due to increased sophistication of machine learning, which helps analyse those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques. The process flow depicted here represents how machine learning works

Figure 1.2.1: The Process Flow

1.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let's consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyse huge volumes of data

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

1.4 TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

1.4.1 Supervised Learning :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to "learn" how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

1.4.2 Unsupervised Learning:

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

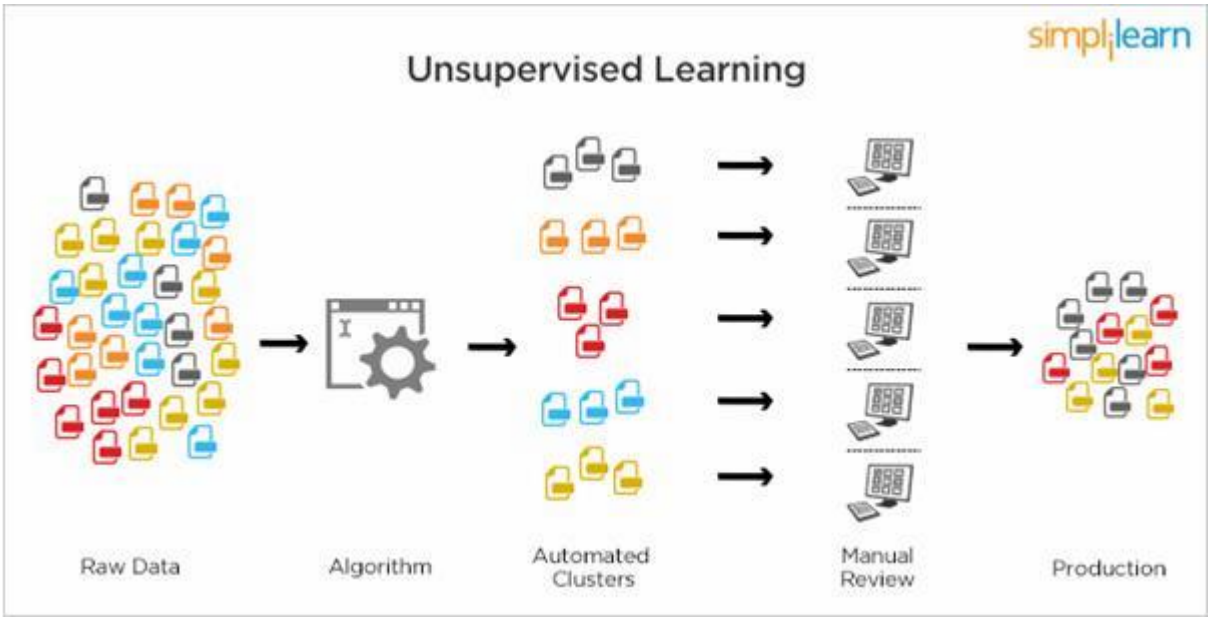

Figure 1.4.2.1: Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbour mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

1.4.3 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labelled and unlabelled data for training. In a typical scenario, the algorithm would use a small amount of labelled data with a large amount of unlabelled data.

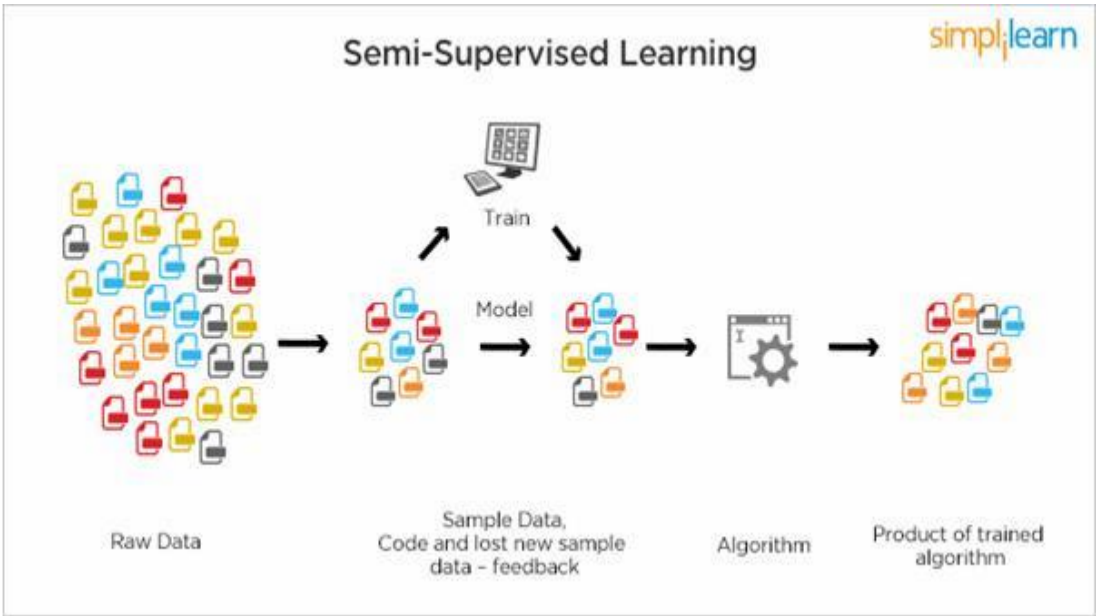

Figure 1.4.3.1: Semi Supervised Learning

CHAPTER 2: DEEP LEARNING

2.1 Deep Learning and It's Importance:

Deep learning algorithms run data through several "layers" of neural network algorithms, each of which passes a simplified representation of the data to the next layer. Most machine learning algorithms work well on datasets that have up to a few hundred features, or columns.

Basically deep learning is itself a subset of machine learning but in this case the machine learns in a way in which humans are supposed to learn. The structure of deep learning model is highly similar to a human brain with large number of neurons and nodes like neurons in human brain thus resulting in artificial neural network. In applying traditional machine learning algorithms we have to manually select input features from complex data set and then train them which becomes a very tedious job for ML scientist but in neural networks we don't have to manually select useful input features, there are various layers of neural networks for handling complexity of the data set and algorithm as well. In my recent project on human activity recognition , when we applied traditional machine learning algorithm like K-NN then we have to separately detect human and its activity also had to select impactful input parameters manually which became a very tedious task as data set was way too complex but the complexity dramatically reduced on applying artificial neural network, such is the power of deep learning. Yes it's correct that deep learning algorithms take lots of time for training sometimes even weeks as well but its execution on new data is so fast that its not even comparable with traditional ML algorithms. Deep learning has enabled Industrial Experts to overcome challenges which were impossible, a decades ago like Speech and Image recognition and Natural Language Processing. Majority of the Industries are currently depending on it , be it Journalism, Entertainment, Online Retail Store, Automobile, Banking and Finance, Healthcare, Manufacturing or even Digital Sector. Video recommendations, Mail Services, Self Driving cars, Intelligent Chat bots, Voice Assistants are just trending achievements of Deep Learning.

Furthermore, Deep learning can most profoundly be considered as future of Artificial Intelligence due to constant rapid increase in amount of data as well as the gradual development in hardware field as well, resulting in better computational power.

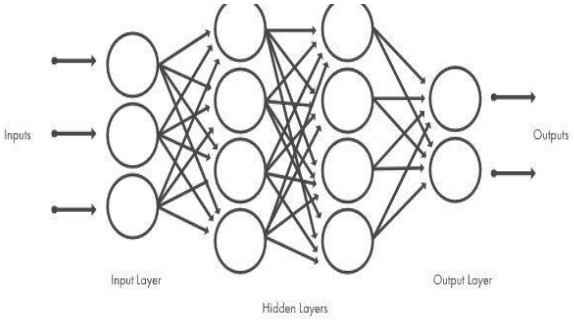

Fig 2.1.1: Deep Neural network

2.2 Uses of Deep Learning:

i. Translations:

Although automatic machine translation isn't new, deep learning is helping enhance automatic translation of text by using stacked networks of neural networks and allowing translations from images.

ii. Adding colour to black-and-white images and videos:

It is used to be a very time-consuming process where humans had to_add colour to black-and-white images and videos by hand can now be automatically done with deep-learning models.

iii. Language recognition:

Deep learning machines are beginning to differentiate dialects of a language. A machine decides that someone is speaking English and then engages an AI that is learning to tell the differences between dialects. Once the dialect is determined, another AI will step in that specializes in that particular dialect. All of this happens without involvement from a human.

iv. Autonomous vehicles:

There's not just one AI model at work as an autonomous vehicle drives down the street. Some deep-learning models specialize in streets signs while others are trained to recognize pedestrians. As a car navigates down the road, it can be informed by up to millions of individual AI models that allow the car to act.

v. Computer vision:

Deep learning has delivered super-human accuracy for image classification, object detection, image restoration and image segmentation—even handwritten digits can be recognized. Deep learning using enormous neural networks is teaching machines to automate the tasks performed by human visual systems.

vi. Text generation:

The machines learn the punctuation, grammar and style of a piece of text and can use the model it developed to_ automatically create entirely new text with the proper spelling, grammar and style of the example text. Everything from Shakespeare to Wikipedia entries have been created.

vii. Deep-learning robots:

Deep-learning applications for robots are plentiful and powerful from an impressive deep-learning system that can teach__a robot just by observing the actions of a human completing a task to a_ housekeeping robot that's provided with input from several other AIs in order to take action. Just like how a human brain processes input from past experiences, current input from senses and any additional data that is provided, deep-learning models will help robots execute tasks based on the input of many different AI opinions.

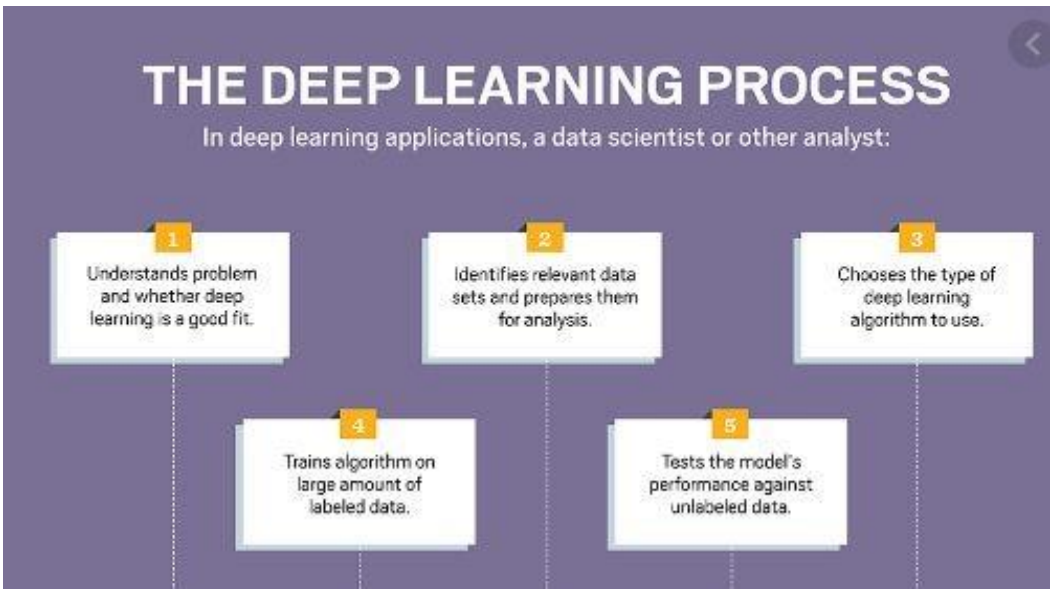

Fig 2.2.1: The Deep Learning Process

2.3 Relation between Data Mining, Machine Learning and Deep Learning:

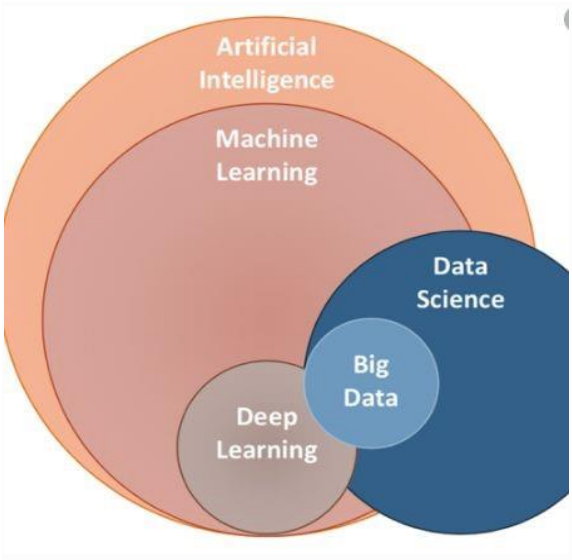

Fig 2.3.1: Relation between DM,ML,DL

The deep learning, data mining and machine learning share a foundation in data science, and there certainly is overlap between the two. Data mining can use machine learning algorithms to improve the accuracy and depth of analysis, and vice-versa; machine learning can use mined data as its foundation, refining the dataset to achieve better results.

You could also argue that data mining and machine learning are similar in that they both seek to address the question of how we can learn from data. However, the way in which they achieve this end, and their applications, form the basis of some significant differences.

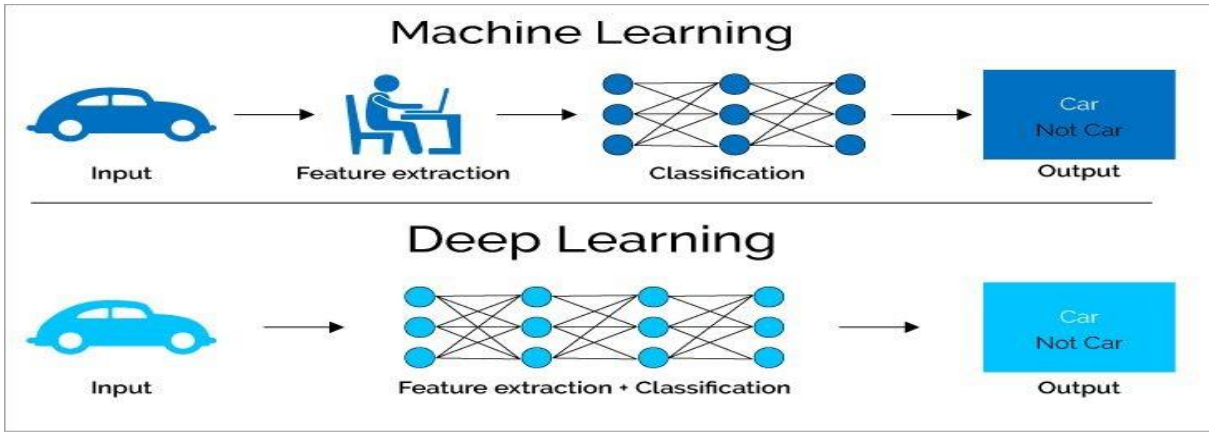

Fig 2.3.2: Process in machine learning and deep learning

Machine Learning comprises of the ability of the machine to learn from trained data set and predict the outcome automatically. It I a subset of artificial intelligence.

Deep Learning is a subset of machine learning. It works in the same way on the machine just like how the human brain processes information. Like a brain can identify the patterns by comparing it with previously memorized patterns, deep learning also uses this concept.

Deep learning can automatically find out the attributes from raw data while machine learning selects these features manually which further needs processing. It also employs artificial neural networks with many hidden layers, big data, and high computer resources.

Data Mining is a process of discovering hidden patterns and rules from the existing data. It uses relatively simple rules such as association, correlation rules for the decision-making process, etc. Deep Learning is used for complex problem processing such as voice recognition etc. It uses Artificial Neural Networks with many hidden layers for processing. At times data mining also uses deep learning algorithms for processing the data.

CHAPTER 3: PYTHON

Basic programming language used for machine learning is : PYTHON

3.1 Introduction to python:

- Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Python is a general purpose programming language that is often applied in scripting roles
- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

3.1.1 History of python:

- Python was developed by GUIDO VAN ROSSUM in early 1990's
- Its latest version is 3.7 , it is generally called as python3

3.2 How to setup python:

- Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.
- The most up-to-date and current source code, binaries, documentation, news, etc., is

available on the official website of Python.

3.2.1 Installation(using python IDLE):

- Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
- Download python from www.python.org

- When the download is completed, double click the file and follow the instructions to install it.
- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

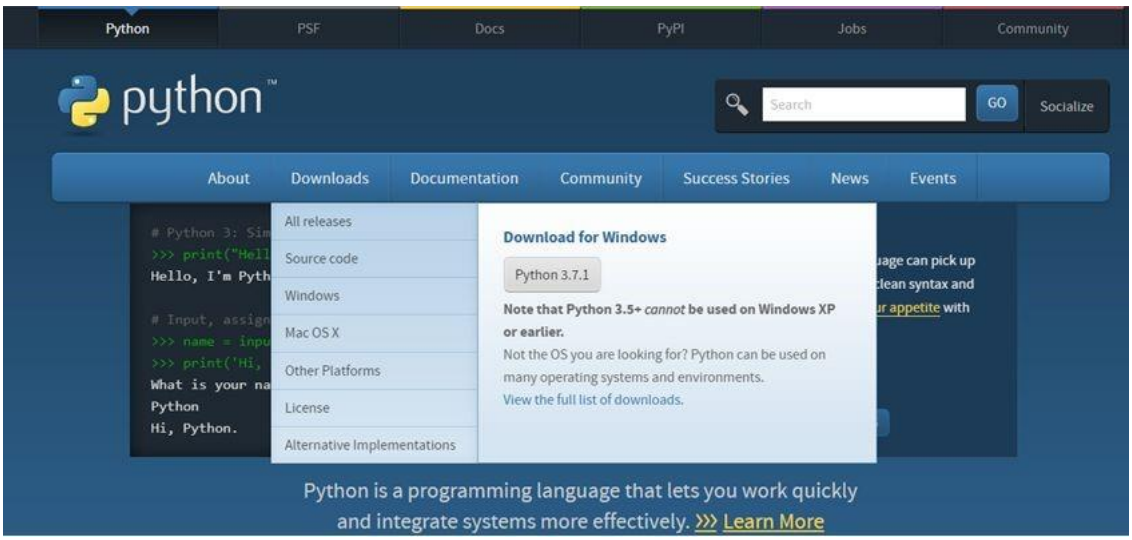

Figure 3.2.1.1: Python download

3.2.2 Installation(using Anaconda):

- Python programs are also executed using Anaconda.
- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
- Conda is a package manager quickly installs and manages packages.
- In WINDOWS:
- Step 1: Open Anaconda.com/downloads in web browser.
- Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
- Step 3: select installation type(all users)
- Step 4: Select path(i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish
- Step 5: Open jupyter notebook (it opens in default browser)

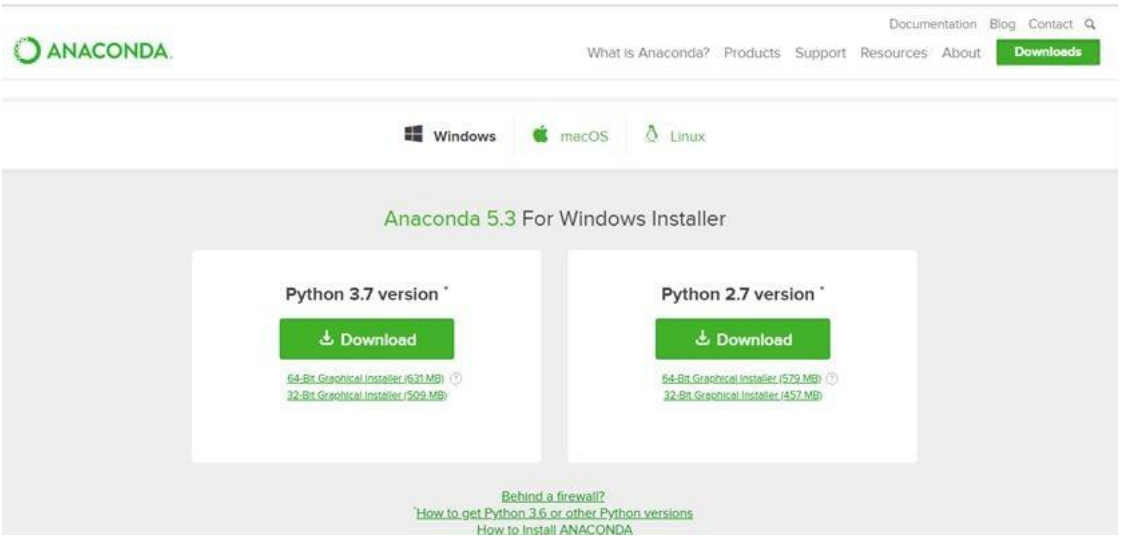

Figure 3.2.2.1: Anaconda download

Figure 3.2.2.2: Jupyter notebook

3.3 Features of python:

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax, This allows the student to pick up the language quickly.
- Easy-to-read: Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain: Python's source code is fairly easy-to-maintaining.
- A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

- Databases: Python provides interfaces to all major commercial databases.
- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

3.4 PYTHON VARIABLE TYPES:

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory. Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Python has five standard data types –

- Numbers
- Strings
- Lists
- Tuples
- Dictionary

3.4.1 Python Numbers:

- Number data types store numeric values. Number objects are created when you assign a value to them.
- Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

3.4.2 Python Strings:

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.

- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

3.4.3 Python Lists:

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square brackets ([]).
- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

3.4.4 Python Tuples:

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.
- Tuples can be thought of as read-only lists.
- For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

3.4.5 Python Dictionary:

- Python's dictionaries are kind of hash table type.
- They work like associative arrays or hashes found in Perl and consist of key-value pairs.

- A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).
- You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.
- What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

3.5 PYTHON FUNCTION:

3.5.1 Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

3.5.2 Calling a Function:

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

3.6 PYTHON USING OOP's CONCEPTS:

3.6.1 Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

- Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- Data member: A class variable or instance variable that holds data associated with a class and its objects.
- Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.
- Defining a Class:
- We define a class in a very similar way how we define a function.
- Just like a function ,we use parentheses and a colon after the class name(i.e. ():) when we define a class. Similarly, the body of our class is indented like a functions body is.

```
def my_function():
    # the details of the
    # function go here
```

```
class MyClass():
    # the details of the
    # class go here
```

Figure 3.6.1.1: Defining a Class

3.6.2 __init__ method in Class:

- The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.
- The init method has a special name that starts and ends with two underscores:__init__().

CHAPTER 4: TO IDENTIFY SHIPS IN A SATELLITE IMAGE

4.1Project Requirements:

4.1.1. Packages used:

Numpy: In Python we have lists that serve the purpose of arrays, but they are slow to process. Numpy aims to provide an array object that is up to 50x faster that traditional Python lists. The array object in Numpy is called ndata, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

Pandas: Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib: Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of Numpy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with Numpy can be considered as the open source equivalent of MATLAB.

TensorFlow: TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

OS: The OS module in python provides functions for interacting with the operating system. OS, comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality.

JSON: Python has a built-in package called json, which can be used to work with JSON data.

```
[ ]  import os
     import json
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import tensorflow as tf
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense,Conv2D,MaxPooling2D,Flatten
     from tensorflow.keras.preprocessing import image
```

Fig 4.1.1: Packages used

4.1.2. Versions of the packages:

The versions of the packages are found by following command

```
#Versions Of Packages
print("numpy",np.__version__)
print("pandas",pd.__version__)
print("matplotlib",matplotlib.__version__)
print("tensorflow",tf.__version__)
```

```
numpy 1.18.5
pandas 1.0.5
matplotlib 3.2.2
tensorflow 2.2.0
```

Fig 4.1.2.1: Versions of packages

4.1.3. Algorithms used:

Convolutional Neural Network (CNN): The CNN algorithm is efficient at recognition and highly adaptable. It's also easy to train because there are fewer training parameters, and is scalable when coupled with backpropagation.

4.2 Problem Statement:

The aim of this dataset is to help difficult task of detecting the ships in satellite images.

4.3 Dataset Description:

The dataset is also distributed as a JSON formatted text file shipsnet.json. The loaded object contains data, label, scene_ ids, and location lists.

Data: It contains image data in the form of array

labels: Valued 1 or 0, representing the “ship” class and “no-ship” class, respectively.

Scene id: The unique identifier of the PlanetScope visual scene the image chip was extracted from. The scene id can be used with the Planet API to discover and download the entire scene.

Longitude_latitude: The longitude and latitude coordinates of the image centre point, with values separated by a single underscore.

The pixel value data for each 80x80 RGB image is stored as a list of 19200 integers within the data list. The first 6400 entries contain the red channel values, the next 6400 the green, and the final 6400 the blue. The image is stored in row-major order, so that the first 80 entries of the array are the red channel values of the first row of the image.

The “ship” class includes 1000 images. Images in this class are near-centred on the body of a single ship. Ships of different sizes, orientations, and atmospheric collection conditions are included.

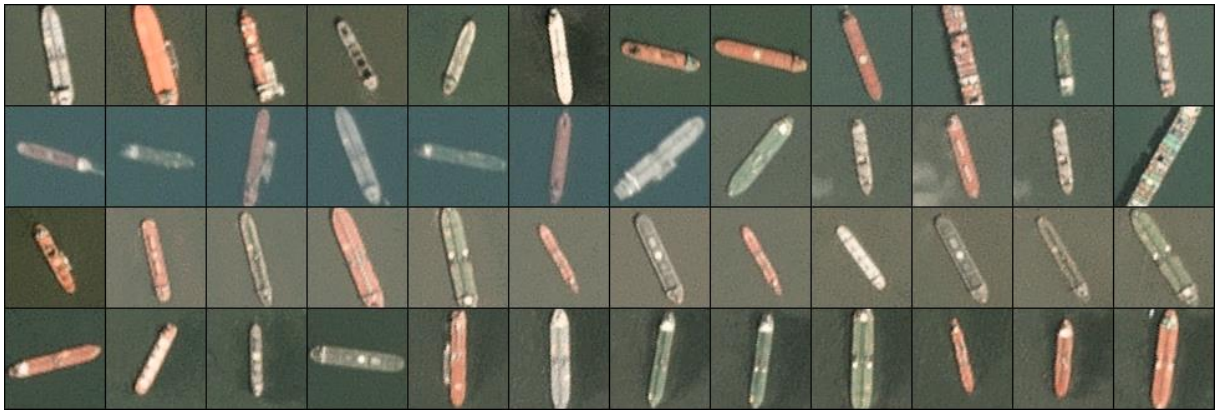

Fig 4.3.1: Sample ship images

The "no-ship" class includes 3000 images. A third of these are a random sampling of different land cover features - water, vegetation, bare earth, buildings, etc. - that do not include any portion of an ship. The next third are "partial ships" that contain only a portion of an ship, but not enough to meet the full definition of the "ship" class. The last third are images that have previously been mislabelled by machine learning models, typically caused by bright pixels or strong linear features.

Fig 4.3.2: Sample no-ship images

4.4. Objective of the Case Study:

Objective of the problem is to identify ships in satellite images. To classify whether ships are present or not present.

CHAPTER 5: DATA PREPROCESSING/FEATURE ENGINEERING AND EDA

5.1 Loading the data:

Python has a built-in open() function to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly.

We can specify the mode while opening a file. In mode, we specify whether we want to read , write or append to the file. We can also specify if we want to open the file in text mode or binary mode.

The default is reading in text mode. In this mode, we get strings when reading from the file. Python has a built-in package called json, which can be used to work with JSON data. Contents of json file are stored in dataset variable.

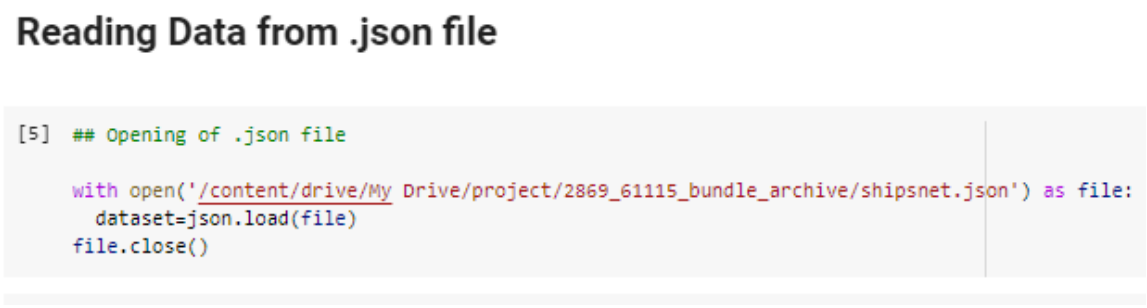

Fig 5.1.1: Reading data from json file

Pandas DataFrame to which all the operations can be performed which helps us to access each and every row as well as columns and each and every value can be access using the dataframe. Any missing value or NaN value have to be cleaned.

```
# Creating DataFrame for data in .json file

all_ships=pd.DataFrame(dataset)
all_ships.head()
```

	data	labels	locations	scene_ids
0	[82, 89, 91, 87, 89, 87, 86, 86, 86, 86, 84, 8...	1	[-118.2254694333423, 33.73803725920789]	20180708_180909_0f47
1	[76, 75, 67, 62, 68, 72, 73, 73, 68, 69, 69, 6...	1	[-122.33222866289329, 37.7491755586813]	20170705_180816_103e
2	[125, 127, 129, 130, 126, 125, 129, 133, 132, ...	1	[-118.14283073363218, 33.736016066914175]	20180712_211331_0f06
3	[102, 99, 113, 106, 96, 102, 105, 105, 103, 10...	1	[-122.34784341495181, 37.76648707436548]	20170609_180756_103a
4	[78, 76, 74, 78, 79, 79, 79, 82, 86, 85, 83, 8...	1	[-122.34852408322172, 37.75878462398653]	20170515_180653_1007

Fig 5.1.2: Loading data set

5.2 Statistical Analysis:

Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding nan values. Analyses both numeric and object series, as well as DataFrame column sets of mixed data types. The output will vary depending on what is provided.

For numeric data, the result's index will include count, mean, std, min, max as well as lower, 50 and upper percentiles. By default the lower percentile is 25 and the upper percentile is 75.The 50 percentile is the same as the median.

For object data (e.g. strings or timestamps), the result's index will include count, unique, top and freq. The top is the most common value. The freq is the most common value's frequency. Timestamps also include the first and last items.

If multiple object values have the highest count, then the count and top results will be arbitrarily chosen from among those with the highest count.

For mixed data types provided via a DataFrame , the default is to return only an analysis of numeric columns. If the dataframe consists only of object and categorical data without any numeric columns, the default is to return an analysis of both the object and categorical columns. If include='all' is provided as an option, the result will include a union of attributes of each type.

```
#Description of labels column
all_ships.describe()
```

	labels
count	4000.000000
mean	0.250000
std	0.433067
min	0.000000
25%	0.000000
50%	0.000000
75%	0.250000
max	1.000000

Fig 5.2.1: Statistical data of labels

Checking for total number of ships images pesent in dataset

```
[ ]  all_ships.labels.value_counts()
```

```
0    3000
1    1000
Name: labels, dtype: int64
```

Fig 5.2.2: Checking for total number of ship and non ship images

Observations:-

1. There are total 4000 images.
2. By observing the mean of the labels there are 1000 ship containing images and 3000 no ship images.

5.3 Visualisation of images

To know the image names from the main folder these following steps are done accordingly.

1. Checking for the contents in main data containing folder

```
## checking contents in the main folder folder

os.listdir("/content/drive/My Drive/project/2869_61115_bundle_archive")
```

```
['shipsnet.json', 'scenes', 'shipsnet']
```

Fig 5.3.1: Contents in file

2. Here base directory is created so that it makes joining the other paths easy.

Here ship_imgs contains all the images

```
## creating base directory,ships directory,ship images directory.

base_dir="/content/drive/My Drive/project/2869_61115_bundle_archive"
ships_dir=os.path.join(base_dir,'shipsnet')
ships_imgs=os.path.join(ships_dir,'shipsnet1')
```

Fig 5.3.2: Creating directories

3. Displaying 5 images names

```
# going through 5 image names in shipsnet1

imgships=os.listdir(ships_imgs)
imgships[:5]
```

```
['1__20170830_181003_0f4e__-122.35157982161317_37.78327205920199.png',
 '1__20170903_181304_1041__-122.3359293751746_37.7580383438423.png',
 '1__20170830_181004_0f4e__-122.33472265432373_37.759680938376434.png',
 '1__20170905_181214_0f12__-122.32764083883828_37.736846543182324.png',
 '1__20170901_181520_0e14__-122.35466805228293_37.75722310404933.png']
```

Fig 5.3.3: Displaying 5 images names

4. By, using one of the image name and displaying a sample image using matplotlib package.

```
#Displaying a sample image

plt.imshow(plt.imread(ships_imgs+'/1__20170830_181003_0f4e__-122.35157982161317_37.78327205920199.png'))
```

```
<matplotlib.image.AxesImage at 0x7fe8b473c080>
```

Fig 5.3.4: Displaying Sample image

5. Visualization of 15 images with title as the image shapes

```
#Visualisation of images

plt.figure(figsize=(12,12))
j=1
for i in range(1,3000,200):
  img=plt.imread(os.path.join(ships_imgs,imgships[i]))
  plt.subplot(3,5,j)
  plt.imshow(img)
  plt.title(img.shape)
  plt.axis('off')
  j+=1
```

Fig 5.3.5: Input for visualization of 15 images

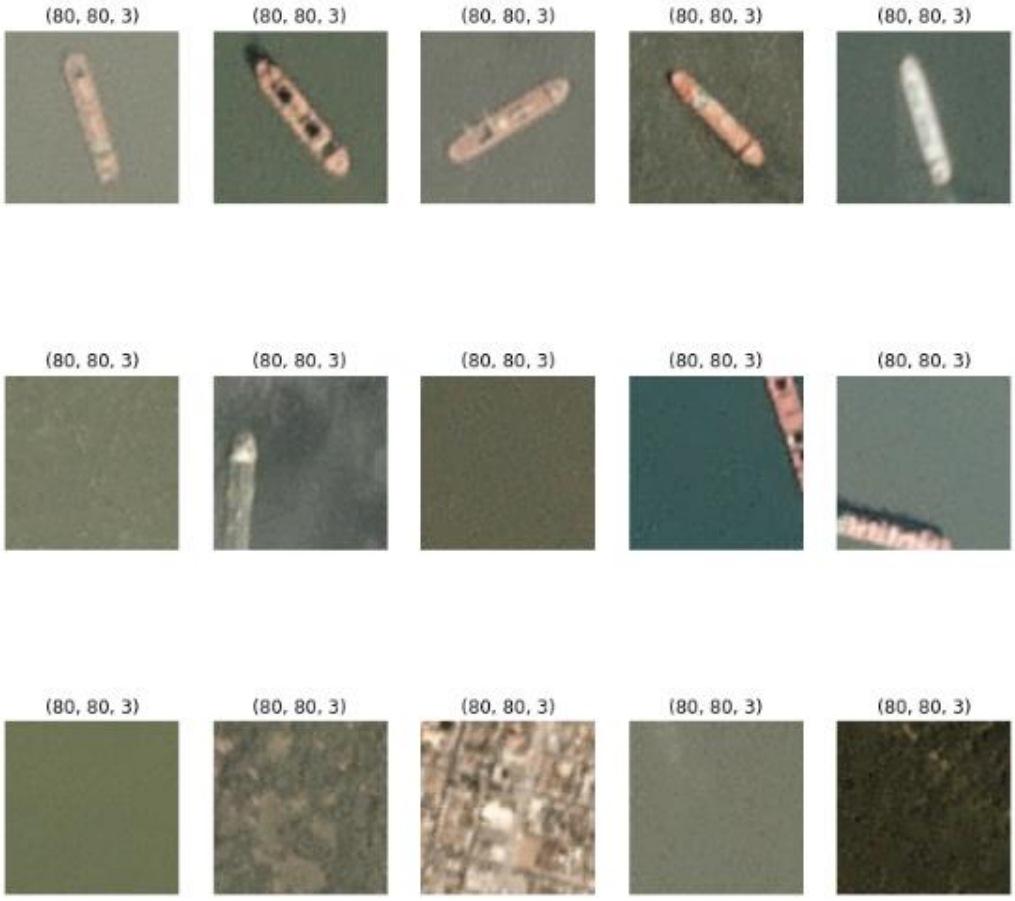

Fig 5.3.6: Output for visualization of 15 images

Observation: Images are of size 80 x 80 x 3.

5.4 Handling Missing Values:

There are a number of schemes that have been developed to indicate the presence of missing data in a table or DataFrame. Generally, they revolve around one of two strategies: using a mask that globally indicates missing values, or choosing a sentinel value that indicates a missing entry

```
# checking missing values in all_ships dataset columns
all_ships.isnull().sum()
```

```
data         0
labels       0
locations    0
scene_ids    0
dtype: int64
```

Fig 5.4.1: No Missing Values in the data set

CHAPTER 6: FEATURE SELECTION

6.1 Select relevant features for the analysis:

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features. In Classification of images input should be of numpy array or image.

```
# Creating DataFrame for data in .json file

all_ships=pd.DataFrame(dataset)
all_ships.head()
```

	data	labels	locations	scene_ids
0	[82, 89, 91, 87, 89, 87, 86, 86, 86, 86, 84, 8...	1	[-118.2254694333423, 33.73803725920789]	20180708_180909_0f47
1	[76, 75, 67, 62, 68, 72, 73, 73, 68, 69, 69, 6...	1	[-122.33222866289329, 37.7491755586813]	20170705_180816_103e
2	[125, 127, 129, 130, 126, 125, 129, 133, 132, ...	1	[-118.14283073363218, 33.736016066914175]	20180712_211331_0f06
3	[102, 99, 113, 106, 96, 102, 105, 105, 103, 10...	1	[-122.34784341495181, 37.76648707436548]	20170609_180756_103a
4	[78, 76, 74, 78, 79, 79, 79, 82, 86, 85, 83, 8...	1	[-122.34852408322172, 37.75878462398653]	20170515_180653_1007

Fig 5.1.2: Loading data set

Here data column can be considered as input and labels column can be considered as output. Remaining columns are irrelevant.

6.2 Drop irrelevant features:

In the dataset locations, scene_ids columns are irrelevant so these columns are to be dropped.

```
# Dropping irrelevent columns
all_ships=all_ships.drop(['locations','scene_ids'],axis=1)
all_ships
```

	data	labels
0	[82, 89, 91, 87, 89, 87, 86, 86, 86, 86, 84, 8...	1
1	[76, 75, 67, 62, 68, 72, 73, 73, 68, 69, 69, 6...	1
2	[125, 127, 129, 130, 126, 125, 129, 133, 132, ...	1
3	[102, 99, 113, 106, 96, 102, 105, 105, 103, 10...	1
4	[78, 76, 74, 78, 79, 79, 79, 82, 86, 85, 83, 8...	1
...	...	...
3995	[126, 122, 124, 138, 165, 186, 195, 199, 203, ...	0
3996	[130, 134, 139, 128, 117, 126, 141, 147, 142, ...	0
3997	[171, 135, 118, 140, 145, 144, 154, 165, 139, ...	0
3998	[85, 90, 94, 95, 94, 92, 93, 96, 93, 94, 94, 9...	0
3999	[122, 122, 126, 126, 142, 153, 174, 190, 185, ...	0

4000 rows × 2 columns

Fig 6.2.1: Dropping columns

6.3 Converting Data and reshaping:

In CNN image is defined in the form of array based on that it will classify the images. So, as the project data is not directly involving the images so data should be in the array format. As the data is in not array format it should be converted to array format of building the model.

Extracting data and labels columns from dataset and converting it into numpy array format

```
[16] items = np.array(dataset['data']).astype('int64')
     output_labels = np.array(dataset['labels']).astype('int64')

     #checking datatypes of items and output_labels

     print("items",type(items))
     print("output_labels",type(output_labels))
```

```
items <class 'numpy.ndarray'>
output_labels <class 'numpy.ndarray'>
```

Fig 6.3.1: Conversion of data into numpy array

Here, the data column which is in the numpy array format is named as items and labels column which is in the numpy array format is named as output_labels.

When doing image classification, it is important to make sure you are using the correct size of images; otherwise you may get unexpected results or errors.

```
# Checking shape of items

items.shape
```

```
(4000, 19200)
```

Fig 6.3.2: Checking shape of data

Here, the shape of the items data is not in correct size format so reshaping should be done for correct format to proceed further.

Reshaping the items into 80 x 80 x 3 Format

```
[ ]  items=items.reshape([-1,3,80,80]).transpose([0,2,3,1])
     items.shape
```

```
(4000, 80, 80, 3)
```

Fig 6.3.3: Reshaping of data

Now, data is of 80 x 80 x 3 size.

6.4 Feature Scaling:

It is a step of Data Pre Processing which is applied to independent variables or features of data. It basically helps to normalise the data within a particular range. Sometimes, it also helps in speeding up the calculations in an algorithm. Real world dataset contains features that highly vary in magnitudes, units, and range. Normalisation should be performed when the scale of a feature is irrelevant or misleading and not should Normalise when the scale is meaningful.

Scaling for items column

```
[17] items=items/ 255
```

Fig 6.4.1: Scaling of data

Scaling is done only for items column where image data is represented in numpy array format. The items column is divided by 255 because every digital image is formed by pixel having value in range 0~255. 0 is black and 255 is white. For colourful image, it contains three maps: Red, Green and Blue, and all the pixel still in the range 0~255. Since 255 is the maximin pixel value. Rescale 1./255 is to transform every pixel value from range [0,255] -> [0,1].which helps treating all images in the same manner.

CHAPTER 7: MODEL BUILDING AND EVALUATION:

7.1 Brief about the algorithms used:

Convolutional Neural Networks:

A breakthrough in building models for image classification came with the discovery that a convolutional neural network (CNN) could be used to progressively extract higher- and higher-level representations of the image content. Instead of pre-processing the data to derive features like textures and shapes, a CNN takes just the image's raw pixel data as input and "learns" how to extract these features, and ultimately infer what object they constitute. CNN receives an input feature map: a three-dimensional matrix where the size of the first two dimensions corresponds to the length and width of the images in pixels. The size of the third dimension is 3 (corresponding to the 3 channels of a colour image: red, green, and blue). The CNN comprises a stack of modules, each of which performs three operations.

1. Convolution:

A convolution extracts tiles of the input feature map, and applies filters to them to compute new features, producing an output feature map, or convolved feature (which may have a different size and depth than the input feature map).Convolutions are defined by two parameters:

- Size of the tiles that are extracted (typically 3x3 or 5x5 pixels).
- The depth of the output feature map, which corresponds to the number of filters that are applied.

During a convolution, the filters (matrices the same size as the tile size) effectively slide over the input feature map's grid horizontally and vertically, one pixel at a time, extracting each corresponding tile.

For each filter-tile pair, the CNN performs element-wise multiplication of the filter matrix and the tile matrix, and then sums all the elements of the resulting matrix to get a single value. Each of these resulting values for every filter-tile pair is then output in the convolved feature matrix.

During training, the CNN "learns" the optimal values for the filter matrices that enable it to extract meaningful features (textures, edges, shapes) from the input feature map. As the number of filters (output feature map depth) applied to the input increases, so does the number of features the CNN can extract. However, the trade-off is that filters compose the majority of resources expended by the CNN, so training time also increases as more filters are added. Additionally, each filter added to the network provides less incremental value than the previous one, so engineers aim to construct networks that use the minimum number of filters needed to extract the features necessary for accurate image classification.

2. ReLU

Following each convolution operation, the CNN applies a Rectified Linear Unit (ReLU) transformation to the convolved feature, in order to introduce nonlinearity into the model. The ReLU function,

$F(x)=\max(0,x)$ returns x for all values of $x > 0$, and returns 0 for all values of $x \leq 0$.

3. Pooling

After ReLU comes a pooling step, in which the CNN down samples the convolved feature (to save on processing time), reducing the number of dimensions of the feature map, while still preserving the most critical feature information. A common algorithm used for this process is called max pooling.

Max pooling operates in a similar fashion to convolution. We slide over the feature map and extract tiles of a specified size. For each tile, the maximum value is output to a new feature map, and all other values are discarded. Max pooling operations take two parameters:

Size of the max-pooling filter (typically 2x2 pixels)

Stride: the distance, in pixels, separating each extracted tile. Unlike with convolution, where filters slide over the feature map pixel by pixel, in max pooling, the stride determines the locations where each tile is extracted.

4. Fully Connected Layers

At the end of a convolutional neural network are one or more fully connected layers (when two layers are "fully connected," every node in the first layer is connected to every node in the second layer). Their job is to perform classification based on the features extracted by the convolutions. Typically, the final fully connected layer contains a softmax activation

function, which outputs a probability value from 0 to 1 for each of the classification labels the model is trying to predict.

Preventing Overfitting

As with any machine learning model, a key concern when training a convolutional neural network is overfitting: a model so tuned to the specifics of the training data that it is unable to generalize to new examples.

Two techniques to prevent overfitting when building a CNN are:

Data augmentation: artificially boosting the diversity and number of training examples by performing random transformations to existing images to create a set of new variants. Data augmentation is especially useful when the original training data set is relatively small.

Dropout regularization: Randomly removing units from the neural network during a training gradient step.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Conv2D,MaxPooling2D,Flatten
```

Fig 7.1.1: Importing model and other parameters

- CNN Parameters are implemented in sequential model. As the model is binary classification the output dense value is 1 and using sigmoid as activation function for the last dense layer. Sigmoid activation function gives output for binary classification in range 0 to 1 whereas for the value less than 0.5 its classifies to 0 and for the value greater than 0.5 its classifies to 1.

- Flatten is used to convert data into 1D array.

- Same padding: It applies padding to the input image so that input image gets fully covered by the filter and specified stride. It is same because, for stride 1,the output is same as the input.

```
model = Sequential() # Sequential model
#  adding Convolution Layer followed by Max pooling layer
model.add(Conv2D(16,3, padding='same', input_shape=(80,80,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
#  adding Convolution Layer followed by Max pooling layer
model.add(Conv2D(32,3,padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
#  adding Convolution Layer followed by Max pooling layer
model.add(Conv2D(64,3, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
#  adding Convolution Layer followed by Max pooling layer
model.add(Conv2D(200,3, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
# Converting feature map into 1-D array
model.add(Flatten())
# Fully connected layes with 512 neurons
model.add(Dense(512, activation='relu'))
# Final Output layer
model.add(Dense(1, activation='sigmoid'))
# Summary
model.summary()
```

Fig 7.1.2: Building model

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_24 (Conv2D)           (None, 80, 80, 16)        448
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 40, 40, 16)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 40, 40, 32)        4640
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 20, 20, 64)        18496
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 10, 10, 200)       115400
_________________________________________________________________
max_pooling2d_27 (MaxPooling (None, 5, 5, 200)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 5000)              0
_________________________________________________________________
dense_12 (Dense)             (None, 512)               2560512
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 513
=================================================================
Total params: 2,700,009
Trainable params: 2,700,009
Non-trainable params: 0
```

Fig 7.1.3: Output of Built model

Here,total 2,700,009 parameters to trained in model. After building compiling should be done.

Adam optimizer : Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

Binary crossentropy: Binary crossentropy is a loss function that is used in binary classification tasks. These are tasks that answer a question with only two choices (yes or no, A or B, 0 or 1, left or right).

Accuracy metric: Calculates how often predictions equals labels.

Compiling the model

```
[27] model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

Fig 7.1.4: Compiling the model

7.2 Train the Models

Splitting the data : after the pre-processing is done then the data is split into train and Validation sets

- In deep learning in order to access the performance of the classifier. You train the classifier using 'training set' and then test the performance of your classifier on unseen 'validation set'. An important point to note is that during training the classifier only uses the training set. The validation set must not be used during training the classifier. The validation set will only be available during testing the classifier.
- Training set - a subset to train a model.(Model learns patterns between Input and Output)
- Validation set - a subset to test the trained model.(To test whether the model has correctly learnt)
- The amount or percentage of Splitting can be taken as specified.
- First we need to identify the input and output variables and we need to separate the input set and output set.
- An epoch is a term used in machine learning and indicates the number of passes of the entire training dataset the machine learning algorithm has completed. Datasets are usually grouped into batches (especially when the amount of data is very large). Some

people use the term iteration loosely and refer to putting one batch through the model as an iteration.. This helps in attaining good accuracy to the model.

- Here, the model has been split into 80% for trained data and rest 20% of the data is of validation data by using valididation_split parameter.

```
# Training the model where 20% of data is used for validaton and 80% pf data used for training

history=model.fit(items,output_labels,epochs=25,batch_size=32,validation_split=0.2)
```

Fig 7.2.1: Training the model

```
Epoch 1/25
100/100 [==============================] - 2s 20ms/step - loss: 0.3170 - accuracy: 0.8666 - val_loss: 0.3829 - val_accuracy: 0.8138
Epoch 2/25
100/100 [==============================] - 2s 18ms/step - loss: 0.1494 - accuracy: 0.9362 - val_loss: 0.6701 - val_accuracy: 0.7000
Epoch 3/25
100/100 [==============================] - 2s 18ms/step - loss: 0.0979 - accuracy: 0.9641 - val_loss: 0.3768 - val_accuracy: 0.8250
Epoch 4/25
100/100 [==============================] - 2s 18ms/step - loss: 0.0577 - accuracy: 0.9834 - val_loss: 0.1879 - val_accuracy: 0.9262
Epoch 5/25
100/100 [==============================] - 2s 18ms/step - loss: 0.0589 - accuracy: 0.9784 - val_loss: 0.2348 - val_accuracy: 0.9175
Epoch 6/25
100/100 [==============================] - 2s 18ms/step - loss: 0.0264 - accuracy: 0.9912 - val_loss: 0.3019 - val_accuracy: 0.8950
Epoch 7/25
100/100 [==============================] - 2s 18ms/step - loss: 0.0216 - accuracy: 0.9922 - val_loss: 0.2135 - val_accuracy: 0.9150
Epoch 8/25
100/100 [==============================] - 2s 18ms/step - loss: 0.0082 - accuracy: 0.9975 - val_loss: 0.2287 - val_accuracy: 0.9325
Epoch 9/25
100/100 [==============================] - 2s 18ms/step - loss: 0.0208 - accuracy: 0.9931 - val_loss: 0.1835 - val_accuracy: 0.9425
Epoch 10/25
100/100 [==============================] - 2s 18ms/step - loss: 0.0119 - accuracy: 0.9956 - val_loss: 0.1116 - val_accuracy: 0.9688
Epoch 11/25
100/100 [==============================] - 2s 18ms/step - loss: 0.0061 - accuracy: 0.9987 - val_loss: 0.1531 - val_accuracy: 0.9500
Epoch 12/25
100/100 [==============================] - 2s 18ms/step - loss: 0.0079 - accuracy: 0.9981 - val_loss: 0.2419 - val_accuracy: 0.9212
Epoch 13/25
100/100 [==============================] - 2s 18ms/step - loss: 8.6357e-04 - accuracy: 1.0000 - val_loss: 0.2271 - val_accuracy: 0.9400
Epoch 14/25
100/100 [==============================] - 2s 18ms/step - loss: 1.6838e-04 - accuracy: 1.0000 - val_loss: 0.2359 - val_accuracy: 0.9413
Epoch 15/25
100/100 [==============================] - 2s 18ms/step - loss: 9.3761e-05 - accuracy: 1.0000 - val_loss: 0.2091 - val_accuracy: 0.9500
Epoch 16/25
100/100 [==============================] - 2s 18ms/step - loss: 6.6832e-05 - accuracy: 1.0000 - val_loss: 0.2373 - val_accuracy: 0.9463
Epoch 17/25
100/100 [==============================] - 2s 18ms/step - loss: 4.8759e-05 - accuracy: 1.0000 - val_loss: 0.2396 - val_accuracy: 0.9450
Epoch 18/25
100/100 [==============================] - 2s 18ms/step - loss: 4.0178e-05 - accuracy: 1.0000 - val_loss: 0.2354 - val_accuracy: 0.9475
Epoch 19/25
100/100 [==============================] - 2s 18ms/step - loss: 3.3344e-05 - accuracy: 1.0000 - val_loss: 0.2416 - val_accuracy: 0.9450
Epoch 20/25
100/100 [==============================] - 2s 18ms/step - loss: 2.7642e-05 - accuracy: 1.0000 - val_loss: 0.2280 - val_accuracy: 0.9500

Epoch 21/25
100/100 [==============================] - 2s 18ms/step - loss: 2.4027e-05 - accuracy: 1.0000 - val_loss: 0.2400 - val_accuracy: 0.9488
Epoch 22/25
100/100 [==============================] - 2s 18ms/step - loss: 2.1265e-05 - accuracy: 1.0000 - val_loss: 0.2463 - val_accuracy: 0.9463
Epoch 23/25
100/100 [==============================] - 2s 18ms/step - loss: 1.8208e-05 - accuracy: 1.0000 - val_loss: 0.2605 - val_accuracy: 0.9425
Epoch 24/25
100/100 [==============================] - 2s 18ms/step - loss: 1.6509e-05 - accuracy: 1.0000 - val_loss: 0.2463 - val_accuracy: 0.9475
Epoch 25/25
100/100 [==============================] - 2s 18ms/step - loss: 1.4449e-05 - accuracy: 1.0000 - val_loss: 0.2454 - val_accuracy: 0.9475
```

Fig 7.2.2: Output of training the model

7.3 Validate the Models:

Model validation is the process of evaluating a trained model on test data set. This provides the generalization ability of a trained model. Here I provide a step by step approach to complete first iteration of model validation in minutes.

- The model are validate after completion of training and testing the model.
- Checking the accuracy scores as metrics to validate the models.

```
train_acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
train_loss = history.history['loss']
val_loss = history.history['val_loss']
# Visualisation of graph between train accuracy and validation accuracy
plt.figure(figsize=(12,12))
epochs = list(range(1,26))
plt.plot(epochs,train_acc,label='train_acc',)
plt.plot(epochs,val_acc,label='val_acc')
plt.xlabel("epochs",fontsize=20)
plt.ylabel("accuracy pecentage",fontsize=20)
plt.title('Accuracy graph',fontsize=30)
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f5e32be3ef0>
```

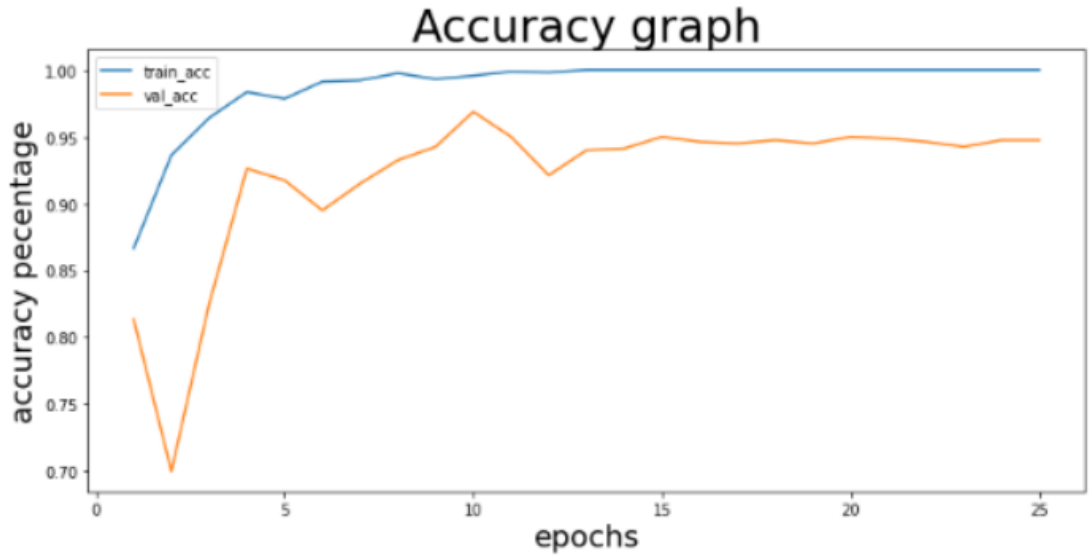

Fig 7.3.1: Train accuracy vs Validation accuracy

```
# Visualisation of graph between train loss and validation loss
plt.figure(figsize=(8,8))
plt.plot(epochs,train_loss,label='train_loss')
plt.plot(epochs,val_loss,label='val_loss')
plt.xlabel("epochs",fontsize=20)
plt.ylabel("loss percentage",fontsize=20)
plt.title('loss graph',fontsize=30)
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f5e32ba9470>
```

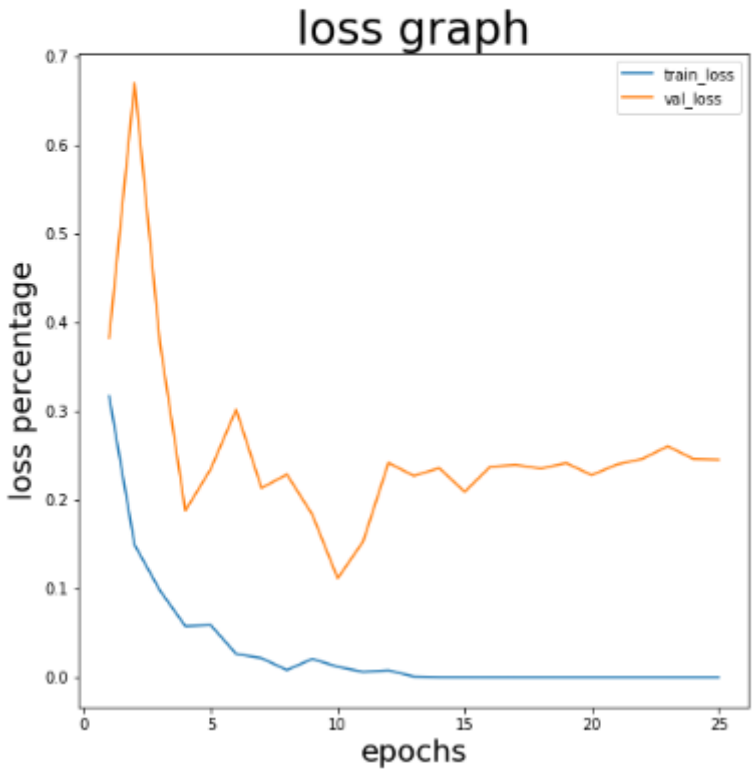

Fig 7.3.2: Train loss vs Validation loss

1. Here, training accuracy is 100%.

2. Validation accuracy is 94.75%

By observing from the above graph test accuracy and validation accuracy doesn't have much variation.

So, the model doesn't over fit or under fit.

7.4 Make Predictions:

- We have a method called predict , using this method we need to predict the output for given input and we need to compare that the model is predicting correctly or not. If the

model is predicting correctly then the model is good otherwise some changes should be done.

- Now, I am trying to predict an unknown image which is not present in the training or validation set.

Below image is an unknown image with a different size. Now it should predict the image is consisting of ship or not.

Fig 7.4.1: Unknown sample of ship image

In order to predict the new image we need to resize and scale the new image. So, that image is classified without any errors.

```
# Predicting unknown ship image
new_image = image.load_img('/content/drive/My Drive/project/1.png')
print(type(new_image))

new_image = tf.keras.preprocessing.image.img_to_array(new_image)
print("shape of image :",new_image.shape)
print("Type of image :",type(new_image))

# resizing
new_image = tf.image.resize(new_image,(80,80))

## Scaling
new_image = new_image/255
print('After resizing the image shape :',new_image.shape)
new_image = np.expand_dims(new_image,axis=0)
print("image shape",new_image.shape)
```

```
<class 'PIL.PngImagePlugin.PngImageFile'>
shape of image : (101, 95, 3)
Type of image : <class 'numpy.ndarray'>
After resizing the image shape : (80, 80, 3)
image shape (1, 80, 80, 3)
```

Fig 7.4.2: Resizing and scaling unknown sample image

- Predicting the scaled image

```
#Predicting
print('Output predicted',model.predict(new_image))
array_num = model.predict(new_image)
num_list = array_num.tolist()
x=num_list[0][0]
if x > 0.5:
  print('the given image contains SHIP')
else:
  print('the given image does NOT contains SHIP')
```

```
Output predicted [[1.]]
the given image contains SHIP
```

Fig 7.4.3: Predicting the ship image

The output array([1.]], datatype=float32) represents the ship is present in the image

Similarly, Predicting for no ship image from unknown satellite image where image does not consists of ship.

Fig 7.4.4: Unknown sample of no ship image

```
# predicting of unknown image where the image does not consists ship
new_image = image.load_img('/content/drive/My Drive/project/0.png')
print(type(new_image))

new_image = tf.keras.preprocessing.image.img_to_array(new_image)
print("shape of image :",new_image.shape)
print("Type of image :",type(new_image))

# resizing
new_image = tf.image.resize(new_image,(80,80))

## Scaling
new_image = new_image/255
print('After resizing the image shape :',new_image.shape)
new_image = np.expand_dims(new_image,axis=0)
print("image shape",new_image.shape)
```

```
<class 'PIL.PngImagePlugin.PngImageFile'>
shape of image : (76, 78, 3)
Type of image : <class 'numpy.ndarray'>
After resizing the image shape : (80, 80, 3)
image shape (1, 80, 80, 3)
```

```
#Predicting
print('Output predicted',model.predict(new_image))
array_num = model.predict(new_image)
num_list = array_num.tolist()
x=num_list[0][0]
if x > 0.5:
  print('the given image contains SHIP')
else:
  print('the given image does NOT contains SHIP')
```

```
Output predicted [[1.7355088e-10]]
the given image does NOT contains SHIP
```

Fig 7.4.5: Predicting the no ship image

The output array([[1.7355088e-10]], dtype=float32)this values is less than 0.5 and it is almost equal to 0.

From this it can be confirmed that the built model classifies images correctly.

CONCLUSION

It is concluded after performing thorough building model using CNN, the model is computed to get good accuracy for unknown data that detects ship images and non-ship images correctly which leads to clear understanding of the how the model parameters should be defined to get best accuracy and not leading to under fit or over fit. Thus, it leads to point of getting the solution for the problem statement of identification of ships through satellite images.

REFERENCES

- https://en.wikipedia.org/wiki/Deep_learning
- https://en.wikipedia.org/wiki/Machine_learning
- https://www.kaggle.com/rhammell/ships-in-satellite-imagery
- https://www.tensorflow.org/tutorials/images/cnn
- https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras
- **GITHUB LINK:** https://github.com/sahithreddy567/Project-AIML