FINTECH: ADVANCED FISH SPECIES IDENTIFICATION

A PROJECT REPORT

Submitted by

RISHIKA S	**[8115U20AD035]**
DEEPIKA B	**[8115U20AD008]**
THARSHNE K	**[8115U20AD044]**
VINODHINI C	**[8115U20AD048]**

in partial fulfilment of the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

SAMAYAPURAM, TRICHY – 621 112.

ANNA UNIVERSITY : CHENNAI 600 025
APRIL/MAY 2024

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **FINTECH: ADVANCED FISH SPECIES IDENTIFICATION** is the bonafide work of **RISHIKA S(8115U20AD035), DEEPIKA B(8115U20AD008), THARSHNE K(8115U20AD044), VINODHINI C(8115U20AD048)** who carried out the project work under my supervision.

SIGNATURE
Dr. B. Kiran Bala, M.E.,M.B.A.,Ph.D.,

HEAD OF THE DEPARTMENT
Artificial Intelligence and Data Science
K.Ramakrishnan College of Engineering
(Autonomous), Samayapuram
Trichy – 621112

SIGNATURE
Mrs. G. USHA DEVI, M.Tech.,
Assistant Professor.

SUPERVISOR
Artificial Intelligence and Data Science
K.Ramakrishnan College of Engineering
(Autonomous), Samayapuram
Trichy – 621112

Submitted for the Project Viva-Voce Examination held on …….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We thank the almighty GOD, without whom it would not have been possible for us to complete our project.

We wish to address our profound gratitude to **Dr.K. RAMAKRISHNAN,** Chairman, K.Ramakrishnan College of Engineering (Autonomous), who encouraged and gave us all help throughout the course.

We express our hearty gratitude and thanks to our honourable and grateful Executive Director **Dr.S. KUPPUSAMY, B.Sc., MBA., Ph.D.,** K.Ramakrishnan College of Engineering (Autonomous).

We are glad to thank our principal **Dr.D. SRINIVASAN, M.E., Ph.D., FIE.,MIIW.,MISTE.,MISAE.,C.Engg,** for giving us permission to carry out this project.

We wish to convey our sincere thanks to **Dr. B. KIRAN BALA, B.Tech., M.E., M.B.A., Ph.D.,** Head of the Department, Artificial Intelligence and Data Science**,** K.Ramakrishnan College of Engineering (Autonomous), for giving us constants encouragement and advice throughout the course.

We are grateful to **Mrs.G. USHA DEVI, M.Tech.,** Assistant Professor in the Department of Artificial Intelligence and Data Science, K.Ramakrishnan College of Engineering (Autonomous), for her guidance and valuable suggestions during the course of study.

Finally, we sincerely acknowledged in no less term for all our staff members, colleagues, our parents and friends for their co-operation and help at various stages of this project work.

DECLARATION

I hereby declare that the work entitled **"FINTECH: ADVANCED FISH SPECIES IDENTIFICATION"** is submitted in partial fulfilment of the requirement for the reward of the degree in B.Tech., Anna University, Chennai, is a record of our own work carried out by us during the academic year 2023-2024 under the supervision and guidance of **Mrs.G. USHA DEVI, M.Tech,** Assistant professor, **Department of Artificial Intelligence and Data Science, K. Ramakrishnan College of Engineering (Autonomous).** The extent and source of information are derived from the existing literature and have been indicated through the dissertation at the appropriate places. The matter embodied in this work is original and has not been submitted for the award of any degree or diploma, either in this or any other University.

RISHIKA S [8115U20AD035]
DEEPIKA B [8115U20AD008]
THARSHNE K [8115U20AD044]
VINODHINI C [8115U20AD048]

I certify that the declarations made by the above candidates are true.

Mrs. G. USHA DEVI, M.TECH.,
Assistant Professor/AI&DS

ABSTRACT

Monitoring fish populations and assessing species abundance is essential for marine scientists and conservationists. Traditional manual methods are time-consuming, prompting the development of computer-based alternatives. However, challenges like unclear images persist. Our solution leverages the MobileNet v2 Architecture integrated with web development to automate fish identification. By combining deep learning algorithms with web functionality, our system offers real-time analysis of underwater data. This innovative module streamlines data collection, enhances species identification accuracy, and aids marine conservation efforts effectively. With its user-friendly interface and scalable design, our technology revolutionizes underwater research, paving the way for more efficient marine ecosystem management.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
API	Application Programming Interface
ANN	Artificial Neural Networks
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
CV	Computer Vision
DNN	Deep Neural Networks
EDA	Exploratory Data Analysis
FC	Fully Connected Layer
GB	Gigabyte
GHz	Gigahertz
GPU	Graphical Processing Unit
HTML	Hypertext Markup Language
ML	Machine Learning
MRI	Magnetic Resonance Imaging
NLP	Natural Language Processing
OOP	Object-Oriented Programming
ROC	Receiver Operating Characteristic
VGG16	Visual Geometry Group 16

CHAPTER 1

INTRODUCTION

1.1 DEEP LEARNING

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don't need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs). These neural networks are inspired by the structure and function of the human brain's biological neurons, and they are designed to learn from large amounts of data.

1.1.1 TASKS ASSOCIATED WITH DEEP LEARNING

Deep learning models can be taught to perform classification tasks and recognize patterns in photos, text, audio and other various data. These tasks are outlined in Fig. 1.1.1.

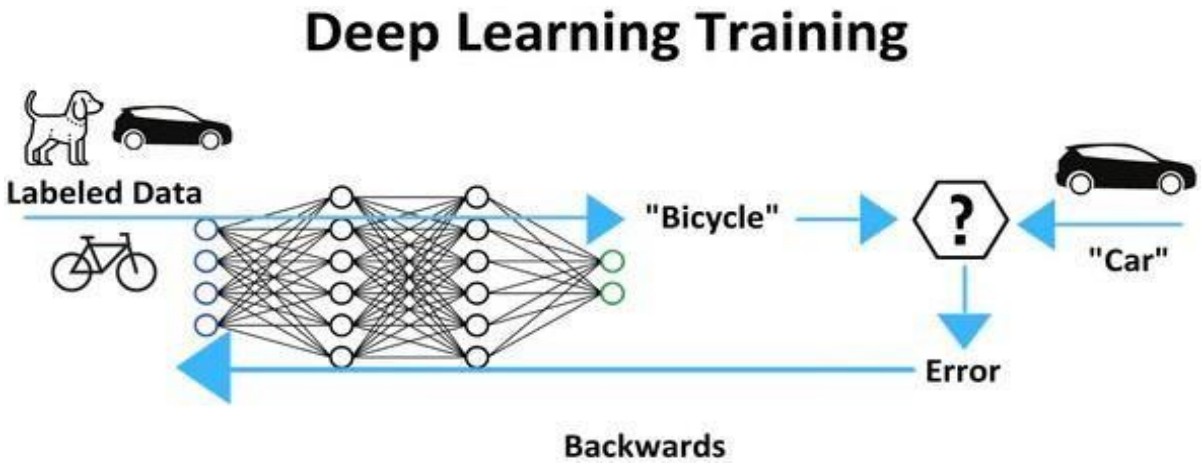


Fig. 1.1.1 Deep Learning Task

1.1.2 AN EXPLANATION OF NEURAL NETWORKS

A neural network is structured like the human brain and consists of artificial neurons, also known as nodes. These nodes are stacked next to each other in three layers:

- The input layer
- The hidden layer(s)
- The output layer

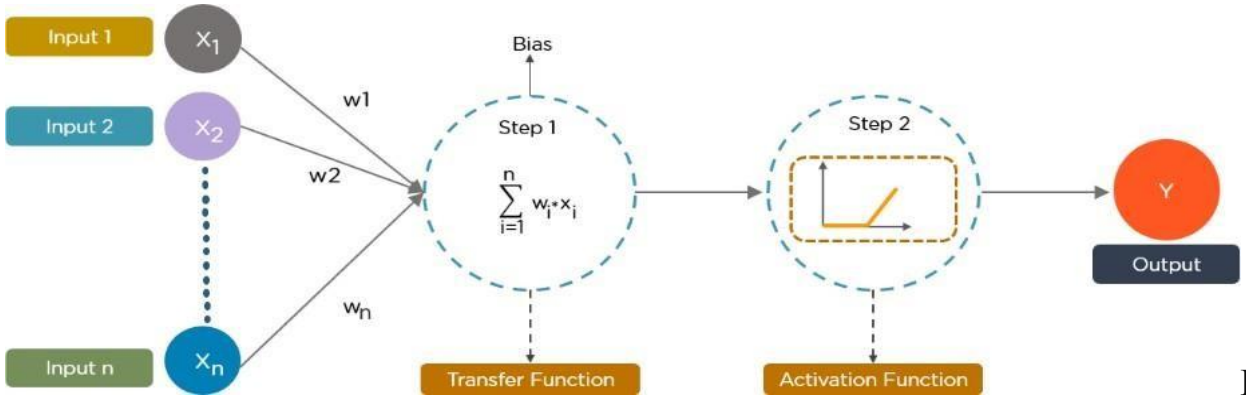


Fig.1.1.2 Neural Network

Every node receives information from data in the form of inputs. The node computes the inputs, multiplies them by random weights, and then adds a bias. Lastly, to decide which neuron to fire, nonlinear functions—also referred to as activation functions— are used.

1.1.3 DEEP LEARNING METHODOLOGY

- Deep learning algorithms rely on artificial neural networks (ANNs) to mimic the way the brain processes information, even though they include self-learning representations.
- Algorithms use unknown elements in the input distribution to group objects, extract features, and find meaningful data patterns during training.
- This happens on several levels, employing the algorithms to create the models, much like when machines are trained for self-learning.
- Multiple algorithms are used by deep learning models. Even though no network is flawless, some algorithms work better for particular jobs than others. It's helpful to have a firm grasp of each major algorithm in order to make the appropriate decisions.

1.1.4 CONVOLUTIONAL NEURAL NETWORKS (CNNS)

One kind of Deep Learning neural network architecture that's frequently used in computer vision is the convolutional neural network (CNN). Convolutional neural networks perform better with image, speech, or audio signal inputs than other types of neural networks. There are three primary categories of layers in them:

Convolution Layer

CNN has a convolution layer that has several filters to perform the convolution operation.

Pooling Layer

- The rectified feature map next feeds into a pooling layer. Pooling is a down-sampling operation that reduces the dimensions of the feature map.
- The pooling layer then converts the resulting two-dimensional arrays from the pooled feature map into a single, long, continuous, linear vector by flattening it.

Fully Connected Layer

A fully connected layer forms when the flattened matrix from the pooling layer is fed as an input, which classifies and identifies the images. Below is an example of an image processed via CNN.

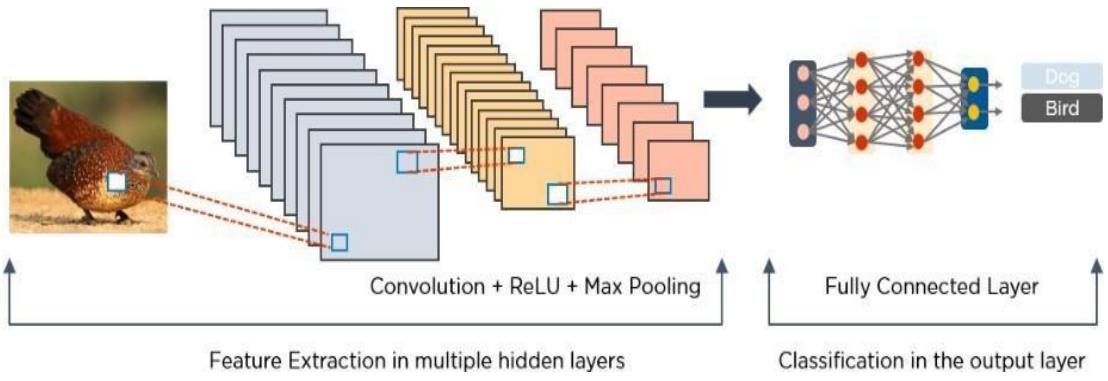


Fig. 1.1.4 Fully Connected Layer

1.1.5 APPLICATIONS

1.1.5.1 COMPUTER VISION

Deep learning models in computer vision can help machines recognise and comprehend visual data. The following are a few of the primary computer vision applications of deep learning:

Object Detection and Recognition:

Deep learning model can be used to identify and locate objects within images and videos, making it possible for machines to perform tasks such as self-driving cars, surveillance, and robotics.

Image Classification:

Deep learning models can be used to classify images into categories such as animals, plants, and buildings. This is used in applications such as medical imaging, quality control, and image retrieval.

1.1.5.2 NATURAL LANGUAGE PROCESSING (NLP)

Deep learning model can enable machines to understand and generate human language. Some of the main applications of deep learning in NLP include:

1. **Automatic Text Generation:** Using trained deep learning models, new text such as essays and summaries can be automatically generated. These models are capable of learning a corpus of text

2. **Language translation**: Text can be translated between languages using deep learning models, facilitating communication with speakers of various languages.

1.1.5.3 REINFORCEMENT LEARNING

- **Reinforcement learning:** Cognitive gaming: In games like Go, Chess, and Atari, deep reinforcement learning models have outperformed human experts.
- **Robotics:** Robots can be trained to carry out intricate tasks like manipulating, navigating, and grasping objects by using deep reinforcement learning models.
- **Control systems:** Complex systems, including supply chain optimization, traffic management, and power grids, can be managed by deep reinforcement learning models.

1.1.6 CHALLENGES

Deep learning has made significant advancements in various fields, but there are still some challenges that need to be addressed. Here are some of the main challenges in deep learning:

- **Data availability:** Learning from vast volumes of data is necessary. Acquiring as much training data as possible is crucial for deep learning applications.
- **Computational Resources:** The deep learning model requires specialized hardware, such as GPUs and TPUs, which makes it computationally expensive to train.

- **Time-consuming:** Depending on the computational resources available, working with sequential data can take many days or even months.
- **Interpretability:** The inner workings of deep learning models are mysterious and intricate. The result is very hard to interpret.

1.2 OVERVIEW OF MACHINE LEARNING

Machine learning is a subfield of artificial intelligence that involves the development of algorithms and statistical models that enable computers to improve their performance in tasks through experience. These algorithms and models are designed to learn from data and make predictions or decisions without explicit instructions. There are several types of machine learning, including supervised learning, unsupervised learning, and reinforcement learning

1.2.1 TASKS ASSOCIATED WITH MACHINE LEARNING

A machine learning task is the type of prediction or inference being made, based on the problem or question that is being asked, and the available data. These tasks are outlined in Fig. 1.2.1.

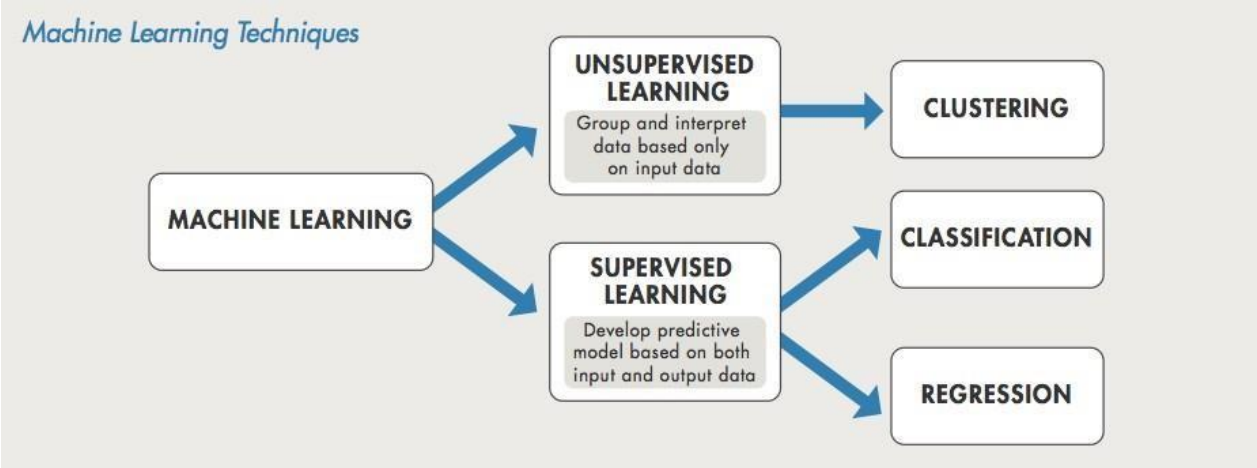


Fig. 1.2.1 Machine Learning Task

1.2.1.1 SUPERVISED MACHINE LEARNING

The process of giving the machine learning model accurate input and output data is known as supervised learning. Finding a mapping function to link the input variable (x) and the output variable (y) is the goal of a supervised learning algorithm.

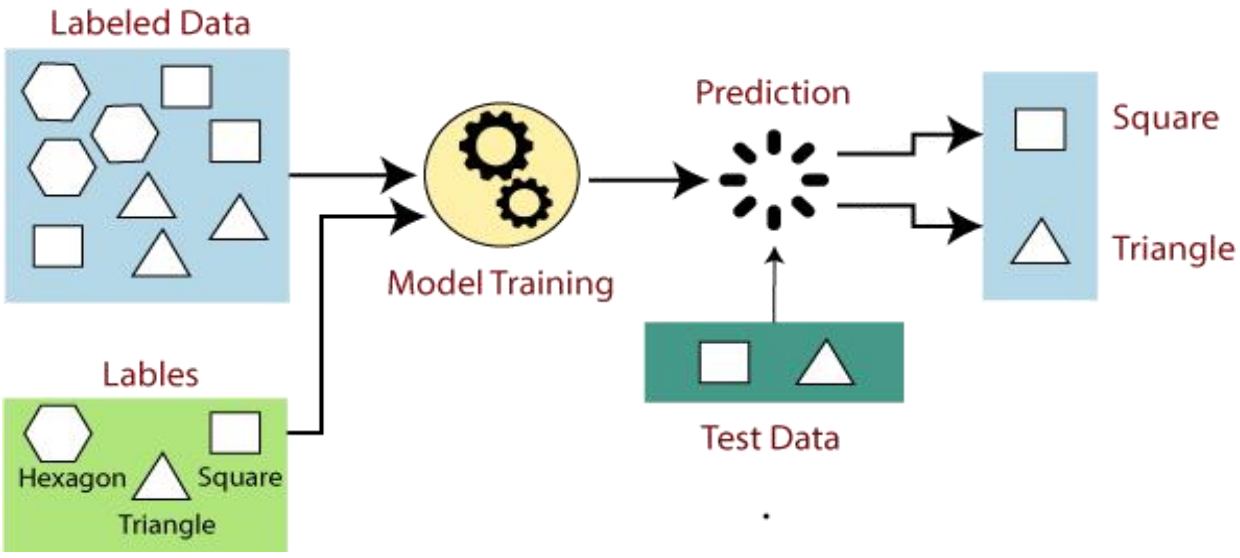


Fig. 1.2.1.1 Supervised learning

Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc.

Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

1.2.1.2 UNSUPERVISED MACHINE LEARNING

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of the dataset, group that data according to similarities, and represent that dataset in a compressed format.

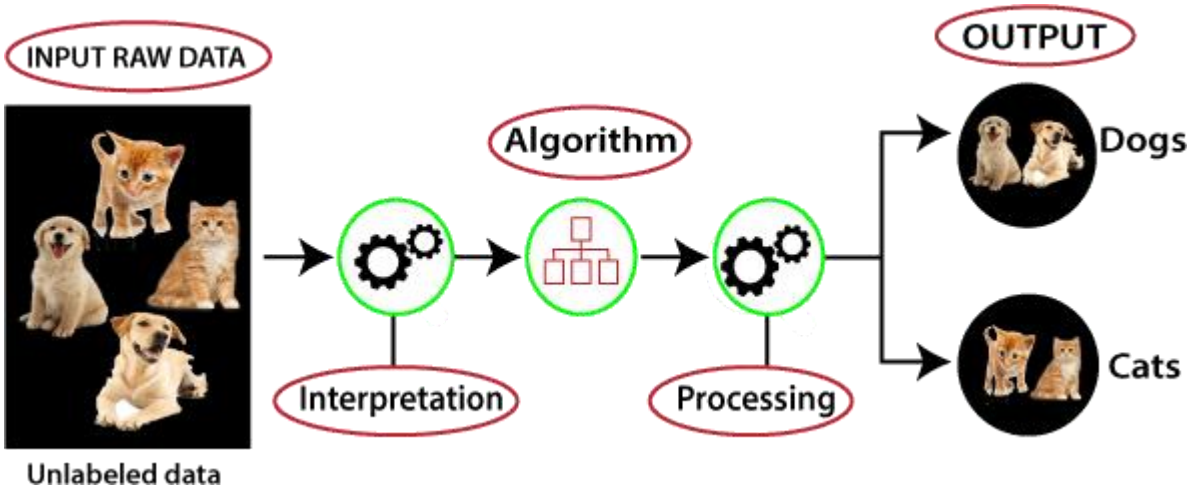


Fig. 1.2.1.2 Unsupervised learning

Clustering

Clustering is a type of unsupervised learning that is used to group similar data points together. Clustering algorithms work by iteratively moving data points closer to their cluster centers and further away from data points in other clusters.

Association

Association rule learning is a type of unsupervised learning that is used to identify patterns in a data. Association rule learning algorithms work by finding relationships between different items in a dataset.

1.2.2 RANDOM FOREST ALGORITHM

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

Accuracy is increased and overfitting is avoided when there are more trees in the forest.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset" Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The below diagram explains the working of the Random Forest algorithm:

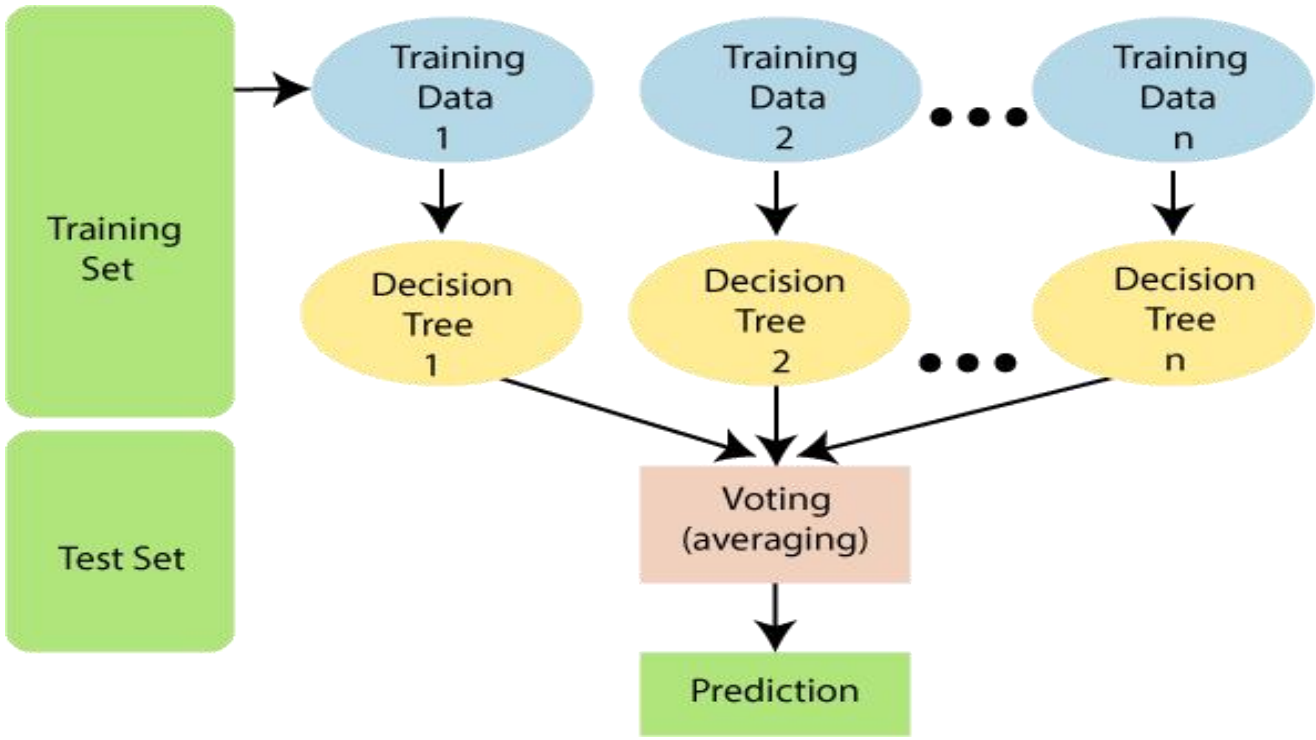


Fig. 1.2.2 Random Forest Algorithm (Train_test_split)

Random Forest Algorithm Operation

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

1.2.3 TRANSFER LEARNING

Transfer learning is a technique in machine learning where a model trained on one task is used as the starting point for a model on a second task. When there is insufficient data available for the second task or when the tasks are comparable to each other, this can be helpful. The model can learn more efficiently and fast on the second challenge by starting with the learnt features from the first task. As a result of the model having previously acquired generic features that will probably be helpful in the second assignment, this can also aid in preventing overfitting.

1.2.3.1 TASKS ASSOCIATED WITH MACHINE LEARNING

The general process of transfer learning is as follows:

Pre-trained Model: Begin with a model that has already undergone extensive training on a sizable dataset for a particular task. This model has been trained on large datasets on a regular basis, and it has found common features and patterns that apply to many comparable tasks.

Base Model: The pre-trained model is referred to as the base model. It consists of layers that have learned hierarchical feature representations by using the input data.

Transfer Layers: Identify a group of layers in the pre-trained model that record general data pertinent to both the new and the prior task. These layers are usually found near the top of the network since they are prone to learning low-level information.

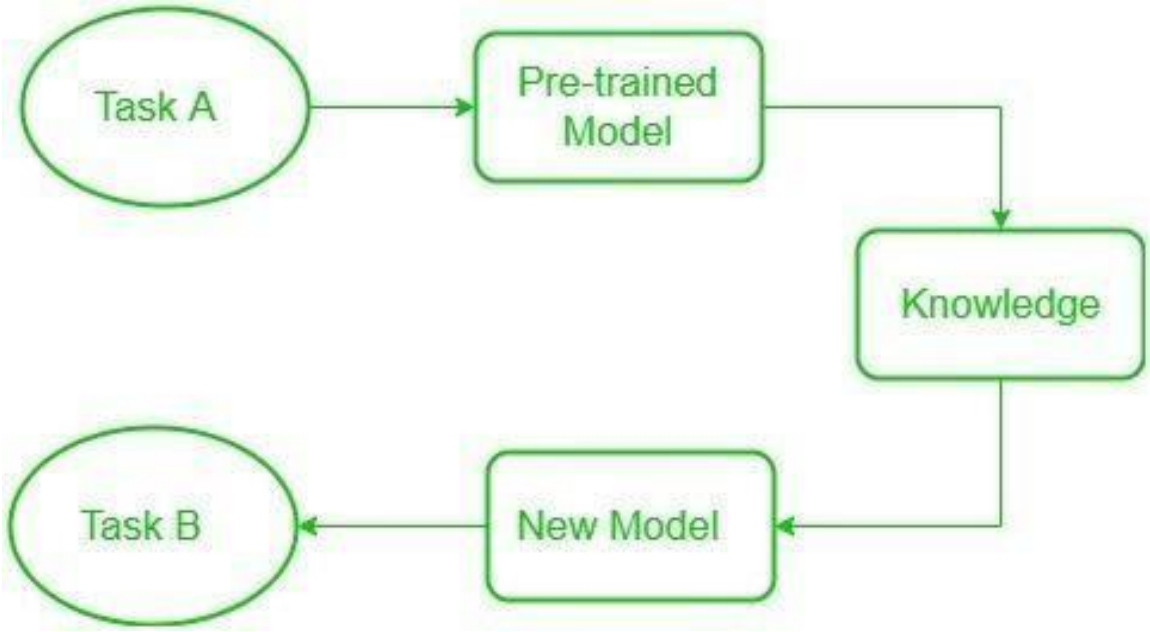


Fig. 1.2.3.1. Pre-Trained Model

Learning low-level features for task A should help with learning the model for task B. This is the essence of transfer learning. These days, it's rare to see someone building a whole convolutional neural network from start; instead, it's typical to use a pre-trained model that was trained on a range of images for a comparable purpose.

1.2.4 APPLICATION

1. Image Recognition

Image recognition, which is an approach for cataloging and detecting a feature or an object in the digital image, is one of the most significant and notable machine learning and AI techniques. This technique is being adopted for further analysis, such as pattern recognition, face detection, and face recognition.

2. Sentiment Analysis

Sentiment analysis is one of the most necessary applications of machine learning. Sentiment analysis is a real-time machine learning application that determines the emotion or opinion of the speaker or the writer.

1.2.5 CHALLENGES

1. Poor Data Quality

In the process of machine learning, data is crucial. The lack of high-quality data is one of the major problems that machine learning experts deal with. A messy or noisy set of data might make the entire procedure quite taxing. We do not want our system to produce erroneous or imprecise predictions. Therefore, to improve the result, the quality of the data is crucial.

2. Underfitting of Training Data

This process occurs when data is unable to establish an accurate relationship between input and output variables. It simply means trying to fit in undersized jeans.
It signifies the data is too simple to establish a precise relationship.

3. Overfitting of Training Data

Overfitting refers to a machine learning model trained with a massive amount of data that negatively affect its performance. It is like trying to fit in

Oversized jeans. Unfortunately, this is one of the significant issues faced by machine learning professionals. This means that the algorithm is trained with noisy and biased data, which will affect its overall performance. Let's understand this with the help of an example.

4. Overfitting of Training Data

Overfitting refers to a machine learning model trained with a massive amount of data that negatively affect its performance. It is like trying to fit in Oversized jeans. Unfortunately, this is one of the significant issues faced by machine learning professionals. This means that the algorithm is trained with noisy and biased data, which will affect its overall performance.

1.3 COMPUTER VISION

Computer vision is one of the fields of artificial intelligence that trains and enables computers to understand the visual world. Computers can use digital images and deep learning models to accurately identify and classify objects and react to them.

Computer vision in AI is dedicated to the development of automated systems that can interpret visual data (such as photographs or motion pictures) in the same manner as people do. The idea behind computer vision is to instruct computers to interpret and comprehend images on a pixel-by-pixel basis.

1.3.1 HOW DOES COMPUTER VISION WORK?

Massive amounts of information are required for computer vision. Repeated data analyses are performed until the system can differentiate between objects and identify visuals. Deep learning, a specific kind of machine learning,

and convolutional neural networks, an important form of a neural network, are the two key techniques that are used to achieve this goal.

With the help of pre-programmed algorithmic frameworks, a machine learning system may automatically learn about the interpretation of visual data. The model can learn to distinguish between similar pictures if it is given a large enough dataset. Algorithms make it possible for the system to learn on its own, so that it may replace human labour in tasks like image recognition.

1.3.2 HISTORY

For almost 60 years, researchers and developers have sought to teach computers how to perceive and make sense of visual information. In 1959, neurophysiologists started showing a cat a variety of sights in an effort to correlate a reaction in the animal's brain. They found that it was particularly sensitive to sharp corners and lines, which technically indicates that straight lines and other basic forms are the foundation upon which image analysis is built.

Around the same period, the first image-scanning technology emerged that enabled computers to scan images and obtain digital copies of them. This gave computers the ability to digitize and store images. In the 1960s, artificial intelligence (AI) emerged as an area of research, and the effort to address AI's inability to mimic human vision began.

1.3.3 DEEP LEARNING REVOLUTION

Examining the algorithms upon which modern computer vision technology is based is essential to understanding its development. Deep learning is a kind of machine learning that modern computer vision utilizes to get data-based insights.

When it comes to computer vision, deep learning is the way to go. An algorithm known as a neural network is used. Patterns in the data are extracted using neural networks.

1.3.4 COMPUTER VISION APPLICATIONS

One field of Machine Learning where fundamental ideas are already included in mainstream products is computer vision. The applications include:

Self-Driving Cars

With the use of computer vision, autonomous vehicles can understand their environment. Multiple cameras record the environment surrounding the vehicle, which is then sent into computer vision algorithms that analyzes the photos in perfect sync to locate road edges, decipher signposts, and see other vehicles, obstacles, and people.

Facial Recognition

Facial recognition programs, which use computer vision to recognize individuals in photographs, rely heavily on this field of study. Facial traits in photos are identified by computer vision algorithms, which then match those aspects to stored face profiles.

Augmented & Mixed Reality

Augmented reality, which allows computers like smartphones and wearable technology to superimpose or embed digital content onto real-world environments, also relies heavily on computer vision. Virtual items may be placed in the actual environment through computer vision in augmented reality equipment.

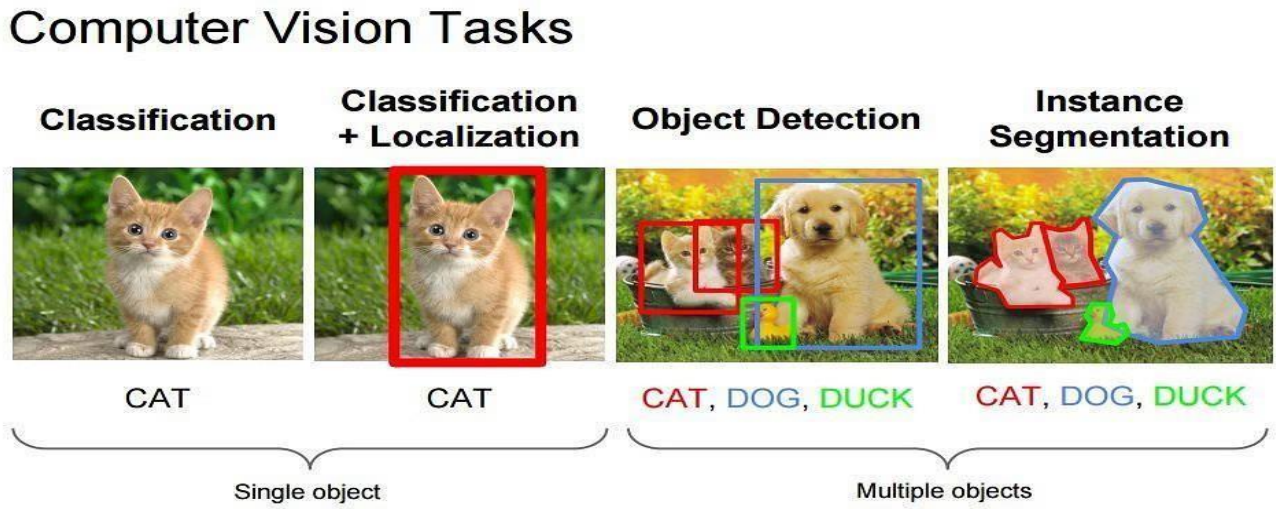


Fig. 1.3.5 Computer Vision Task

1.3.5 COMPUTER VISION AND IMAGE PROCESSING

Computer vision (CV) and image processing are two closely related fields that utilize techniques from artificial intelligence (AI) and pattern recognition to derive meaningful information from images, videos, and other visual inputs.

Image processing is at the helm of the technical revolution that is being called Industry 4.0, laying the framework for new technologies in image data processing and object recognition.

Computer vision uses image processing to recognize and categorize image data. In fact, CV is becoming more adept at identifying patterns from images than

the human visual cognitive system. CV technology is being used across industries from healthcare and media to gaming and transportation.

1.4 EVALUATION CRITERIA

Evaluation metrics are quantitative measures used to assess the performance and effectiveness of a statistical or machine learning model. These metrics provide insights into how well the model is performing and help in comparing different models or algorithms.

1.4.1 ACCURACY SCORE

Accuracy is used in classification problems to tell the percentage of correct predictions made by a model. Accuracy score in machine learning is an evaluation metric that measures the number of correct predictions made by a model in relation to the total number of predictions made.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

1.4.2 PRECISION

Precision is one indicator of a machine learning model's performance – the quality of a positive prediction made by the model. Precision refers to the number of true positives divided by the total number of positive predictions (i.e., the number of true positives plus the number of false positives).

1.4.3 CONFUSION MATRIX

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing.

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

The above table has the following cases:

- **True Negative:** Model has given prediction No, and the real or actual value was also No.
- **True Positive:** The model has predicted yes, and the actual value was also true.
- **False Negative:** The model has predicted no, but the actual value was Yes, it is also called as **Type-II error**.

- **False Positive:** The model has predicted Yes, but the actual value was No. It is also called a **Type-I error.**

1.4.4 HEATMAP

A Heat map is a graphical representation of multivariate data that is structured as a matrix of columns and rows. Heat maps are very useful in

describing correlation among several numerical variables, visualizing patterns and anomalies.

1.4.5 SUPPORT

Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing.

1.5 PYTHON LIBRARIES

NumPy

NumPy is an essential tool for anyone involved in scientific computing with Python. Its powerful features, high performance, and extensive functionalities make it a versatile and indispensable resource for scientific research, data analysis, and engineering applications.

Pandas

A Python package called Pandas is used to work with data sets. Its features include data manipulation, cleaning, analysis, and exploration. Using statistical theories, pandas enable us to analyze large amounts of data and draw conclusions. Pandas can tidy up unstructured data sets, making them more meaningful and readable.

OS

Python's OS module offers functions to communicate with the operating system. Python's standard utility modules include OS. This module offers an approach to use operating system-dependent functionality that is portable.

Matplotlib

A complete Python visualisation toolkit for static, animated, and interactive graphics is called Matplotlib. Matplotlib enables both difficult and easy tasks.

Make plots fit for a publication. Create dynamic figures with zoom, pan, and update capabilities.

Personalise the layout and visual style.

Seaborn

A Python library for data visualisation called Seaborn is built on top of matplotlib. It offers an advanced graphical user interface for creating eye-catching and educational statistical illustrations.You can read the introduction notes or the paper for a quick overview of the concepts underlying the library. channel for general support queries.

Tensorflow

An open-source software library is called TensorFlow. Although TensorFlow was initially created for machine learning and deep neural network research by scientists and engineers on the Google Brain Team within Google's Machine Intelligence research organisation, the system is sufficiently versatile to be used in a wide range of other fields as well!

Keras

Keras is a high-level deep learning API developed by Google for implementing sneural networks. It is used to simplify the implementation of neural networks and is written in Python. Multiple backend neural network computation is also supported.

CHAPTER 2

LITERATURE REVIEW

1. UNDERWATER FISH SPECIES CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK AND DEEP LEARNING

PUBLICATION YEAR: 2017

AUTHORS: S. Indu,S. Jain, and D. Rathi

SUMMARY

The target of this paper is to recommend a way for Automated classification of Fish species. A high accuracy fish classification is required for greater understanding of fish behaviour in Ichthyology and by marine biologists. Maintaining a ledger of the number of fishes per species and marking the endangered ssspecies in large and small water bodies is required by concerned institutions. Majority of available methods focus on classification of fishes outside of water because underwater classification poses challenges such as background noises, distortion of images, the presence of other water bodies in images, image quality and occlusion. This method uses a novel technique based on Convolutional Neural Networks, Deep Learning and Image Processing to achieve an accuracy of 96.29%. This method ensures considerably discrimination accuracy improvements than the previously proposed methods.

2. UNDERWATER FISH DETECTION WITH WEAK MULTI-DOMAIN SUPERVISION

PUBLICATION YEAR: 2019

AUTHORS: M.Bradley , D. A. Konovalov, S. Marini A. Saleh, M. Sankupellay, M. Sheave

SUMMARY

Modern convolutional neural networks are reasonably simple to train given a sizable enough training dataset. A necessary image classifier is a neural network (CNN). But for the purpose of classifying fish and/or Fish detection: When applied to new or unseen fish species and/or fish habitats, a CNN that was trained to identify or categorise specific fish species in specific background habitats shows relatively decreased accuracy. environments. As a result, in order to handle new fish species or habitats unique to a project, the CNN must be regularly adjusted to increase its classification accuracy. In this study, we introduce an effective labeling technique for training a CNN-based fish detector (using the Exception CNN as the foundation) using a very small dataset (4,000) of underwater fish and non-fish photos from 20 distinct habitats in the project domain.

3. A REALISTIC FISH-HABITAT DATASET TO EVALUATE ALGORITHMS FOR UNDERWATER VISUAL ANALYSIS

PUBLICATION YEAR: 2020

AUTHORS: Alzayat, A.Konovalov, H.Laradji,Dmitry, Michael Bradley, Saleh Issam

SUMMARY

A critical first step toward sustainable fisheries for human consumption and Environmental protection is the visual study of complex fish habitats. When trained on large-scale datasets, deep learning techniques have demonstrated considerable potential for scene interpretation. Nevheless, the majority of fish analysis datasets available today concentrate on the classification problem in limited, simple contexts, failing to adequately represent the complexity of fish habitats below the surface. We provide DeepFish as a benchmark suite with a large-scale dataset to train and test techniques for various computer vision tasks in order to address this constraint. About 40,000 underwater photos from 20 different green habitats in the marine environments of tropical Australia make up the dataset. At first, the dataset merely included labels for classification. We therefore gathered segmentation and point- level labels to create a more thorough fish

4. FISH SURVEYS ON THE MOVE: ADAPTING AUTOMATED FISH DETECTION AND CLASSIFICATION FRAMEWORKS FOR VIDEOS ON A REMOTELY OPERATED VEHICLE IN SHALLOW MARINE WATERS

PUBLICATION YEAR: 2022

AUTHORS: Cesar Herrera, Kristen I. Jinks, Rod M. Connolly, Sebastián Lopez-Marcano

SUMMARY

Fisheries monitoring has seen a rise in the use of mobile underwater cameras, either on submersible vehicles or controlled by divers. There are several benefits to replacing divers with cameras, including the ability to enter waters

inaccessible to divers and make permanent records. However, using cameras usually results in massive amounts of video that require a lot of time to manually process. Deep learning techniques for automated underwater video analysis from stationary cameras have come a long way in the last several years, but the utilisation of mobile cameras may provide new difficulties for current approaches. Using a mobile camera, we evaluated the efficacy of three automation protocols designed for stationary underwater cameras. These processes prioritise object-centricity over background-centricity while conducting fish surveys. Using drone footage of seagrass and reef environments, we counted and identified two marine fish species.

5. ANALYSIS OF UNDERWATER IMAGE PROCESSING METHODS FOR ANNOTATION IN DEEP LEARNING BASED FISH DETECTION

PUBLICATION YEAR: 2022

AUTHORS: Jose-Luis, Lisani,Ana-Belen Petro,Catalina Sbert, Amaya Alvarez-Ellacuria

SUMMARY

Fisheries monitoring has seen a rise in the use of mobile underwater cameras, either on submersible vehicles or controlled by divers. There are many benefits to replacing divers with cameras, including the ability to enter waters inaccessible to divers and make permanent records. However, using cameras usually results in massive amounts of video that require a lot of time to manually process. While deep learning techniques have made significant progress in recent years in automating the analysis of underwater recordings from

stationary cameras, the usage of mobile cameras may provide new hurdles for current approaches. Using a mobile camera, we evaluated the effectiveness of three automation protocols designed for stationary underwater cameras. These processes prioritise object-centricity over background-centricity while conducting fish surveys. We identified and counted two marine fisheries species by examining underwear drone footage from reef and seagrass area.

6. AUTOMATIC DETECTION AND CLASSIFICATION OF COASTAL MEDITERRANEAN FISH FROM UNDERWATER IMAGES: GOOD PRACTICES FOR ROBUST TRAINING

PUBLICATION YEAR: 2023

AUTHORS: Ignacio A. Catala, Amaya Alvarez-Ellacurıa, JoseLuis, Lisani Josep Sanchez

SUMMARY

Fisheries monitoring has seen a rise in the use of mobile underwater cameras, either on submersible vehicles or controlled by divers. There are many benefits to replacing divers with cameras, including the ability to enter waters inaccessible to divers and make permanent records. However, using cameras usually results in massive amounts of video that require a lot of time to manually process. While deep learning techniques have made significant progress in recent years in automating the analysis of underwater recordings from stationary cameras, the usage of mobile cameras may provide new hurdles for current approaches. Using a mobile camera, we evaluated the effectiveness of three automation protocols designed for stationary underwater cameras. These processes prioritise object-centricity over background- centricity while conducting fish surveys. We identified and counted two marine fisheries species by examining underwear drone footage from reef and seagrass area.

CHAPTER 3
SYSTEM ANALYSIS

The process of analyzing the system that existed and alterations that are made in the proposed system is stated in system analysis

3.1 EXISTING SYSTEM

In the existing system, various components and techniques have been included to detect and concentrate on freshwater fish species identification. The identification of fish species is one of the most in-demand applications of underwater imagery processing. This activity is crucial for education purposes in addition to being important for scientists, biologists, and fishermen. According to reports, Malaysia is home to over 200 different species of freshwater fish. Many efforts have been made to improve fish recognition and classification using image processing techniques; however, the majority of the work that has already been done has been developed for the identification of saltwater fish species and is intended for a particular user base. The main goal of this research project is to create a prototype system called FishDeTec that uses image processing to identify the freshwater fish species that can be found in Malaysia.

The VGG16 deep convolutional neural network model (CNN), An artificial neural network that is a type of convolutional neural network is called a ConvNet. An input layer, an output layer, and several hidden layers make up a convolutional neural network. Convolutional neural networks, or CNNs, like the VGG16 model are thought to be among the best computer vision models available today. The model's developers assessed the networks and used an architecture with minuscule (3 × 3) convolution filters to increase the depth, demonstrating a notable advance

over previous state-of-the-art configurations. They increased the depth to roughly 138 trainable parameters by pushing it to 16–19 weight layers.

VGG16, while powerful, suffers from limitations in speed and memory usage compared to MobileNet v2. Its larger number of parameters and traditional convolutional layers make it computationally expensive and require more memory, hindering its suitability for resource-constrained environments and real-time applications. In contrast, MobileNet v2's efficient architecture, including depth wise separable convolutions, significantly reduces computational cost and memory footprint, making it ideal for deployment on mobile devices and edge computing platforms.

In Random Forest, three subsets are created at random with replacement of original data. The variables are selected at random and the best split to the node is made. In this method, there is no danger of over-fitting.

3.1.1 DISADVANTAGES

- High Computational Cost
- High Memory Usage
- Susceptibility to Overfitting
- Limited Efficiency
- Less Suitable for Real-Time Applications
- Outdated Architecture

3.2 PROPOSED SYSTEM

The Proposed system is fully computerised, which removes all the drawbacks of the existing system. Our proposed system takes into consideration of all factor such as Precision, Recall, F1 score in order to provide the accuracy.

Marine scientists and conservationists have important responsibilities, such as tracking population fluctuations and routinely evaluating the relative abundance of fish species in their environs. In place of labor-intensive manual collection, a variety of autonomous computer-based techniques for gathering fish samples have been illustrated in underwater images and videos.

Nevertheless, there is still no foolproof system for automatically categorizing and recognizing fish. This is primarily because it can be challenging to capture clear underwater photos and videos due to a variety of factors, such as shifting lighting, fish hiding, shifting backgrounds, murky water, low resolution, deformed fish, and subtle variations amongst some fish species. We developed our technology to overcome these limitations by effectively identifying and recognizing fish using the MobileNet v2 Architecture. A contemporary convolutional neural network (CNN) architecture created especially for mobile and embedded devices is called MobileNet v2. It prioritizes resource optimization and efficiency, thereby addressing the shortcomings of the VGG16 and other conventional architectures.

Depth wise separable convolutions, which drastically cut down on computations and parameters over conventional convolutions, are used by MobileNet v2. Because of the smaller model size and faster inference speed that arise from this, it can be used in environments with limited resources. MobileNet v2 has a much smaller model size (~10MB) than VGG16, which has a massive size (~530MB).

This enables effective deployment and storage on low-power and mobile devices. Compared to VGG16, MobileNet v2's efficient architecture results in faster inference speed and less memory usage.

MobileNetv2 is chosen for its lightweight architecture, making it more efficient in terms of speed and memory usage compared to VGG16.

3.2.1 ADVANTAGES

- Transfer learning
- High Accuracy
- Fast Inference Speed
- Small Model Size
- Scalability
- Open-Source Availability

CHAPTER 4

ARCHITECTURE

4.1 DESIGN

Project design is an early phase of the project lifecycle where ideas, processes, resources, and deliverables are planned out. A project design comes before a project plan as it's a broad overview whereas a project plan includes more detailed information. Project design is an important piece of executing a successful project. From gathering the necessary information and resources to coordinating with team members, your job is to bring the details to life. With the right project design, you and your team can tackle anything that comes your way.

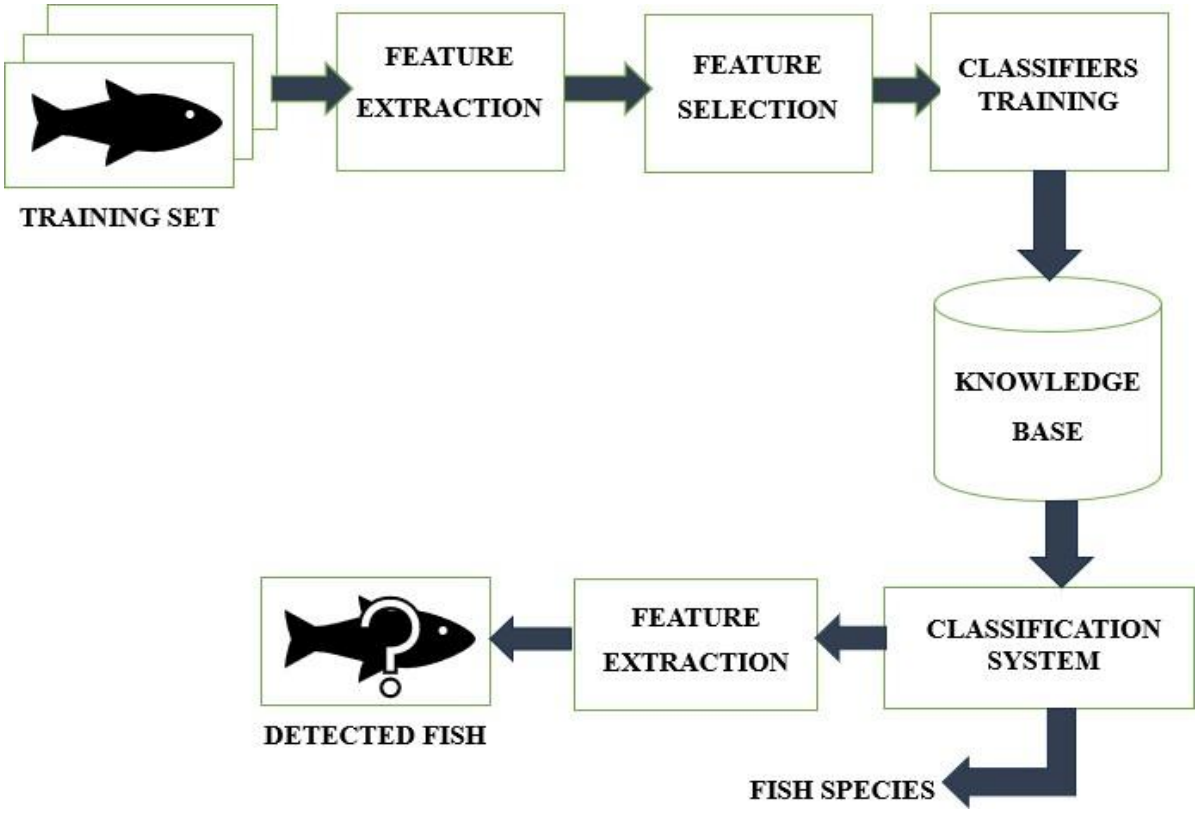


Fig. 4.1 Design

4.2 FLOW CHART

A flowchart (or flow chart) is a diagram that shows the steps in a process. Flowcharts are often used for visualizing the sequence of actions or information needed for training, documenting, planning, and decision-making. They often use symbols, shapes, and arrows to illustrate how one step leads to another.

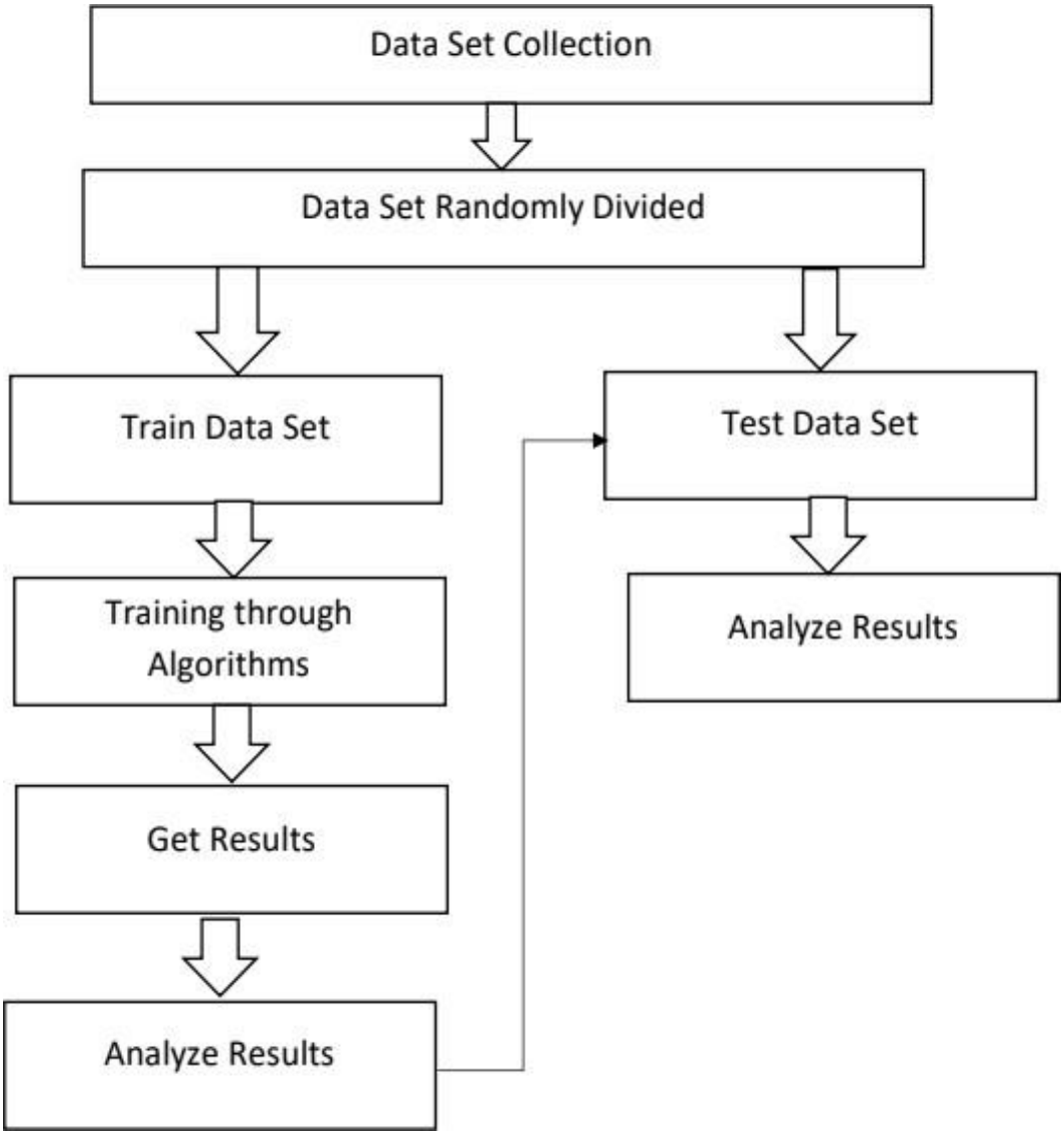


Fig. 4.2 Flow Chart

4.3 UML DIAGRAM

4.3.1 USE CASE DIAGRAM

Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

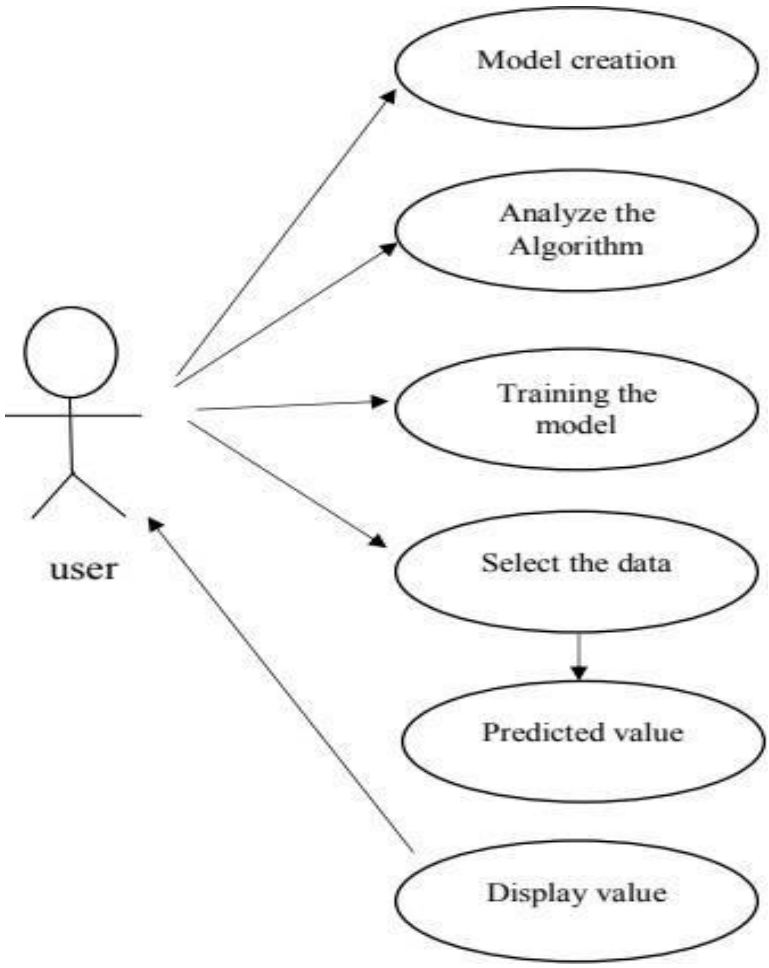


Fig. 4.3.1 Use Case Diagram

4.3.2 ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So, the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

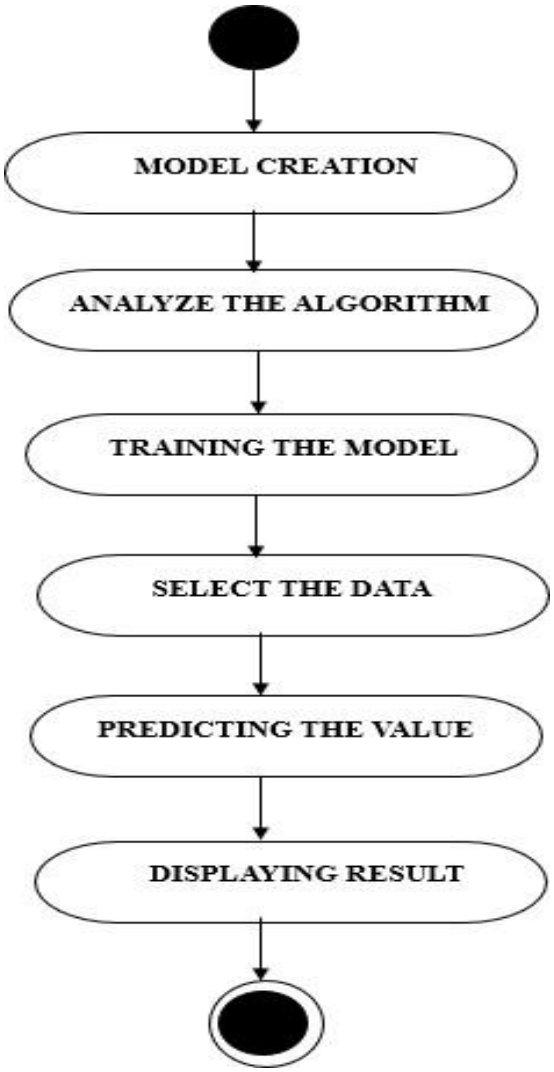


Fig. 4.3.2 Activity Diagram

4.3.3 SEQUENCE DIAGRAM

A Sequence diagram is an interaction diagram that's how show objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

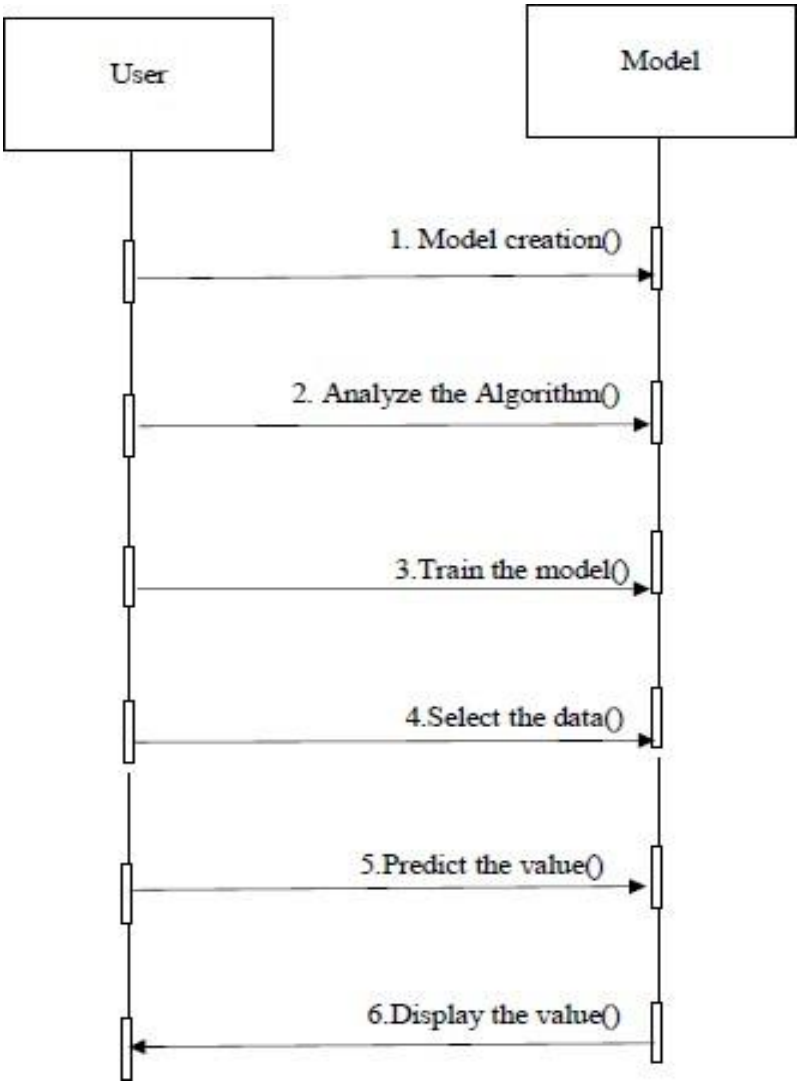


Fig. 4.3.3 Sequence Diagram

CHAPTER 5
SYSTEM CONFIGURATION

5.1 HARDWARE REQUIREMENTS

- **Processor :** 11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz 2.90 GHz
- **RAM :** 8.00 GB
- **Hard disk :** 237 GB

5.2 SOFTWARE REQUIREMENTS

- **Operating System :** Windows OS ● **Front End :** Python
- **IDE :** Google colab

5.3 PYTHON

Python is a versatile, high-level programming language known for its readability and ease of use. Developed by Guido van Rossum in 1991, Python is widely used for various purposes, including web development, data science, artificial intelligence, machine learning, and scientific computing. Noteworthy features include its clear syntax, dynamic typing, and emphasis on code readability. In Python, indentation plays a crucial role in indicating code blocks, and the language supports a range of data types such as integers, floats, strings, lists, and dictionaries. Control flow is facilitated through constructs like if statements and loops, and the language supports object-oriented programming principles with classes and objects.

5.3.1 KEY ASPECTS OF PYTHON

- **Readability:** Python's syntax is designed to be clear and readable, with an emphasis on code readability and a clean, visual layout.
- **Versatility:** Python is a general-purpose language used for web development, data science, artificial intelligence, machine learning, automation, scientific computing, and more.
- **Interpreted:** Python is an interpreted language, meaning that the source code is executed line by line by the Python interpreter without the need for compilation.
- **Syntax:** Python uses indentation to indicate code blocks (whitespace matters). Code blocks are defined using colons (`:`), and indentation is typically four spaces.
- **Data Types:** Common data types include integers, floats, strings, lists, tuples, sets, and dictionaries. Python has dynamic typing, allowing variables to change types during execution.

5.4 GOOGLE COLAB

Google Colab, short for Google Colaboratory, is a cloud-based platform provided by Google that allows users to write and execute Python code in a collaborative environment. It's particularly popular among data scientists, machine learning researchers, and educators. Here are some,

Key features and aspects of Google Colab

- **Integration with Google Drive:**Colab is integrated with Google Drive, allowing users to save their work directly to Google Drive. This makes it easy to share and collaborate on notebooks.
- **Pre-installed Libraries:** Colab comes with many pre-installed Python libraries commonly used in data science and machine learning, such as NumPy, pandas, TensorFlow, and Matplotlib. This saves users the time and effort of installing these libraries manually.
- **Easy Sharing and Collaboration:**Notebooks can be easily shared with others, similar to Google Docs. Multiple users can collaborate on the same notebook in real-time, making it a powerful tool for collaborative work.
- **Access to BigQuery:**Colab provides integration with Google BigQuery, allowing users to query and analyze large datasets directly from their notebooks.
- **Support for Markdown and HTML:** In addition to code cells, Colab supports cells with Markdown and HTML, allowing users to create rich-text documents with embedded code and visualizations.
- **Interactive Visualizations:** Colab supports interactive visualizations with libraries like Matplotlib, Plotly, and others. This makes it easy to create and explore data visualizations within the notebook.
- **Form Features:** Colab supports form features, allowing users to create interactive forms within their notebooks. This can be useful for parameterizing code and making it more user-friendly.

CHAPTER 6

MODULES

6.1 COLLECTING AND ANALYZING DATA SETS

Collecting and analyzing datasets is a fundamental aspect of data science and various other fields, including machine learning, statistics, and business intelligence. The process involves gathering relevant data, organizing it, and extracting meaningful insights. In this module datasets are collected from Kaggle for Fish species identification & Detection. The data set consists of 9 varieties of fish species each species with 1000 datasets.

6.2 EXPLORATORY ANALYSIS

Exploratory Data Analysis (EDA) is a crucial phase in the data analysis process where the primary goal is to gain insights into the dataset and understand 13its characteristics. EDA involves a variety of techniques and methods to summarize, visualize, and interpret key features and patterns within the data.

6.3 TEST-TRAIN SPLIT AND PREPROCESSING

The test-train split is essential in machine learning, dividing the dataset into training and evaluation subsets for assessing the model's generalization capabilities. Preprocessing is crucial in computer vision tasks like image classification and object detection, involving image transformations to optimize the dataset for training models.

6.4 MODEL TRAINING AND EVALUATION- (MOBILENETV2 ARCHITECTURE)

MobileNetV2 is a convolutional neural network architecture that seeks to perform well on mobile devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depth wise convolutions to filter features as a source of non-linearity. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers.

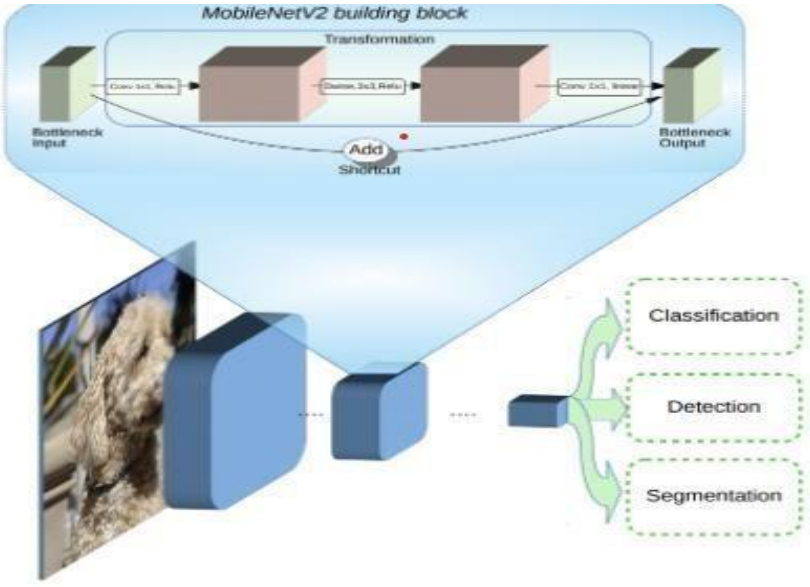
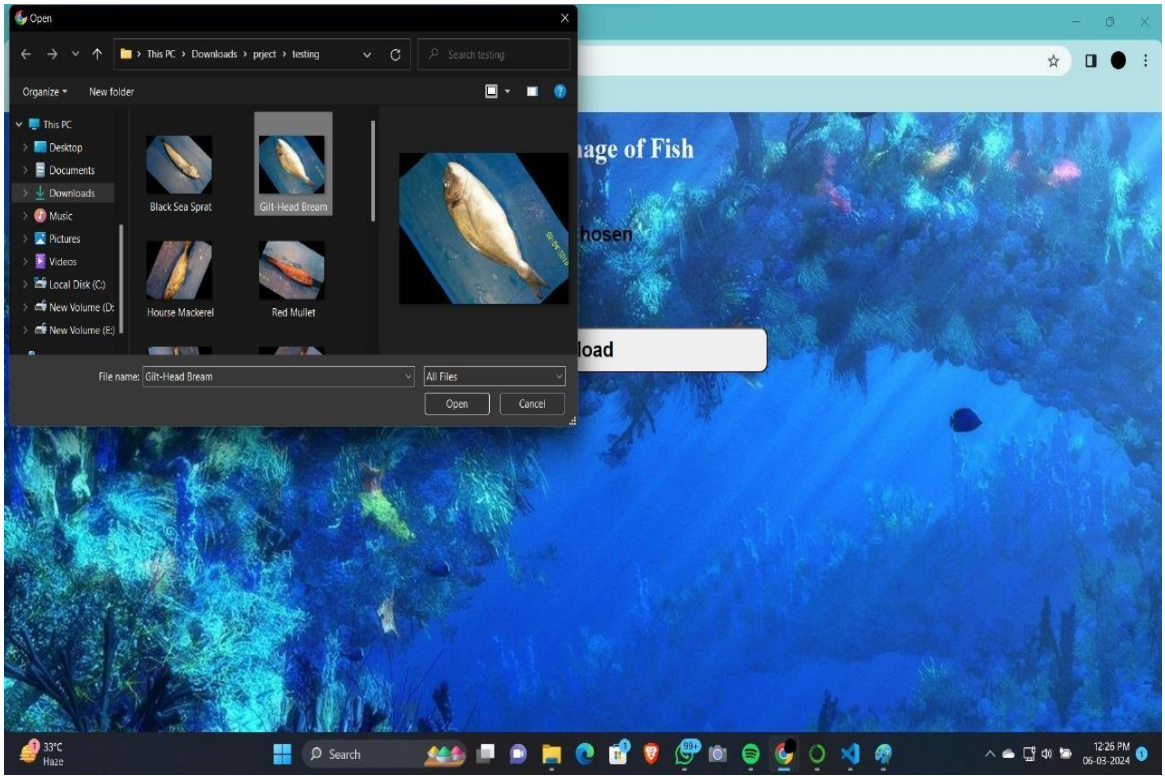


Fig. 6.5 MobileNetV2 Architecture

6.5 Integrated web development

The "Image Upload and Fish Species Identification" module develops a web application allowing users to upload fish images for species identification. Frontend development creates an intuitive interface for image upload, while backend logic securely handles uploads. The module includes dataset preprocessing and integrates machine learning models for classification. Upon upload, the system predicts the species and displays the image alongside its classification and additional species information. User interaction features enable species selection and feedback. Testing ensures accuracy before deployment, with reflections on challenges and future improvements.



CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

In this section, we assess the performance accuracy of our model, now enhanced with integrated web development features. Accuracy remains a crucial metric for evaluating classification models, depicted as a percentage to gauge the alignment of predictions with actual values. Meanwhile, loss functions like entropy loss and log loss quantify deviations between predicted and true values, offering essential insights for both regression and classification tasks. With the integration of web development functionalities, our model aims to streamline data analysis and visualization processes. Real-time data visualization and interactive features enhance user engagement and facilitate informed decision-making. Moreover, the inclusion of web technologies enables seamless deployment and accessibility across platforms, expanding the reach and usability of our solution. Ongoing iterations seek to improve accuracy and reduce loss, empowering researchers and conservationists with a comprehensive toolset for monitoring fish populations and advancing marine conservation efforts.

7.2 FUTURE ENHANCEMENT

In future, Allowing users to contribute their own fish images to expand the dataset and improve the system's ability to identify a wider variety of fish species.

APPENDIX

SAMPLE CODING

Importing necessary libraries and modules for this Notebook

```
'''Data Handling & Linear Algebra'''
import numpy as np import pandas
as pd import os
'''Visualisation''' import
matplotlib. pyplot as plt
import seaborn as sns
'''Manipulating Data and Model Building''' import
tensorflow as tf from tensorflow import keras from
tensorflow.keras import layers from
tensorflow.python.keras.layers import Dense from
tensorflow.keras.models import Sequential from
tensorflow.keras.optimizers import Adam from
sklearn.model_selection import train_test_split
'''Implementation of Transfer Learning''' from
tensorflow.keras.applications import MobileNetV2
'''Image generation and preprocessing''' from
tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import
ImageDataGenerator '''Result and Performance Analysis''' from
sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
```

'''Ignoring Warnings''' import
warnings
warnings.filterwarnings("ignore")

Exploratory Analysis

DIR = '/kaggle/input/a-large-scale-fish-dataset/Fish_Dataset/Fish_Dataset'
#Creating a master directory classes = [i for i
in os.listdir(DIR) if '.' not in i] #Creating
classes out of the required folders classes

label = [] path = [] for dirname, _,filenames in
os.walk(DIR): for filename in filenames: if
os.path.splitext(filename)[-1]=='.png': # If
filename contains .png if dirname.split()[-
1]!='GT': # If directory doesn't contain GT
label.append(os.path.split(dirname)[-1]) #
Append the directory name to label
path.append(os.path.join(dirname,filename)) #
Append all the png files to path of that
directory # Ready to create the df df =
pd.DataFrame(columns=['path','label'])
df['path']=path df['label']=label df.head()
df.path[0] df.info() df['label'].value_counts()
idx = 0 plt.figure(figsize=(15,12)) for
unique_label in df['label'].unique():

```
plt.subplot(3, 3, idx+1)
plt.imshow(plt.imread(df[df['label']==unique_label].iloc[0,0]))
plt.title(unique_label) plt.axis('off') idx+=1
```

Test-Train Split

```
train_df, test_df = train_test_split(df, train_size=0.8, shuffle=True, random_state=42) print(train_df.shape) print(test_df.shape)
```

Image Generation and Preprocessing

```
train_generator = ImageDataGenerator(preprocessing_function=preprocess_input, validation_split=0.2) test_generator =ImageDataGenerator(preprocessing_function=preprocess_input) train_images = train_generator.flow_from_dataframe(dataframe=train_df, x_col='path', y_col='label', target_size=(224, 224), color_mode='rgb',class_mode='categorical', batch_size=32, shuffle=True, seed=42,subset='training') val_images = train_generator.flow_from_dataframe(dataframe=train_df, x_col='path', y_col='label', target_size=(224, 224), color_mode='rgb', class_mode='categorical', batch_size=32, shuffle=True, seed=42,subset='validation' ) test_images = test_generator.flow_from_dataframe(dataframe=test_df, x_col='path', y_col='label', target_size=(224, 224), color_mode='rgb', class_mode='categorical', batch_size=32, shuffle=False ) display(train_images.class_indices) display(val_images.class_indices) display(test_images.class_indices) fit, ax= plt.subplots(nrows=2, ncols=3, figsize=(15,8)) ax=ax.flatten() j=0 for _ in range(6):
```

```
img, label = test_images.next()
ax[j].imshow(img[0],)
ax[j].set_title(label[0]) j=j+1
```

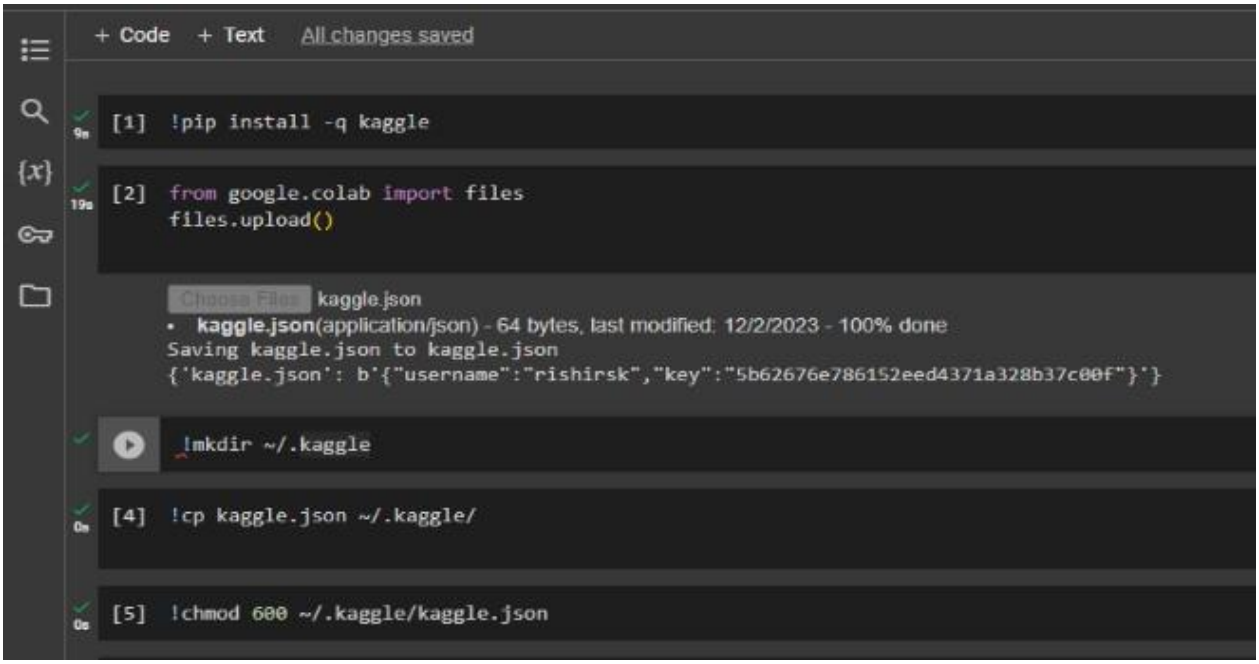
Model Training and Evaluation - MobileNet v2 Architecture

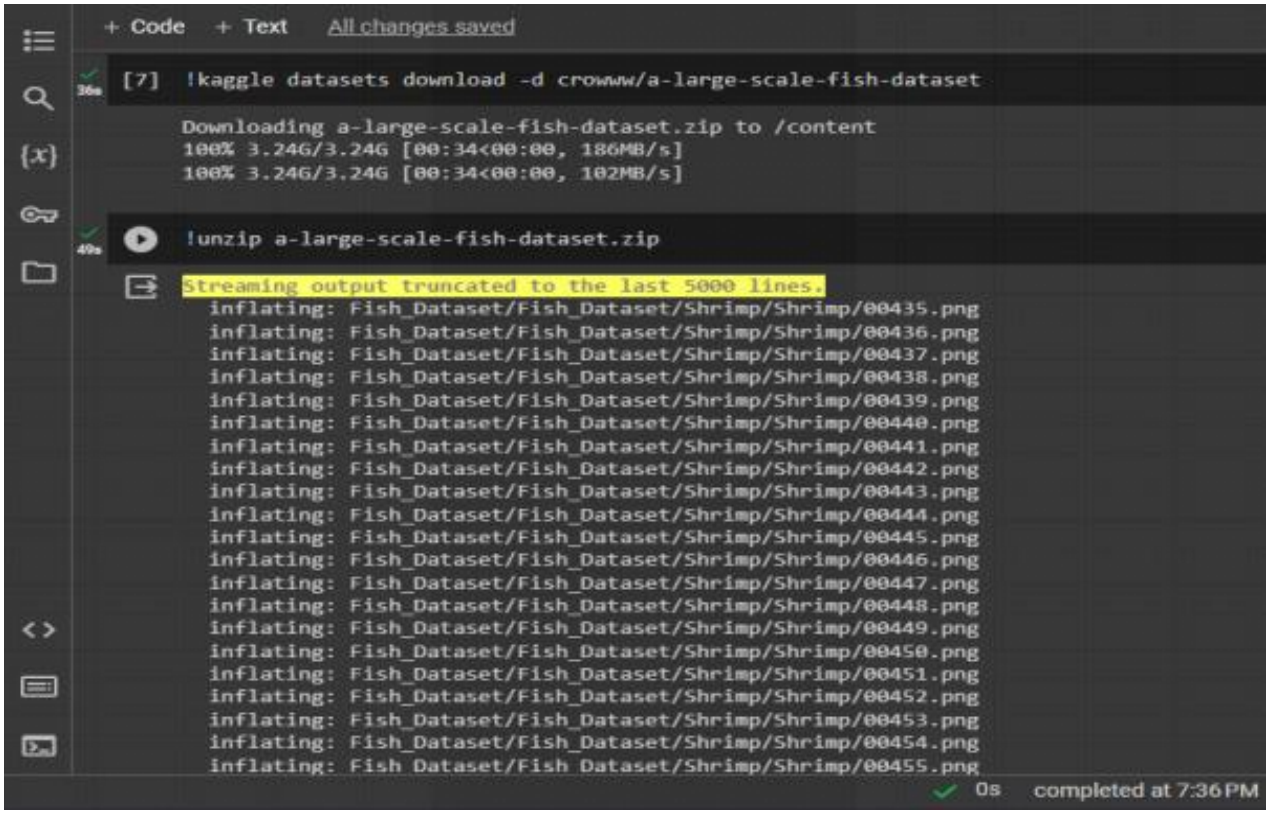
```
pretrained_model = tf.keras.applications.MobileNetV2(input_shape=(224, 224,
3),include_top=False, weights='imagenet',pooling='avg')
pretrained_model.trainable = False


inputs = pretrained_model.input x =
tf.keras.layers.Dense(128, activation='relu')
(pretrained_model.output) x = tf.keras.layers.Dense(128,
activation='relu')(x) outputs = tf.keras.layers.Dense(9,
activation='softmax')(x) model =
tf.keras.Model(inputs=inputs, outputs=outputs)

model.summary() model.compile(optimizer='adam',
loss='categorical_crossentropy',metrics=['accuracy'])
history = model.fit(train_images,validation_data=val_images,epochs=5)


results = model.evaluate(test_images, verbose=0)
print("Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

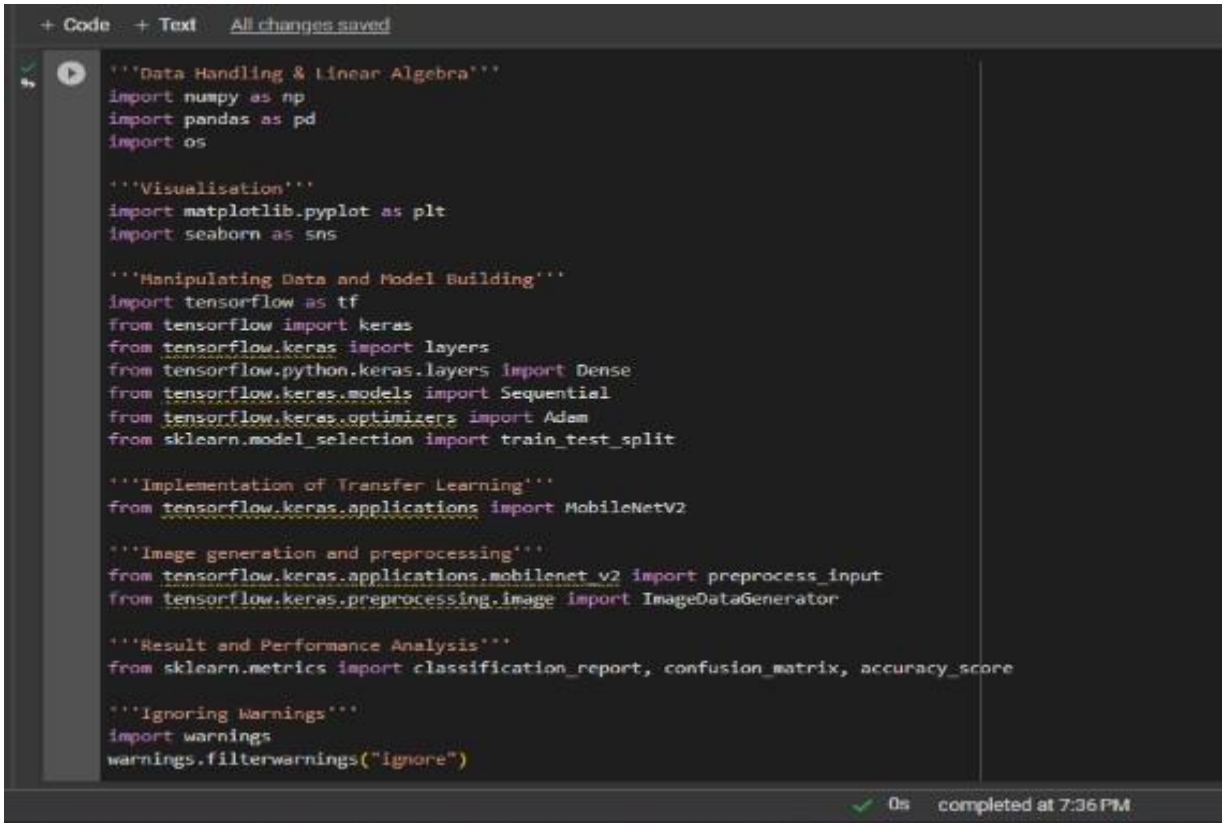
```
pred = model.predict(test_images)
pred=np.argmax(pred,axis=1) pred_df=test_df.copy()
labels={} for l,v in test_images.class_indices.items():
labels.update({v:l}) pred_df['pred']=pred
pred_df['pred']=pred_df['pred'].apply(lambda x: labels[x])
# Visualize the pred_df
pred_df=pred_df.reset_index(drop=True)
pred_df.head(10)

pred_df[pred_df['label']!=pred_df['pred']] pred_df.path[1604]
plt.figure(figsize=(15,8))
plt.imshow(plt.imread(pred_df.path[1604])) title1 =
pred_df.path[1604].split('/')[-2] title2 =
pred_df.path[1604].split('/')[-1] title3 = pred_df.pred[1604]
plt.title(f'Image:- {title2}\nTrue Class:- {title1}\nPredicted
Class:-
{title3}', color = 'r', weight = 'bold', fontsize = 15) plt.show()
print(f"Accuracy Score:{accuracy_score(pred_df['label'],pred_df['pred'])}")
plt.figure(figsize=(8,6))
sns.heatmap(confusion_matrix(pred_df['label'],pred_df['pred']),annot=True,
fmt='2d') print(classification_report(pred_df['label'],pred_df['pred']))
```

OUTPUT

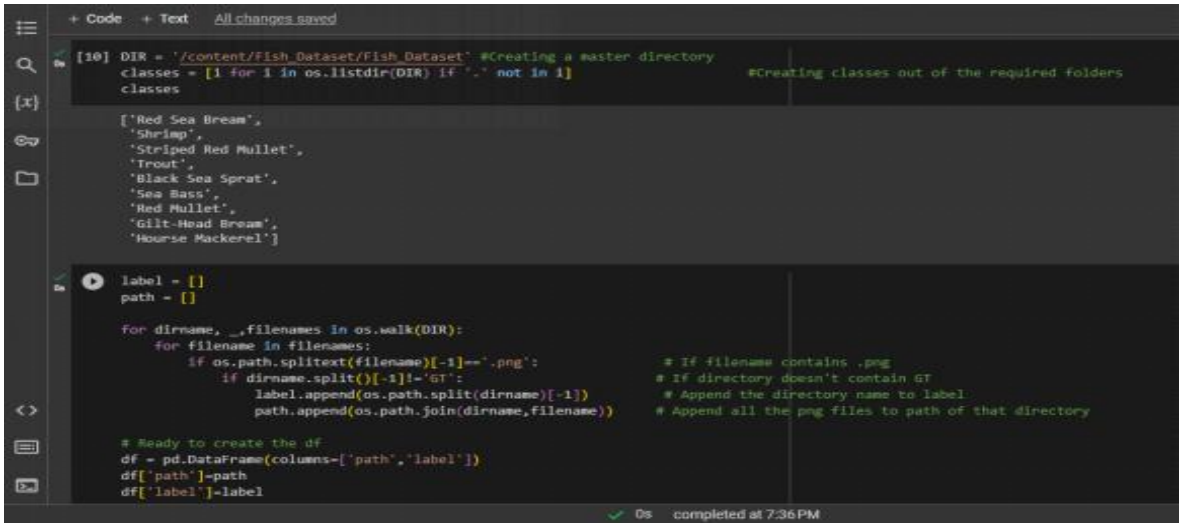
Collecting and Analyzing Datasets



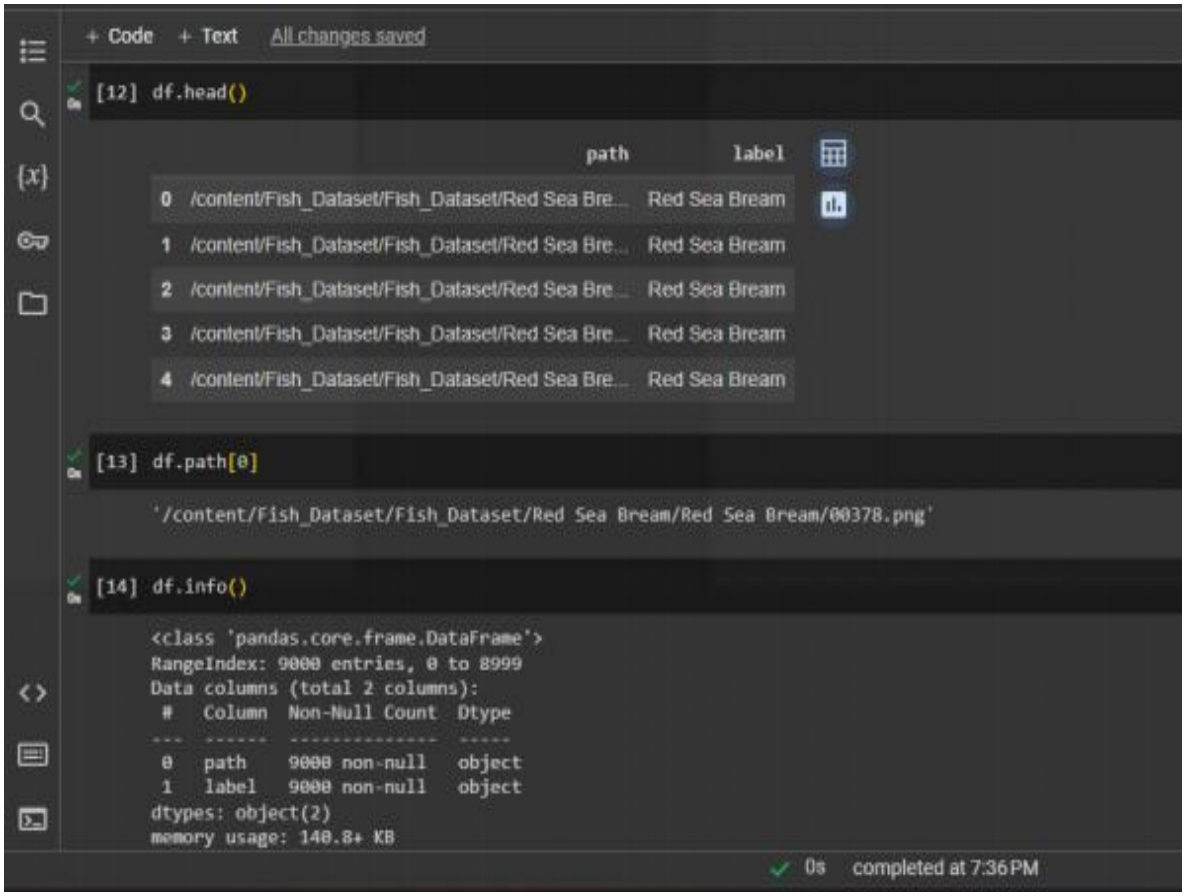

+ Code
+ Text
All changes saved
[7] !kaggle datasets download -d crowww/a-large-scale-fish-dataset
Downloading a-large-scale-fish-dataset.zip to /content
100% 3.24G/3.24G [00:34<00:00, 186MB/s]
100% 3.24G/3.24G [00:34<00:00, 102MB/s]
!unzip a-large-scale-fish-dataset.zip
Streaming output truncated to the last 5000 lines.
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00435.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00436.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00437.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00438.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00439.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00440.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00441.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00442.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00443.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00444.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00445.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00446.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00447.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00448.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00449.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00450.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00451.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00452.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00453.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00454.png
inflating: Fish_Dataset/Fish_Dataset/Shrimp/Shrimp/00455.png
0s completed at 7:36 PM



Exploratory Analysis



```
+ Code + Text  All changes saved

[10] DIR = '/content/Fish_Dataset/Fish_Dataset' #Creating a master directory
     classes = [i for i in os.listdir(DIR) if '.' not in i]          #Creating classes out of the required folders
     classes

     ['Red Sea Bream',
      'Shrimp',
      'Striped Red Mullet',
      'Trout',
      'Black Sea Sprat',
      'Sea Bass',
      'Red Mullet',
      'Gilt-Head Bream',
      'Hourse Mackerel']

     label = []
     path = []

     for dirname, _,filenames in os.walk(DIR):
         for filename in filenames:
             if os.path.splitext(filename)[-1]=='.png':              # If filename contains .png
                 if dirname.split()[-1]!='GT':                       # If directory doesn't contain GT
                     label.append(os.path.split(dirname)[-1])        # Append the directory name to label
                     path.append(os.path.join(dirname,filename))     # Append all the png files to path of that directory

     # Ready to create the df
     df = pd.DataFrame(columns=['path','label'])
     df['path']=path
     df['label']=label

✓ 0s  completed at 7:36 PM
```



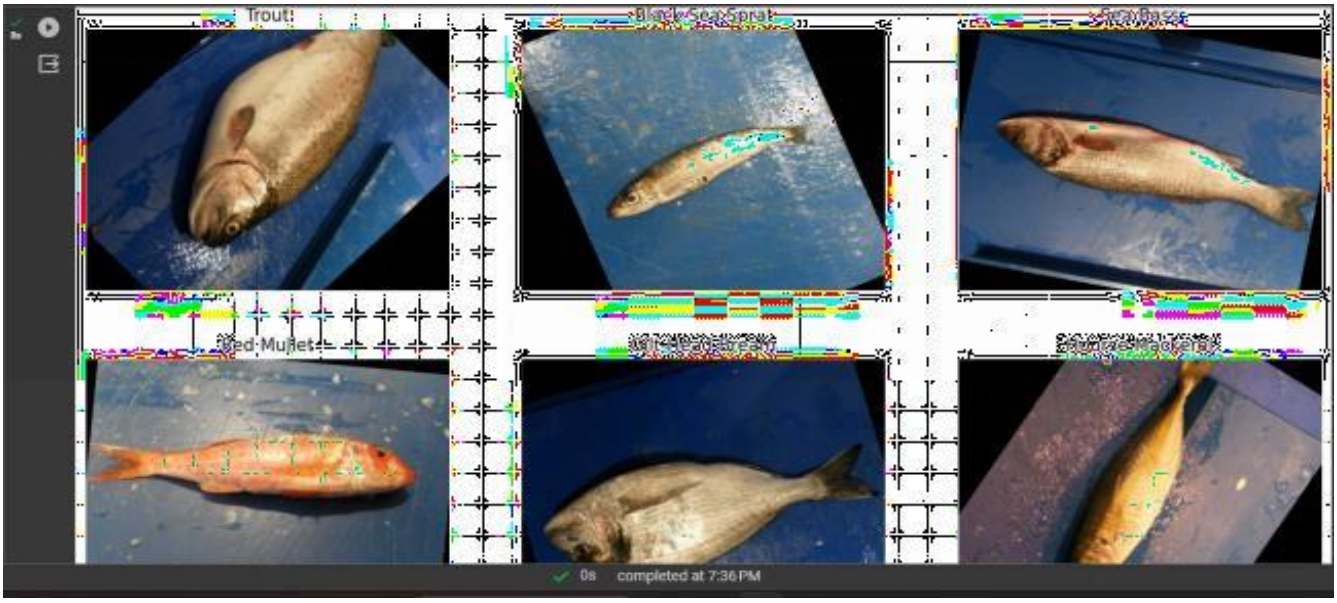
```
+ Code + Text  All changes saved

[12] df.head()
```

	path	label
0	/content/Fish_Dataset/Fish_Dataset/Red Sea Bre...	Red Sea Bream
1	/content/Fish_Dataset/Fish_Dataset/Red Sea Bre...	Red Sea Bream
2	/content/Fish_Dataset/Fish_Dataset/Red Sea Bre...	Red Sea Bream
3	/content/Fish_Dataset/Fish_Dataset/Red Sea Bre...	Red Sea Bream
4	/content/Fish_Dataset/Fish_Dataset/Red Sea Bre...	Red Sea Bream

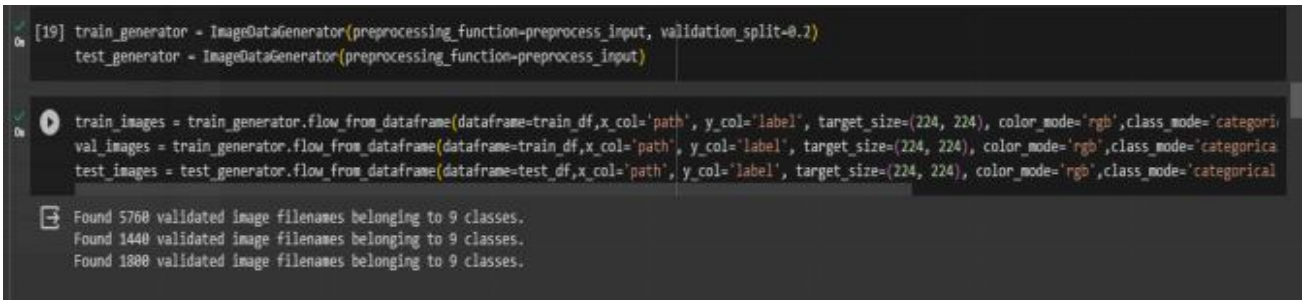
```
[13] df.path[0]

     '/content/Fish_Dataset/Fish_Dataset/Red Sea Bream/Red Sea Bream/00378.png'

[14] df.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 9000 entries, 0 to 8999
     Data columns (total 2 columns):
      #   Column  Non-Null Count  Dtype
     ---  ------  --------------  -----
      0   path    9000 non-null   object
      1   label   9000 non-null   object
     dtypes: object(2)
     memory usage: 140.8+ KB

✓ 0s  completed at 7:36 PM
```

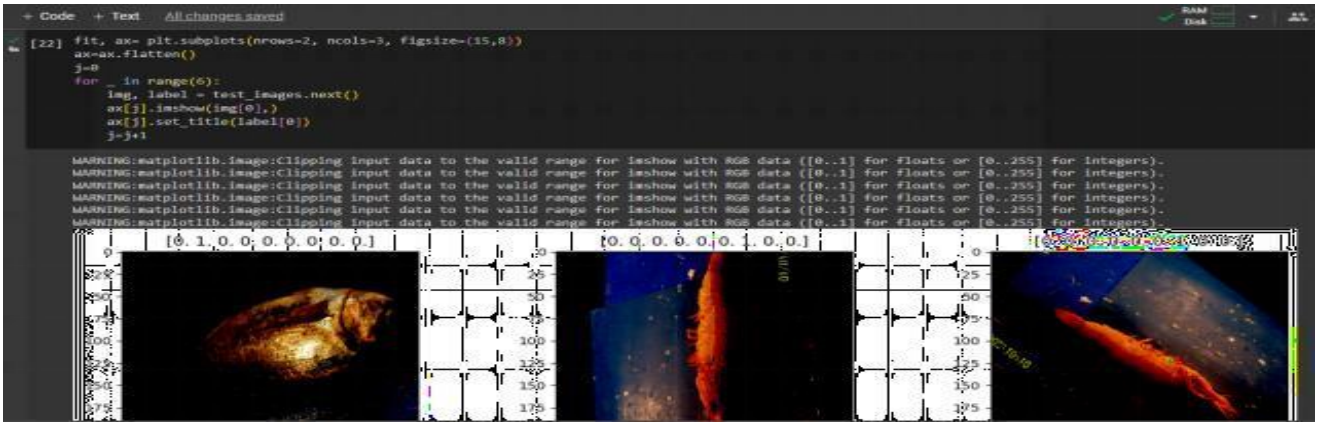
```
[15] df['label'].value_counts()

Red Sea Bream          1000
Shrimp                 1000
Striped Red Mullet     1000
Trout                  1000
Black Sea Sprat        1000
Sea Bass               1000
Red Mullet             1000
Gilt-Head Bream        1000
Hourse Mackerel        1000
Name: label, dtype: int64
```

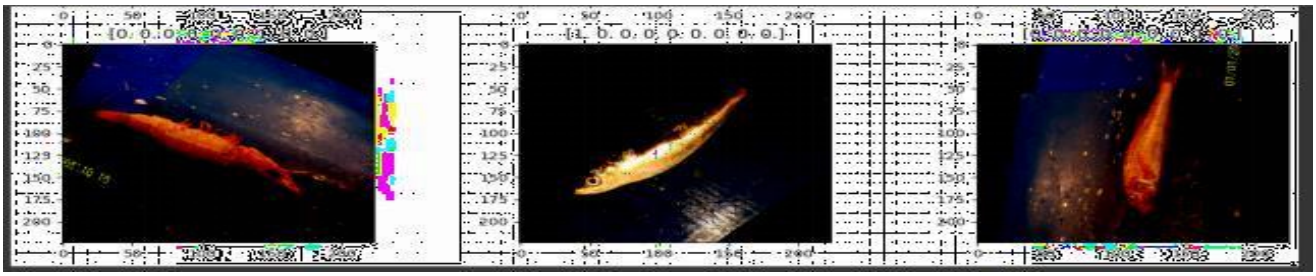
```
[16] idx = 0
     plt.figure(figsize=(15,12))
     for unique_label in df['label'].unique():
         plt.subplot(3, 3, idx+1)
         plt.imshow(plt.imread(df[df['label']==unique_label].iloc[0,0]))
         plt.title(unique_label)
         plt.axis('off')
         idx+=1
```



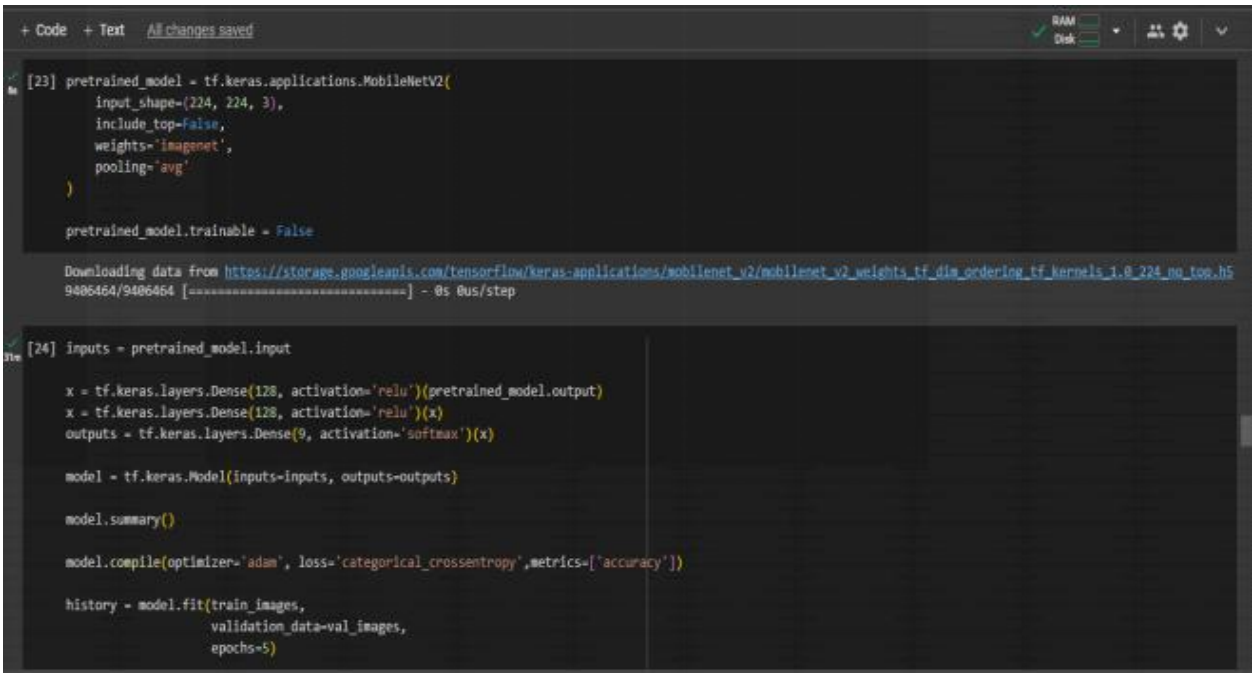
Test-Train Split and Preprocessing

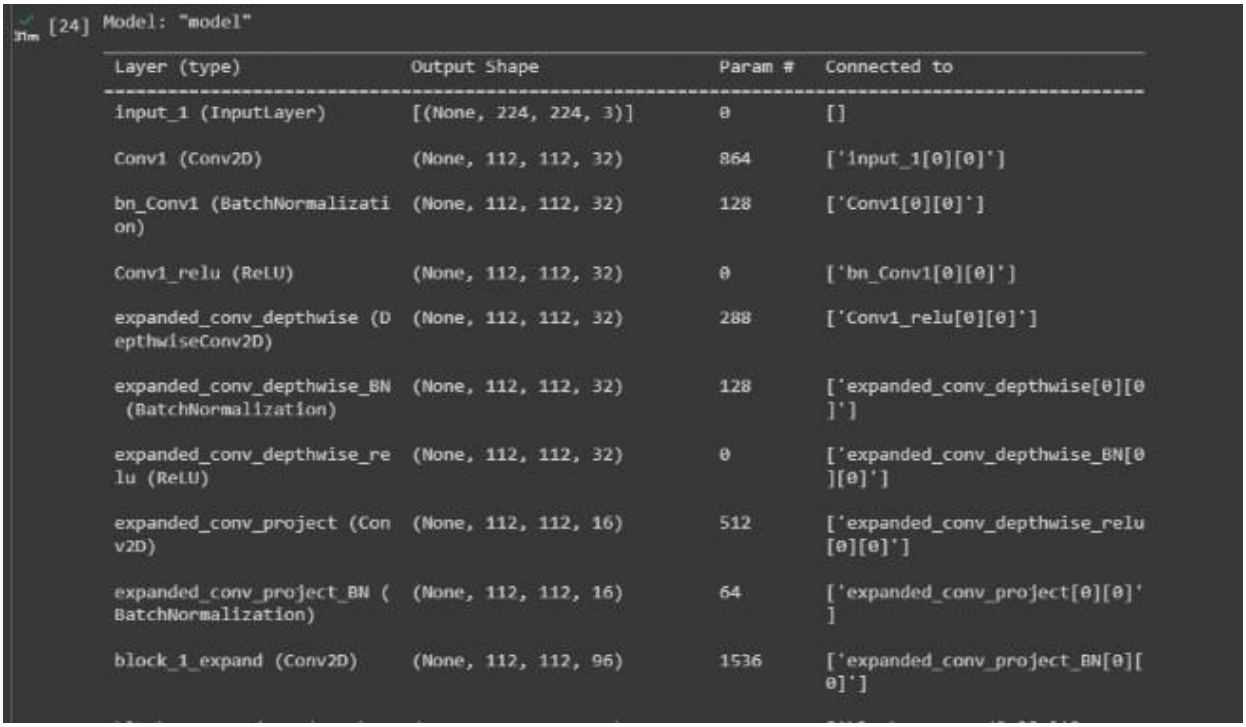






Model Training and Evaluation- (MobileNetv2 Architecture)





[24] Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)]	0	[]
Conv1 (Conv2D)	(None, 112, 112, 32)	864	['input_1[0][0]']
bn_Conv1 (BatchNormalization)	(None, 112, 112, 32)	128	['Conv1[0][0]']
Conv1_relu (ReLU)	(None, 112, 112, 32)	0	['bn_Conv1[0][0]']
expanded_conv_depthwise (DepthwiseConv2D)	(None, 112, 112, 32)	288	['Conv1_relu[0][0]']
expanded_conv_depthwise_BN (BatchNormalization)	(None, 112, 112, 32)	128	['expanded_conv_depthwise[0][0]']
expanded_conv_depthwise_relu (ReLU)	(None, 112, 112, 32)	0	['expanded_conv_depthwise_BN[0][0]']
expanded_conv_project (Conv2D)	(None, 112, 112, 16)	512	['expanded_conv_depthwise_relu[0][0]']
expanded_conv_project_BN (BatchNormalization)	(None, 112, 112, 16)	64	['expanded_conv_project[0][0]']
block_1_expand (Conv2D)	(None, 112, 112, 96)	1536	['expanded_conv_project_BN[0][0]']

```
[24]  block_1_expand_BN (BatchNo  (None, 112, 112, 96)   384   ['block_1_expand[0][0]']
      rmalization)

      block_1_expand_relu (ReLU)  (None, 112, 112, 96)   0     ['block_1_expand_BN[0][0]']

      block_1_pad (ZeroPadding2D  (None, 113, 113, 96)   0     ['block_1_expand_relu[0][0]']
      )

      block_1_depthwise (Depthwi  (None, 56, 56, 96)     864   ['block_1_pad[0][0]']
      seConv2D)

      block_1_depthwise_BN (Batc  (None, 56, 56, 96)     384   ['block_1_depthwise[0][0]']
      hNormalization)

      block_1_depthwise_relu (Re  (None, 56, 56, 96)     0     ['block_1_depthwise_BN[0][0]']
      LU)

      block_1_project (Conv2D)    (None, 56, 56, 24)     2304  ['block_1_depthwise_relu[0][0]
                                                               ']

      block_1_project_BN (BatchN  (None, 56, 56, 24)     96    ['block_1_project[0][0]']
      ormalization)

      block_2_expand (Conv2D)     (None, 56, 56, 144)    3456  ['block_1_project_BN[0][0]']

      block_2_expand_BN (BatchNo  (None, 56, 56, 144)    576   ['block_2_expand[0][0]']
      rmalization)

      block_2_expand_relu (ReLU)  (None, 56, 56, 144)    0     ['block_2_expand_BN[0][0]']

      block_2_depthwise (Depthwi  (None, 56, 56, 144)    1296  ['block_2_expand_relu[0][0]']
```

```
[24]  block_2_depthwise_BN (Batc  (None, 56, 56, 144)    576   ['block_2_depthwise[0][0]']
      hNormalization)

      block_2_depthwise_relu (Re  (None, 56, 56, 144)    0     ['block_2_depthwise_BN[0][0]']
      LU)

      block_2_project (Conv2D)    (None, 56, 56, 24)     3456  ['block_2_depthwise_relu[0][0]
                                                               ']

      block_2_project_BN (BatchN  (None, 56, 56, 24)     96    ['block_2_project[0][0]']
      ormalization)

      block_2_add (Add)           (None, 56, 56, 24)     0     ['block_1_project_BN[0][0]',
                                                                'block_2_project_BN[0][0]']

      block_3_expand (Conv2D)     (None, 56, 56, 144)    3456  ['block_2_add[0][0]']

      block_3_expand_BN (BatchNo  (None, 56, 56, 144)    576   ['block_3_expand[0][0]']
      rmalization)

      block_3_expand_relu (ReLU)  (None, 56, 56, 144)    0     ['block_3_expand_BN[0][0]']

      block_3_pad (ZeroPadding2D  (None, 57, 57, 144)    0     ['block_3_expand_relu[0][0]']
      )

      block_3_depthwise (Depthwi  (None, 28, 28, 144)    1296  ['block_3_pad[0][0]']
      seConv2D)

      block_3_depthwise_BN (Batc  (None, 28, 28, 144)    576   ['block_3_depthwise[0][0]']
      hNormalization)
```

```
block_3_depthwise_relu (Re  (None, 28, 28, 144)        0          ['block_3_depthwise_BN[0][0]']
LU)

block_3_project (Conv2D)    (None, 28, 28, 32)         4608       ['block_3_depthwise_relu[0][0]
                                                                  ']

block_3_project_BN (BatchN  (None, 28, 28, 32)         128        ['block_3_project[0][0]']
ormalization)

block_4_expand (Conv2D)     (None, 28, 28, 192)        6144       ['block_3_project_BN[0][0]']

block_4_expand_BN (BatchNo  (None, 28, 28, 192)        768        ['block_4_expand[0][0]']
rmalization)

block_4_expand_relu (ReLU)  (None, 28, 28, 192)        0          ['block_4_expand_BN[0][0]']

block_4_depthwise (Depthwi  (None, 28, 28, 192)        1728       ['block_4_expand_relu[0][0]']
seConv2D)

block_4_depthwise_BN (Batc  (None, 28, 28, 192)        768        ['block_4_depthwise[0][0]']
hNormalization)

block_4_depthwise_relu (Re  (None, 28, 28, 192)        0          ['block_4_depthwise_BN[0][0]']
LU)

block_4_project (Conv2D)    (None, 28, 28, 32)         6144       ['block_4_depthwise_relu[0][0]
                                                                  ']

block_4_project_BN (BatchN  (None, 28, 28, 32)         128        ['block_4_project[0][0]']
ormalization)
```

```
[24] block_4_add (Add)           (None, 28, 28, 32)         0          ['block_3_project_BN[0][0]',
                                                                       'block_4_project_BN[0][0]']

     block_5_expand (Conv2D)     (None, 28, 28, 192)        6144       ['block_4_add[0][0]']

     block_5_expand_BN (BatchNo  (None, 28, 28, 192)        768        ['block_5_expand[0][0]']
     rmalization)

     block_5_expand_relu (ReLU)  (None, 28, 28, 192)        0          ['block_5_expand_BN[0][0]']

     block_5_depthwise (Depthwi  (None, 28, 28, 192)        1728       ['block_5_expand_relu[0][0]']
     seConv2D)

     block_5_depthwise_BN (Batc  (None, 28, 28, 192)        768        ['block_5_depthwise[0][0]']
     hNormalization)

     block_5_depthwise_relu (Re  (None, 28, 28, 192)        0          ['block_5_depthwise_BN[0][0]']
     LU)

     block_5_project (Conv2D)    (None, 28, 28, 32)         6144       ['block_5_depthwise_relu[0][0]
                                                                       ']

     block_5_project_BN (BatchN  (None, 28, 28, 32)         128        ['block_5_project[0][0]']
     ormalization)

     block_5_add (Add)           (None, 28, 28, 32)         0          ['block_4_add[0][0]',
                                                                       'block_5_project_BN[0][0]']

     block_6_expand (Conv2D)     (None, 28, 28, 192)        6144       ['block_5_add[0][0]']

     block_6_expand_BN (BatchNo  (None, 28, 28, 192)        768        ['block_6_expand[0][0]']
```

```
[24]  block_6_expand_relu (ReLU)  (None, 28, 28, 192)   0     ['block_6_expand_BN[0][0]']
      block_6_pad (ZeroPadding2D  (None, 29, 29, 192)   0     ['block_6_expand_relu[0][0]']
      )
      block_6_depthwise (Depthwi  (None, 14, 14, 192)   1728  ['block_6_pad[0][0]']
      seConv2D)
      block_6_depthwise_BN (Batc  (None, 14, 14, 192)   768   ['block_6_depthwise[0][0]']
      hNormalization)
      block_6_depthwise_relu (Re  (None, 14, 14, 192)   0     ['block_6_depthwise_BN[0][0]']
      LU)
      block_6_project (Conv2D)    (None, 14, 14, 64)    12288 ['block_6_depthwise_relu[0][0]
                                                               ']
      block_6_project_BN (BatchN  (None, 14, 14, 64)    256   ['block_6_project[0][0]']
      ormalization)
      block_7_expand (Conv2D)     (None, 14, 14, 384)   24576 ['block_6_project_BN[0][0]']
      block_7_expand_BN (BatchNo  (None, 14, 14, 384)   1536  ['block_7_expand[0][0]']
      rmalization)
      block_7_expand_relu (ReLU)  (None, 14, 14, 384)   0     ['block_7_expand_BN[0][0]']
      block_7_depthwise (Depthwi  (None, 14, 14, 384)   3456  ['block_7_expand_relu[0][0]']
      seConv2D)
      block_7_depthwise_BN (Batc  (None, 14, 14, 384)   1536  ['block_7_depthwise[0][0]']
```

```
[24]  block_7_depthwise_relu (Re  (None, 14, 14, 384)   0     ['block_7_depthwise_BN[0][0]']
      LU)
      block_7_project (Conv2D)    (None, 14, 14, 64)    24576 ['block_7_depthwise_relu[0][0]
                                                               ']
      block_7_project_BN (BatchN  (None, 14, 14, 64)    256   ['block_7_project[0][0]']
      ormalization)
      block_7_add (Add)           (None, 14, 14, 64)    0     ['block_6_project_BN[0][0]',
                                                               'block_7_project_BN[0][0]']
      block_8_expand (Conv2D)     (None, 14, 14, 384)   24576 ['block_7_add[0][0]']
      block_8_expand_BN (BatchNo  (None, 14, 14, 384)   1536  ['block_8_expand[0][0]']
      rmalization)
      block_8_expand_relu (ReLU)  (None, 14, 14, 384)   0     ['block_8_expand_BN[0][0]']
      block_8_depthwise (Depthwi  (None, 14, 14, 384)   3456  ['block_8_expand_relu[0][0]']
      seConv2D)
      block_8_depthwise_BN (Batc  (None, 14, 14, 384)   1536  ['block_8_depthwise[0][0]']
      hNormalization)
      block_8_depthwise_relu (Re  (None, 14, 14, 384)   0     ['block_8_depthwise_BN[0][0]']
      LU)
      block_8_project (Conv2D)    (None, 14, 14, 64)    24576 ['block_8_depthwise_relu[0][0]
                                                               ']
```

```
[24]  block_8_project_BN (BatchN  (None, 14, 14, 64)    256   ['block_8_project[0][0]']
      ormalization)
      block_8_add (Add)           (None, 14, 14, 64)    0     ['block_7_add[0][0]',
                                                               'block_8_project_BN[0][0]']
      block_9_expand (Conv2D)     (None, 14, 14, 384)   24576 ['block_8_add[0][0]']
      block_9_expand_BN (BatchNo  (None, 14, 14, 384)   1536  ['block_9_expand[0][0]']
      rmalization)
      block_9_expand_relu (ReLU)  (None, 14, 14, 384)   0     ['block_9_expand_BN[0][0]']
      block_9_depthwise (Depthwi  (None, 14, 14, 384)   3456  ['block_9_expand_relu[0][0]']
      seConv2D)
      block_9_depthwise_BN (Batc  (None, 14, 14, 384)   1536  ['block_9_depthwise[0][0]']
      hNormalization)
      block_9_depthwise_relu (Re  (None, 14, 14, 384)   0     ['block_9_depthwise_BN[0][0]']
      LU)
      block_9_project (Conv2D)    (None, 14, 14, 64)    24576 ['block_9_depthwise_relu[0][0]
                                                               ']
      block_9_project_BN (BatchN  (None, 14, 14, 64)    256   ['block_9_project[0][0]']
      ormalization)
      block_9_add (Add)           (None, 14, 14, 64)    0     ['block_8_add[0][0]',
                                                               'block_9_project_BN[0][0]']
```

```
[24]
block_10_expand (Conv2D)          (None, 14, 14, 384)      24576     ['block_9_add[0][0]']
block_10_expand_BN (BatchNormalization)  (None, 14, 14, 384)  1536  ['block_10_expand[0][0]']
block_10_expand_relu (ReLU)       (None, 14, 14, 384)      0         ['block_10_expand_BN[0][0]']
block_10_depthwise (DepthwiseConv2D)  (None, 14, 14, 384)  3456  ['block_10_expand_relu[0][0]']
block_10_depthwise_BN (BatchNormalization)  (None, 14, 14, 384)  1536  ['block_10_depthwise[0][0]']
block_10_depthwise_relu (ReLU)    (None, 14, 14, 384)      0         ['block_10_depthwise_BN[0][0]']
block_10_project (Conv2D)         (None, 14, 14, 96)       36864     ['block_10_depthwise_relu[0][0]']
block_10_project_BN (BatchNormalization)  (None, 14, 14, 96)  384  ['block_10_project[0][0]']
block_11_expand (Conv2D)          (None, 14, 14, 576)      55296     ['block_10_project_BN[0][0]']
block_11_expand_BN (BatchNormalization)  (None, 14, 14, 576)  2304  ['block_11_expand[0][0]']
block_11_expand_relu (ReLU)       (None, 14, 14, 576)      0         ['block_11_expand_BN[0][0]']
```

```
[24]
block_11_depthwise_BN (BatchNormalization)  (None, 14, 14, 576)  2304  ['block_11_depthwise[0][0]']
block_11_depthwise_relu (ReLU)    (None, 14, 14, 576)      0         ['block_11_depthwise_BN[0][0]']
block_11_project (Conv2D)         (None, 14, 14, 96)       55296     ['block_11_depthwise_relu[0][0]']
block_11_project_BN (BatchNormalization)  (None, 14, 14, 96)  384  ['block_11_project[0][0]']
block_11_add (Add)                (None, 14, 14, 96)       0         ['block_10_project_BN[0][0]',
                                                                     'block_11_project_BN[0][0]']
block_12_expand (Conv2D)          (None, 14, 14, 576)      55296     ['block_11_add[0][0]']
block_12_expand_BN (BatchNormalization)  (None, 14, 14, 576)  2304  ['block_12_expand[0][0]']
block_12_expand_relu (ReLU)       (None, 14, 14, 576)      0         ['block_12_expand_BN[0][0]']
block_12_depthwise (DepthwiseConv2D)  (None, 14, 14, 576)  5184  ['block_12_expand_relu[0][0]']
block_12_depthwise_BN (BatchNormalization)  (None, 14, 14, 576)  2304  ['block_12_depthwise[0][0]']
block_12_depthwise_relu (ReLU)    (None, 14, 14, 576)      0         ['block_12_depthwise_BN[0][0]']
```

```
[24]
block_12_project_BN (BatchNormalization)  (None, 14, 14, 96)  384  ['block_12_project[0][0]']
block_12_add (Add)                (None, 14, 14, 96)       0         ['block_11_add[0][0]',
                                                                     'block_12_project_BN[0][0]']
block_13_expand (Conv2D)          (None, 14, 14, 576)      55296     ['block_12_add[0][0]']
block_13_expand_BN (BatchNormalization)  (None, 14, 14, 576)  2304  ['block_13_expand[0][0]']
block_13_expand_relu (ReLU)       (None, 14, 14, 576)      0         ['block_13_expand_BN[0][0]']
block_13_pad (ZeroPadding2D)      (None, 15, 15, 576)      0         ['block_13_expand_relu[0][0]']
block_13_depthwise (DepthwiseConv2D)  (None, 7, 7, 576)    5184  ['block_13_pad[0][0]']
block_13_depthwise_BN (BatchNormalization)  (None, 7, 7, 576)  2304  ['block_13_depthwise[0][0]']
block_13_depthwise_relu (ReLU)    (None, 7, 7, 576)        0         ['block_13_depthwise_BN[0][0]']
block_13_project (Conv2D)         (None, 7, 7, 160)        92160     ['block_13_depthwise_relu[0][0]']
block_13_project_BN (BatchNormalization)  (None, 7, 7, 160)  640  ['block_13_project[0][0]']
```

```
[24]  block_14_expand_BN (BatchN  (None, 7, 7, 960)        3840      ['block_14_expand[0][0]']
      ormalization)

      block_14_expand_relu (ReLU  (None, 7, 7, 960)        0         ['block_14_expand_BN[0][0]']
      )

      block_14_depthwise (Depthw  (None, 7, 7, 960)        8640      ['block_14_expand_relu[0][0]']
      iseConv2D)

      block_14_depthwise_BN (Bat  (None, 7, 7, 960)        3840      ['block_14_depthwise[0][0]']
      chNormalization)

      block_14_depthwise_relu (R  (None, 7, 7, 960)        0         ['block_14_depthwise_BN[0][0]'
      eLU)                                                           ]

      block_14_project (Conv2D)   (None, 7, 7, 160)        153600    ['block_14_depthwise_relu[0][0
                                                                     ]']

      block_14_project_BN (Batch  (None, 7, 7, 160)        640       ['block_14_project[0][0]']
      Normalization)

      block_14_add (Add)          (None, 7, 7, 160)        0         ['block_13_project_BN[0][0]',
                                                                      'block_14_project_BN[0][0]']

      block_15_expand (Conv2D)    (None, 7, 7, 960)        153600    ['block_14_add[0][0]']

      block_15_expand_BN (BatchN  (None, 7, 7, 960)        3840      ['block_15_expand[0][0]']
      ormalization)

      block_15_expand_relu (ReLU  (None, 7, 7, 960)        0         ['block_15_expand_BN[0][0]']
      )
```

```
[24]  Conv_1_bn (BatchNormalizat  (None, 7, 7, 1280)       5120      ['Conv_1[0][0]']
      ion)

      out_relu (ReLU)             (None, 7, 7, 1280)       0         ['Conv_1_bn[0][0]']

      global_average_pooling2d (  (None, 1280)             0         ['out_relu[0][0]']
      GlobalAveragePooling2D)

      dense (Dense)               (None, 128)              163968    ['global_average_pooling2d[0][
                                                                     0]']

      dense_1 (Dense)             (None, 128)              16512     ['dense[0][0]']

      dense_2 (Dense)             (None, 9)                1161      ['dense_1[0][0]']

==================================================================================================
Total params: 2439625 (9.31 MB)
Trainable params: 181641 (709.54 KB)
Non-trainable params: 2257984 (8.61 MB)
__________________________________________________________________________________________________
Epoch 1/5
180/180 [==============================] - 386s 2s/step - loss: 0.1889 - accuracy: 0.9431 - val_loss: 0.0164 - val_accuracy: 0.9958
Epoch 2/5
180/180 [==============================] - 377s 2s/step - loss: 0.0146 - accuracy: 0.9951 - val_loss: 0.0349 - val_accuracy: 0.9854
Epoch 3/5
180/180 [==============================] - 362s 2s/step - loss: 0.0110 - accuracy: 0.9972 - val_loss: 0.0132 - val_accuracy: 0.9972
Epoch 4/5
180/180 [==============================] - 380s 2s/step - loss: 0.0024 - accuracy: 0.9993 - val_loss: 0.0044 - val_accuracy: 0.9986
Epoch 5/5
180/180 [==============================] - 381s 2s/step - loss: 6.0469e-04 - accuracy: 1.0000 - val_loss: 0.0030 - val_accuracy: 0.9993
```

```
[25] results = model.evaluate(test_images, verbose=0)
     print("Test Loss: {:.5f}".format(results[0]))
     print("Test Accuracy: {:.2f}%".format(results[1] * 100))

     Test Loss: 0.00273
     Test Accuracy: 99.94%

[26] pred = model.predict(test_images)
     pred=np.argmax(pred,axis=1)

     57/57 [==============================] - 96s 2s/step

[27] pred_df=test_df.copy()
     labels={}
     for l,v in test_images.class_indices.items():
         labels.update({v:l})
     pred_df['pred']=pred
     pred_df['pred']=pred_df['pred'].apply(lambda x: labels[x])
```

```
[28] pred_df=pred_df.reset_index(drop=True)
     pred_df.head(10)
```

	path	label	pred
0	/content/Fish_Dataset/Fish_Dataset/Gilt-Head B...	Gilt-Head Bream	Gilt-Head Bream
1	/content/Fish_Dataset/Fish_Dataset/Shrimp/Shri...	Shrimp	Shrimp
2	/content/Fish_Dataset/Fish_Dataset/Red Sea Bre...	Red Sea Bream	Red Sea Bream
3	/content/Fish_Dataset/Fish_Dataset/Black Sea S...	Black Sea Sprat	Black Sea Sprat
4	/content/Fish_Dataset/Fish_Dataset/Hourse Mack...	Hourse Mackerel	Hourse Mackerel
5	/content/Fish_Dataset/Fish_Dataset/Hourse Mack...	Hourse Mackerel	Hourse Mackerel
6	/content/Fish_Dataset/Fish_Dataset/Sea Bass/Se...	Sea Bass	Sea Bass
7	/content/Fish_Dataset/Fish_Dataset/Hourse Mack...	Hourse Mackerel	Hourse Mackerel
8	/content/Fish_Dataset/Fish_Dataset/Trout/Trout...	Trout	Trout
9	/content/Fish_Dataset/Fish_Dataset/Black Sea S...	Black Sea Sprat	Black Sea Sprat

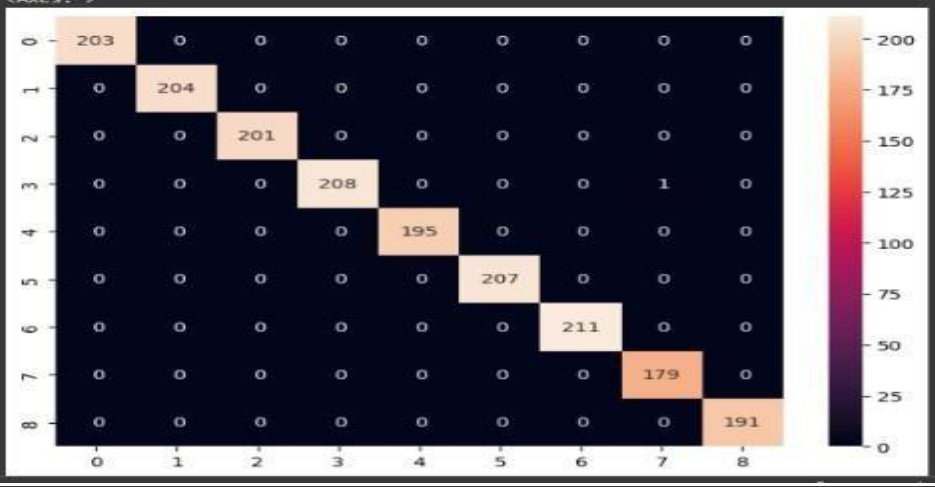
```
[29] pred_df[pred_df['label']!=pred_df['pred']]
```

	path	label	pred
756	/content/Fish_Dataset/Fish_Dataset/Red Mullet/...	Red Mullet	Striped Red Mullet

```
[30] pred_df.path[1604]

     '/content/Fish_Dataset/Fish_Dataset/Sea Bass/Sea Bass/00493.png'

[31] plt.figure(figsize=(15,8))
     plt.imshow(plt.imread(pred_df.path[1604]))
     title1 = pred_df.path[1604].split('/')[-2]
     title2 = pred_df.path[1604].split('/')[-1]
     title3 = pred_df.pred[1604]
     plt.title(f'Image:- {title2}\nTrue Class:- {title1}\nPredicted Class:- {title3}', color = 'r', weight = 'bold', fontsize = 15)
     plt.show()
```


```
print(f"Accuracy Score: {accuracy_score(pred_df['label'],pred_df['pred'])}")
plt.figure(figsize=(8,6))
sns.heatmap(confusion_matrix(pred_df['label'],pred_df['pred']), annot=True, fmt='2d')
```

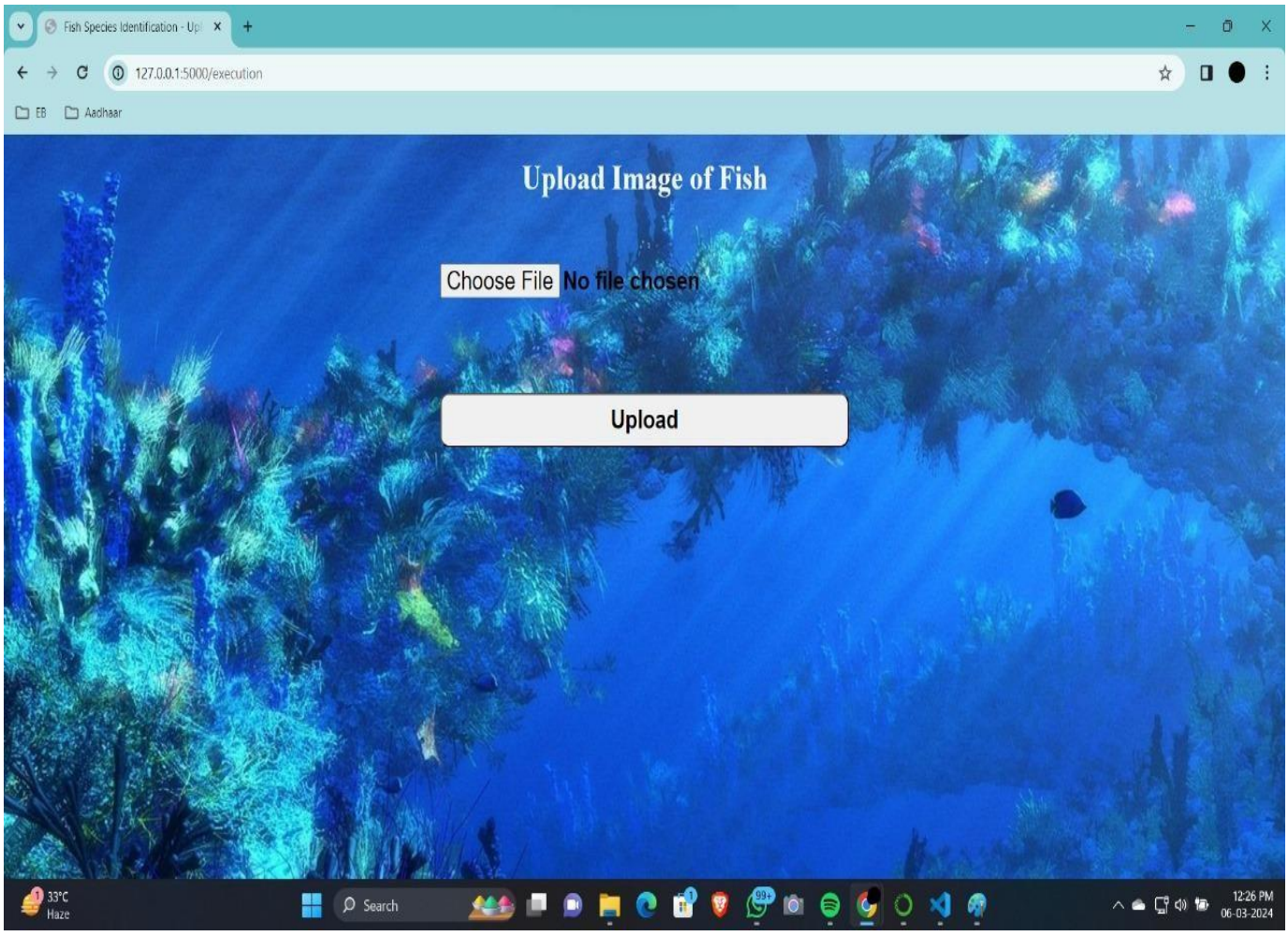
```
Accuracy Score: 0.9994444444444445
<Axes: >
```

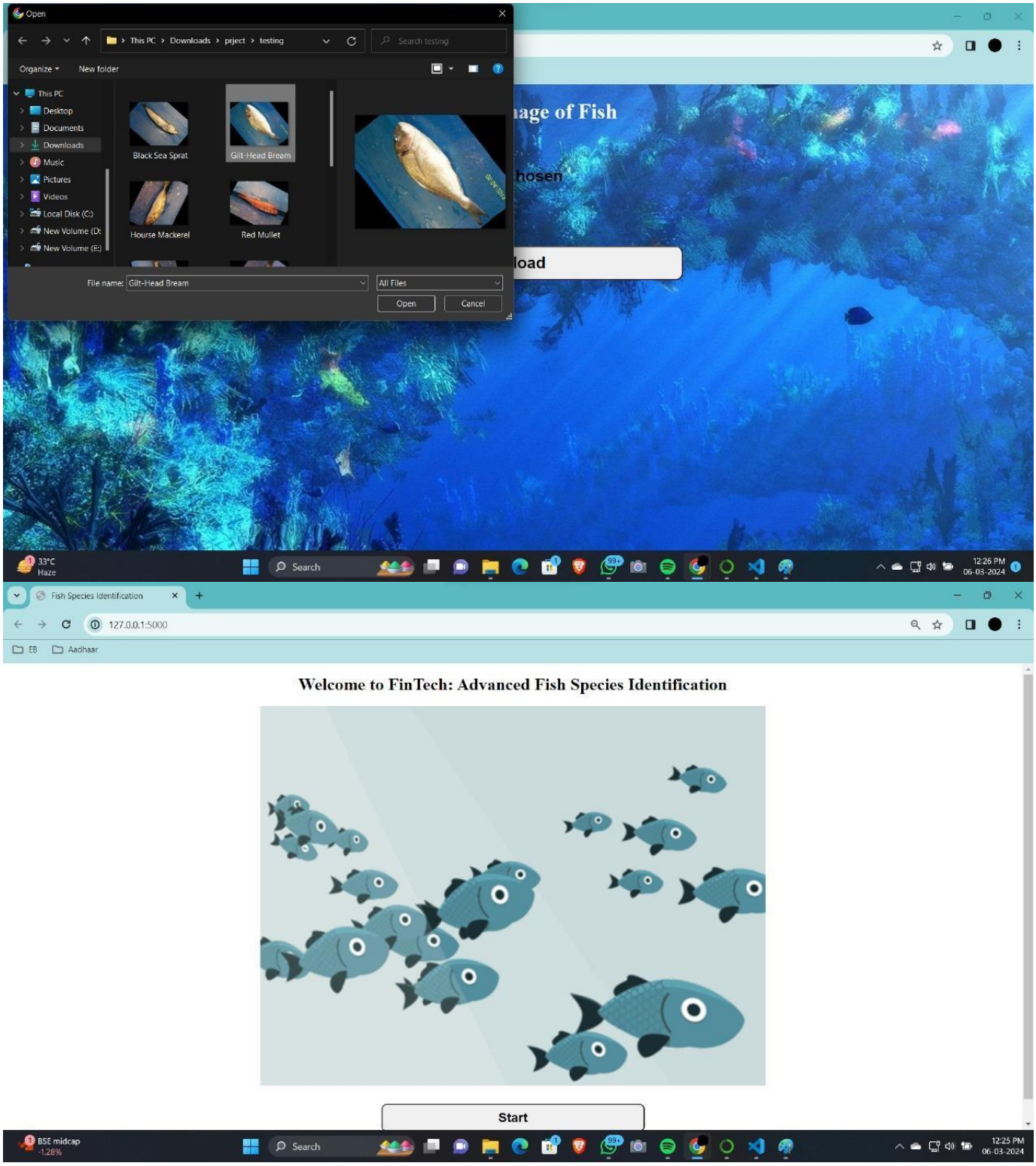


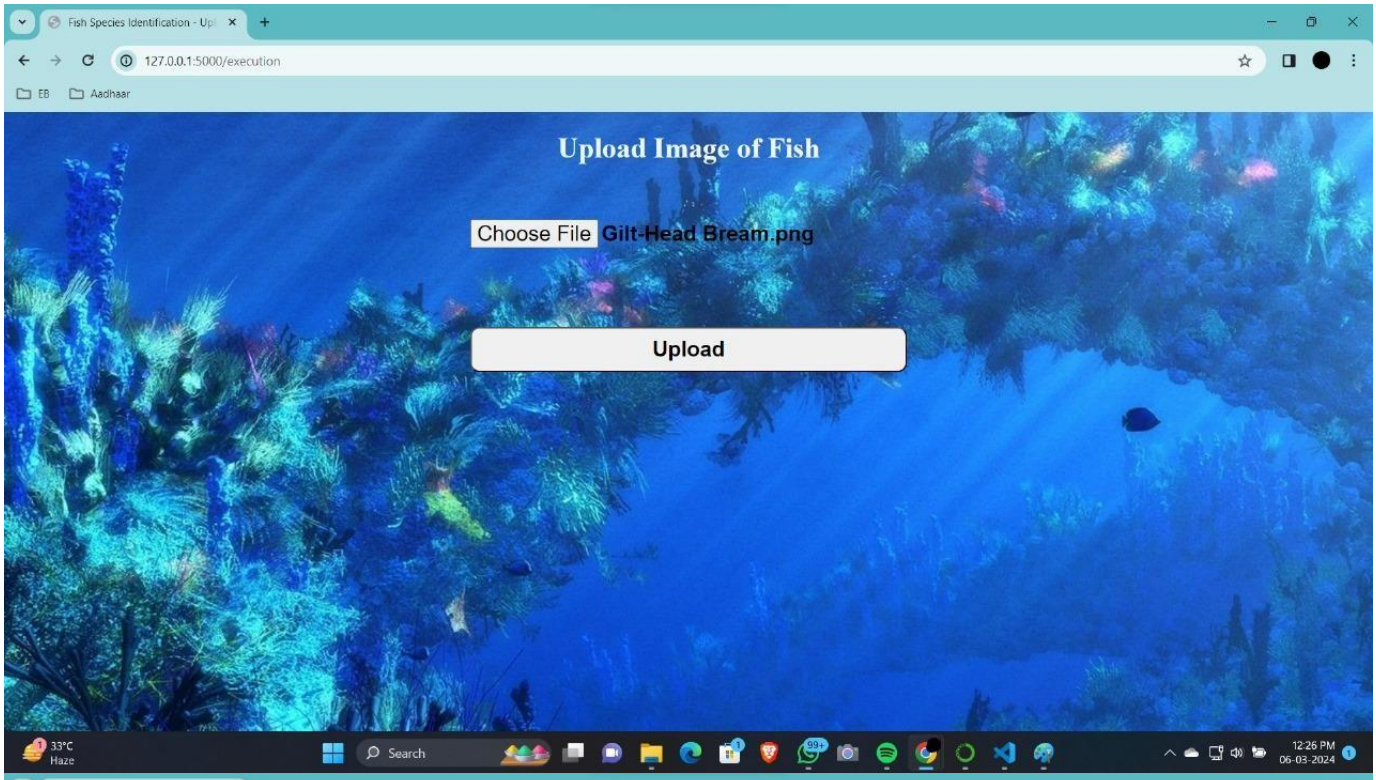

```
print(classification_report(pred_df['label'],pred_df['pred']))
```

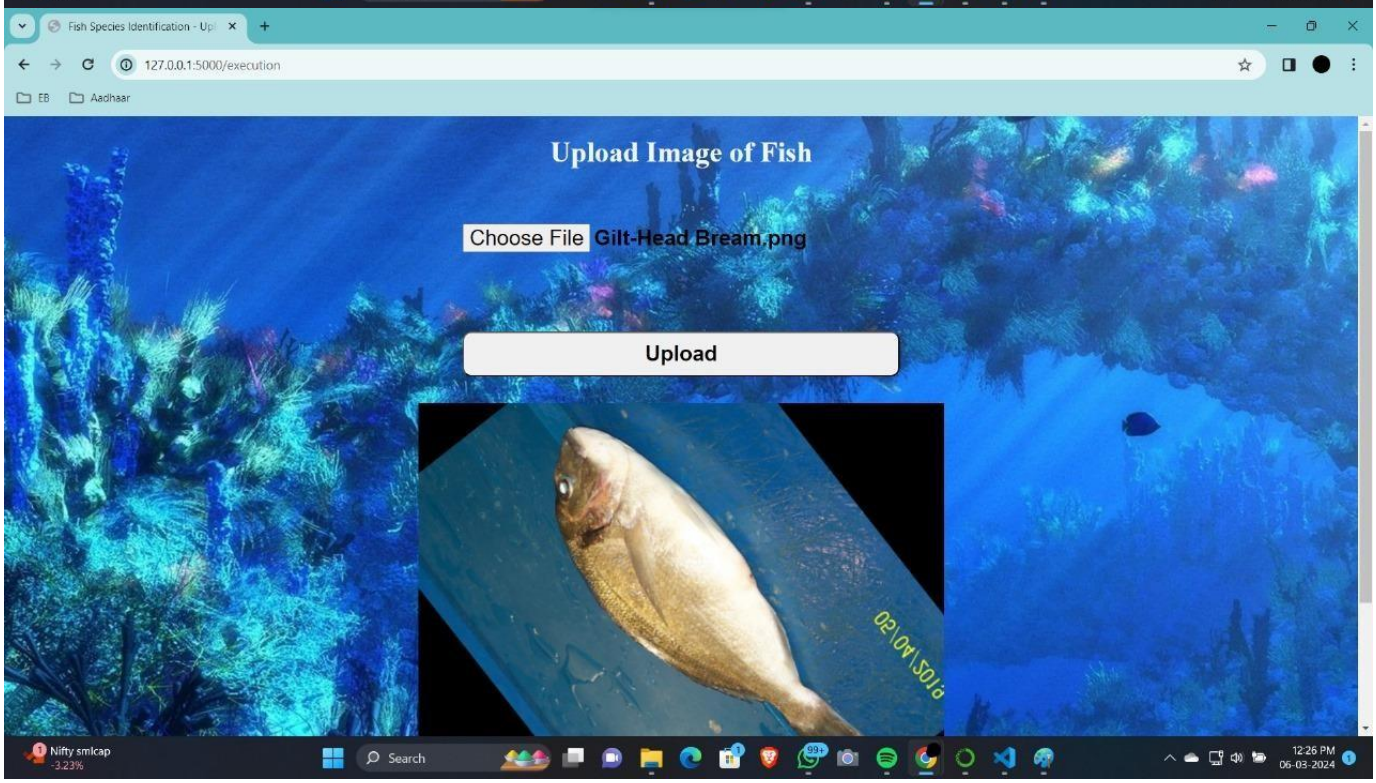
	precision	recall	f1-score	support
Black Sea Sprat	1.00	1.00	1.00	203
Gilt-Head Bream	1.00	1.00	1.00	204
Hourse Mackerel	1.00	1.00	1.00	201
Red Mullet	1.00	1.00	1.00	209
Red Sea Bream	1.00	1.00	1.00	195
Sea Bass	1.00	1.00	1.00	207
Shrimp	1.00	1.00	1.00	211
Striped Red Mullet	0.99	1.00	1.00	179
Trout	1.00	1.00	1.00	191
accuracy			1.00	1800
macro avg	1.00	1.00	1.00	1800
weighted avg	1.00	1.00	1.00	1800

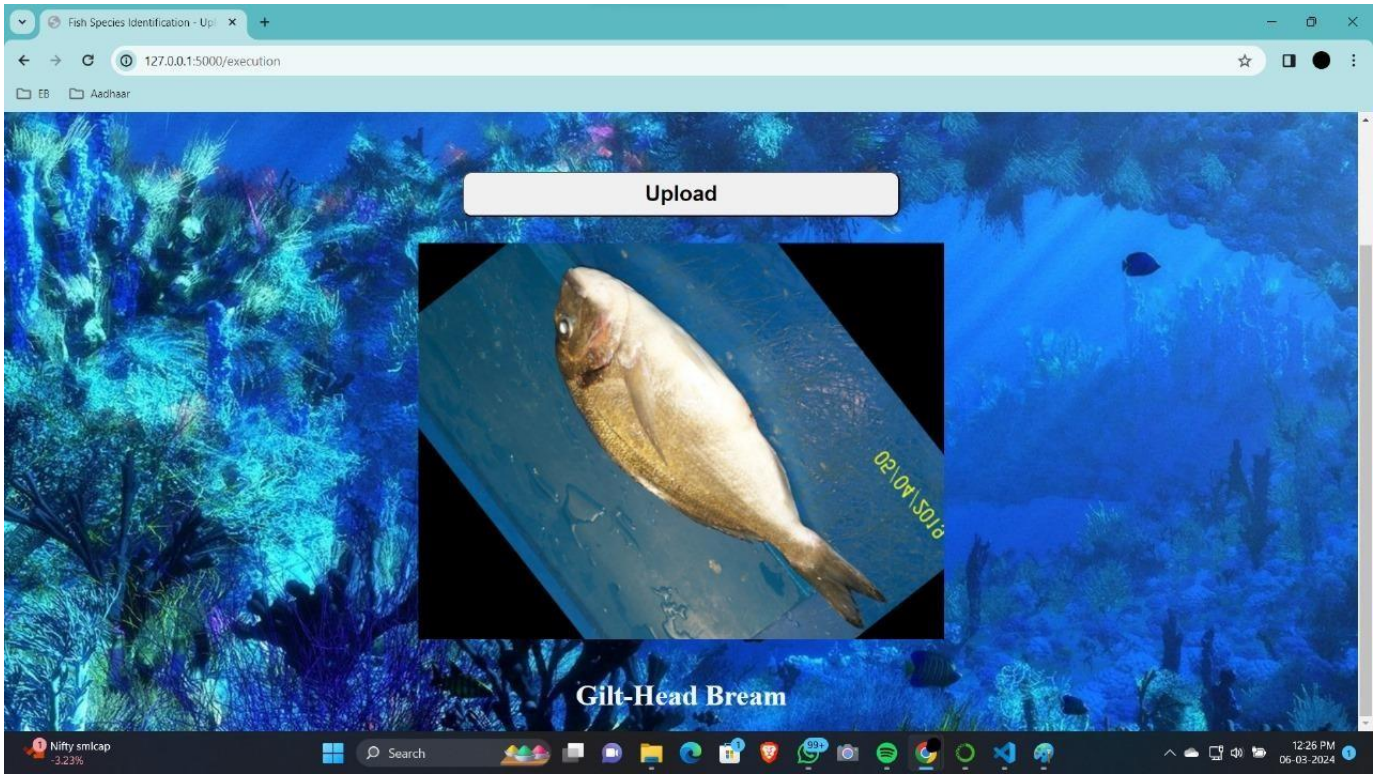
Integrated Web Development




Open
This PC › Downloads › prject › testing
Search testing
Organize
New folder
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
New Volume (D:
New Volume (E:)
Black Sea Sprat
Gilt-Head Bream
Hourse Mackerel
Red Mullet
File name: Gilt-Head Bream
All Files
Open
Cancel
age of Fish
hosen
load
33°C
Haze
Search
12:26 PM
06-03-2024
Fish Species Identification
127.0.0.1:5000
EB
Aadhaar
Welcome to FinTech: Advanced Fish Species Identification
Start
BSE midcap
-1.28%
Search
12:25 PM
06-03-2024


Fish Species Identification - Up
127.0.0.1:5000/execution
EB
Aadhaar
Upload Image of Fish
Choose File
Gilt-Head Bream.png
Upload
33°C
Haze
Search
12:26 PM
06-03-2024


Fish Species Identification - Up
127.0.0.1:5000/execution
EB
Aadhaar
Upload Image of Fish
Choose File
Gilt-Head Bream.png
Upload
Nifty smlcap
-3.23%
Search
12:26 PM
06-03-2024


Fish Species Identification - Up
127.0.0.1:5000/execution
EB
Aadhaar
Upload
Gilt-Head Bream
Nifty smlcap
-3.23%
Search
12:26 PM
06-03-2024

REFERENCES

1. Bai, X., Yang, X., and Latecki, L. J. (2008). "Detection and recognition of contour parts based on shape similarity," Pattern Recognition, vol. 41, pp. 2189-2199.

2. Chen, L., Li, Z., and Zhao, Z. (1999). "Forensic medical identification of death due to poisoning of tetrodotoxin in puffer fish," Fa yi xue za zhi, vol. 15, pp. 131-2, 189.

3. Hong, K.-S., and Kim, J.-S. (2009). "Color–texture segmentation using unsupervised graph cuts," Pattern Recognition, vol. 42, pp. 735-750.

4. Khalifa, N. E. M., Taha, M. H. N., and Hassanien, A. E. (2018). "Aquarium family fish species identification system using deep neural networks," International Conference on Advanced Intelligent Systems and Informatics, pp. 347-356.

5. Knausgård, K. M., Wiklund, A., Sørdalen, T. K., Halvorsen, K., Kleiven, A. R., Jiao, L., et al. (2020). "Temperate Fish Detection and Classification: a Deep Learning based Approach," arXiv preprint arXiv:2005.07518.

6. Konovalov, D. A., Saleh, A., Bradley, M., Sankupellay, M., Marini, S., and Sheaves, M. (2019). "Underwater fish detection with weak multi-domain supervision," 2019 International Joint Conference on Neural Networks (IJCNN), pp.
1-8.

7. LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., et al. (1989). "Backpropagation applied to handwritten zip code recognition," Neural computation, vol. 1, pp. 541-551.

8. Lim, E., Ngo, A., Quek, L., and Yong, Y. (2013). "A case report of puffer fish poisoning in Singapore," Case reports in medicine, vol. 2013.

9. Matthews, W. J. (2012). "Patterns in freshwater fish ecology: Springer Science & Business Media."

10. Ostman, D. C., and Sims, J. K. (1986). "Pufferfish poisoning: emergency diagnosis and management of mild human tetrodotoxication," Annals of emergency medicine, vol. 15, pp. 1094-1098.

11. Saleh, A., Bradley, M., Sankupellay, M., Marini, S., Sheaves, M., and Konovalov, D. A. (2019). "Underwater fish detection with weak multi-domain supervision," 2019 International Joint Conference on Neural Networks (IJCNN), pp.
1-8.

12. Simonyan, K., and Zisserman, A. (2014). "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556.

13. Wurts, W. A. (1998). "Why can some fish live in freshwater, some in salt water, and some in both," World Aquaculture, vol. 29, p. 65.