

OIL SPILLS DETECTION ON A SEA SURFACE BY USING SENTINAL-1 SAR IMAGES

A Project Report Phase – II

**Submitted in partial fulfilment of requirements for the award of
Bachelor of Engineering degree in Electronics and Communication Engineering**

by

K.Jahnavi (39130236)

K.M.Pushpa latha (39130228)



**DEPERTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119

APRIL – 2023



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI
- 600 119**

www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **K.JAHNAVI** (39130236) and **K.M.PUSHPA LATHA** (39130228), who carried out the project entitled "**OIL SPILLS DETECTION ON A SEA SURFACE BY USING SENTINAL-1 SAR IMAGES**" under our supervision from NOVEMBER 2022 to APRIL 2023.

Dr.R.M.JOANY, M.E., Ph.D.,

Internal Guide

Dr. T. RAVI, M.E., Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on 20/04/2023

Internal Examiner

External Examiner

DECLARATION

We, **K. JAHNAVI** (39130236) and **K.M. PUSHPA LATHA** (39130228), hereby declare that the Project Report entitled **OIL SPILLS DETECTION ON A SEA SURFACE BY USING SENTINAL-1 SAR IMAGES** is done by us under the guidance of, **Dr.R.M.Joany, M.E., Ph.D.**, is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in **Electronics and Communication Engineering**.

DATE: 20/04/2023

PLACE: Chennai

1.



2. *Pushpa latha*

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N. M. NANDHITHA, M.E., Ph.D., Professor & Dean, School of Electrical and Electronics** and **Dr. T. RAVI, Ph.D., Professor & Head, Department of Electronics and Communication Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our **Project Guide Dr.R.M.JOANY,M.E.,Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of Electronics and Communication Engineering who were helpful in many ways for the completion of the project.

ABSTRACT

A major and main reason for marine pollution is discharge of oil into the water bodies and this has become common. It has much impact on aqua life. This project proposes an algorithm to detect the oil spills on the water bodies like ocean and sea surfaces by using sentinel SAR images. In this project SAR (Synthetic Aperture Radar) images are taken and analysed. SAR is a common imagery medium for detecting oil spills. Object detection is a frequent and important field of study in image processing. Marine oil spill pollution poses a serious threat to the ecology of the world's oceans. Thousands of tons of oil are spilled into the oceans every year due to both anthropogenic causes, such as tanker accidents, rupture of rigs/pipelines or malfunctioning of oil extraction platforms, and natural events, such as natural seepage from seabed oil structures. As discussed, we can find the oil spills on the surfaces by many of the methods. Laser fluoro sensors are the best sensors for oil spill detection, and have the capability of identifying oil on shores, ice and snow, and determining what type of oil has been spilled. However, they require relatively cloud free conditions to detect the oil spill. Oil spills can destroy marine life as well as damage habitat for land animals and humans. The majority of marine oil spills result from ships emptying their ballast tanks before or after entering port. Large area oil spills result from tanker ruptures or collisions with reefs, rocky shoals, or other ships.

A good handling operation of oil spills detection depends on the accelerated response from the time of oil spills is recognized. This approach is implemented, and results are verified in WEBSERVER. Once the oil spills are detected they are useful to clean water bodies and water pollution is reduced.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	V
	LIST OF FIGURES	Viii
	LIST OF TABLES	Ix
	LIST OF GRAPHS	x
	LIST OF ABBREVIATIONS	xi
1	INTRODUCTION	1
	1.1 Oil spills	1
	1.2 Synthetic-aperture radar (SAR)	3
	1.3 Machine Learning	5
	1.4 Deep learning	6
	1.5 Convolutional neural networks (CNNs)	8
	1.6 Detection of oil spills	11
	1.7 Environmental consequences due to oil spills	12 13
	1.8 Harmful effects caused to other creatures.	14
	1.9 Problem statement	
2	LITERATURE SURVEY	16
3	PROPOSED METHODOLOGY	25
	3.1 Existing work	25
	3.2 Proposed work	27
4	EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED	30
	4.1 Installation process of visual studio	30
	4.2 Convolutional neural networks method	32
	4.3 Algorithm used	33
5	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	49
	5.1 Result	49

	5.2 Discussion	51
	5.3 Trained and tested data	54
	5.4 Performance analysis with CNN model	56
6	SUMMARY AND CONCLUSIONS	61
	REFERENCES	62
	APPENDIX	64
	A. Source code	64

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
Fig: 1.1	CNN model.	9
Fig: 3.1	1D CFAR (left) and nested (2D) CFAR (right) for radar image processing.	27
Fig: 3.2	Flow chart of proposed work.	29
Fig: 4.1	Downloading the Visual Studio Code.	30
Fig: 4.2	Installing the Visual Studio Code.	31
Fig: 4.3	Installation of python in Visual Studio Code.	31
Fig: 4.4	Installation of Jupyter Notebook in Visual Studio Code.	31
Fig: 4.5	CNN architecture.	32
Fig: 4.6	Image processing in CNN.	33
Fig: 4.7	Dataset division into spill and no spill.	34
Fig: 4.8	Resizing the images.	36
Fig: 4.9	Splitting of dataset.	37
Fig: 4.10	Convolution of the image.	41
Fig: 4.11	Optimisation.	43
Fig: 4.12	Training the dataset.	45
Fig: 4.13	Creating a Webpage.	46
Fig: 4.14	Webpage design for Oil spill detection.	47
Fig: 5.1	No oil spill has been detected	49
Fig: 5.2	Oil spill has been detected	50
Fig: 5.3	Oil spill areas in the world	50
Fig: 5.4	A typical workflow of a model development	54
Fig: 5.5	Architecture of VGG	58

LIST OF TABLES

TABLE No.	TABLE NAME	PAGE No.
Table: 5.1	Confusion matrix	52

LIST OF GRAPHS

GRAPH No.	GRAPH NAME	PAGE No.
Graph: 5.1	Oil spills from the year 1970-2021	51
Graph: 5.2	Confusion matrix	53
Graph: 5.3	Confusion matrix analysis	53
Graph: 5.4	Model accuracy	55
Graph: 5.5	Model loss	56
Graph: 5.6	CNN Vs ANN accuracy comparison	57
Graph: 5.7	CNN Vs VGG accuracy comparison	59
Graph: 5.8	CNN Vs Resnet 50 accuracy comparison.	60

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
SAR	Synthetic-Aperture Radar
DEM	Digital Elevation Model
PSI	Persistent Scatterer Interferometry
LDA	Linear Discriminant Analysis
ANOVA	Analysis Of Variance
ANN	Artificial Neural Networks
SVM	Support Vector Machine
ML	Machine Learning
AI	Artificial Intelligence
CNN	Convolutional Neural Networks
RNN	Recurrent Neural Network
CV	Computer Vision
NLP	Natural Language Processing
UI	User Interface
RESNET 50	Residual Network
VGG	Visual Geometry Group

CHAPTER -1

INTRODUCTION

Oil is an important source of energy in the world. It is transferred from one place to another by the ships in the sea and through the pipes. Due to some kinds of problems, the pipes leaked, and the oil spread in the oceans. This is called an oil spill. The release of petroleum in the marine ecosystem is pollution. It occurs in the oceans but also spreads to nearby lands. This happens because of several reasons like transferring oil to other containers, breaking pipelines through which oil is transported from one place to another and at the time of drilling in the earth's crust. Oil is a mixture of hydrocarbons, and it causes harm to the surrounding environment. It can destroy animals and plants.

1.1 OIL SPILLS

The catastrophic and massive spills are highly considered as their damage to the environment is high in quantity. But the smaller and chronic oil spills occur on anormal basis. For millions of years, under high pressure and temperature, the remains of fossils changed into hydrocarbons which are called petroleum. It is also known as crude oil. It is composed of gas, kerosene and light gas which can cause hazardous effects on ecosystems. If it is consumed by any other species of the ocean, then it will cause high damage to them. The production of oil rose in the millions. Due to massive transportation, it results in oil spills. A large number of waste oils are entered into the ocean which comes from the land and the other waste disposal. It can cause a great impact on the health of the people who consume seafood. Strict measurements will be taken in this regard. So that it cannot cause an impact on the environment.

1.1.1 Oil leak

If we talk about spill meaning, according to the Oxford Learner Dictionary, "It is an act of letting a liquid come or fall out of a container." An oil spill occurs because of leakage of oil especially in the ocean waters and this leakage creates oil spill pollution which is a kind of ocean pollution because it leaves greater impacts on the oceans and when

this oil spill occurs at a larger level which leads to leaving impacts on the water body, environment, marine life as well as the economy of the nation for many years, then this is termed as oil spill disaster. The following graph shows the average number of oil spills in the ocean per year by decade which clearly shows that it has been decreased with time.

1.1.2 Causes of oil spills

There are some causes from which oil spills occurred, a few of which are:

- It happens on the land or sea because of the leaks from ships, wells, pipelines and other carriers of the oil.
- It can be caused by some accidents which include tankers, refineries and drilling rigs.
- It can be caused by the storage facilities.
- Due to the production in millions, it will result in massive transportation and oil spills.
- Spills can be caused by the people who make mistakes and sometimes they become careless.
- It happens because of the breaking of equipment.
- It can also happen because of disasters.
- It can also occur because of the people who do illegal work.

1.1.3 Effects of oil spills

Oil spills are damaging in a variety of ways. There are some effects of oil spills which are 3 When oil spills occur in the ocean, the oil does not mix with the water, and it spreads on the surface. Because of the thick layer present in the ocean which is also known as slick expands because of which the sunlight does not reach the ocean animals and plants.

- It affects the food chain of the whole ecosystem.
- When oil spills occur, it will break down a moderate amount of oil and deposit on the bottom.

· When oil spreads in the ocean, it breaks and sinks into the ocean which deteriorates the health of the ocean life.

- It contaminates the natural habitat of the ocean.
- Oil spills have harsh effects on the coastlines. The birds and the sea mammals are covered by the oil.
- Oil is so damaging to the whole ecosystem as it covers everything which is nearby the coastlines like sand, animals, grass and soil etc.
- It destroys everything near the coastline.

1.2 SYNTHETIC-APERTURE RADAR (SAR)

Synthetic-aperture radar (SAR) is a form of radar that is used to create two dimensional images or three-dimensional reconstructions of objects, such as landscapes. SAR uses the motion of the radar antenna over a target region to provide finer spatial resolution than conventional stationary beam-scanning radars. SAR is typically mounted on a moving platform, such as an aircraft or spacecraft, and has its origins in an advanced form of side looking airborne radar (SLAR). The distance the SAR device travels over a target during the period when the target scene is illuminated creates the large synthetic antenna aperture (the size of the antenna). Typically, the larger the aperture, the higher the image resolution will be, regardless of whether the aperture is physical (a large antenna) or synthetic (a moving antenna) – this allows SAR to create high-resolution images with comparatively small physical antennas. For a fixed antenna size and orientation, objects which are further away remain illuminated longer - therefore SAR has the property of creating larger synthetic apertures for more distant objects, which results in a consistent spatial resolution over a range of viewing distances. To create a SAR image, successive pulses of radio waves are transmitted to "illuminate" a target scene, and the echo of each pulse is received and recorded. The pulses are transmitted, and the echoes received using a single beam- forming antenna, with wavelengths of a meter down to several millimetres. As the SAR device on board the aircraft or spacecraft moves, the antenna location relative to the target changes with time. Signal processing of the successive recorded radar echoes allows the combining of the recordings from these multiple antenna positions. This process

forms the synthetic antenna aperture and allows the creation of higher-resolution images than would otherwise be possible with a given physical antenna.

1.2.1 Basic Principle of SAR

A synthetic-aperture radar is an imaging radar mounted on an instant moving platform. Electromagnetic waves are transmitted sequentially, the echoes are collected, and the system electronics digitizes and stores the data for subsequent processing. As transmission and reception occur at different times, they map to different small positions. The well-ordered combination of the received signals builds a virtual aperture that is much longer than the physical antenna width. That is the source of the term "synthetic aperture," giving it the property of an imaging radar. The range direction is perpendicular to the flight track and perpendicular to the azimuth direction, which is also known as the along-track direction because it is in line with the position of the object within the antenna's field of view. The 3D processing is done in two stages. The azimuth and range direction are focused for the generation of 2D (azimuth-range) high-resolution images, after which a digital elevation model (DEM) is used to measure the phase differences between complex images, which is determined from different look angles to recover the height information. This height information, along with the azimuth- range coordinates provided by 2-D SAR focusing, gives the third dimension, which is the elevation. The first step requires only standard processing algorithms, for the second step, additional pre-processing such as image co- registration and phase calibration is used. In addition, multiple baselines can be used to extend 3D imaging to the time dimension. 4D and multi-D SAR imaging allows imaging of complex scenarios, such as urban areas, and has improved performance with respect to classical interferometric techniques such as persistent scatterer interferometry (PSI).

1.2.2 Applications of SAR

SAR is capable of high-resolution remote sensing, independent of flight altitude, and independent of weather, as SAR can select frequencies to avoid weather- caused signal attenuation. SAR has day and night imaging capability as illumination is provided by the SAR. SAR images have wide applications in remote sensing and mapping of surfaces of the Earth and other planets. Applications of SAR are numerous. Examples include topography, oceanography, and glaciology, geology (for

example, terrain discrimination and subsurface imaging). SAR can also be used in forestry to determine forest height, biomass, and deforestation. Volcano and earthquake monitoring use differential interferometry. SAR can also be applied for monitoring civil infrastructure stability such as bridges. SAR is useful in environment monitoring such as oil spills, flooding, urban growth, military surveillance: including strategic policy and tactical assessment. SAR can be implemented as inverse SAR by observing a moving target over a substantial time with a stationary antenna.

1.3 MACHINE LEARNING

Machine learning is a very hot topic for many key reasons, and because it provides the ability to automatically obtain deep insights, recognize unknown patterns, and create high performing predictive models from data, all without requiring explicit programming instructions. This high-level understanding is critical if ever involved in a decision-making process surrounding the usage of machine learning, how it can help achieve business and project goals, which machine learning techniques to use, potential pitfalls, and how to interpret the results. The most common machine learning tasks that one may come across while trying to solve a machine learning problem. Under each task are also listed a set of machine learning methods that could be used to resolve these tasks. Please feel free to comment/suggest if I missed mentioning one or more important points. Following are the key machine learning tasks briefed later in this article: Feature Selection, Regression, Classification, Clustering, Multivariate querying, Density estimation, Dimension reduction, Testing and matching. Following are top 8 most common machine learning tasks that one could come across most frequently while solving an advanced analytics problem: Feature Selection: Feature selection is one of the critical tasks which would be used when building machine learning models. Feature selection is important because selecting right features would not only help build models of higher accuracy but also help achieve objectives related to building simpler models, reduce overfitting etc. The following are some of the techniques which could be used for feature selection: Filter methods which help in selecting features based on the outcomes of statistical tests. The following are some of the statistical tests which are used: Pearson's correlation, Linear discriminant analysis (LDA), Analysis of Variance (ANOVA), Chi-square tests 4 Wrapper methods

which help in feature selection by using a subset of features and determining the model accuracy. The following are some of the algorithms used: Forward selection, backward elimination.

1.3.1 Classification

Classification tasks is simply related with predicting a category of a data (discrete variables). One of the most common examples is predicting whether or not an email is spam or ham. Some of the common use cases could be found in the area of healthcare such as whether a person is suffering from a particular disease or not. It also has its application in financial use cases such as determining whether a transaction is fraud or not. The ML methods such as following could be applied to solve classification tasks:

Kernel discriminant analysis (Higher accuracy), K-Nearest Neighbours (Higher accuracy), Artificial neural networks (ANN) (Higher accuracy), Support vector machine (SVM) (Higher accuracy), Random forests (Higher accuracy), Decision trees, Boosted trees, Logistic regression, naive Bayes.

1.4 DEEP LEARNING

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behaviour of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

1.4.1 Deep learning vs. machine learning

If deep learning is a subset of machine learning, how do they differ? Deep learning distinguishes itself from classical machine learning by the type of data that it works with and the methods in which it learns.

Machine learning algorithms leverage structured, labelled data to make predictions—meaning that specific features are defined from the input data for the model and organized into tables. This does not necessarily mean that it doesn't use unstructured data; it just means that if it does, it generally goes through some pre-processing to organize it into a structured format.

Deep learning eliminates some of data pre-processing that is typically involved with machine learning. These algorithms can ingest and process unstructured data, like text and images, and it automates feature extraction, removing some of the dependency on human experts. For example, let us say that we had a set of photos of different pets, and we wanted to categorize by “cat”, “dog”, “hamster”, et cetera. Deep learning algorithms can determine which features (e.g., ears) are most important to distinguish each animal from another. In machine learning, this hierarchy of features is established manually by a human expert.

Then, through the processes of gradient descent and backpropagation, the deep learning algorithm adjusts and fits itself for accuracy, allowing it to make predictions about a new photo of an animal with increased precision.

Machine learning and deep learning models are capable of different types of learning as well, which are usually categorized as supervised learning, unsupervised learning, and reinforcement learning. Supervised learning utilizes labelled datasets to categorize or make predictions; this requires some kind of human intervention to label input data correctly. In contrast, unsupervised learning doesn't require labelled datasets, and instead, it detects patterns in the data, clustering them by any distinguishing characteristics. Reinforcement learning is a process in which a model learns to become more accurate for performing an action in an environment based on feedback in order to maximize the reward.

1.4.2 How deep learning works

Deep learning neural networks, or artificial neural networks, attempts to mimic the human brain through a combination of data inputs, weights, and bias. These elements work together to accurately recognize, classify, and describe objects within the data.

Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called *visible* layers. The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made.

Another process called backpropagation uses algorithms, like gradient descent, to calculate errors in predictions and then adjusts the weights and biases of the function by moving backwards through the layers in an effort to train the model. Together, forward propagation and backpropagation allow a neural network to make predictions and correct for any errors accordingly. Over time, the algorithm becomes gradually more accurate.

The above describes the simplest type of deep neural network in the simplest terms. However, deep learning algorithms are incredibly complex, and there are different types of neural networks to address specific problems or datasets. For example,

- **Convolutional neural networks (CNNs)**, used primarily in computer vision and image classification applications, can detect features and patterns within an image, enabling tasks, like object detection or recognition. In 2015, a CNN bested a human in an object recognition challenge for the first time.
- **Recurrent neural network (RNNs)** are typically used in natural language and speech recognition applications as it leverages sequential or times series data.

1.5 CONVOLUTIONAL NEURAL NETWORKS (CNNS)

Within Deep Learning, a Convolutional Neural Network or CNN is a type of artificial neural network, which is widely used for image/object recognition and classification. Deep Learning thus recognizes objects in an image by using a CNN. CNNs are playing

a major role in diverse tasks/functions like image processing problems, computer vision tasks like localization and segmentation, video analysis, to recognize obstacles in self-driving cars, as well as speech recognition in natural language processing. As CNNs are playing a significant role in these fast-growing and emerging areas, they are very popular in Deep Learning.

1.5.1 Brain's Architecture Inspired CNNs

A typical neural network will have an input layer, hidden layers, and an output layer. CNNs are inspired by the architecture of the brain. Just like a neuron in the brain processes and transmits information throughout the body, artificial neurons or nodes in CNNs take inputs, processes them and sends the result as output. The image is fed as input. The input layer accepts the image pixels as input in the form of arrays. In CNNs, there could be multiple hidden layers, which perform feature extraction from the image by doing calculations. This could include convolution, pooling, rectified linear units, and fully connected layers. Convolution is the first layer that does feature extraction from an input image. The fully connected layer classifies the object and identifies it in the output layer. "CNNs are feedforward networks in that information flow takes place in one direction only, from their inputs to their outputs. Just as artificial neural networks (ANN) are biologically inspired, so are CNNs. The visual cortex in the brain, which consists of alternating layers of simple and complex cells, motivates their architecture. CNN architectures come in several variations; however, in general, they consist of convolutional and pooling (or subsampling) layers, which are grouped into modules.

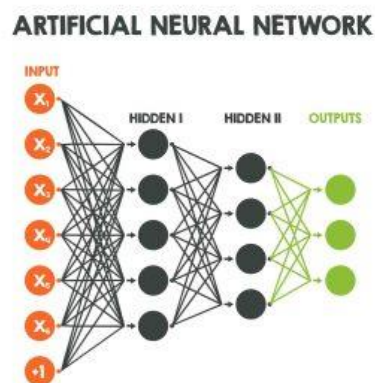


Fig: 1.1 CNN model

1.5.2 CNN in image processing

A convolutional neural network (CNN or convnet) is a subset of machine_learning. It is one of the various types of artificial neural_networks which are used for different applications and data types. A CNN is a kind of network architecture for deep learning algorithms and is specifically used for image_recognition and tasks that involve the processing of pixel data.

There are other types of neural networks in deep learning, but for identifying and recognizing objects, CNNs are the network architecture of choice. This makes them highly suitable for computer vision (CV) tasks and for applications where object recognition is vital, such as self-driving-cars and facial recognition.

Artificial neural networks (ANNs) are a core element of deep learning algorithms. One type of an ANN is a recurrent neural network (RNN) that uses sequential or time series data as input. It is suitable for applications involving natural language processing (NLP), language translation, speech recognition and image captioning.

The CNN is another type of neural network that can uncover key information in both time series and image data. For this reason, it is highly valuable for image-related tasks, such as image recognition, object classification and pattern recognition. To identify patterns within an image, a CNN leverages principles from linear algebra, such as matrix multiplication. CNNs can also classify audio and signal data.

A CNN's architecture is analogous to the connectivity pattern of the human brain. Just like the brain consists of billions of neurons, CNNs also have neurons arranged in a specific way. In fact, a CNN's neurons are arranged like the brain's frontal lobe, the area responsible for processing visual stimuli. This arrangement ensures that the entire visual field is covered, thus avoiding the piecemeal image processing problem of traditional neural networks, which must be fed images in reduced-resolution pieces. Compared to the older networks, a CNN delivers better performance with image inputs, and also with speech or audio signal inputs.

1.6 DETECTION OF OIL SPILLS

Oil pollutions of the oceans are mainly caused by accidents in offshore industry and oil tankers as well as by operative discharges of oil products from ships. According to 29% of pollutions in the ocean annually are crude oil and about 48% - fuels. They both are a great threat to the marine environment and the eco system of the oceans and that is why it is of a great importance to have a good surveillance system for observation of the sea surface. Spaceborne SAR systems are now commonly used as a complement to existing ground-based monitoring systems, ship and aircraft observation with their advantages, including wide area coverage and capabilities to monitor ocean surface day and night in all-weather conditions. Ships on the sea surface can be detected as bright spots on the SAR images, while oil spills can be detected as dark spots. Ships signatures in SAR images are relatively brighter on the darker noisier background of sea surface because of backscattering coherence from the ships due to the presence of metal corner reflectors on their hulls and superstructures. Oil spills appear as dark spots due to the changes they cause in Bragg backscatter by damping of the resonant sea surface capillary waves and changing the surface roughness. The contrast between oil spills and sea background depends on the wavelength of radar systems. X-band radars provide best contrast, C-band radars provide moderate and L-band – smallest. Polarization of radio waves also influences the quality of radar pictures. According to, best results are provided by using VV polarization. Detection of ships on the sea surface, based on the pixel value intensities is studied relatively well and carried out mostly by using algorithms for automatic detection, while image analysis for detection of oil spills is mostly carried out by using visual methods. Reasons are related to the influence of hydro-meteorological conditions and presence of phenomena that look like oil spills on the SAR images, such as ice, algal blooms, freshwater slicks and rain cells, ships' wakes, etc. A ship bright echo and an oil spill dark spot on the sea surface of Sentinel – 1 SAR image Wind and sea state are important factors, that influence on the ability of SAR to detect oil spills. In calm sea and light wind conditions as well as in stormy weather conditions, it is not possible to detect oil spills on water surface. Ideal conditions to identify oil spill are as follows: wind speed between 1.5m/s and 10.0m/s and wave height between 30cm and 3m. Sometimes it is hard even for well-trained operators to distinguish oil slicks and look-alike phenomena, by using only SAR intensity. However,

some differences between oil spills and natural slicks can be detected by using phase differences between the VV and HH co-polarized channels in complex SAR imagery. It is also possible to use data, obtained from optical sensors as a complement to SAR, in particular for areas closer to the coast, but it is well known that the quality of this data strongly depends on atmospheric conditions in the area of observation.

1.7 ENVIRONMENTAL CONSEQUENCES DUE TO OIL SPILLS

Generally, oil spills harm ocean life in two ways

Fouling or oiling: Fouling or oiling occurs when oil physically harms a plant or animal. Oil can coat a bird's wings and leave it unable to fly or strip away the insulating properties of a sea otter's fur, putting it at risk of hypothermia. The degree of oiling often impacts the animal's chances of survival.

Oil toxicity: Oil consists of many different toxic compounds. These toxic compounds can cause severe health problems like heart damage, stunted growth, Immune system effects, and even death.

Oil spills caused by damaged tankers, pipelines, or offshore oil rigs often result in immediate and long-term environmental damage that can last for decades. These are among the most notable areas of environmental damage caused by spills

- Beaches
- Marshlands
- Fragile Aquatic Ecosystems

Oil spills coat everything they touch and become unwelcome but long-term parts of every ecosystem they enter.

When an oil slick from a large spill reaches a beach, oil coats and clings to every rock and grain of sand. If the oil washes into coastal marshes, mangrove forests, or other

wetlands, fibrous plants and grasses absorb oil, which can damage plants and make the area unsuitable as wildlife habitat.

When oil eventually stops floating on the water's surface and begins to sink into the marine environment, it can have similar damaging effects on fragile underwater ecosystems, killing or contaminating fish and smaller organisms that are essential links in the global food chain.

1.8 HARMFUL EFFECTS CAUSED TO OTHER CREATURES

Birds

Oil-covered birds are a universal symbol of environmental damage wreaked by oil spills. Some species of shore birds might escape by relocating if they sense danger in time, but sea birds that swim and dive for their food are most likely to be covered in oil following a spill. Oil spills also damage nesting grounds, potentially causing serious long-term effects on entire species.

Even a small amount of oil can be deadly to a bird. By coating feathers, oil not only makes flying impossible but also destroys birds' natural waterproofing and insulation, leaving them vulnerable to hypothermia or overheating. As birds frantically preen their feathers to restore their natural protections, they often swallow oil, which can severely damage their internal organs and lead to death.

Marine mammals

Oil spills frequently kill marine mammals such as whales, dolphins, seals, and sea otters. Oil can clog blowholes of whales and dolphins, making it impossible for them to breathe properly and disrupting their ability to communicate. Oil coats fur of otters and seals, leaving them vulnerable to hypothermia.

Even when marine mammals escape the immediate effects, an oil spill can contaminate their food supply. Marine mammals that eat fish or other food exposed to an oil spill may be poisoned by oil and die or experience other problems.

Fish

Oil spills often take a deadly toll on fish, shellfish, and other marine life, particularly if many fish eggs or larvae are exposed to oil. Fish, shellfish, and corals may not be exposed immediately, but can come into contact with oil if it is mixed into the water column shellfish can also be exposed in the intertidal zone. When exposed to oil, adult fish may experience reduced growth, enlarged livers, and changes in heart and respiration rates, fin erosion, and reproduction impairment. Fish eggs and larvae can be especially sensitive to lethal and sub lethal impacts. Even when lethal impacts are not observed, oil can make fish and shellfish unsafe for humans to eat.

Wildlife habitat and breeding grounds

Long-term damage to species and their habitats and nesting or breeding grounds is one of the most far-reaching environmental effects caused by oil spills. Even species that spend most of their lives at sea, such as various species of sea turtles, must come ashore to nest. Sea turtles can be harmed by oil they encounter in the water or on the beach where they lay their eggs, their eggs can be damaged by oil and fail to develop properly, and newly hatched turtles may be oiled as they scurry toward the ocean across an oily beach.

Ultimately, the severity of environmental damage caused by an oil spill depends on many factors, including the amount of oil spilled, type and weight of oil, location of the spill, species of wildlife in the area, timing of breeding cycles and seasonal migrations, and even the weather at sea during and after the oil spill.

1.9 PROBLEM STATEMENT

A main reason for marine pollution is discharge of oil into the water bodies and this has become common. Oil pollutions of the oceans are mainly caused by accidents in offshore industry and oil tankers as well as by operative discharges of oil products from ships. According to 29% of pollutions in the ocean annually are crude oil and about 48% - fuels. They both are a great threat to the marine environment and the ecosystem of the oceans and that is why it is of a great importance to have a good surveillance system for observation of the sea surface. As it has much impact on aqua

life. When oil spills occur in the ocean, the oil does not mix with the water, and it spreads on the surface. Because of the thick layer present in the ocean which is also known as slick expands because of which the sunlight does not reach the ocean animals and plants. It affects the food chain of the whole ecosystem. When oil spills occur, it will break down a moderate amount of oil and deposit on the bottom. When oil spreads in the ocean, it breaks and sinks into the ocean which deteriorates the health of the ocean life. It contaminates the natural habitat of the ocean. Oil spills have harsh effects on the coastlines. The birds and the sea mammals are covered by the oil. Oil is damaging to the whole ecosystem as it covers everything which is nearby the coastlines like sand, animals, grass and soil etc. It destroys everything near the coastline. The oil that spills into the ocean can have a great impact on people's health who consume seafood. It can be damaging for those marine animals who swim in the ocean and if they ingest oil then it will be destroying them.

To collect the SAR images from the satellite and store them as the data. The SAR images are converted to grey scale images so it will be easy to examine the oil spills on the sea or ocean surface. Develop an algorithm to detect the oil spills. Develop the algorithm in deep learning platform. The detected information is stored and used for the future purposes.

CHAPTER-2

LITERATURE SURVEY

2.1 PROJECT TITLE: Oil spills detection in MATHLAB software using SAR-images.

AUTHOR NAME: Mital K. Sorde, Mrs Aruna sharma.

YEAR OF PUBLISHING: October 2018 at International Research Journal of Engineering and Technology (IRJET). A major aspect of marine pollution, oil discharge into the sea has become a frequent phenomenon. The main aim of this research is to detect oil spills using satellite images. SAR (Synthetic Aperture Radar) imagery is a common medium for detecting oil spills. Every year, several thousand tonnes of oil are spilled or illegally dumped into the ocean. Finding them with an automatic arrangement could give an early warning so it may be cleared up before doing too much harm. The method consists of four stages, namely: 1) RGB to Grayscale 2) Image Filtering 3) Image Thresholding; and 4) Object understanding of the segmented objects as oil spills. This approach is implemented, and results are verified in MATLAB software. By this technique we can obtain proper detection of oil spill Edge detection, Shape detection and thresholding gives accuracy for the output result which is the main advantage of this technique. Such detection can be useful during assessment of clean-up operations.

2.2 PROJECT TITLE: Oil spills detection on sea surface by using sentinel–1 SAR images.

AUTHOR NAME: Chavdar Alexandrov, Nikolay Kolev, Yordan Sivkov, Avgustin Hristov.

YEAR OF PUBLISHING: 2018 by Dept. of Electronics Nikola Vaptsarov Naval Academy. To validate the detection, an official report, provided by European satellitebased oil spill monitoring service Clean Sea Net has been used. According to 29% of pollutions in the ocean annually are crude oil and about 48% - fuels. Ships on the sea surface can be detected as bright spots on the SAR images, while oil spills can be detected as dark spots. The contrast between oil spills and sea background

depends on the wavelength of radar systems. X-band radars provide best contrast, C-band radars provide moderate and L-band – smallest. Polarization of radio waves also influences the quality of radar pictures. According to, best results are provided by using VV polarization. Ideal conditions to identify oil spill are as follows: wind speed between 1.5m/s and 10.0m/s and wave height between 30cm and 3m. In order to test the ability of automatic detection of oil spills by using satellite remote sensing we use the following process. (A.) SAR Image Processing First the image was changed by an upside down reflection (flipping) and shifting up to the maximum pixel value of the image: $UFlippl(i, j) = MaxVal - I(i, j)$, for each i, j , (1) where $I(M \times N)$ is the Sentinel-1 SAR image with size $M \times N$ pixels, $UFlippl(M \times N)$ – upside down flipped image, and $MaxVal = \max(I(i, j))$, for each i, j , is the maximum pixel value of the SAR image $I(M \times N)$. B. Oil Spills Detection to make an appropriate oil spill detection after the preprocessing, the CFAR detector is applied to adapt the threshold constantly to the mean value of the sea clutter return. s. If the background statistics have a Gaussian distribution, the detection test is: $0 \leq y \leq \mu_B + T \cdot \sigma_B + E_0$ (2) where μ_B is the mean value of background, σ_B is the background standard deviation, E_0 is an initial value, set by operator and T is the design parameter. Satellite-based SAR systems are one of the best sensors for oil slicks detection in wide areas, not only for daytime but also for night-time observation. Although there are some wind and sea state limitations as well as many look-alikes, SAR is nowadays the standard tool. For oil spills detection and monitoring offshore. It is also possible to use data, obtained from optical sensors as a complement to SAR, in particular for areas closer to the coast, but it is well known that the quality of this data strongly depends on atmospheric conditions in the area of observation is one of the drawback in this proposed survey.

2.3 PROJECT TITLE: Oil spill detection using space-borne sentinel-1 SAR imagery.

AUTHOR NAME: K. Topouzelis, and S. Singha.

PUBLISHED BY: University of the Aegean, Mytilini, Greece German Aerospace Centre (DLR), Bremen, Germany. Operational monitoring of oil discharges using synthetic aperture radar (SAR) images contributes significantly toward fewer occurrences of oil spills on the sea surface was proposed by K. Topouzelis, and S. Singha University of the Aegean, Mytilini, Greece German Aerospace Centre (DLR),

Bremen, Germany. The proposed SAR images analysis is based on five basic stages as described and (1) SAR pre-processing where images are prepared for the processing scheme; (2) ROI definition where parts of the processed image are defined to be further processed; (3) dark area detection where the borders of the possible oil spills are defined; (4) parameter extraction where characteristics of the detected dark areas are calculated; and (5) classification where dark objects are classified into oil spills or look-alikes. Apart from the ROI definition which requires human assistance, the rest stages are fully automatic. The processing chain using the proposed methodology and depending on the product type and the number of the detected dark areas, takes approximately 7–8 min to process on a 2.7 GHz quad core–based system for SM and 10–12 min for IW and EW products. The algorithm's processing speed performance is well under the NRT time requirement and allows sufficient time to transfer the data to end users. Results show us that the ANN classifier is sufficient for detecting oil spill on different imaging modes of Sentinel-1, as most of the oil spills were successfully separated from look-alikes using the above-mentioned characteristics. In the present work, oil spills were detected using Sentinel 1A images and dark areas were accurately classified using an object-based approach. It is concluded that Sentinel-1A products are reliable, accurate, and can successfully support oil spill detection from operational ship activities, due to the robustness of the SAR dataset and the quality of the imagery in the different modes. Despite the large numbers of available algorithms, no inter comparison can be made mainly because of the diverse size and accuracy of the training and testing datasets which have been used.

2.4 PROJECT TITLE: Oil spill mapping with sentinel 1.

AUTHOR NAME: By a public page.

PUBLISHED BY: RUS-Research and User Support for Sentinel Core Products. Ocean pollution due to oil spills remains a major environmental hazard. Although oil tanker accidents are well known, they are not the main cause for this type of event. Illegal discharges from ships or offshore platforms, drilling rigs, pipeline accidents or natural leaks amongst others bring together most of the sources for oil pollution in the ocean and a solution was proposed by RUS-Research and User Support for Sentinel

Core Products. 1) Data download – ESA SCIHUB In this step, we will download the Sentinel-2A level 2A image from the Copernicus Open Access Hub using the online interface. 2) SNAP – open and explore data Open SNAP (Applications -> Processing). To import the Sentinel-1 image, click File -> Open product (), navigate to the following path and open the product by double clicking on it. The opened product will appear in Product Explorer. Click+ to expand the contents of the file, then expand the Bands folder and double click on the Amplitude_VV band to visualize it. To reduce the processing time of the algorithm, we subset the image to our area of interest. Click on Raster -> Subset. In the Spatial Subset tab, set the following parameters in the Pixel Coordinates tab and click OK. Speckle filter will take place. The Oil spill profile plots will be plotted. Accordingly, mapping takes place. Ellipsoid correction after the oil spill detection is completed, we can reproject our data into a specific coordinate reference system. To perform this step, we will use the Ellipsoid Correction. Download/Upload Virtual Machine/Local Computer to download outputs from the RUS Virtual Machine to your local computer press Ctrl+Alt+Shift in your keyboard. In this way we can also download the results and make them use for future purpose.

2.5 PROJECT TITLE: An assessment of oil spill detection using sentinel 1 Sar-c image.

AUTHOR NAME: Sudhir Kumar Chaturvedi A, Saikat Banerjee b, Shashank Lelec.

PUBLISHED BY: Department of Aerospace Engineering, University of Petroleum and Energy Studies. The present examination studies an oil spill occurred in the Al Khaffi region by applying Sentinel 1 SAR-C images was proposed by the Sudhir Kumar Chaturvedi A, Saikat Banerjee b, Shashank Lelec a Department of Aerospace Engineering, University of Petroleum and Energy Studies. Currently, there is a wide-ranging phenomenon that radar polarimetry can give further data for conservational remote sensing applications. In addition, a few Earth Observation satellite missions will convey on board polarimetric SAR. The techniques used here are spreading, evaporation, dispersion, emulsification, dissolution, oxidation, sedimentation, biodegradation. Sentinel 1A and 1B information were utilized for oil slick examination. Distinctive polarization plans can be utilized for checking oil spills. Sentinel Application Platform (SNAP) programming is together created by Brockmann Consult and is

utilized to process Sentinel 1 Level 1 IW GRD images. The subset of the territory of intrigue is modified from the terrain-corrected image and changed over to db. Satellite-based remote sensing is a viable instrument for crisis circumstances where it is beyond the land of imagination to expect to perform quick and viable on-site estimation. It is also necessary to further study what reduces the noise level of the texture map of the intensity image effectively.

2.6 PROJECT TITLE: Marine oil spill detection using synthetic aperture radar over Indian Ocean.

AUTHOR NAME: Saima Naza, Muhammad Farooq Iqbala, Irfan Mahmooda, Mona Allamb.

PUBLISHED BY: Geo-Informatics Research Group, Department of Meteorology, COMSATS University Islamabad (CUI), Islamabad, Pakistan, Environment and Climate Change Research Institute. Four oil spill events over the Indian Ocean including Chennai, Sharjah, Al Khiran and Mubarak Village are analysed using Sentinel-1 satellite data and this was proposed by Saima Naza, Muhammad Farooq Iqbala, Irfan Mahmooda, Mona Allamb a Applied Geo - Informatics Research Group, Department of Meteorology, COMSATS University Islamabad (CUI), Islamabad, Pakistan, Environment and Climate Change Research Institute, The Indian Ocean is the third largest ocean and is connected with Pacific Ocean through world major shipping routes and this region is prosperous in seabed oil reservoir with massive oil exploitation. The Indian Ocean is the third largest ocean and is connected with Pacific Ocean through world major shipping routes and this region is prosperous in seabed oil reservoir with massive oil exploitation. The Persian Gulf is another part of the Indian Ocean and is located between 24° to 30° N and 48° to 56° E. It is a semi enclosed sea in arid region. In this study S-1 satellite data is utilized for oil spills detection from Torres Malinowski Schubert, Topouzelis and Singha. S-1 is the first space mission of European Space Agency (ESA), contains two satellite constellations S-1(A) and S1(B). After data acquisition, image pre-processing technique includes image sub setting, calibration, speckle filtering, multi-looking, ellipsoid correction and land/sea masking were applied. Oil spill trajectory production –The General National Oceanic and Atmospheric Administration (NOAA) Operational Modelling Environment

(GNOME) model is used for trajectories production for oil spill events and monitoring the stretch and movement of oil spills in different parts of the Indian Ocean. Oil spill weathering-Weathering processes of each oil spill event is checked using ADIOS model. The model was run with the appropriate weather parameters to check the weathering processes for all oil spills. In this study, an improved methodology for oil spills detection is utilized using S-1 SAR satellite data for four reported oil spill events in the Indian Ocean. Oil spill movement and weathering processes are analysed for four events in different time periods. Influence of wind and currents on oil spills spread movement and degradation are also checked for different oil types along with varying atmospheric conditions.

2.7 PROJECT TITLE: A deep learning based oil spill detector using Sentinel -1 SAR imagery.

AUTHOR NAME: Mayerle and Suman singha.

PUBLISHED BY: A public page. The Eastern Mediterranean Sea has been known as an oil pollution hotspot due to its heavy marine traffic and an increasing number of oil and gas exploration activities. To provide automatic detection of oil pollution from not only maritime accidents but also deliberate discharges in this region, a deep learning-based object detector was developed by Roberto Mayerle and Suman singha. Oil spills in different areas might have varying patterns due to different origins, meteorological conditions, currents and circulation systems. Visualization of study area with heat map of the amount of oil spills collected inside. The blue outline marks the study area. Sentinel-1 SAR data was first pre-processed with a series of corrections, and oil spills in the images were labelled as oil objects. Then the pre-processed results were cropped into smaller images to fit the image input size of the object detector. Deep learning-based object detection algorithms work in three essential stages: feature extraction, object localization and classification. The results obtained from training the deep learning model for oil spill detection. The comparison between predictions and ground truth can be summarized by a confusion matrix with four components: True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN). Oil spills near look alike were also distinguished well, but oil spills covered by look-alikes were not always clearly detected. It appeared that the model was misled as in these

cases, it detected the whole look-alike and the oil spill covered by it as one big oil spill. The results of this study are the basis of an improved oil spill detector, which lays the ground for building an efficient early warning system in the near future.

2.8 PROJECT TITLE: Oil spill detection by imaging radars.

AUTHOR NAME: Werner Alpers, Benjamin Holt, Kan Zeng.

PUBLISHED BY: Jet Propulsion Laboratory, California Institute of Technology. Institute of oceanography, University of Hamburg, Germany. Ocean Remote Sensing Institute, Ocean University of China. Criteria for discriminating between radar signatures of oil films and biogenic slicks visible on synthetic aperture radar (SAR) images of the sea surface as dark patches are critically reviewed and this was often claimed high success rate of oil spill detection algorithms using single-polarization SARs because the SAR images used to train these algorithms are based usually on subjective interpretation and are not validated by on-site inspections or multi-sensor measurements carried out from oil pollution surveillance planes. Furthermore, we doubt that polar metric parameters derived from fully polar metric SAR data, like entropy, anisotropy, and mean scattering angle, are beneficial for discriminating between mineral oil films and biogenic slicks. We critically review methods that are being applied to detect mineral oil films on radar images. Such images are acquired by real aperture radars (RARs) or synthetic aperture radars (SARs) from airplanes or satellites. Mineral oil spills become visible on radar images as dark areas relative to the surrounding area. Identifying oil films by the position/shape of the dark areas is the most viable method. The methods which have been proposed to discriminate between mineral oil films and biogenic (monomolecular) surface films that are based on differences in the statistical properties of the radar backscatter. The first method uses single-polarization SAR data, and the second one dual-polarization SAR data were also proposed. It is a highly demanding task to determine whether dark areas visible on SAR images result from mineral oil spills or oil spill look-alikes. Biogenic slicks are the most intricate oil spill look-alikes since their radar signatures can be quite similar to those of mineral oil films. Fully polar metric or quad-polar metric space borne SARs do not have the anticipated potential to significantly improve oil spill detection, particularly the separation of mineral oil from natural biogenic films. The same applies

for compact-polarimetric or hybrid-polarity spaceborne SARs. In order to improve oil spill detection using SAR, we recommend upgrading conventional oil spill detection algorithms using single-polarization SARs by including more ancillary information.

2.9 PROJECT NAME: Oil spill identification based on textural information of SAR images.

AUTHOR NAME: Fengli Zhang, Yun Shao, Wei Tian, Shiang Wang.

PUBLISHED BY: Institute of Remote Sensing Applications, Chinese Academy of Sciences. Texture information reflects the spatial variation of image, and especially useful for single channel SAR image. According to the characteristics of oil spill in SAR image we extract textural features and then use them for identification using SVM method and was proposed by Fengli Zhang, Yun Shao, Wei Tian, Shiang Wang Institute of Remote Sensing Applications, Chinese Academy of Sciences. Identification of oil spill in SAR image always includes the first and basic step detection of dark features. Texture information can reflect the spatial variation of image and may be especially useful for SAR image when there is only one frequency or one polarization information available. Here co-occurrence matrix is used for texture analysis. The statistical properties of patterns that jointly occur in the neighbourhood of each pixel and have been widely used in texture classification or texture segmentation. SVM is a classification method derived from statistical learning theory and can provide good classification results from complex and noisy data. Experiments on several SAR images show that textural information contained in SAR images is very useful for SAR image analysis. Oil spills in SAR image can be more distinctly displayed using textual feature combinations, and SVM method following textual analysis can make the detection of oil spills more effectively. If combined with morphological filtering and false alarm discrimination, oil spill detection based on textual information could be more satisfying.

2.10 PROJECT NAME: SAR images detection.

AUTHOR NAME: V. Karathanassi, P. Pavlakis, D. Rokos.

PUBLISHED BY: By a public page. Synthetic Aperture Radar (SAR) images are extensively used for the detection of oil spills in the marine environment, as they are not affected by local weather conditions and cloudiness. However, radar backscatter values for oil spills are very similar to backscatter values for very calm sea areas and other ocean phenomena was developed by K. Topouzelis a, V. Karathanassi, P. Pavlakis, D. Rokos. Neural networks have been employed to process remote sensing images and have achieved improved accuracy compared to traditional statistical methods. A single neuron can be compared with a multi variance linear regression model, which works without any a priori assumptions concerning the statistical nature of the data set. In the present study two different networks are tested in order to evaluate the one most suitable for oil spill detection: Multilayer Perceptron (MLP) and Radial Basis Function (RBF) networks. The method developed was applied on an ERS 1 image. The main aspects considered for oil spill detection using neural networks were data preparation, network architecture decision, parameters estimation and network performance accession. For both MPL and RBF neural networks an initial network topology was selected. The selection of the best suited topology for each NN was designed through the hill- climbing approach. The contractive algorithm was used, in which initial topology was the simplest one and nodes were added afterwards. The performance of each topology was evaluated, and the process was repeated iteratively until a predetermined stopping criterion was achieved. Neural network approach for oil spill identification was investigated using SAR images. Two types of neural network were used: the feed forward Multilayer Perceptron and Radial Basis Function. Original images and other images generated from them were used as n puts to a neural network. The method was tested on SAR image windows, containing oil spills. Further examination is needed using images containing different sea states and different types of oils spills. Moreover, the performance of other neural network types like Support Vector Machines (SVM) and Recurrent networks (like Hopfield) need to be investigated.

CHAPTER-3

EXISTING AND PROPOSED WORK

3.1 EXISTING WORK

In order to test the ability of automatic detection of oil spills by using satellite remote sensing, an image from Sentinel – 1 (A) was examined for the same date and time as EMSA report sent to Varna MRCC. The image was produced in Interferometric Wide. A software application based on MATLAB has been used to implement SAR image processing, oil spill detection and visualization.

SAR Image Processing

First the image was changed by an upside-down reflection (flipping) and shifting up to the maximum pixel value of the image:

$$UFlipI(i, j) = \text{MaxVal} - I(i, j), \text{ for each } i, j, \text{ -----(1)}$$

Where I (MXN) is the Sentinel – 1 SAR image with size MXN pixels, $UFlipI$ (MXN) – upside down flipped image,

And $\text{MaxVal} = \max(I(i, j))$, for each i, j ,

Is the maximum pixel value of the SAR image I (MXN).

After this changes the oils, spills become brighter on the background of sea surface and ships echoes change to dark spots, while background statistics remain unchanged because they usually have a Gaussian distribution. To reduce level of speckle noise and to enhance picture quality in order to make further detection of objects more accurate, the flipped image is pre-processed by using two types of moving window image processing: Neighbourhood – averaging (linear) filters and Median, or non – linear filters. The size of the window depends on the purpose of the application and on the image resolution. For oil spill applications, the pre-processing should suppress the noise, but still preserve small and thin oil spills. If, for example the SAR image includes a small oil spill with size of NXN pixel area and the windows size is also NXN, in a smoothing operation using a neighbourhood – averaging (linear) filter this spill will get

smoothed. But if the window is twice the size of the spill or larger, then the spill may even disappear.

Oil Spills Detection

To make an appropriate oil spill detection after the pre-processing, the CFAR detector is applied to adapt the threshold constantly to the mean value of the sea clutter return. CFAR processing compares the value of current pixels by a weighted-average value of the pixels covered by a window. A reference, or background window is formed of M cells that surround the cell under test. A guard interval is included around the tested cell in order to avoid objects that are close each other and could affect noise estimation. The multiplier T is a design parameter, influencing probability of false alarm together with an initial offset E0. Block diagram of basic (1D) CFAR detector.

2D CFAR processing compares the value of current cells by a weighted average of the pixels covered by the window. The algorithm uses 3 nested windows – background, guard and signal windows, as shown on right. According to the processing algorithm, statistics of signal window is compared to statistics of background window. As in 1D CFAR guard window is included in order to avoid influence of objects that are close each other.

Sizes of nested windows depend on image resolution. According to, for low resolution images sizes have to be as follows: 13X13 for background window, 5X5 for guard window and 3X3 for signal window. For high resolution images sizes have to be 25X25, 7X7 and 5X5 respectively. During the thresholding procedure the value of the current pixel is compared to the weighted mean value of the background pixels. If the background statistics have a Gaussian distribution, the detection test is:

$$0 \leq y \leq \mu_B + T \cdot \sigma_B + E_0$$

Where μ_B is the mean value of background, σ_B is the background standard deviation, E_0 is an initial value, set by operator and T is the design parameter.

To maximize spills detection abilities in low resolution images, the value of T should be within the interval (5.0 ÷ 5.5) and for high resolution images the value should be within (5.5 ÷ 6.0).

3.1.1 Block diagram of existing work

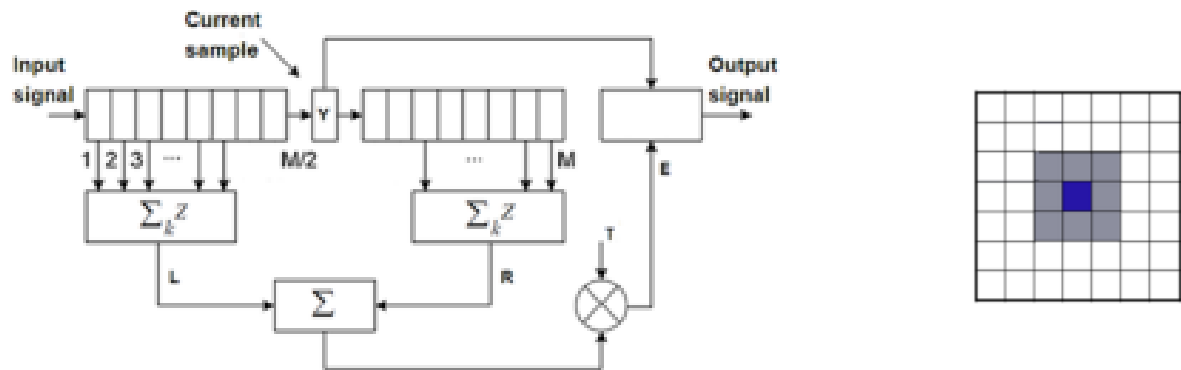


Fig: 3.1 1D CFAR (left) and nested (2D) CFAR (right) for radar image processing

3.2 PROPOSED WORK

3.2.1 Data Collection and Pre-processing

Data collection refers to the process of gathering information from various sources, which is then used to develop deep learning. The information should be saved in a way that is appropriate for the challenge. We collect data from Kaggle. We train and test our model using this dataset.

The oil spill dataset, oil slicks dataset, or just oil will be used for this research due to its widespread availability and familiarity as an unbalanced machine learning dataset. The satellite photographs of the ocean, some of which show an oil spill and some of which do not, served as the basis for the dataset's development.

The stage of data pre-processing, also known as data purification, is an important one. Prior to developing the model, most machine learning engineers spend a significant amount of effort on data pre-processing. Outlier detection, missing value treatments, and noisy data removal are some examples of things that can be done during the pre-processing stage of data management. In the same vein, the phrase "image pre-processing" refers to actions performed on images at the most fundamental level of abstraction. If information is measured in terms of entropy, then these actions do not

make the image more informative; rather, they make it less informative. The purpose of pre-processing is to improve the picture data by eliminating unwanted distortions or enhancing certain visual elements that are important for the subsequent processing and analysis tasks. This is the goal of the process.

3.2.2 Model Making and Training

CNN or ConvNets is one of the deep learning algorithms, designed to extract features of images which help in classifying images. Convolutional Neural Networks are a subcategory of feed-forward artificial neural networks. The connectivity pattern between the neuron in a convolutional neural network is modelled after the connectivity pattern found in the visual cortex. A relatively small number of cells in the visual cortex are located in an area that is particularly sensitive to visual fields. Some neurons respond as and when they are exposed to the vertical edges, whereas others respond when they are presented to the horizontal or diagonal edges. This is nothing more than the impetus behind convolutional neural networks. Convolutional Neural Networks, or convnets for short, are simply neural networks that share the parameters that define them with other neural networks.

Also, these networks have performed fairly in various deep learning competitions with very less error percentage. They also solve the problem generally observed in large networks that is with increase in depth of networks, saturation and degradation in accuracy start taking place. We have tried to employ the best practices for training the models. We have used cyclic learning rates that change their values cyclically between epochs and differential learning rates that change their values according to layers such that the training rate varies layer-wise as initial layers usually represent primitive features and inner layers represent high-level features. This helps in improving the accuracy greatly and reducing overfitting if any.

3.2.3 Creating A Web App Using Streamlit

The fields of data science and analytics are becoming increasingly popular as time goes on. The deployment of models is one of the most crucial processes in the data science pipeline. It is essential to have a nice user interface because if we merely show people code snippets, it will be too abstract for them to understand. People are able to obtain a better understanding of what is going on with the help of graphical

user interfaces. Everything is different once an interface is present because it makes it simpler for individuals to comprehend what is going on. Machine learning is characterized by its frequently abstract nature. Firstly, there is an input, then there is a workflow, and finally, there is the output. If you have a user interface, you will be able to demonstrate the effects of your actions in a straightforward manner.

3.2.4 Block diagram of proposed work

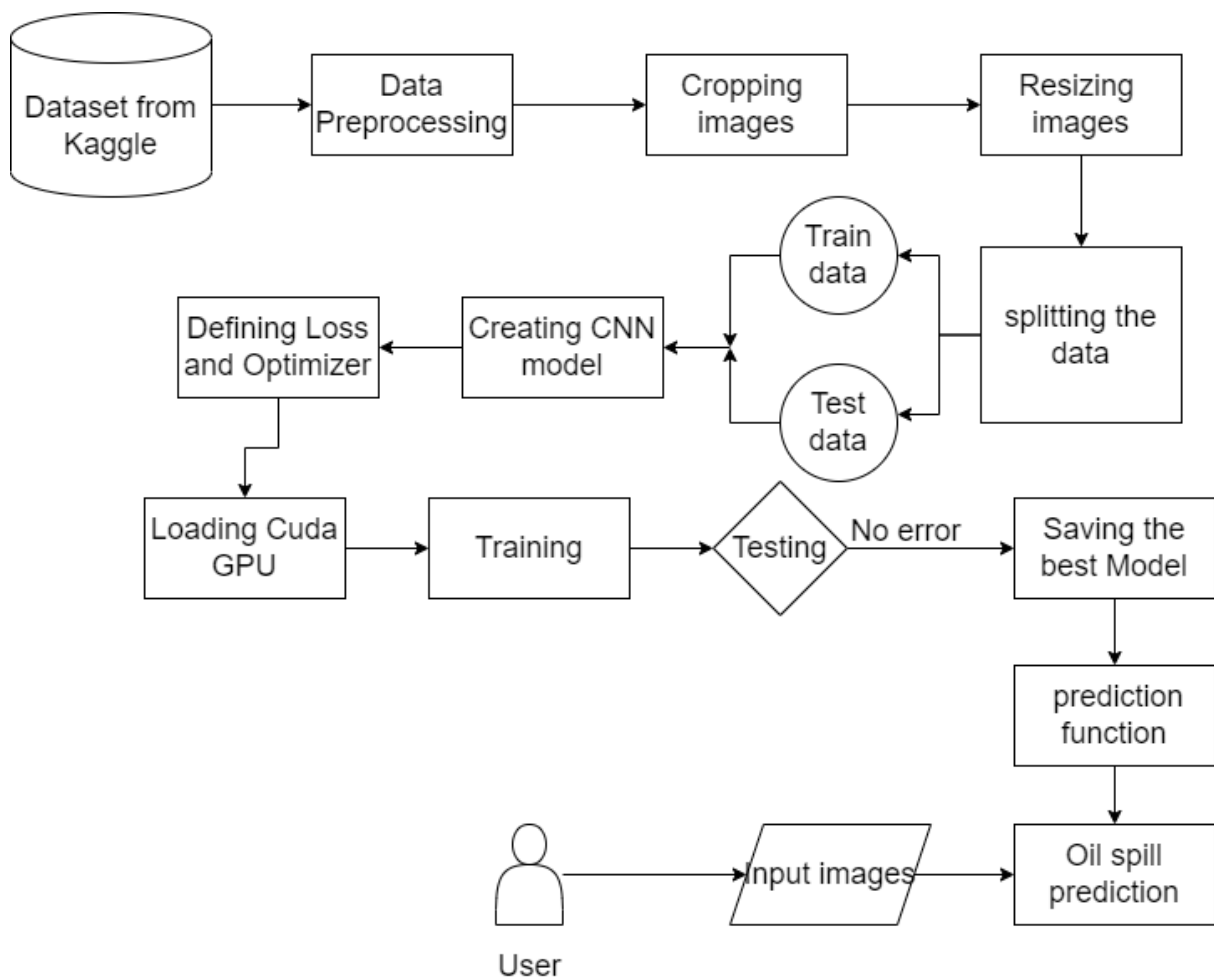


Fig: 3.2 Flow chart of proposed work

CHAPTER- 4

EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED

4.1 INSTALLATION PROCESS OF VISUAL STUDIO

Before you begin installing Visual Studio:

1. Check the system requirements. These requirements help you know whether your computer supports Visual Studio 2022.
2. Apply the latest Windows updates. These updates ensure that your computer has both the latest security updates and the required system components for Visual Studio.
3. Reboot. The reboot ensures that any pending installs or updates don't hinder your Visual Studio install.
4. Free up space. Remove unneeded files and applications from your system drive by, for example, running the Disk Clean up app.

STEP-1 Download Visual Studio

Next, download the Visual Studio bootstrapper file.

To do so, select the following button, choose the edition of Visual Studio that you want, and then save to your Downloads folder.

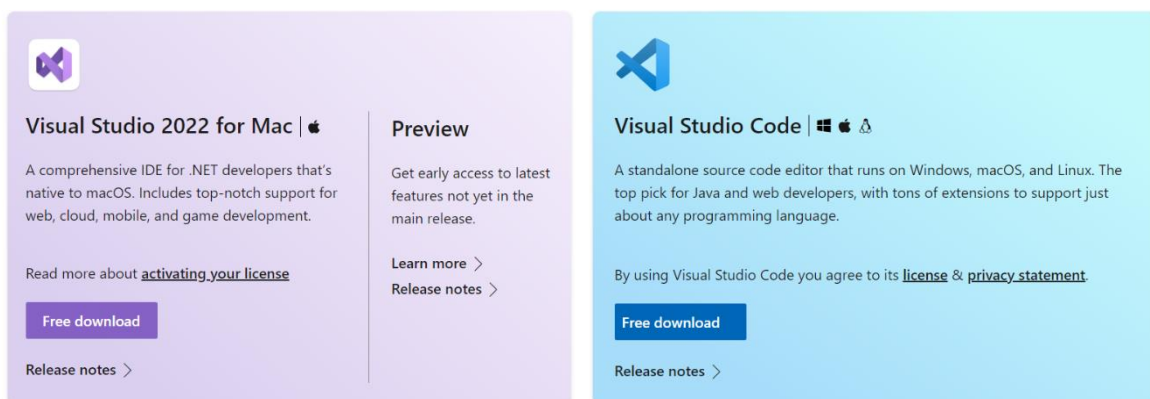


Fig: 4.1 Downloading the Visual Studio Code

STEP-2 Install the Visual Studio Installer

Run the bootstrapper file to install the Visual Studio Installer. This new lightweight installer includes everything you need to both install and customize Visual Studio. Acknowledge the Microsoft License Terms and the Microsoft Privacy Statement. Choose Continue.

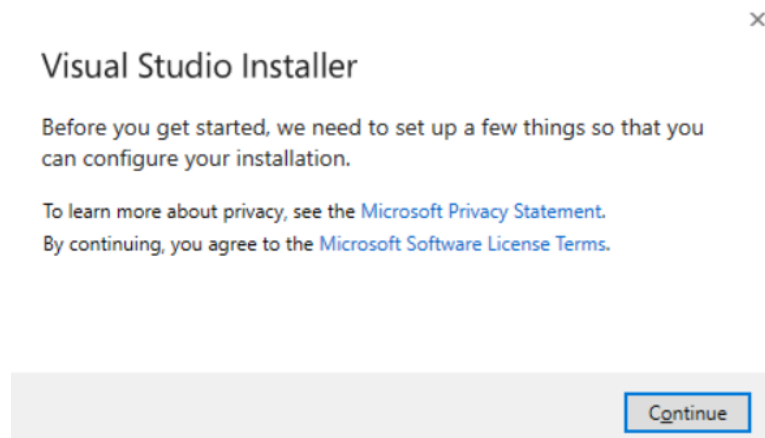


Fig: 4.2 Installing the Visual Studio Code

STEP-3 Choose workloads

After the installer is installed, you can use it to customize your installation by selecting the feature sets—or workloads—that you want.

Now install Python extension workload in Visual Studio.

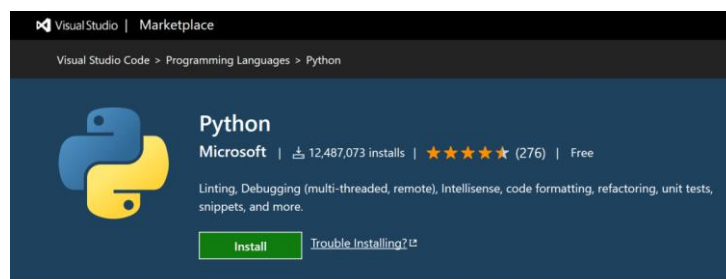


Fig: 4.3 Installation of python in Visual Studio Code

Install Jupyter Notebook workload in Visual Studio.

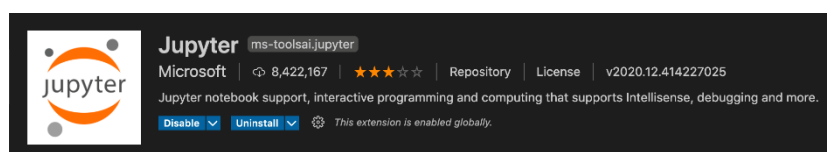


Fig 4.4 Installation of Jupyter Notebook in Visual Studio Code

4.2 CONVOLUTIONAL NEURAL NETWORKS METHOD

CNN is a type of deep learning model for processing data that has a grid pattern, such as images, which is inspired by the organization of animal visual cortex and designed to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns. CNN is a mathematical construct that is typically composed of three types of layers (or building blocks): convolution, pooling, and fully connected layers. The first two, convolution and pooling layers, perform feature extraction, whereas the third, a fully connected layer, maps the extracted features into final output, such as classification. A convolution layer plays a key role in CNN, which is composed of a stack of mathematical operations, such as convolution, a specialized type of linear operation. In digital images, pixel values are stored in a two-dimensional (2D) grid, i.e., an array of numbers, and a small grid of parameters called kernel, an optimizable feature extractor, is applied at each image position, which makes CNNs highly efficient for image processing, since a feature may occur anywhere in the image. As one layer feeds its output into the next layer, extracted features can hierarchically and progressively become more complex. The process of optimizing parameters such as kernels is called training, which is performed so as to minimize the difference between outputs and ground truth labels through an optimization algorithm called backpropagation and gradient descent, among others.

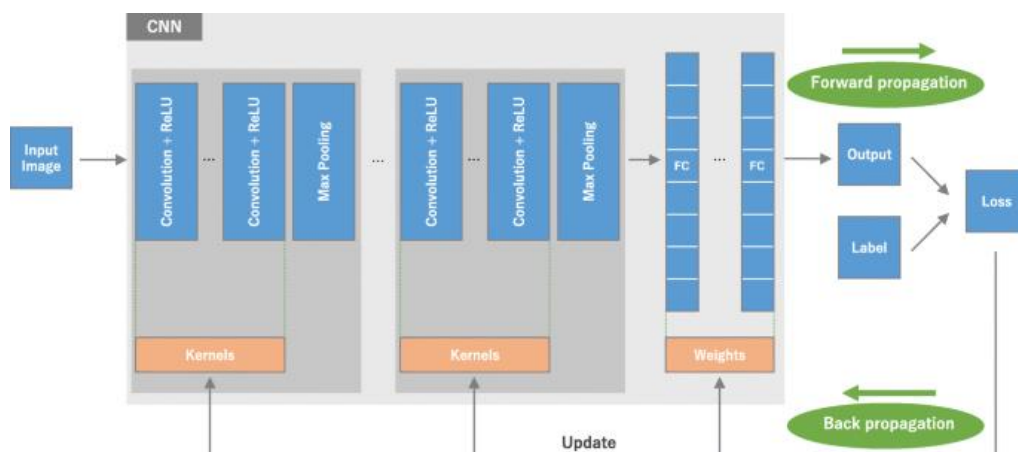


Fig.no 4.5 CNN architecture

An overview of a convolutional neural network (CNN) architecture and the training process. A CNN is composed of a stacking of several building blocks: convolution layers, pooling layers (e.g., max pooling), and fully connected (FC) layers. A model's performance under particular kernels and weights is calculated with a loss function

through forward propagation on a training dataset, and learnable parameters, i.e., kernels and weights, are updated according to the loss value through backpropagation with gradient descent optimization algorithm.

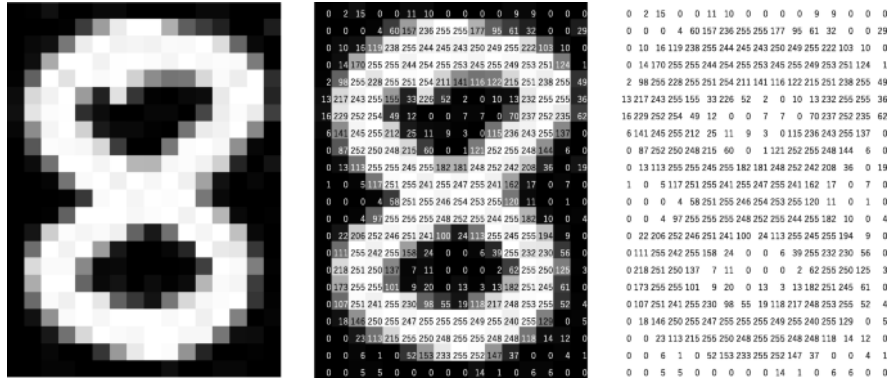


Fig 4.6 Image processing in CNN

4.3 ALGORITHM USED

MODULE 1: DATA COLLECTION AND PRE-PROCESSING

Data Collection

Data collection refers to the process of gathering information from various sources, which is then used to develop deep learning. The information should be saved in a way that is appropriate for the challenge. We collect data from Kaggle. We train and test our model using this dataset.

The oil spill dataset, oil slicks dataset, or just oil will be used for this research due to its widespread availability and familiarity as an unbalanced machine learning dataset. The satellite photographs of the ocean, some of which show an oil spill and some of which do not, served as the basis for the dataset's development. Computer vision algorithms were applied to the segmented images in order to derive a vector of characteristics that characterize the content of the image patch.

The dataset was constructed by beginning with satellite images of the ocean; some of the images show an oil spill, while others do not. Images were first patched together and then cut into sections before being processed with computer vision algorithms to produce a vector of features that described the contents of each image patch or

section. The objective of this task is to determine, given a vector that describes the contents of a patch of an image obtained from a satellite, whether or not the patch contains an oil spill, such as one that may have resulted from the unintentional or accidental dumping of oil in the ocean.

There are two different classes, and the objective is to differentiate between spill and non-spill using the characteristics of a particular ocean patch.

Non-spill: Case that does not spill over, also known as the majority class.

Oil spill: Positive case, or a member of a minority group.

```
[ ] oilspill = os.listdir(r"/content/drive/MyDrive/oilspill (1)/oilspill/dataset/oil_spill")
print(oilspill[:10]) # the output we get are the .jpg files

nospill = os.listdir(r"/content/drive/MyDrive/oilspill (1)/oilspill/dataset/no_spill")
print('\n')
print(nospill[:10])

['82864176_628860854349272_5353757726973906242_n.jpg', '117326751_296497461422762_7032561698141121852_n.jpg', '117334874_167149291548480_1869926664632554801_n.jpg', '122117459_32917936',
['137002886_3048161735402857_3726706303954180611_n.jpg', '136992735_396875208282222_1813019295886707905_n.jpg', '122194042_3420238221364951_2905020661909101747_n.jpg', '134528271_15312']
```

Fig 4.7 Dataset division into spill and no spill

Data pre-processing

The stage of data pre-processing, also known as data purification, is an important one. Prior to developing the model, most machine learning engineers spend a significant amount of effort on data pre-processing. Outlier detection, missing value treatments, and noisy data removal are some examples of things that can be done during the pre-processing stage of data management. In the same vein, the phrase "image pre-processing" refers to actions performed on images at the most fundamental level of abstraction. If information is measured in terms of entropy, then these actions do not make the image more informative; rather, they make it less informative. The purpose of pre-processing is to improve the picture data by eliminating unwanted distortions or enhancing certain visual elements that are important for the subsequent processing and analysis tasks. This is the goal of the process.

Data pre-processing is a data mining technique used to turn the raw data into a format that is both practical and effective. Data pre-processing changes the data into a format that can be processed in data mining, machine learning, and other data science tasks more quickly and efficiently.

To ensure reliable findings, the techniques are typically applied at the very beginning of the machine learning and AI development pipeline. Although there are many various tools and strategies for preparing data, we have chosen to concentrate on the following measures for our dataset:

Cropping The Images

PIL stands for the Python Imaging Library, and it is responsible for giving the Python interpreter the ability to modify images. A rectangular section of an image can be extracted with the help of the PIL. `Image.crop()` method.

Syntax: `PIL.Image.crop(box=None)`

Box: is a 4-tuple that defines the left, upper, right, and bottom pixel coordinates. It is one of the parameters.

Return Image: Image is the type of return. (Returns a rectangular region as a tuple consisting of the left, upper, right, and bottom sides.)

Returns: An instance of the Image class.

Resizing The Images

The Python Imaging Library, or PIL, gives the Python interpreter the ability to modify images. A class with the same name is provided by the Image module and is used to represent a PIL image. The module also offers a number of factory functions, including tools for generating new images and loading images from files.

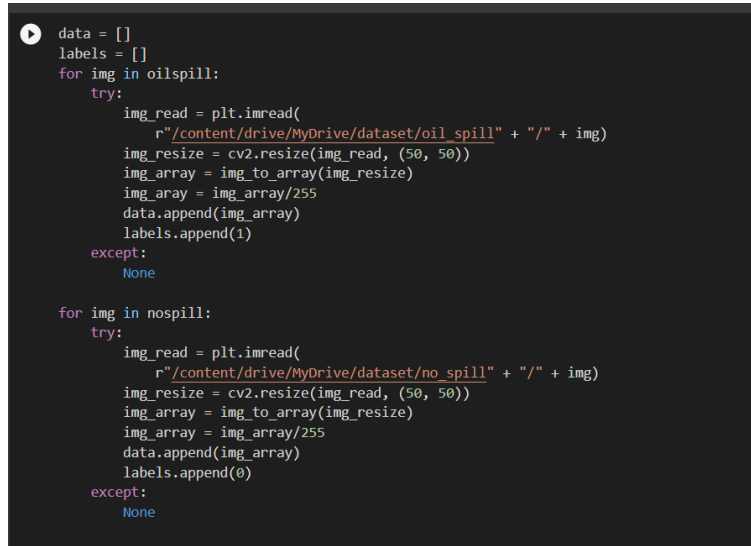
An image's new size is returned by the function `image.resize()`. The following parameters are taken into account:

Size: as a 2-tuple, represents the intended size in pixels. (Length, Width)

Resample: is a resampling filter that is optional. Among `PIL.Image.NEAREST` (use nearest neighbour), `PIL.Image.BILINEAR` (linear interpolation), `PIL.Image.BICUBIC` (cubic spline interpolation), and `PIL.Image.LANCZOS`, this can be used (a high-quality

down sampling filter). If it is left out or has the mode "1" or "P," it is set to PIL.Image.NEAREST.

Returns type: returns an object of type Image.



```
data = []
labels = []
for img in oilspill:
    try:
        img_read = plt.imread(
            r"/content/drive/MyDrive/dataset/oil_spill" + "/" + img)
        img_resize = cv2.resize(img_read, (50, 50))
        img_array = img_to_array(img_resize)
        img_array = img_array/255
        data.append(img_array)
        labels.append(1)
    except:
        None

for img in nospill:
    try:
        img_read = plt.imread(
            r"/content/drive/MyDrive/dataset/no_spill" + "/" + img)
        img_resize = cv2.resize(img_read, (50, 50))
        img_array = img_to_array(img_resize)
        img_array = img_array/255
        data.append(img_array)
        labels.append(0)
    except:
        None
```

Fig: 4.8 Resizing the images

Splitting The Dataset

The process of data splitting involves dividing the data into two or more parts. When there is a split into two parts, one of those parts is typically used to analyze or test the data, while the other portion is used to train the model. The process of data splitting is an essential component of data science, particularly for the development of models that are founded on data. This method contributes to the production of correct data models and the processes that use data models, such as machine learning, so helping to ensure their precision.

In a simple data split consisting of two halves, the training data set is the one that is utilized to create and train models. Training sets are frequently utilized in order to estimate a variety of parameters or to evaluate the overall performance of various models. After the training phase is complete, the testing data set is utilized. A comparison is made between the training data and the test data to ensure that the final model functions appropriately. Data is typically divided into three or more sets before being processed using machine learning. When there are three sets, the additional set is known as the dev set, and it is used to adjust the parameters of the learning process.

The train-test split is used to estimate the performance of machine learning algorithms suitable for prediction-based Algorithms/Applications. This method is a quick and simple procedure that allows us to compare our own machine learning model results to machine results. By definition, the Test set is made up of 30% actual data and the Training set is made up of 70% raw figures.

To assess how well our machine learning model performs, we must divide a dataset into train and test sets. The train set is used to fit the model, and its figures are recognized. The second set is known as the test data set, and it is only used for forecasts.

To divide our dataset into train and test, we use the following approach. The pandas and sklearn packages are imported. Sklearn is Python's most useful and robust machine learning library. The model selection module of the scikit-learn library includes the splitter function `train_test_split()`.

Further, the `read_csv()` method is then used to import the CSV file. The data frame is now stored in the variable `df`. Then we define the test size as 0.3 which means that 30% of our data is taken for testing and the rest 70% is used for training data. We also put the `random_state=0` so that data is randomly split into these two datasets.

```
[ ] from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(
        image_data, labels, test_size=0.2, train_size=0.8, random_state=42)
```

Fig: 4.9 Splitting of dataset

MODULE 2: MODEL MAKING AND TRAINING

Model Making

CNN or ConvNets is one of the deep learning algorithms, designed to extract features of images which help in classifying images. Convolutional Neural Networks are a subcategory of feed-forward artificial neural networks. The connectivity pattern between the neuron in a convolutional neural network is modelled after the connectivity pattern found in the visual cortex. A relatively small number of cells in the visual cortex are located in an area that is particularly sensitive to visual fields. Some neurons respond as and when they are exposed to the vertical edges, whereas others

respond when they are presented to the horizontal or diagonal edges. This is nothing more than the impetus behind convolutional neural networks. Convolutional Neural Networks, or convnets for short, are simply neural networks that share the parameters that define them with other neural networks. The CNN layers are explained below:

Input

The raw pixel values of an image ($[32 \times 32 \times 3]$) will be read from the input if the image has 32 widths and 32 heights and uses the R, G, and B colour spaces.

Convolution

It computes the output of the neurons linked to particular regions in the input volume, so that each neuron computes the dot product between its weights and the area of the input volume to which it is connected. Let's say we want to use 12 filters, which would give us a volume of $[32 \times 32 \times 12]$.

ReLU Layer

The max (0, x) thresholding at zero is one example of a common application of this technique, which is used to apply activation functions at the element level. It produces a result of $[32 \times 32 \times 12]$, which is equivalent to a volume that is unaltered from its original 32 cubic feet.

Pooling Layer

This layer is utilized in the process of performing a downsampling operation along the spatial dimensions (width, height), which ultimately produces a volume with the dimensions $[16 \times 16 \times 12]$. The Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model.

It is used to extract features that are invariant to rotation and position. Pooling can be categorized into two types i.e.

1. Max Pooling
2. Avg Pooling.

After the convolution and pooling layer, the model is enabled to understand and extract features from an image. Max Pooling returns the maximum value from the portion of

the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.

The Convolutional Layer and the Pooling Layer, together form the i -th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low levels details even further, but at the cost of more computational power. After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

Locally Connected Layer

It is possible to characterize it as a typical neural network layer that takes an input from the layer that comes before it, then computes the class scores, and finally produces a one-dimensional array whose size is proportional to the number of classes. This type of layer can be found in neural networks.

We will begin with an input image, which will then have numerous feature detectors—also known as filters—applied to it in order to generate the feature maps that are part of a Convolution layer. Then, on top of that layer, we will apply the Rectified Linear Unit, often known as ReLU, in order to either remove any linearity from our photos or enhance the amount of non-linearity that they contain.

Next, we will add a Pooling layer on top of our Convolutional layer. This will allow us to generate a Pooled feature map from each individual feature map. The primary objective of the Pooling layer is to ensure that there is spatial invariance in each of the images that we are working with. It helps to lower the size of our photographs and prevents any form of overfitting of our data. In addition to these benefits, it aids. Next,

we will feed these values into our artificial neural network, after which we will flatten all of our pooled photos into a single long vector or column containing all of these data. In the end, we will incorporate it into the layer that is connected locally in order to produce the final product.

Fully Connected Layer

It will learn non-linear features from the output of a convolution layer. For multi perception, that output should be converted to a column vector and fed to a feed-forward neural network with back propagation in every iteration of training. This helps the model extract the dominant and low-level features of an image.

Adding a Fully Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully Connected layer is learning a possibly non-linear function in that space.

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

In turn feature extraction from the model helps to identify and classify images. CNN with ReLu and Dropout ReLu (Rectified Linear Unit) overcomes the gradient problem by propagating a gradient effectively. The ReLu output is the same as the value of the input if it's positive or else its output is zero for negative images. Due to its prominent characteristics, ReLu is used in deep networks. Mathematically, ReLu is represented as shown in the below equation.

$$ReLU(P) = \begin{cases} 0 & \text{if } P < 0 \\ P & \text{if } P \geq 0 \end{cases} \quad (1)$$

Dropout refers to ignoring hidden or visible units during training for a few sets of neurons by considering them randomly. The units that are ignored will be eliminated

in the next layers for forward and backward propagation. Dropout is necessary to avoid inter-dependency between units which over-fit the model.

```
inputShape = (50, 50, 3)
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=inputShape))
model.add(MaxPooling2D(2, 2))
model.add(BatchNormalization(axis=-1))
model.add(Dropout(0.2))

model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(2, 2))
model.add(BatchNormalization(axis=-1))
model.add(Dropout(0.2))

model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(2, 2))
model.add(BatchNormalization(axis=-1))
model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(512, activation='relu'))
model.add(BatchNormalization(axis=-1))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))
```

Fig: 4.10 Convolution of the image

Creating a RESNET Model

Also, these networks have performed fairly in various deep learning competitions with very less error percentage. They also solve the problem generally observed in large networks that is with increase in depth of networks, saturation and degradation in accuracy start taking place. We have tried to employ the best practices for training the models. We have used cyclic learning rates that change their values cyclically between epochs and differential learning rates that change their values according to layers such that the training rate varies layer-wise as initial layers usually represent primitive features and inner layers represent high-level features. This helps in improving the accuracy greatly and reducing overfitting if any.

Defining loss function and optimizer

Loss function and optimizer are two most deciding factors for a model being a great model or an ill-trained model, we have used Binary Cross Entropy Loss as Loss function and Adam as optimizer.

The term "Loss Function" refers to the function that is responsible for computing this inaccuracy. As a result of this, the performance of the model is significantly impacted by the choice of loss function, as different loss functions will provide unique errors for the same prediction. The mean square error is one of the loss functions that is used the most frequently. This function works by calculating the square of the difference between the actual value and the anticipated value. When dealing with different kinds of tasks, such as regression and classification, different loss functions need to be utilized.

Error can be understood as a function of the model's internal parameters, which include weights and bias. It is necessary to reduce the amount of estimated error in order to make accurate predictions. Back propagation is the method that is used to accomplish this in neural networks. Typically, the current error is "propagated backwards" to an earlier layer, where it is utilized to make adjustments to the weights and bias in order to ensure that the error is reduced to its smallest possible value. A function known as the Optimization Function is used to make adjustments to the weights.

Now looking for measurements or a function that can be apply to improve the performance of the model. The loss function indicates the accuracy of the model's predictions. Loss will be at its lowest if model projections are closest to the actual values, and at its highest if predictions are farthest from the original values. Then it should be best served by a binary cross entropy loss function because the model is based on CNN.

The actual class output, which can only be either 0 or 1, is compared to each of the projected probabilities using binary cross entropy. The score that penalizes the probabilities based on how far they are from the predicted value is then calculated. Depending on how near or far the value is to the actual value.

Next, Adam optimizer was used. An algorithm for gradient descent minimization is called adaptive moment estimation. When dealing with complex problems involving a lot of data or factors, the strategy is incredibly effective. It is effective and uses little

memory. It combines the "gradient descent with momentum" algorithm and the "RMSProp" algorithm, intuitively.

```
[ ] # compile the model
    model.compile(loss='binary_crossentropy',
                  optimizer='Adam', metrics=['accuracy'])
```

Fig: 4.11 Optimisation

Preparing CUDA GPU

The alternative paradigm is many-core processors, which are designed to operate on enormous chunks of data. This is a scenario in which CPUs are inefficient, hence many-core processors are the better option. A graphics processing unit (GPU) is made up of many cores, the number of which almost doubles every year, and the clock speed of each core is substantially lower than that of a central processing unit (CPU). The execution throughput of massively parallel applications is the primary emphasis of graphics processing units (GPUs). For instance, the Nvidia GeForce GTX 280 GPU contains 240 cores, each of which is a heavily multithreaded, in-order, single-instruction issue processor (SIMD - single instruction, multiple-data) that shares its control and instruction cache with seven other cores. In this configuration, the GPU can perform a variety of tasks simultaneously. GPUs have maintained their position at the head of the pack for some time now with regard to the overall number of floating-point operations performed per second (FLOPS).

Because deep learning requires the same kinds of calculations that GPUs were built to do, deep learning is particularly well suited to GPUs. When we conduct any operation, like a zoom in effect or a camera rotation, what we are really doing is applying some mathematical transformation to a matrix, which is how images, videos, and other graphics are recorded.

In actuality, this means that GPUs are more adept at performing matrix operations and a variety of other types of complex mathematical transformations than central processing units (CPUs) are. Deep learning algorithms now run on a GPU several times quicker than they would on a CPU. Learning periods can frequently be cut down from days to just a few hours. Therefore, we train our model using our GPU.

Make sure that Nvidia drivers are up to date first. And also install cudatoolkit directly from their page. Install Anaconda next, adding it to the environment as to go. The following commands are entered into the command line after all installs have been completed.

Training The Model

Conda install numba & conda install cudatoolkit is the appropriate code. Then, run the normal function on the CPU on the training dataset. Then, make a function optimized to run on the GPU. In the end, desired to get the results of time with GPU and without GPU. In the model, training the model with GPU provides greater efficiency and performance and speed.

Software may use the CPU and GPU simultaneously thanks to the parallel computing framework called CUDA, which was created by NVIDIA. We use this because NVIDIA is now the most popular GPU provider for cloud computing and machine learning. Also, NVIDIA GPUs can be used with the majority of GPU-enabled Python languages. We load the data on the CUDA GPU. We wish to move our data from the CPU to the GPU RAM during the training period because the system has CUDA. Therefore, if it can be enable pin memory=True, the data will be transferred into page-locked memory, which will speed up training. That's it; the data are now prepared for training. Next, load the data in batches.

Batch processing is a technique for repeatably processing large amounts of data. When computational resources are available, users can process data with little or no human intervention using the batch technique. Batch processing is the method by which a computer completes batches of work in a nonstop, sequential manner, typically concurrently.

It's also a command that ensures large jobs are broken down into smaller chunks for easier debugging. In the model, that is to do this by inserting records in chunks or batches and commit per chunk or batch. The data is transferred to the notebook and then a batch size is set. With the selected batch size, a data loader is created.

Here is the syntax for the same:

```
batch_size = ""  
train_loader=DataLoader()  
val_loader=DataLoader()
```

Finally, create a function to train and validate the dataset. Keep in mind that must move the model to the GPU after it is created; otherwise, the model will be running on the CPU and our data, which we have already moved, will be on the GPU.

```
[ ] def plot_confusion_matrix(y_true, y_pred, classes,  
                             normalize=False,  
                             title=None,  
                             cmap=plt.cm.Blues):  
    from sklearn.metrics import confusion_matrix  
    from sklearn.utils.multiclass import unique_labels  
    """  
    This function prints and plots the confusion matrix.  
    Normalization can be applied by setting 'normalize=True'.  
    """  
    if not title:  
        if normalize:  
            title = 'Normalized confusion matrix'  
        else:  
            title = 'Confusion matrix, without normalization'  
  
    # Compute confusion matrix  
    cm = confusion_matrix(y_true, y_pred)  
    # Only use the labels that appear in the data  
    classes = classes[unique_labels(y_true, y_pred)]  
    if normalize:  
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]  
        print("Normalized confusion matrix")  
    else:  
        print('Confusion matrix, without normalization')  
  
    print(cm)  
  
    fig, ax = plt.subplots()  
    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)  
    ax.figure.colorbar(im, ax=ax)  
    # We want to show all ticks...  
    ax.set(xticks=np.arange(cm.shape[1]),  
          yticks=np.arange(cm.shape[0]),  
          # ... and label them with the respective list entries  
          xticklabels=classes, yticklabels=classes,  
          title=title,  
          ylabel='True label',  
          xlabel='Predicted label')  
  
    # Rotate the tick labels and set their alignment.  
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",  
             rotation_mode="anchor")  
  
    # Loop over data dimensions and create text annotations.  
    fmt = '.2f' if normalize else 'd'  
    thresh = cm.max() / 2.  
    for i in range(cm.shape[0]):  
        for j in range(cm.shape[1]):  
            ax.text(j, i, format(cm[i, j], fmt),  
                    ha="center", va="center",  
                    color="white" if cm[i, j] > thresh else "black")  
    fig.tight_layout()  
    return ax
```

Fig: 4.12 Training the dataset

MODULE 3: CREATING A WEB APP USING STREAMLIT

The fields of data science and analytics are becoming increasingly popular as time goes on. The deployment of models is one of the most crucial processes in the data science pipeline. It is essential to have a nice user interface because if it is merely show people code snippets, it will be too abstract for them to understand. People are able to obtain a better understanding of what is going on with the help of graphical user interfaces. Everything is different once an interface is present because it makes it simpler for individuals to comprehend what is going on. Machine learning is characterized by its frequently abstract nature. Firstly, there is an input, then there is a workflow, and finally, there is the output. If you have a user interface, you will be able to demonstrate the effects of your actions in a straightforward manner.

When it comes to deploying the model, Python provides us with a wide variety of possibilities. Flask and Django are two examples of well-known frameworks. The problem with utilizing these frameworks, however, is that is needed to have some familiarity with HTML, CSS, and JavaScript in order to do so. Adrien Treuille, Thiago Teixeira, and Amanda Kelly came up with the idea for the game "Streamlit" while keeping these requirements in mind.

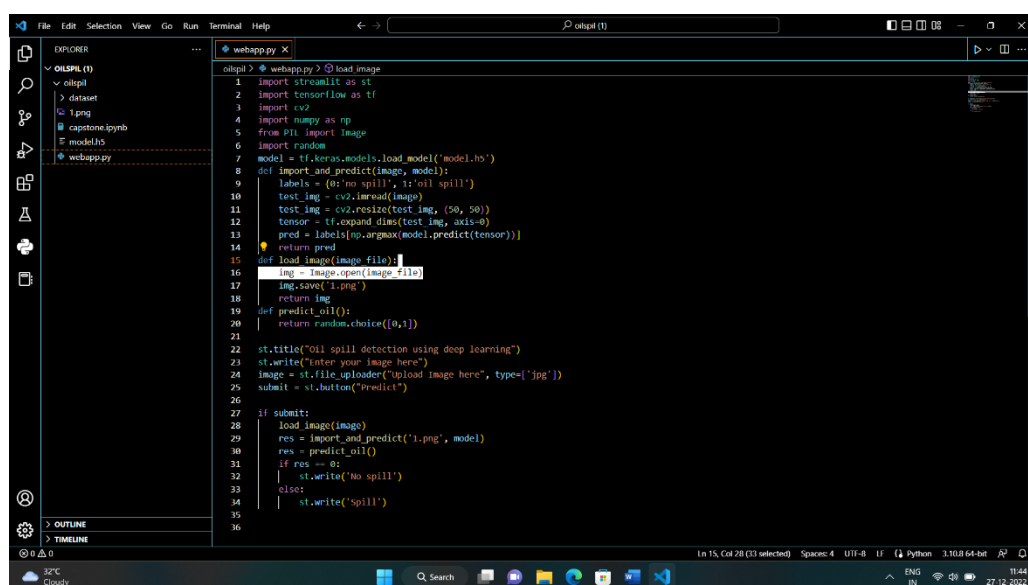


Fig: 4.13 creating a Webpage

Now that have a completely trained model and have saved the weights, it can develop a web application using Streamlit to give the model a user interface (UI). An open-source Python framework called Streamlit is used to create web applications for machine learning and data science.

Using Streamlit, can quickly design and launch web applications. Working on the interactive cycle of coding and watching outcomes on the web app is made simple by Streamlit. Use Streamlit to create apps using Python code. Streamlit gives us the ability to create code for an application in exactly the same way that you would write Python code. Working on the interactive cycle of coding and viewing outcomes on the web app is made to be a smooth experience thanks to Streamlit.

First, install Streamlit using the pip install streamlit command on python. Following this, we generate a python script containing streamlit commands, then run such script using the command streamlit run <ourscript.py>. There are numerous widgets available in streamlit, that was used like st.selectbox, st.checkbox, st.slider, and etc.

Using the st.sidebar function, it can easily arrange the widgets or data. This strategy assisted us in aligning data in the sidebar on the left panel. To display a selectbox in the left panel, it is simply used st.sidebar.selectbox.

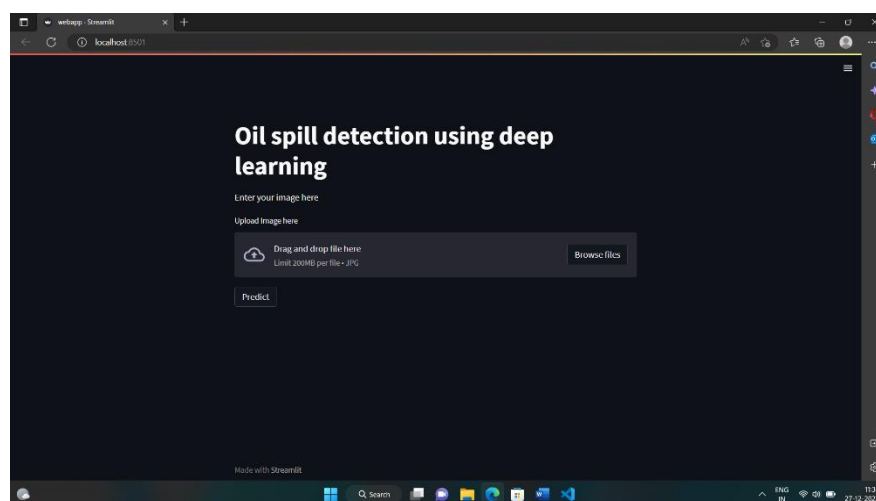


Fig: 4.14 Webpage design for Oil spill detection

Then, in the web app, we use streamlit to create a title and a DataFrame. Now, in the console, type the following command: `streamlit run streamlit app.py`. After the command is executed, so that to get an URL. Clicking on the URL takes us to the webpage or web app.

CHAPTER – 5

Results and Discussion, Performance Analysis

5.1 RESULT

Hence, the data is collected and saved in a data set. The code has been written and analysed through google Collaboratory application. The web page is designed to get the results. There are two types of data stored in the data set one is oil spill and the other is the non-oil spill. This data is trained for a several times using a deep learning algorithm and then tested by the deep learning algorithm. Then the tested data is used as the true data and used to train the data for the further purposes. When we upload an image into the web page then by the algorithm which we developed reads the image and gives the respective result for the sea surface.

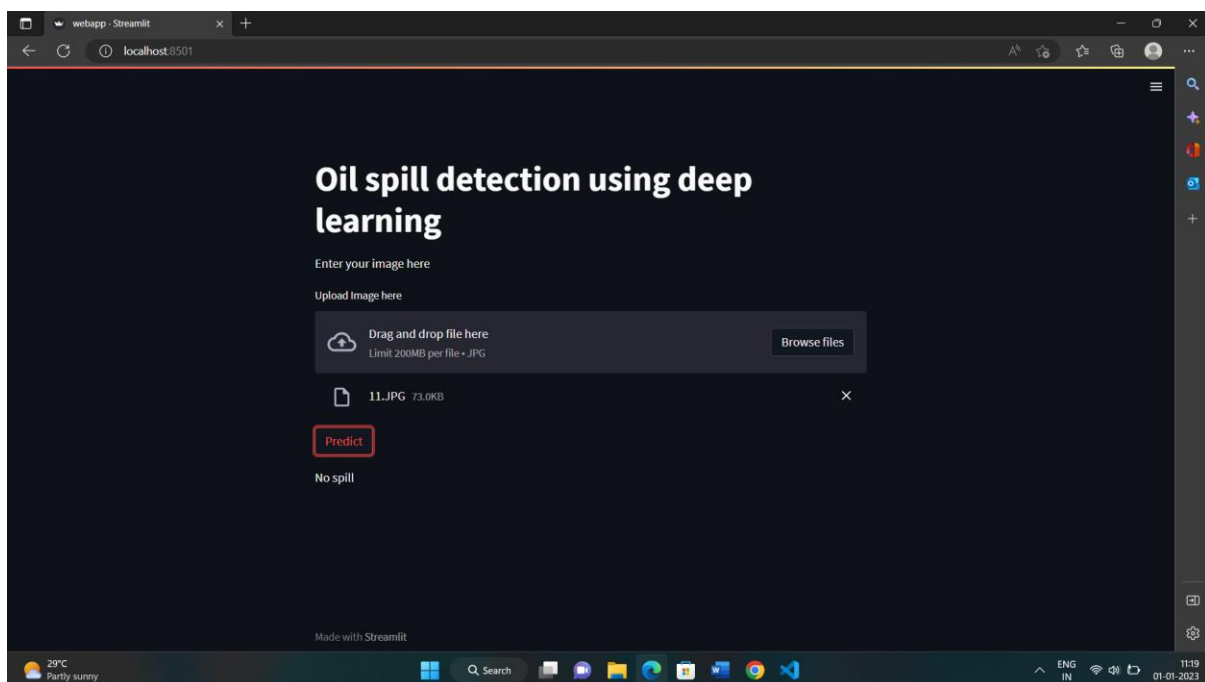


Fig: 5.1 No oil spill has been detected

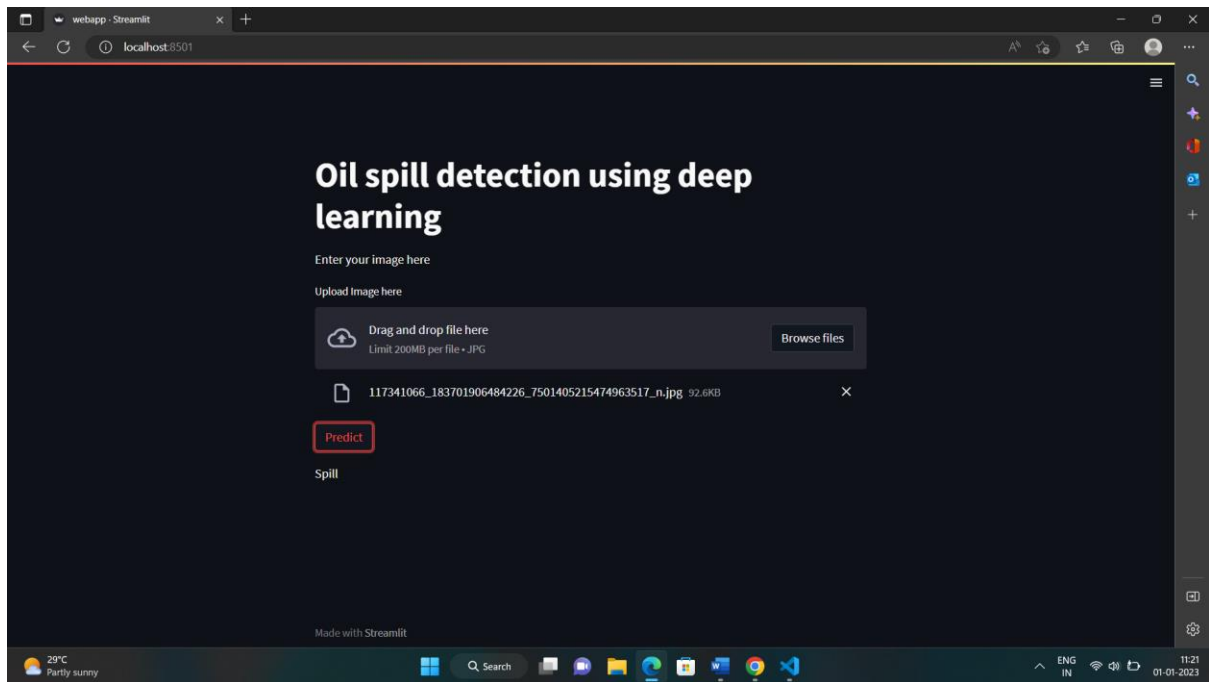


Fig: 5.2 Oil spill has detected

OIL SPILL AREAS THROUGH OUT THE WORLD

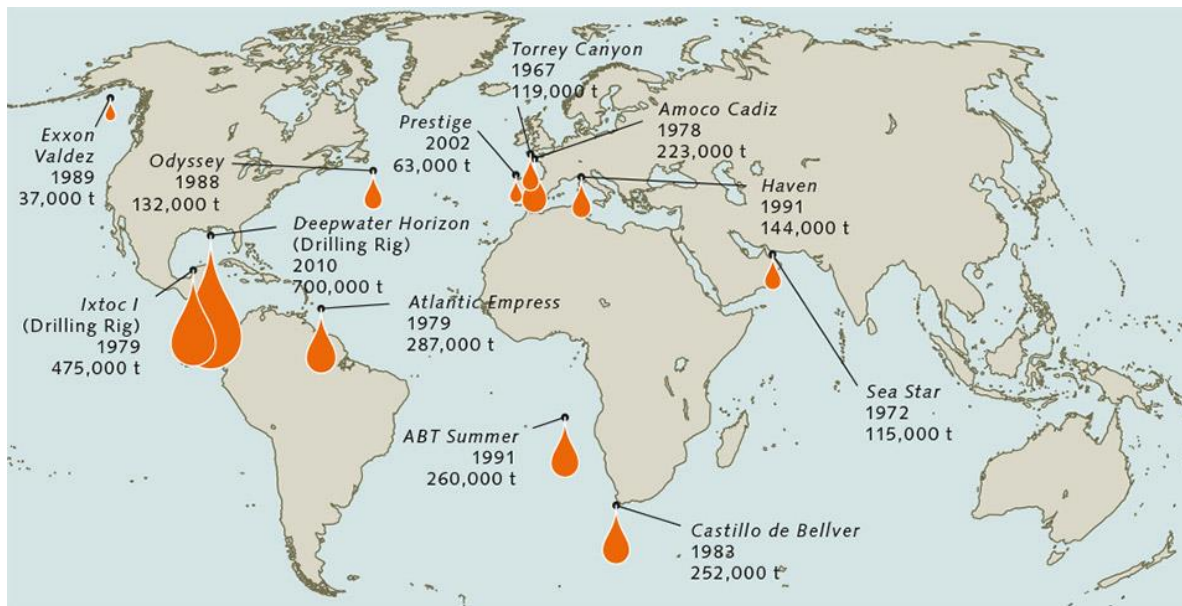
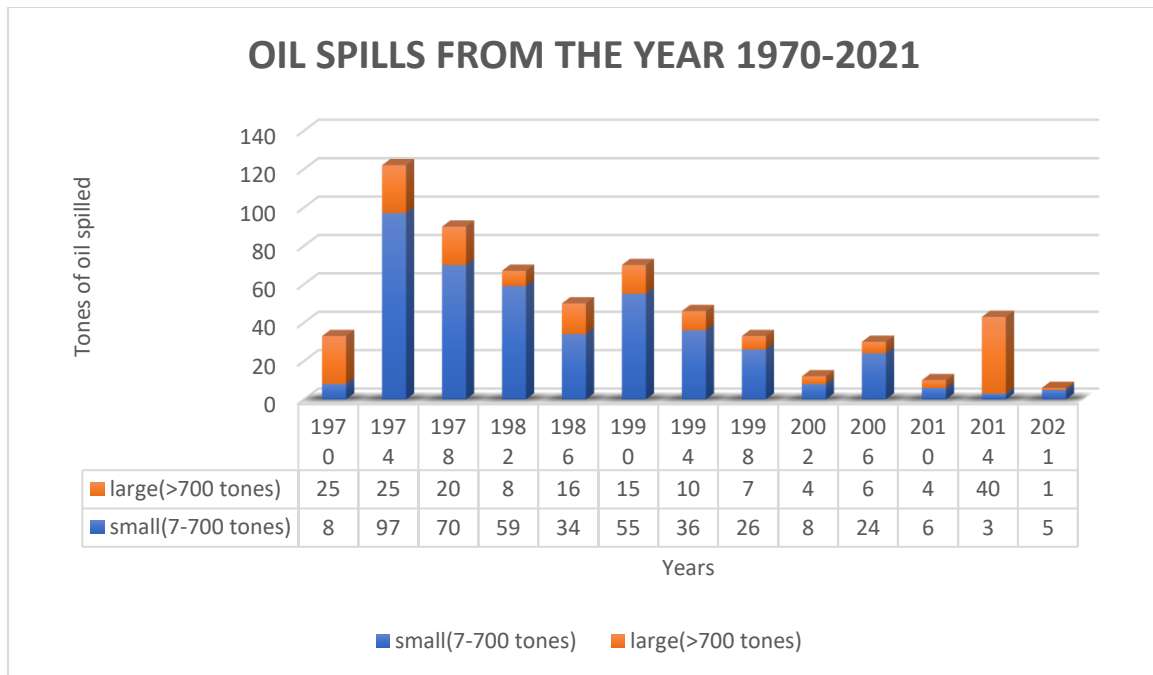


Fig: 5.3 Oil spill areas in the world



Graph: 5.1 Oil spills from the year 1970-2021

5.2 DISCUSSIONS

These choices must be made in all applications, and they undoubtedly have a profound effect on the success, or appropriateness, of learning. Yet the exact nature of this effect is unknown, and a systematic study of these aspects is needed.

The first decision concerned granularity. In our application three different approaches are possible. One of them works with the whole image and its output simply states whether the given image contains an oil slick. The second approach works with the dark regions detected in the images and provides the user with coordinates of those regions that are considered as oil spills. Finally, the third approach classifies individual pixels ("this pixel is part of an oil slick"). The approach operating with pixels represents the finest granularity, whereas the approach operating with images represents the coarsest granularity.

In this project we have used the third approach that is operating with pixels that represent the finest granularity. Finer granularity provides more examples, compare the millions of pixels, with the 937 regions and 9 images. Moreover, in our application a higher misclassification rate can be tolerated at the pixel level. 80% accuracy on the pixels in oil slicks is likely to identify more than 80% of the oil slicks. On the other hand,

pixels can be described only with an impoverished set of features, and the result need not necessarily seem coherent to the user (e.g., if pixels are classified individually there is no guarantee that the "oil slick" pixels will form coherent regions in an image). We decided the system would classify regions.

Our decision to classify regions had important consequences for the general design of the system. Together with the granularity of the interface, which according to Macdonald Detwiler's requirement was at the image level, it has constrained the options concerning the output of our system. It could not have ranked regions according to their probability of being an oil spill, because the unit of output was an image. It could not have classified images, because the system was to support decisions about regions, rather than images.

Now classify the images based on the confusion matrix which gives the matrix for "TRUE" and "PREDICTED" values.

		PREDICTED:	
		Positive	Negative
TRUE	Positive	A	B
	Negative	C	D

Table 5.1 Confusion matrix

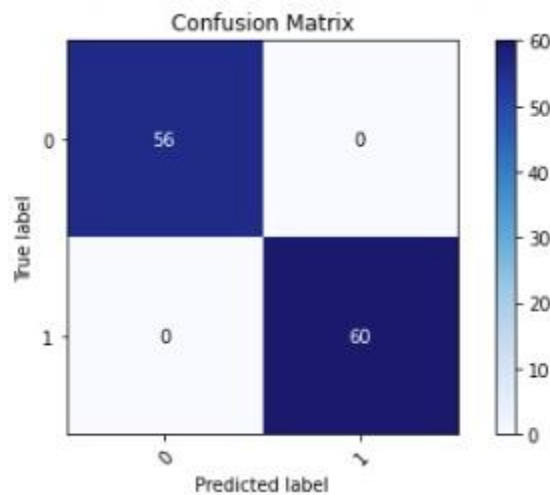
In this confusion matrix the A, B, C and D represents the following:

A → True positive: The number of times our actual positive values are equal to the predicted positive. You predicted a positive value, and it is correct.

B → False negative: The number of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.

C → False positive: The number of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.

D → True negative: The number of times our actual negative values are equal to predicted negative values. You predicted a negative value, and it is actually negative.

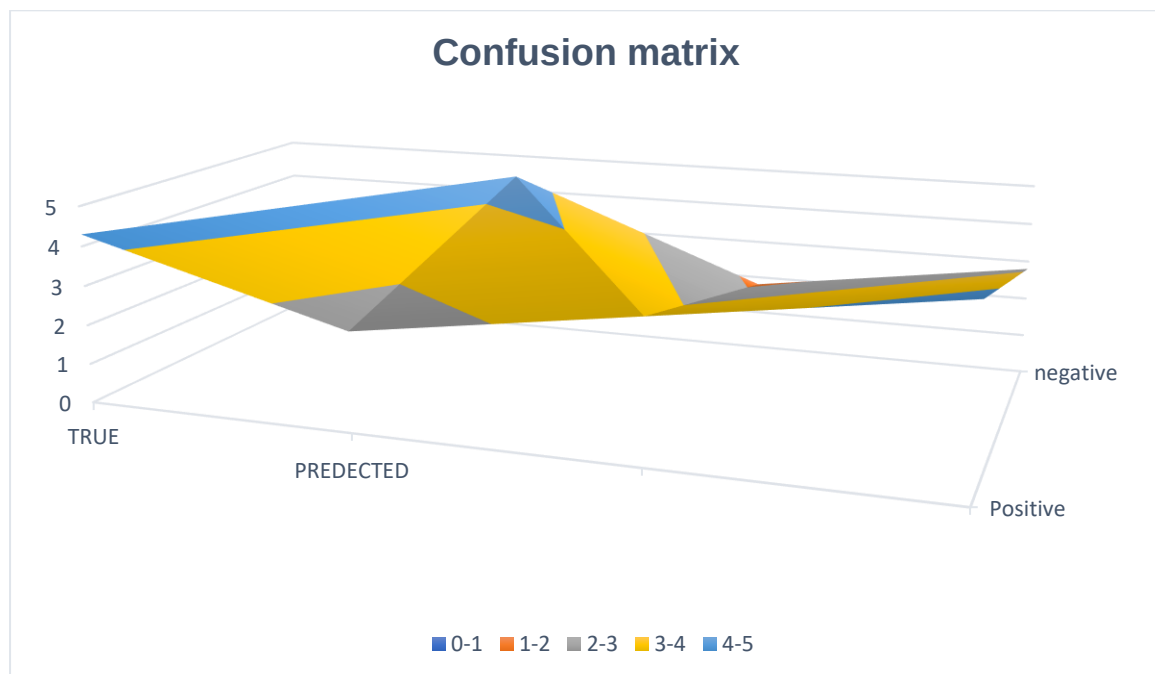


Graph: 5.2 Confusion matrix

By the Fig.no 5.1 we can notice that the ratio of the true and predicted values is 15:14 where, the true value is 15 and the predicted value is 14.

The accuracy of true and predicted values is known by the formula:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$



Graph: 5.3 Confusion matrix analysis

5.3 TRAINED AND TESTED DATA

Supervised Learning (SL) is intended as the way an ML model is trained by a human being. This means that the human (the Data Scientist, the Machine Learning Engineer, or whatever it is) trains the ML model in a set of known data.

Typically, machine learning algorithms accept parameters that can be used to control certain properties of the training process and of the resulting ML model. In Amazon Machine Learning, these are called *training parameters*. You can set these parameters using the Amazon ML console, API, or command line interface (CLI). If you do not set any parameters, Amazon ML will use default values that are known to work well for a large range of machine learning tasks.

You can specify values for the following training parameters:

- Maximum model size
- Maximum number of passes over training data
- Shuffle type
- Regularization type
- Regularization amount

In machine learning, model testing is referred to as the process where the performance of a fully trained model is evaluated on a testing set.

typical software testing suite will include:

- *Unit tests*: which operate on atomic pieces of the codebase and can be run quickly during development,
- *Regression tests*: replicate bugs that we've previously encountered and fixed,
- *Integration tests*: which are typically longer-running tests that observe higher-level behaviors that leverage multiple components in the codebase.

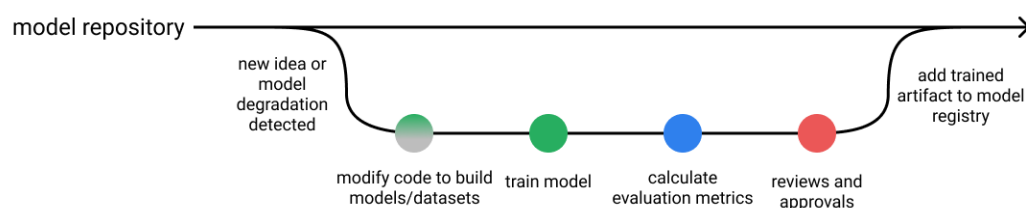
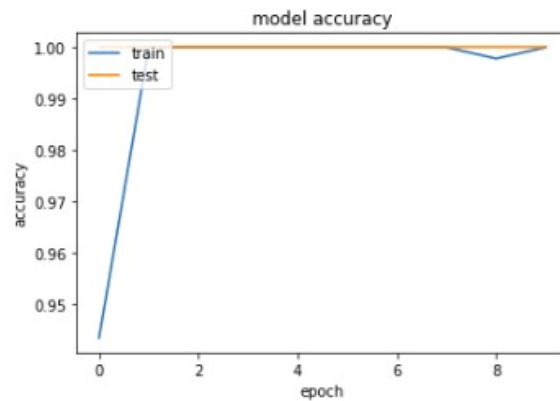


Fig: 5.4 A typical workflow of a model development



Graph 5.4 Model accuracy

Model loss

Loss is the penalty for a bad prediction. That is, **loss** is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater. The goal of training a model is to find a set of weights and biases that have *low* loss, on average.

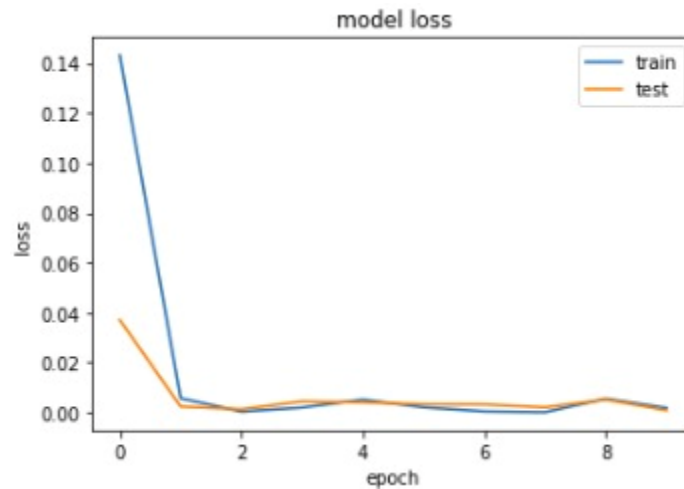
Mean square error (MSE) is the average squared loss per example over the whole dataset. To calculate MSE, sum up all the squared losses for individual examples and then divide by the number of examples:

$$MSE = \frac{1}{N} \sum_{(x,y) \in D} (y - prediction(x))^2$$

where:

- (x, y) is an example in which
 - x is the set of features (for example, chirps/minute, age, gender) that the model uses to make predictions.
 - y is the example's label (for example, temperature).
- prediction(x) is a function of the weights and bias in combination with the set of features x.
- D is a data set containing many labelled examples, which are (x,
- y) pairs.

- N is the number of examples in D .



Graph: 5.5 Model loss

5.4 PERFORMANCE ANALYSIS WITH CNN MODEL

5.4.1 ACCURACY COMPARISON BETWEEN CNN AND ANN

With ANN, concrete data points must be provided. For example, in a model where we are trying to distinguish between oil spill and no oil spill in a sea or ocean surface by using pixels an image where there are dark curve pixels are noticed as the oil spill and remaining area is considered as the non-oil spill area.

When using CNN, these spatial features are extracted from image input. This makes CNN ideal when thousands of features need to be extracted. Instead of having to measure each individual feature, CNN gathers these features on its own.

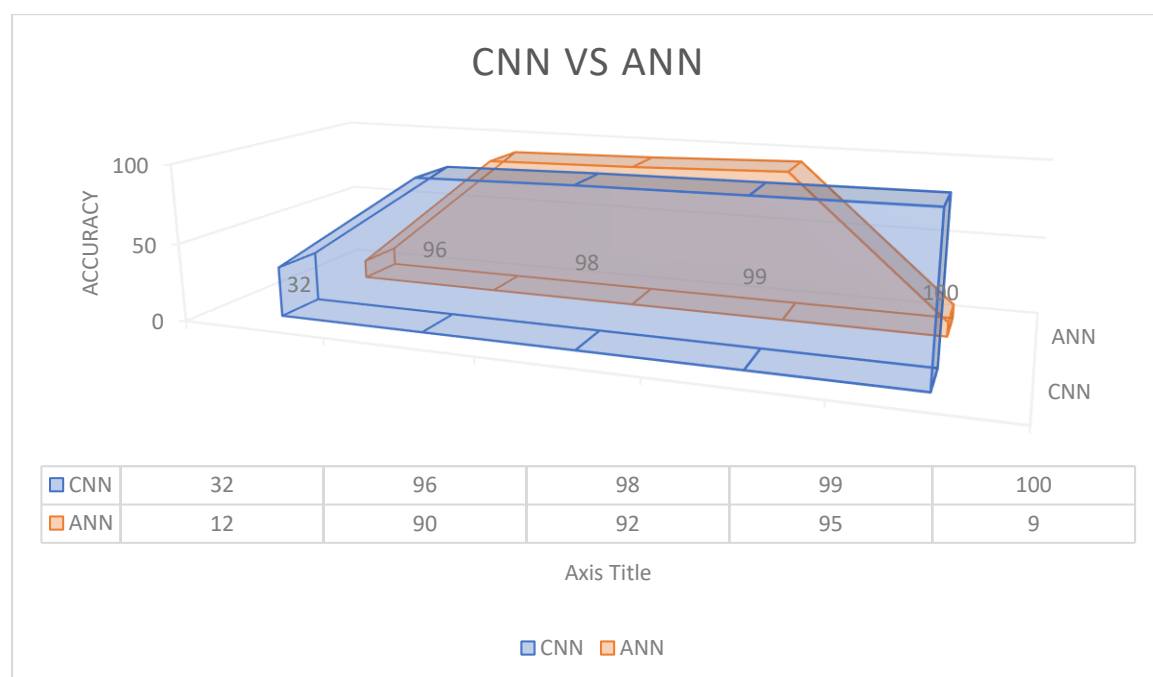
Using ANN, image classification problems become difficult because 2-dimensional images need to be converted to 1-dimensional vectors. This increases the number of trainable parameters exponentially. Increasing trainable parameters takes storage and processing capability.

In other words, it would be expensive. Compared to its predecessors, the main advantage of CNN is that it automatically detects the important features without any human supervision. Therefore, CNN would be an ideal solution to computer vision and image classification problems.

ANN is ideal for solving problems regarding data. Forward-facing algorithms can easily be used to process image data, text data, and tabular data. CNN requires many more data inputs to achieve its novel high accuracy rate.

In many cases, to get the same accuracy as ANN for data processing, you must use multiple data augmentation methods to widen the reaches of your data. In addition, ANN has the ability to implicitly detect complex nonlinear relationships between dependent and independent variables.

It would also get you virtually the same accuracy rate as using CNN for data classification problems. This makes CNN an overkill solution for a data classification problem since you'd have to augment your data to artificially inflate the dataset and deal with the storage and hardware dependencies of CNN. ANN is a comparatively lightweight way of solving data classification problems.



Graph: 5.6 CNN Vs ANN accuracy comparison

A CNN model has an accuracy of 99.78.

An ANN model has an accuracy of 97.03.

5.4.2 ACCURACY COMPARISON BETWEEN CNN AND VGG

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual computer vision competition. Each year, teams compete on two tasks. The first is to detect objects within an image coming from 200 classes, which is called object localization.

The second is to classify images, each labelled with one of 1000 categories, which is called image classification. VGG 16 was proposed by Karen Simonyan and Andrew Zisserman of the Visual Geometry Group Lab of Oxford University in 2014 in the paper “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION”. This model won 1st and 2nd place in the above categories in the 2014 ILSVRC challenge.

VGG Architecture: The input to the network is an image of dimensions (224, 224, 3). The first two layers have 64 channels of a 3*3 filter size and the same padding. Then after a max pool layer of stride (2, 2), two layers have convolution layers of 128 filter size and filter size (3, 3). This is followed by a max-pooling layer of stride (2, 2) which is the same as the previous layer. Then there are 2 convolution layers of filter size (3, 3) and 256 filters. After that, there are 2 sets of 3 convolution layers and a max pool layer. Each has 512 filters of (3, 3) size with the same padding. This image is then passed to the stack of two convolution layers. In these convolution and max-pooling layers, the filters we use are of the size 3*3 instead of 11*11 in AlexNet and 7*7 in ZF-Net. In some of the layers, it also uses 1*1 pixel which is used to manipulate the number of input channels. There is a padding of 1-pixel (same padding) done after each convolution layer to prevent the spatial feature of the image.

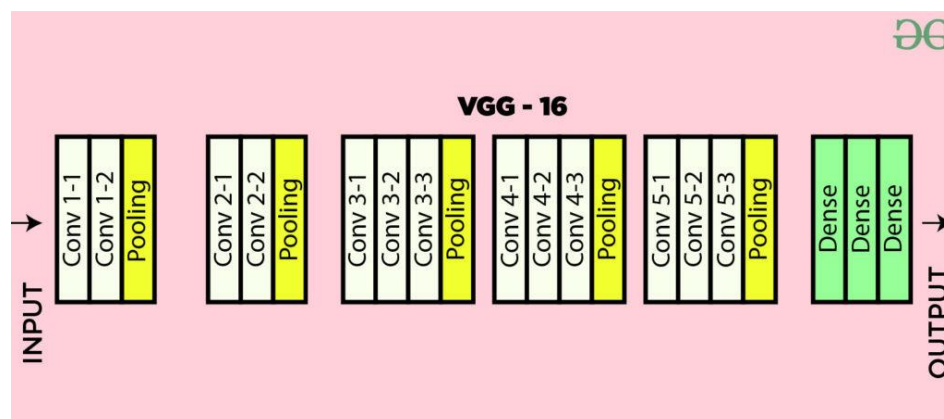
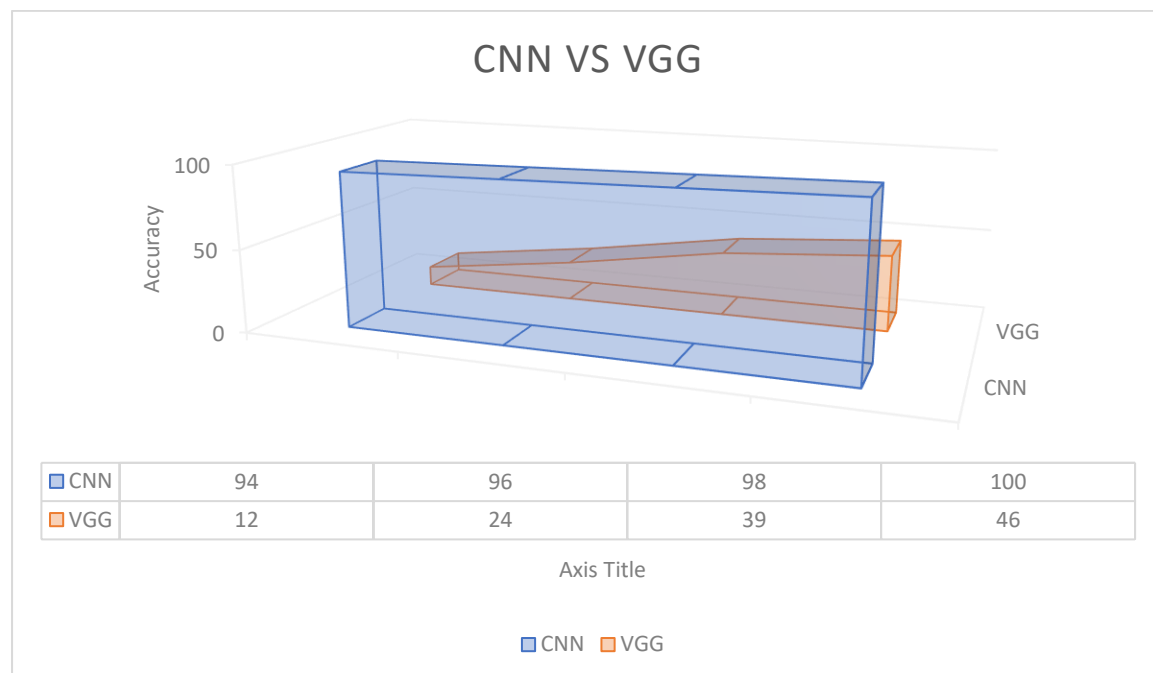


Fig: 5.5 Architecture of VGG

After the stack of convolution and max-pooling layer, we got a (7, 7, 512) feature map. We flatten this output to make it a (1, 25088) feature vector. After this there is 3 fully connected layer, the first layer takes input from the last feature vector and outputs a (1, 4096) vector, the second layer also outputs a vector of size (1, 4096) but the third layer output a 1000 channels for 1000 classes of ILSVRC challenge i.e. 3rd fully

connected layer is used to implement softmax function to classify 1000 classes. All the hidden layers use ReLU as its activation function. ReLU is more computationally efficient because it results in faster learning, and it also decreases the likelihood of vanishing gradient problems.



Graph: 5.7 CNN Vs VGG accuracy comparison

A CNN model has an accuracy of 99.78.

A VGG model has an accuracy of 25.56.

5.4.3 ACCURACY COMPARISON BETWEEN CNN AND Resnet 50

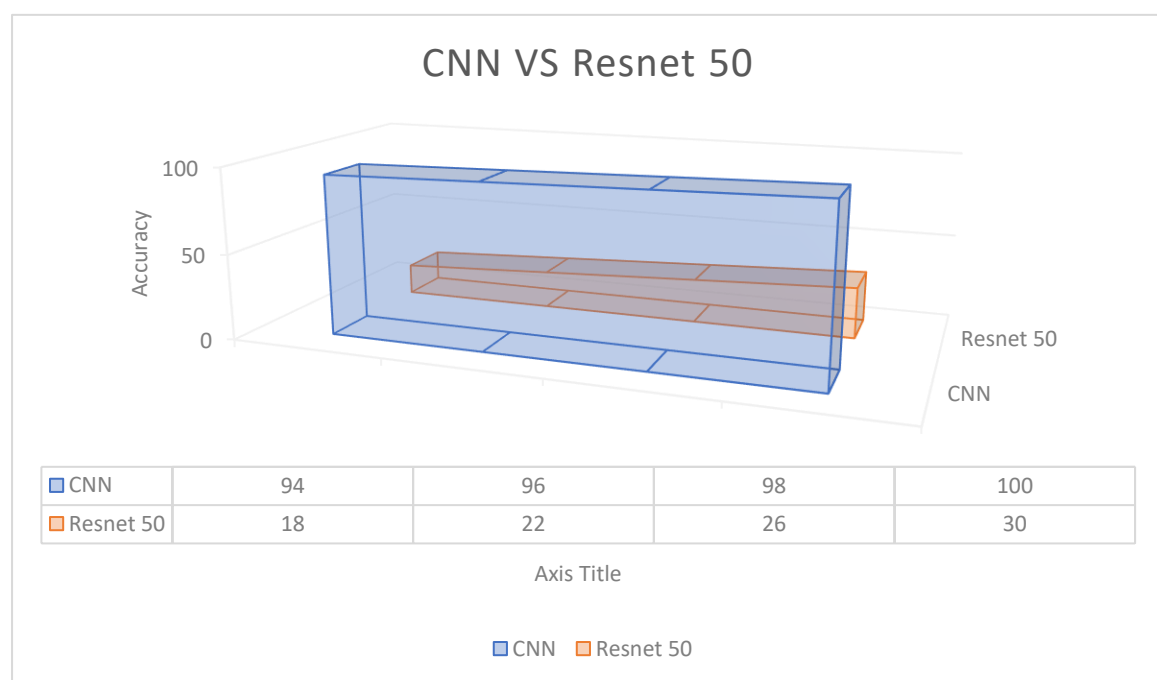
Because of the framework that ResNets presented it was made possible to train ultra-deep neural networks and by that, I mean that a network can contain hundreds or thousands of layers and still achieve great performance.

The ResNets were initially applied to the image recognition task but as it is mentioned in the paper that the framework can also be used for non-computer vision tasks also to achieve better accuracy.

The problem of vanishing/exploding gradients those problems were largely handled by many ways and enabled networks with tens of layers to converge but when deep

neural networks start to converge, we see another problem of the accuracy getting saturated and then degrading rapidly and this was not caused by overfitting as one may guess and adding more layers to a suitable deep model just increased the training error.

This problem was further rectified by taking a shallower model and a deep model that was constructed with the layers from the shallow model and adding identity layers to it and accordingly the deeper model shouldn't have produced any higher training error than its counterpart as the added layers were just the identity layers.



Graph: 5.8 CNN Vs Resnet 50 accuracy comparison

A CNN model has an accuracy of 99.78.

A Resnet 50 has an accuracy of 18.62.

CHAPTER – 6

SUMMARY AND CONCLUSIONS

Satellite-based SAR systems are one of the best sensors for oil slicks detection in wide areas, not only for daytime but also for night-time observation. Although there are some wind and sea state limitations as well as many look-alikes, SAR is nowadays the standard tool for oil spills detection and monitoring offshore. This project describes an algorithm and software application for oil slicks detection on the sea surface. Some experimental results demonstrate application's ability to process SAR images and to detect oil slicks on the sea surface automatically. To validate the detection, to extract the feature maps with simple feature map calculations, a simple CNN model is used. The findings demonstrate the viability of the FCN model provided for oil spill detection. In this project, the present a deep-learning framework to detect occurrences of oil spills in SAR images. Then, T Filtered patches are sent to a fully original 23-layer CNN that was trained for forty epochs on 80% of the patches and validated using a 5-fold and 10-fold cross-validation to identify patches with significant oil spill constituent distribution (i.e., over I Chronicles oil spill). With an accuracy, sensitivity, specificity, and weighted letter of around ninety-nine, the intended model provided exceptional classification performance. Additionally, the planned CNN stage runs the successfully classified patches with more than I Chronicles oil spill through a strong linguistics segmentation stage with a cost-sensitive loss perform (i.e., generalised Dice loss) optimised to reduce the bias among the patches since Oil spill still may represent only a 1/500th of the patch resolution. The research can be extended further for land masking to avoid the influence of the radar echo from land as well as for locating spills with highest accuracy by using edge detection methods to find spot's boundaries for the purposes of oil spills mapping. Remotely piloted aircraft systems, i.e., drones will be used additionally to complement the satellite images by subsequently in-situ observation of detected spills.

REFERENCES

- [1] Alexandrov, Ch., Kolev, N., Tsvetkov, M., Sivkov J. and Hristov, A., Sentinel - 1 SAR image processing for target detection and evaluation by using bulgarian VTMISS, 16th Conference on Electrical Machines, Drives and Power Systems, ELMA 2019.
- [2] Antunes, S., Santos, S. and Lourenco P., CleanSeaNet: 10 years detecting marine pollution in Europe, SeaSAR 2018 International conference, Frascati (Rome), 2018.
- [3] Brekke, C., Solberg, Anne H.S., Oil spill detection by satellite remote sensing, Science direct, Remote sensing of environment, 2005, pp. 1 – 13, Available online at www.sciencedirect.com.
- [4] Chavdar Alexandrov, Nikolay Kolev, Yordan Sivkov, Avgustin Hristov, Oil spills detection on sea surface by using sentinel -1 SAR images, Dept. of Electronics Nikola Vaptsarov Naval Academy.
- [5] Dachev Y., Dimitrakieva Sv., Milev D., Atanasova Kr., Behaviour and fate of oil spills, International Journal of Scientific and Technology Research, Vol. 9, Issue 4, 2020, ISSN 2277-8616.
- [6] Dimitrakiev, D., Y. Dachev, D. Milev, Response organization and control strategies during oil spills, International Journal of Scientific and Technology Research, Vol. 9, Issue 4, 2020, ISSN 2277-8616.
- [7] Fengli Zhang, Yun Shao, Wei Tian, Shiang Wang, Oil spill identification based on textural information of SAR image, Institute of Remote Sensing Applications.
- [8] Fingas, M. and Brown, C. E., A review of oil spill remote sensing, Sensors, 2018, 18, 91, Available online at www.mdpi.com/journals/sensor.
- [9] F. Meyer, S. Hinz. Automatic ship detection in space-born SAR imagery. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences name or conference proceedings, vol. 38, 2009.
- [10] Garvanov, I., Kabakchiev Chr., One- and two- dimensional CFAR processors in the presence of Strong pulse jamming, Cybernetics and information technologies, Vol. 1, pp. 58 – 71, 2002.
- [11] Greidanus H., Santamaria C., First analyses of Sentinel – 1 images for maritime surveillance, JRC Science and Policy Reports, 2014.

- [12] Grover A., Kumar Sh. and Kumar A., Ship detection using Sentinel – SAR data, ISPRS Annals of Photogrammetry, Remote sensing and spatial information sciences, Vol. IV – 5, 2018.
- [13] Hajduch G., V. Kerboal, R. de Joux, Ship detection: from processing to instrument characterisation, European Space Agency, (Special Publication. Proceedings of SeaSAR 2008 Conference, Frascati, ESA SP, 2008.
- [14] Karathanassi V, P. Pavlakis, D. Rokos, Oil spill detection using RBF neural networks and SAR data.
- [15] Kostadinov K., Alexandrov Ch., Tsvetkov M., Applications of TCP/IP based radar – computer interface, International Symposium on electrical apparatus, SIELA2009, Burgas 2009.
- [16] Lehner S., Soloviev A., Kluge J., Schwarz E. and Perrie W., The interaction of crude oil on the sea surface with ocean fronts observed by SAR in NRT, 12-th European Conference on SAR EUSAR 2018, Aachen, June 2018.
- [17] Maio, A. D., Ricci, G., & Tesauro, M. (2001). On CFAR detection of oil slicks on the ocean surface by multifrequency and/or multipolarization SAR. In Radar conference, 2001. Proceedings of the 2001 IEEE (pp. 351– 356).
- [18] Mayerle and Suman singha, A deep learning based oil spill detector using Sentinel - 1 SAR imagery.
- [19] Mital K. Sorde, Mrs.Aruna sharma, Oil spills detection in MATHLAB software using SAR-images, International Research Journal of Engineering and Technology.
- [20] Oil spill mapping with sentinel-1, RUS-Research and User Support for Sentinel Core Products.
- [21] Papila, I., Sertel, E., Kaya, S. and Gazioglu, C., Oil spill detection using remote sensing technologies – SAR, Chapter, Turkish marine research foundation, Publication 47, Istanbul 2018.
- [22] Pelich R., N. Longepe, G. Mercier, G. Hajduch, R. Garelo, AIS based evaluation of target detectors and SAR sensors characteristics for marine surveillance, IEEE Journal of Selected Topics in Applied Earth Observation and Remote Sensing, May 2014, PP(8): 1- 10.
- [23] Saima Naza, Muhammad Farooq Iqbala, Irfan Mahmooda , Mona Allamb, Marine oil spill detection using synthetic aperture radar over Indian Ocean, Geo- infInformatics Research Group, Department of Meteorology, COMSATS University Islamabad d Islamabad, Pakistan , Environment and Climate Change Research Institute.

- [24] SHIRA. Detection and monitoring of sea surface features with radar: Oil slicks, Wave spectra, Currents, Water depth, and Small targets at the surface, Available online at www.argoss.nl/projects/a58_simba/html/fel/shira1.htm
- [25] Sudhir Kumar Chaturvedi A, Saikat Banerjee b, Shashank Lelec, An assessment of oil spill detection using sentinel 1 sar-c image, Department of Aerospace Engineering, University of Petroleum and Energy Studies.
- [26] Topouzelis K, S. Singha, Oil spill detection using space-borne sentinel-1 SAR imagery, RUS-Research and User Support for Sentinel Core Products.
- [27] Tsvetkov, M., Alexandrov, Ch., Software simulator of marine monitoring platform. 19th International Symposium on electrical apparatus, SIELA2016, Burgas 2016.
- [28] Vespe M., Greidanus H., SAR image quality assesment and indicators for vessel and oil spill detection, IEEE Transactions on Geoscience and Remote Sencing, Vol. 50, No. 11, Nov. 2012, pp. 4726 – 4734.
- [29] Wackerman C. C., K. S. Friedman, W. G. Pichel, P. Clemente-Colon and X. Li, Automatic detection of ships in Radar-1 SAR imagery, Canadian journal of remote sensing, Vol. 27, No. 5, pp. 568 – 577, 2001.
- [30] Werner Alpers, Benjamin Holt, Kan Zeng, Oil spill detection by imaging radars, Jet Propulsion Laboratory, California Institute of Technology.

APPENDIX

A. SOURCE CODE

```
from google.colab import drive
drive.mount('/content/drive')
import matplotlib.pyplot as plt
import seaborn as sns

import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam

from sklearn.metrics import classification_report, confusion_matrix

import tensorflow as tf

import cv2
import os

import numpy as np
labels = ['Non Oil Spill', 'Oil Spill']
img_size = 150
def get_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img))[...::-1] #convert BGR to RGB
            format
```

```

resized_arr = cv2.resize(img_arr, (img_size, img_size)) # Reshaping images to
preferred size
data.append([resized_arr, class_num])
except Exception as e:
print(e)
return np.array(data)
train = get_data(r"/content/drive/MyDrive/oil/dataset/Validation")
val = get_data(r"/content/drive/MyDrive/oil/dataset/Validation")
test=get_data(r"/content/drive/MyDrive/oil/dataset/test")
l = []
for i in train:
if(i[1] == 0):
l.append("Non Oil Spill")
else:
l.append("Oil Spill")
sns.set_style('darkgrid')
sns.countplot(l)
a = []
for i in val:
if(i[1] == 0):
a.append("Non Oil Spill")
else:
a.append("Oil Spill")
sns.set_style('darkgrid')
sns.countplot(a)
b = []
for i in test:
if(i[1] == 0):
b.append("Non Oil Spill")
else:
b.append("Oil Spill")
sns.set_style('darkgrid')
sns.countplot(b)
plt.figure(figsize = (5,5))

```

```

plt.imshow(train[1][0])
plt.title(labels[train[0][1]])
plt.figure(figsize = (5,5))
plt.imshow(train[-1][0])
plt.title(labels[train[-1][1]])
#train
#test
#val
x_train = []
y_train = []
x_val = []
y_val = []
x_test = []
y_test = []

for feature, label in train:
x_train.append(feature)
y_train.append(label)

for feature, label in val:
x_val.append(feature)
y_val.append(label)

for feature, label in test:
x_test.append(feature)
y_test.append(label)

# Normalize the data
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255
x_test=np.array(x_test)/255

x_train.reshape(-1, img_size, img_size, 1)
y_train = np.array(y_train)

```

```
x_val.reshape(-1, img_size, img_size, 1)
y_val = np.array(y_val)
```

```
x_test.reshape(-1, img_size, img_size, 1)
y_test = np.array(y_test)
print("x_train:",x_train.shape)
print("y_train:",y_train.shape)
print("x_test:",x_test.shape)
print("y_test:",y_test.shape)
print("x_val:",x_val.shape)
print("y_val:",y_val.shape)
datagen = ImageDataGenerator(
featurewise_center=False,
samplewise_center=False,
featurewise_std_normalization=False,
samplewise_std_normalization=False,
zca_whitening=False,
rotation_range = 90,
zoom_range = 0.5,
width_shift_range=0.2,
height_shift_range=0.2,
horizontal_flip = True,

)
```

```
datagen.fit(x_train)
datagen.fit(x_val)
datagen.fit(x_test)
```

CNN

```
model = Sequential()
model.add(Conv2D(32,(3,3),padding="same",          activation="relu",
input_shape=(150,150,3)))
```

```

model.add(MaxPool2D((2, 2), strides=2))

model.add(Conv2D(32, (3,3), padding="same", activation="relu"))
model.add(MaxPool2D((2, 2), strides=2))

model.add(Conv2D(64, (3,3), padding="same", activation="relu"))
model.add(MaxPool2D((2, 2), strides=2))

model.add(Conv2D(64,(3,3), padding="same", activation="relu"))
model.add(MaxPool2D((2, 2), strides=2))

model.add(Dropout(0.2))
model.add(Flatten())
#model.add(Dense(128,activation="relu"))
"""
model.add(Dense(84,activation="relu"))
model.add(Dense(32,activation="relu"))
"""
model.add(Dense(2, activation="softmax"))

model.summary()
opt = Adam(lr=0.001)
model.compile(optimizer = opt, loss = 'sparse_categorical_crossentropy',
metrics = ['accuracy'])
#tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)'sparse_ca
tegorical_crossentropy'validation_data = (x_val, y_val)
history = model.fit(x_train,y_train,epochs = 25,validation_data = (x_val,
y_val),shuffle=True,batch_size=64)
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

```

```

epochs_range = range(25)

plt.figure(figsize=(15, 15))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
loss = history.history['loss']
epochs_range = range(25)
plt.figure(figsize=(15, 15))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, loss, label='loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
"""

predictions = model.predict_classes(x_test)
predictions = predictions.reshape(1,-1)[0]

print(classification_report(y_val, predictions, target_names = ['Rugby (Class
0)','Soccer (Class 1)']))
"""

predictions = model.predict(x_test)
pred=np.argmax(predictions,axis=1)
pred = pred.reshape(1,-1)[0]
print(classification_report(y_test,pred, target_names = ['Non Oil Spill(Class
0)','Oil Spill (Class 1)']))

```

```

model.evaluate(x_test,y_test,batch_size=64)
pred.shape
y_test.shape

```

VGG

```

from tensorflow.keras.applications import VGG19
vgg19 = VGG19(weights='imagenet',include_top=False,input_shape=(150,150,3))

for layer in vgg19.layers[:19]:
    layer.trainable = False

model = Sequential()
model.add(vgg19)
model.add(MaxPool2D((2,2),strides=2))

model.add(Dense(512, activation='relu'))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(2,activation='sigmoid'))
model.summary()
opt = Adam(lr=0.001)
model.compile(optimizer=opt,loss="sparse_categorical_crossentropy",metrics
="accuracy")
from keras.callbacks import ReduceLROnPlateau
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy',
patience = 2, verbose=1,factor=0.3, min_lr=0.000001)
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

```

```

epochs_range = range(30)

plt.figure(figsize=(15, 15))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()

loss = history.history['loss']
epochs_range = range(30)
plt.figure(figsize=(15, 15))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, loss, label='loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
model.evaluate(x_test,y_test,batch_size=64)

```

RESNET

```

from tensorflow.keras.applications import ResNet101V2
from tensorflow.keras.layers import
Dense,Conv2D,MaxPool2D,Flatten,Dropout,BatchNormalization,Activation
convlayer=ResNet101V2(input_shape=(150,150,3),weights='imagenet',includ
e_top=False)
for layer in convlayer.layers:
layer.trainable=False
model=Sequential()

```

```

model.add(convlayer)
model.add(Dropout(0.5))
model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(1024,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(512,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(2,activation='softmax'))
optimizer = 'adam'
model.compile(loss='sparse_categorical_crossentropy',
optimizer=optimizer,
metrics=['accuracy'])
history=model.fit(x_train,y_train,validation_data=(x_val,y_val),epochs=25,batch_size=32,shuffle=True)
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(25)

plt.figure(figsize=(15, 15))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2, 2, 2)

```

```

plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
model.evaluate(x_test,y_test,batch_size=32)

```

ANN

```

model = Sequential()
model.add(Dense(256,input_shape=(150,150,3), activation='relu'))
model.add(Dense(512, activation='relu'))
model.add(Flatten())
model.add(Dense(2,activation='sigmoid'))
model.summary()
optimizer ='adam'
model.compile(loss='sparse_categorical_crossentropy',
optimizer=optimizer,
metrics=['accuracy'])
history=model.fit(x_train,y_train,validation_data=(x_val,y_val),epochs=10,batch_size=32,shuffle=True)
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(10)

plt.figure(figsize=(15, 15))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

```

```
plt.subplot(2, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
model.evaluate(x_test,y_test,batch_size=32)
```