A Capstone Project Report on

IDENTIFICATION OF FISH SPECIES - USING CNN

Submitted by

KAWIN D

ABSTRACT

This project is based on identification of fish species. In order to develop species-specific advisories, it is necessary to collect the fish-catch information to be categorized at some authentic level with an enhanced accuracy. Often species level catch reporting is having technical hindrances due to several reasons pertaining to manual efforts, one among them would be ignorance about particular species in specific which results to low reporting. Henceforth, Deep Learning based tool for image-based identification of fish-species found in the water resource is to be developed. For creating an application fisher need to take a photo of fishes caught and fish identification will be done automatically after classifying it based on the size, minimizing the manual intervention. This species classifying application is planned to be developed using Convolution Neural Networks (CNN) to achieve maximum accuracy. A CNN is a supervised learning technique which needs both input data and target output data to be supplied. These are classified by using their labels in order to provide a learned model for future data analysis

Dataset Link: https://github.com/DanielKawin/capstone-project-identification-of-fishes-/tree/main/Dataset

Source Code: https://github.com/DanielKawin/capstone-project-identification-of-fishes-/tree/main/code

ACKNOWLEDGEMENTS

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of my capstone project. I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

Further, I have fortunate to have Mr. PRASAD as my mentor. He has readily shared his immense knowledge in data science and guides me in a manner that the outcome resulted in enhancing my data skills.

I certify that the work done by me for conceptualizing and completing this project is original and authentic.

Date: July 10, 2022 Name: KAWIN D

CERTIFICATE OF COMPLETION

I hereby certify that the project titled “Identification of fish species using CNN” was undertaken and completed the project (10th July, 2022).

Mentor	: Mr. Prasad
Date	: 10th July 2022
Place	: Karur

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Most modern deep learning models are based on artificial neural networks, specifically convolution neural networks (CNN), although they can also include propositional formulas or latent variables organized layer-wise in deep generative models such as the nodes in deep belief networks and deep Boltzmann machines. In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw input may be a matrix of pixels.

Figure 1.1: IDENTIFICATION OF FISH SPECIES.

The first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level on its own.

This project is based on identification of the fish species. In order to develop species-specific advisories, it is necessary to collect the fish-catch information to be categorized at some authentic level with an enhanced accuracy. Often species level catch reporting is having technical hindrances due to several reasons pertaining to manual efforts, one among them would be ignorance about particular species in specific which results to low reporting.

Fish species recognition is a multi-class classification problem and is a compelling research field of machine learning and computer vision. The algorithms implemented over individual input images which perform classification mainly using shape and texture feature extraction and matching. All the existing work either deals with a small dataset distinguishing between less number of species or has a low accuracy.

CHAPTER-2

DATA COLLECTION AND DATA PREPARATION

Dataset for identification of fish species is found from various sources such as from Kaggle, web and browser searches. Finally, there are nearly 1000 images of fishes that belong to various species.

For training, I have used totally 750 images belongs to fish species. For Validating, I have used totally 250 images belongs. Some images downloaded from internet browser in order to make prediction by model.

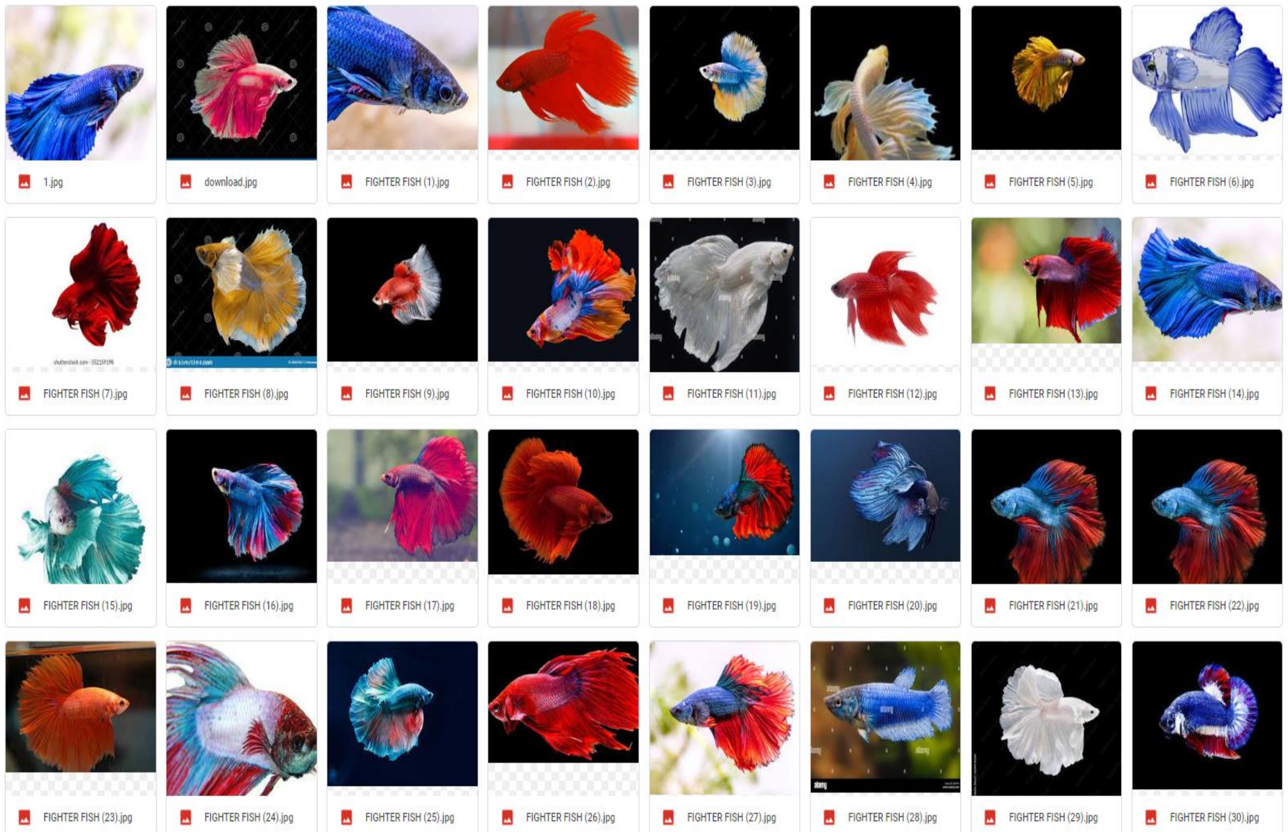
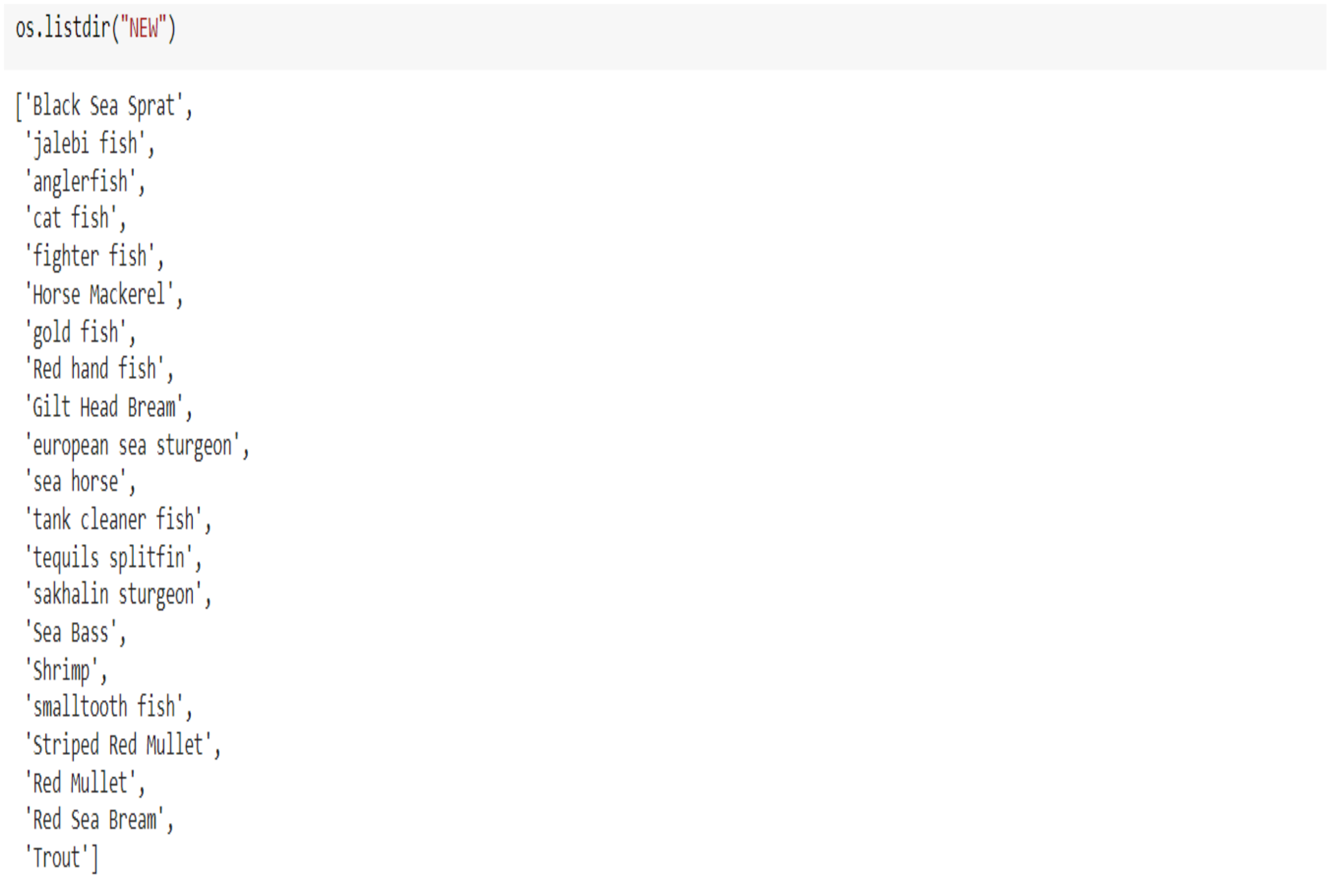


Figure 2.1: FIGHTER FISH TRAINING DATASET

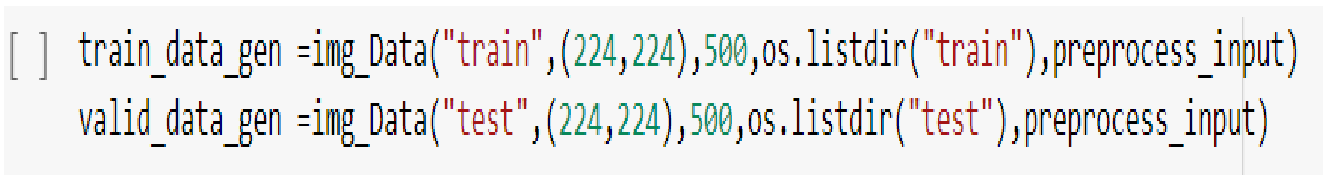


```
os.listdir("NEW")

['Black Sea Sprat',
 'jalebi fish',
 'anglerfish',
 'cat fish',
 'fighter fish',
 'Horse Mackerel',
 'gold fish',
 'Red hand fish',
 'Gilt Head Bream',
 'european sea sturgeon',
 'sea horse',
 'tank cleaner fish',
 'tequils splitfin',
 'sakhalin sturgeon',
 'Sea Bass',
 'Shrimp',
 'smalltooth fish',
 'Striped Red Mullet',
 'Red Mullet',
 'Red Sea Bream',
 'Trout']
```

Figure 2.2: VARIETY OF FISH SPECIES

IMAGE RESHAPING:

Neural networks receive inputs of the same size, all images need to be resized to a fixed size before inputting them to the CNN. So, I have reshaped all the images to the size of 224 * 224 and give them to my neural network model.

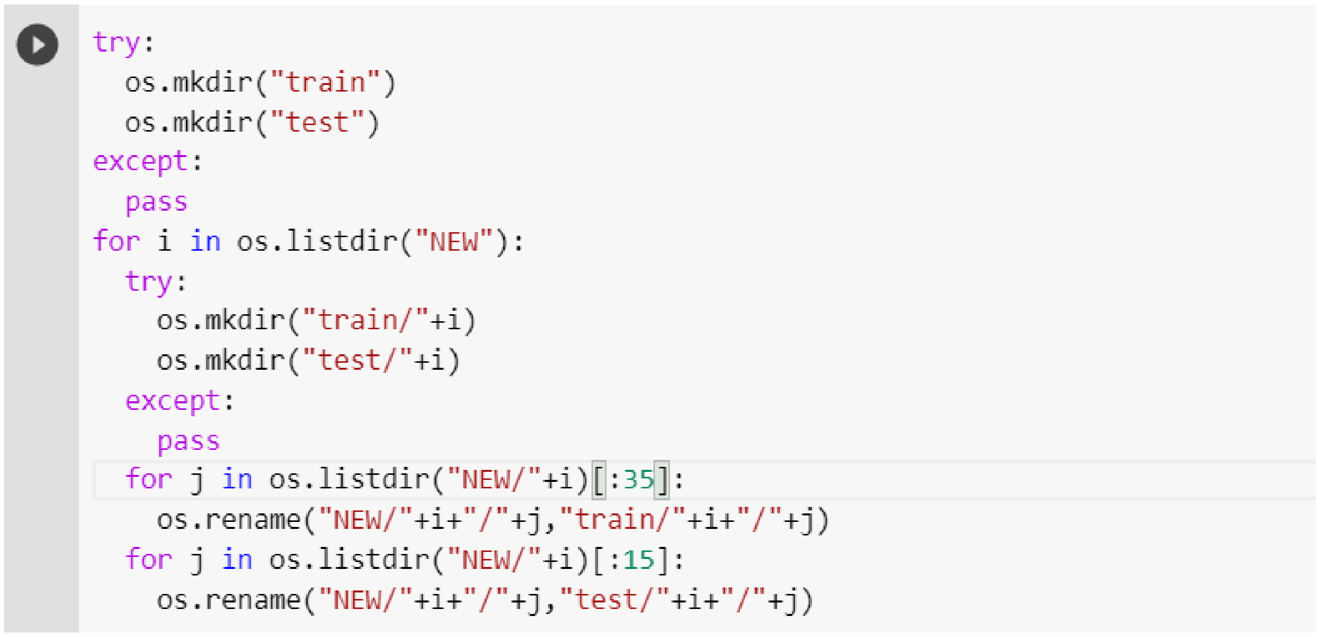


```
[ ]  train_data_gen =img_Data("train",(224,224),500,os.listdir("train"),preprocess_input)
     valid_data_gen =img_Data("test",(224,224),500,os.listdir("test"),preprocess_input)
```

Figure 2.3: RESIZING OF IMAGE

SPLITTING THE DATA:

I have split the total dataset in separate files for training and testing the dataset. The simplest way to split the modelling dataset into training and testing sets is to assign 2/3 data points to the former and the remaining one-third to the latter. Therefore, we train the model using the training set and then apply the model to the test set. In this way, we can evaluate the performance of our model.



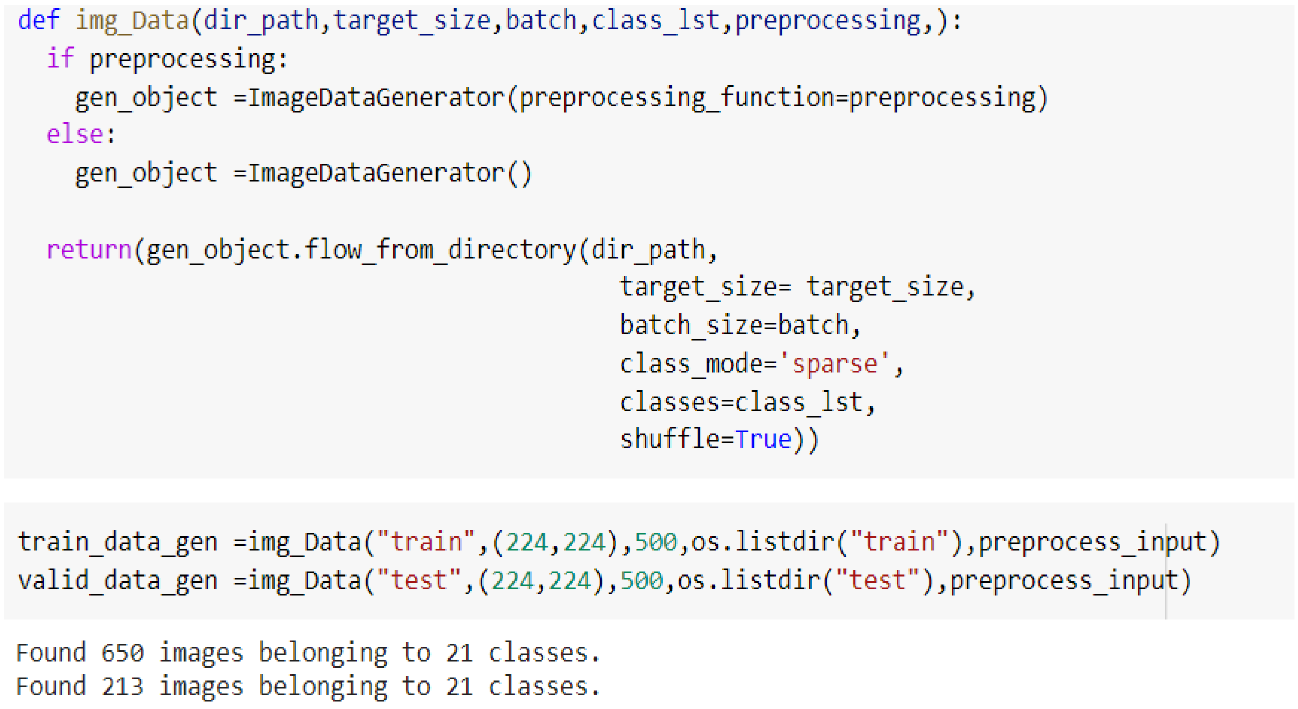
```
try:
  os.mkdir("train")
  os.mkdir("test")
except:
  pass
for i in os.listdir("NEW"):
  try:
    os.mkdir("train/"+i)
    os.mkdir("test/"+i)
  except:
    pass
  for j in os.listdir("NEW/"+i)[:35]:
    os.rename("NEW/"+i+"/"+j,"train/"+i+"/"+j)
  for j in os.listdir("NEW/"+i)[:15]:
    os.rename("NEW/"+i+"/"+j,"test/"+i+"/"+j)
```

Figure 2.4: SPLITTING THE DATA

CHAPTER 3

TRAIN VALIDATION SPLIT

The train dataset and test dataset loaded separately in separate variables. Train dataset contains 750 images belongs to fish species. (i.e., given fish is catfish or golden fish or fighter fish). Test dataset contains 250 images belongs to fish species.



```
def img_Data(dir_path,target_size,batch,class_lst,preprocessing,):
  if preprocessing:
    gen_object =ImageDataGenerator(preprocessing_function=preprocessing)
  else:
    gen_object =ImageDataGenerator()

  return(gen_object.flow_from_directory(dir_path,
                                        target_size= target_size,
                                        batch_size=batch,
                                        class_mode='sparse',
                                        classes=class_lst,
                                        shuffle=True))
```

```
train_data_gen =img_Data("train",(224,224),500,os.listdir("train"),preprocess_input)
valid_data_gen =img_Data("test",(224,224),500,os.listdir("test"),preprocess_input)
```

```
Found 650 images belonging to 21 classes.
Found 213 images belonging to 21 classes.
```

Figure 3.1: TRAINING AND TESTING VALIDATION SPLIT

CHAPTER 4

FITTING MODELS TO DATA

4.1 MOBILENET-V2

In MobileNetV2, there are two types of blocks. One is residual block with stride of 1. Another one is block with stride of 2 for downsizing. There are 3 layers for both types of blocks. This time, the **first layer** is **1×1 convolution with ReLU6.** The **second layer** is the **depthwise convolution**. The **third layer** is another **1×1 convolution but without any non-linearity.** It is claimed that if ReLU is used again, the deep networks only have the power of a linear classifier on the non-zero volume part of the output domain.

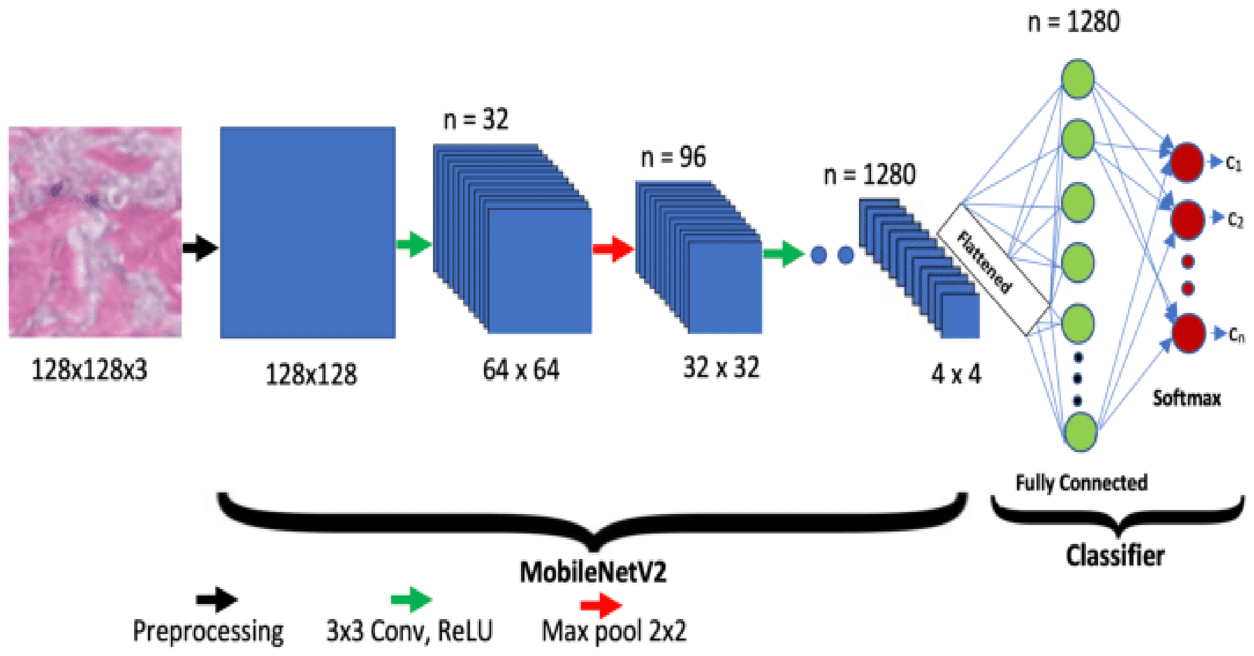


Figure 4.11: ARCHITECTURE OF MOBILENET-V2

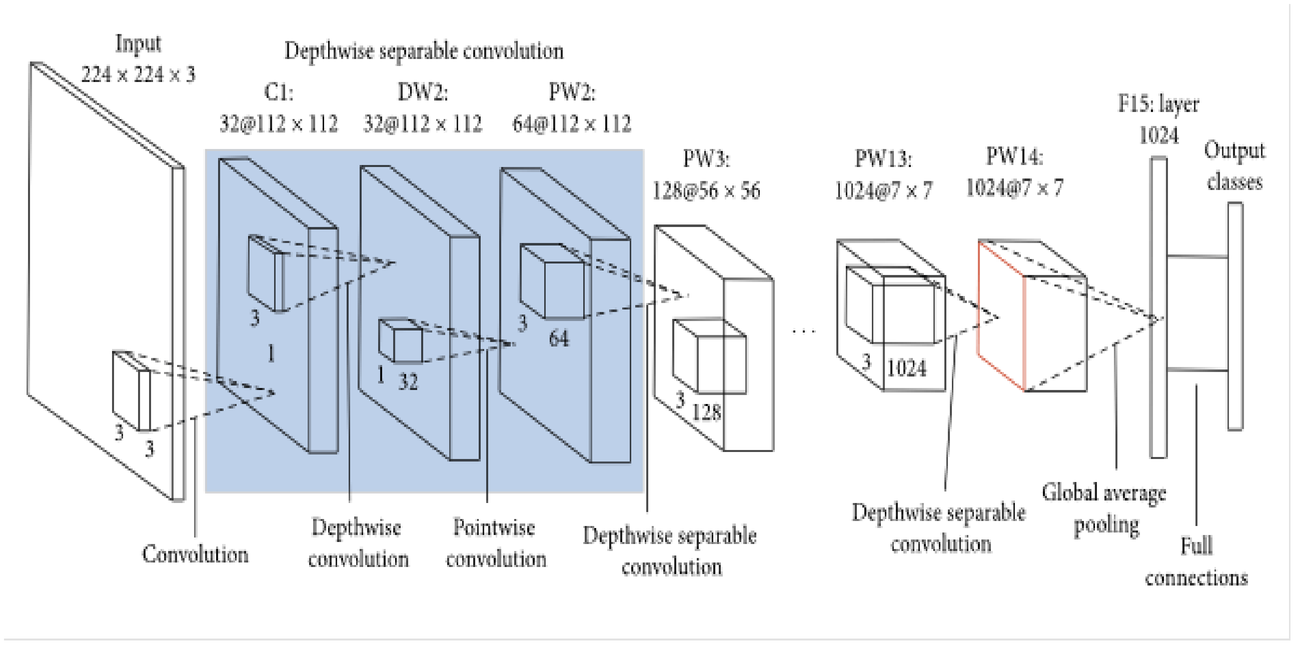
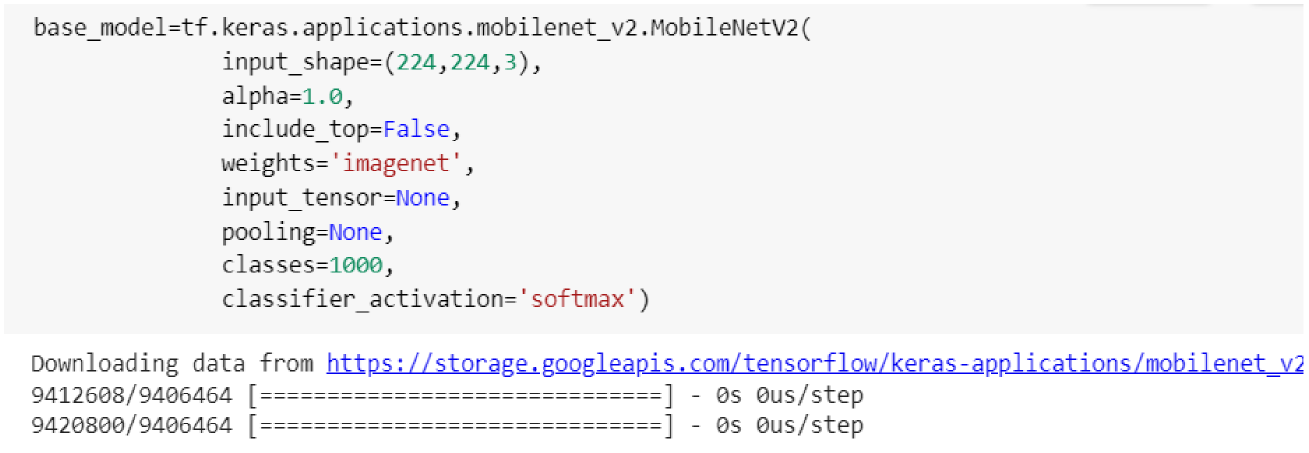


Figure 4.12 MOBILEV2 LAYERS

MobileNetV2 is very similar to the original MobileNet, except that it uses inverted residual blocks with bottlenecking features. It has a drastically lower parameter count than the original MobileNet. It supports any input size greater than 32 x 32, with larger image sizes offering better performance.

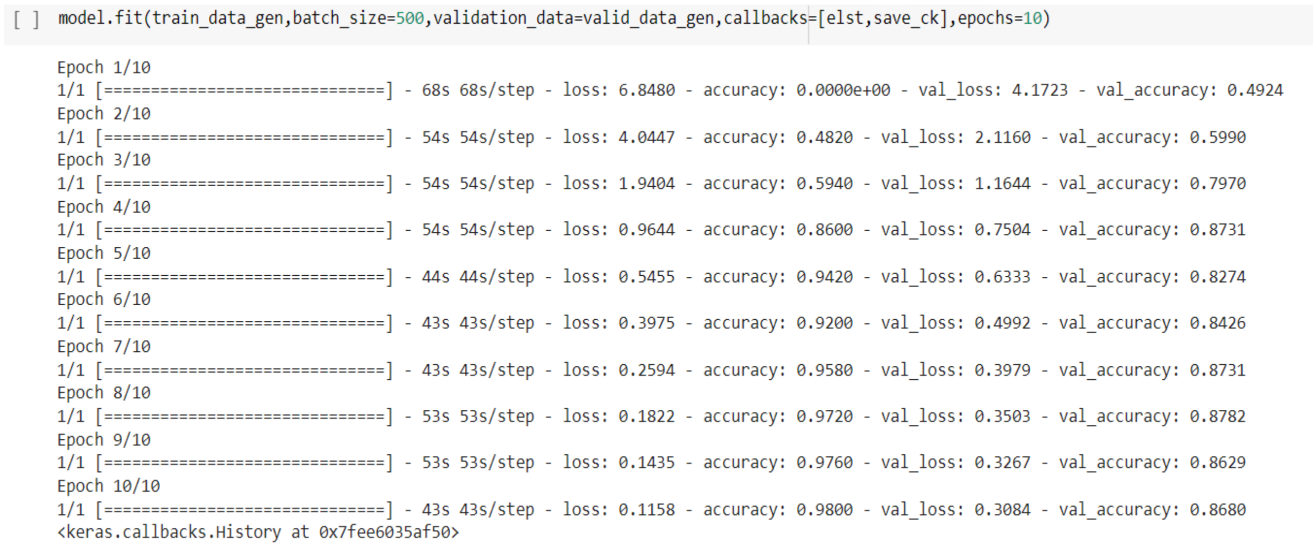


```
base_model=tf.keras.applications.mobilenet_v2.MobileNetV2(
              input_shape=(224,224,3),
              alpha=1.0,
              include_top=False,
              weights='imagenet',
              input_tensor=None,
              pooling=None,
              classes=1000,
              classifier_activation='softmax')
```

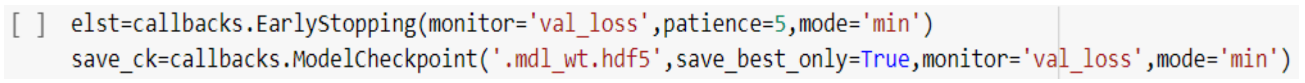
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2
9412608/9406464 [==============================] - 0s 0us/step
9420800/9406464 [==============================] - 0s 0us/step
```

Figure 4.13 DEFINING THE LAYERS OF MOBILENET-V2

I have fitted the model with training dataset for 10 epochs. I have tested my model with validation data which have the given the accuracy of nearly 80%.



```
[ ] model.fit(train_data_gen,batch_size=500,validation_data=valid_data_gen,callbacks=[elst,save_ck],epochs=10)

    Epoch 1/10
    1/1 [==============================] - 68s 68s/step - loss: 6.8480 - accuracy: 0.0000e+00 - val_loss: 4.1723 - val_accuracy: 0.4924
    Epoch 2/10
    1/1 [==============================] - 54s 54s/step - loss: 4.0447 - accuracy: 0.4820 - val_loss: 2.1160 - val_accuracy: 0.5990
    Epoch 3/10
    1/1 [==============================] - 54s 54s/step - loss: 1.9404 - accuracy: 0.5940 - val_loss: 1.1644 - val_accuracy: 0.7970
    Epoch 4/10
    1/1 [==============================] - 54s 54s/step - loss: 0.9644 - accuracy: 0.8600 - val_loss: 0.7504 - val_accuracy: 0.8731
    Epoch 5/10
    1/1 [==============================] - 44s 44s/step - loss: 0.5455 - accuracy: 0.9420 - val_loss: 0.6333 - val_accuracy: 0.8274
    Epoch 6/10
    1/1 [==============================] - 43s 43s/step - loss: 0.3975 - accuracy: 0.9200 - val_loss: 0.4992 - val_accuracy: 0.8426
    Epoch 7/10
    1/1 [==============================] - 43s 43s/step - loss: 0.2594 - accuracy: 0.9580 - val_loss: 0.3979 - val_accuracy: 0.8731
    Epoch 8/10
    1/1 [==============================] - 53s 53s/step - loss: 0.1822 - accuracy: 0.9720 - val_loss: 0.3503 - val_accuracy: 0.8782
    Epoch 9/10
    1/1 [==============================] - 53s 53s/step - loss: 0.1435 - accuracy: 0.9760 - val_loss: 0.3267 - val_accuracy: 0.8629
    Epoch 10/10
    1/1 [==============================] - 43s 43s/step - loss: 0.1158 - accuracy: 0.9800 - val_loss: 0.3084 - val_accuracy: 0.8680
    <keras.callbacks.History at 0x7fee6035af50>
```



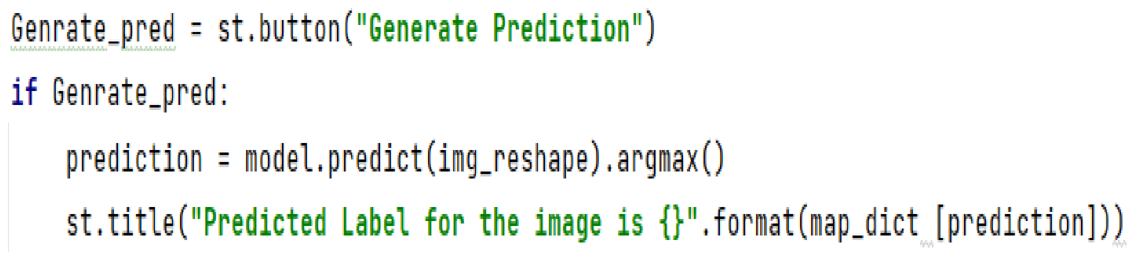
```
[ ] elst=callbacks.EarlyStopping(monitor='val_loss',patience=5,mode='min')
    save_ck=callbacks.ModelCheckpoint('.mdl_wt.hdf5',save_best_only=True,monitor='val_loss',mode='min')
```

Figure 4.14: FITTING DATA TO MOBILENET-V2

CHAPTER 5

PREDICTING NEW DATA

I have downloaded several fish images from internet which is belongs to various classes (i.e., Fighter fish, Cat fish, Red hand fish, Gold fish etc.,) and tested with MobileNetV2 model. I found that my MobileNetV2 model have predicted the types of fishes accurately.



```
Genrate_pred = st.button("Generate Prediction")
if Genrate_pred:
    prediction = model.predict(img_reshape).argmax()
    st.title("Predicted Label for the image is {}".format(map_dict [prediction]))
```

Figure 5.1: PREDICTION WITH NEW DATA

CHAPTER 6

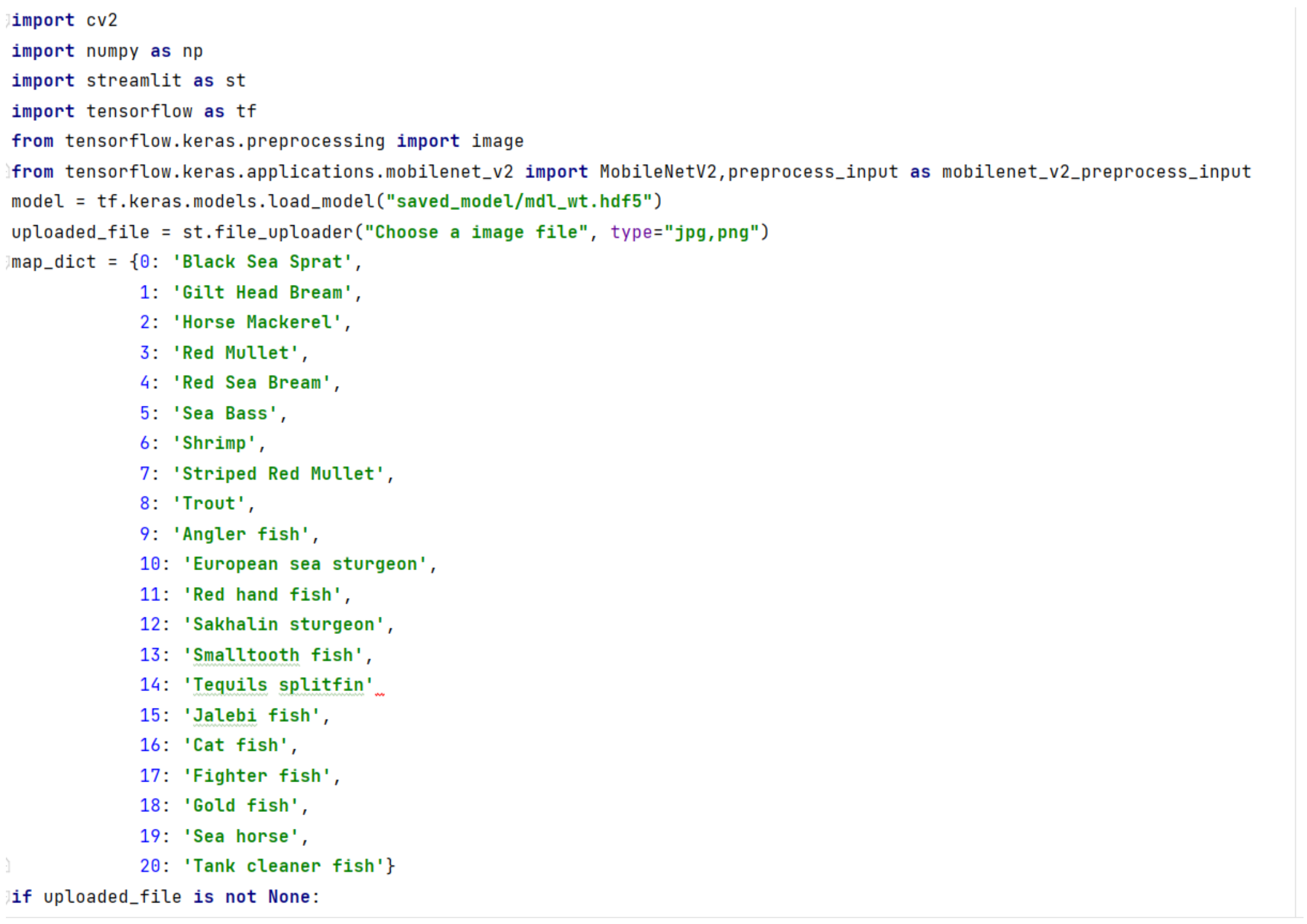
STREAMLIT CONNECTIVITY

6.1 STREAMLIT

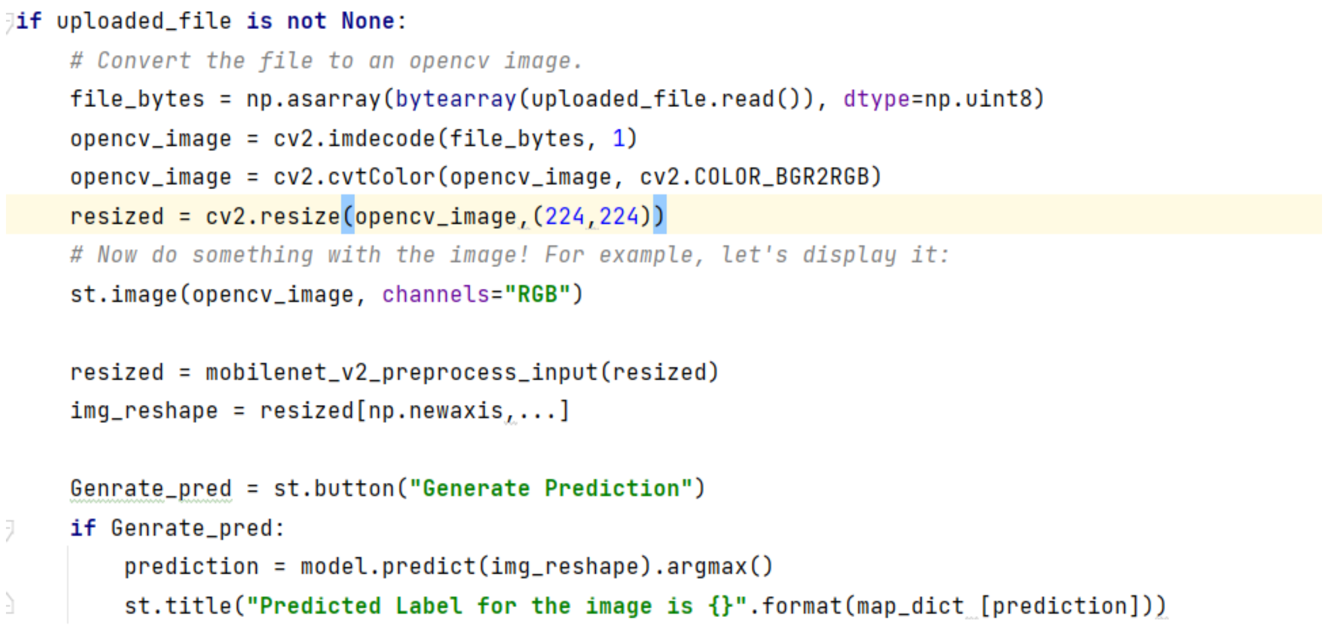
Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science. We can instantly develop web apps and deploy them easily using Streamlit It allows you to write an app the same way you write a python code. Streamlit makes it seamless to work on the interactive loop of coding and viewing results in the web app.

I have connected my backend MobileNet-V2 model with frontend Web pages by using STREAMLIT web python framework. So, we can directly upload the images of fishes and get to know name of the fishes.

The platform uses python scripting, APIs, widgets, instant deployment, team collaboration tools, and application management solutions to help data scientists and machine learning engineers create python-based applications. Applications created using Streamlit range from applications capable of real time object detection, geographic data browsers, deep dream network debuggers, to face-GAN explorers. Frameworks compatible with Streamlit include: Scikit Learn, Keras, OpenCV, PyTorch, NumPy, Seaborn, TensorFlow, Python, Matplotlib and Pandas.



```
import cv2
import numpy as np
import streamlit as st
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2,preprocess_input as mobilenet_v2_preprocess_input
model = tf.keras.models.load_model("saved_model/mdl_wt.hdf5")
uploaded_file = st.file_uploader("Choose a image file", type="jpg,png")
map_dict = {0: 'Black Sea Sprat',
            1: 'Gilt Head Bream',
            2: 'Horse Mackerel',
            3: 'Red Mullet',
            4: 'Red Sea Bream',
            5: 'Sea Bass',
            6: 'Shrimp',
            7: 'Striped Red Mullet',
            8: 'Trout',
            9: 'Angler fish',
            10: 'European sea sturgeon',
            11: 'Red hand fish',
            12: 'Sakhalin sturgeon',
            13: 'Smalltooth fish',
            14: 'Tequils splitfin',
            15: 'Jalebi fish',
            16: 'Cat fish',
            17: 'Fighter fish',
            18: 'Gold fish',
            19: 'Sea horse',
            20: 'Tank cleaner fish'}
if uploaded_file is not None:
```



```
if uploaded_file is not None:
    # Convert the file to an opencv image.
    file_bytes = np.asarray(bytearray(uploaded_file.read()), dtype=np.uint8)
    opencv_image = cv2.imdecode(file_bytes, 1)
    opencv_image = cv2.cvtColor(opencv_image, cv2.COLOR_BGR2RGB)
    resized = cv2.resize(opencv_image,(224,224))
    # Now do something with the image! For example, let's display it:
    st.image(opencv_image, channels="RGB")

    resized = mobilenet_v2_preprocess_input(resized)
    img_reshape = resized[np.newaxis,...]

    Genrate_pred = st.button("Generate Prediction")
    if Genrate_pred:
        prediction = model.predict(img_reshape).argmax()
        st.title("Predicted Label for the image is {}".format(map_dict [prediction]))
```

Figure 6.1: STREAMLIT CONNECTIVITY

6.2 IDENTIFICATION OF FISH SPECIES USING WEB APPLICATION

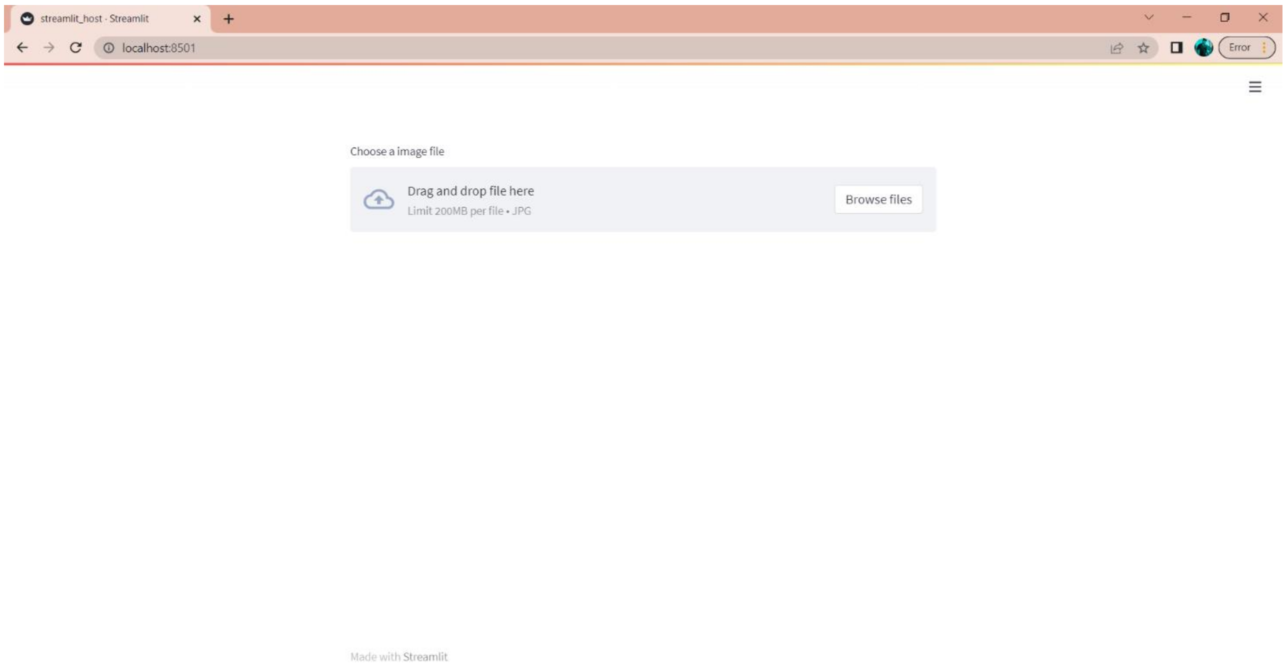


Figure 6.21: WEB PAGE

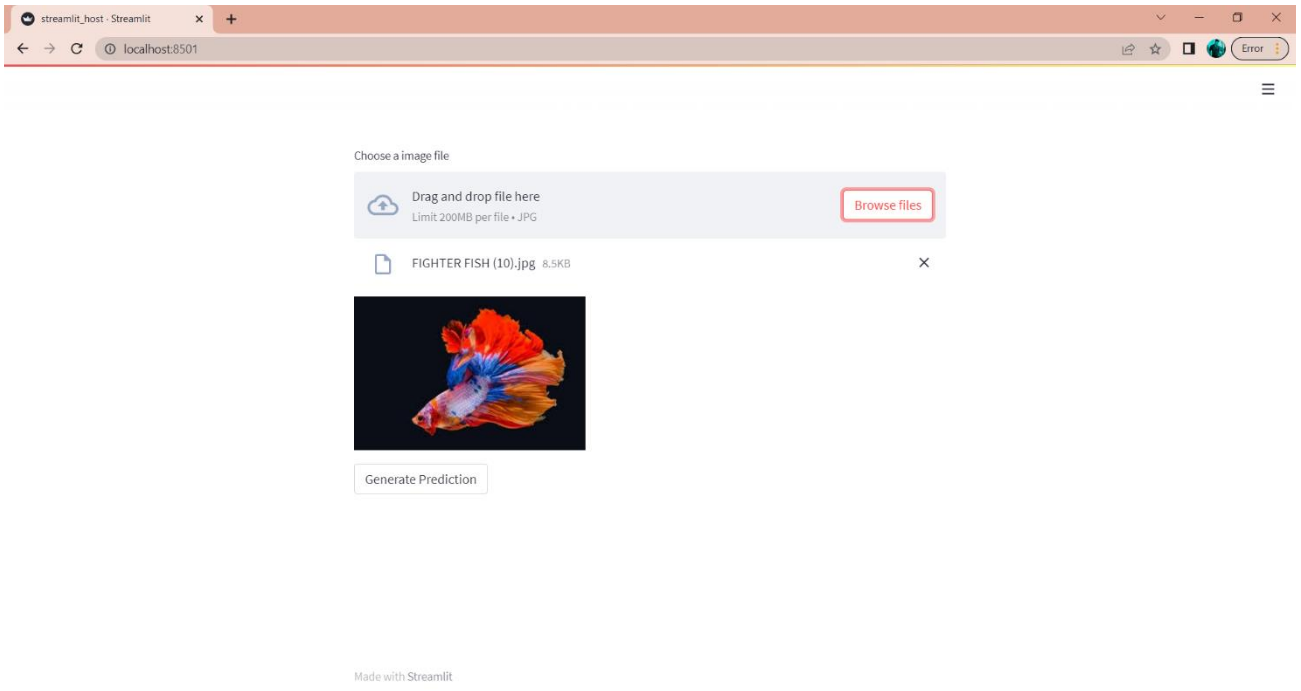
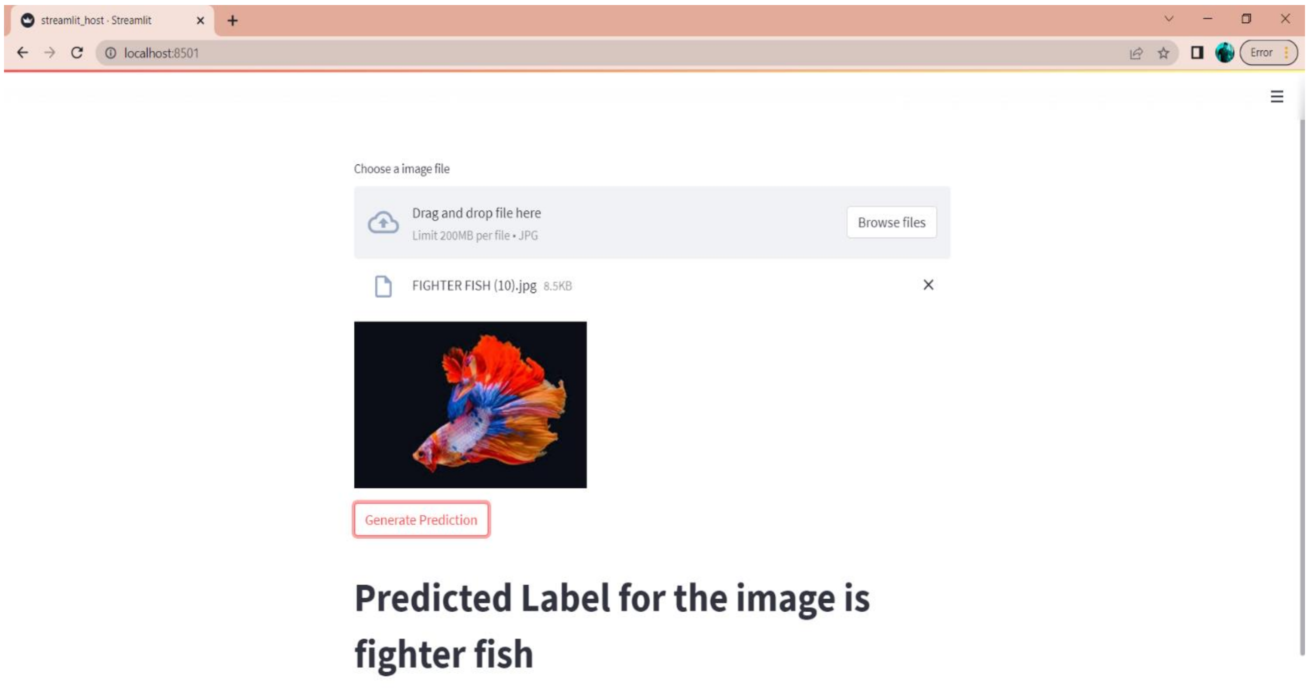


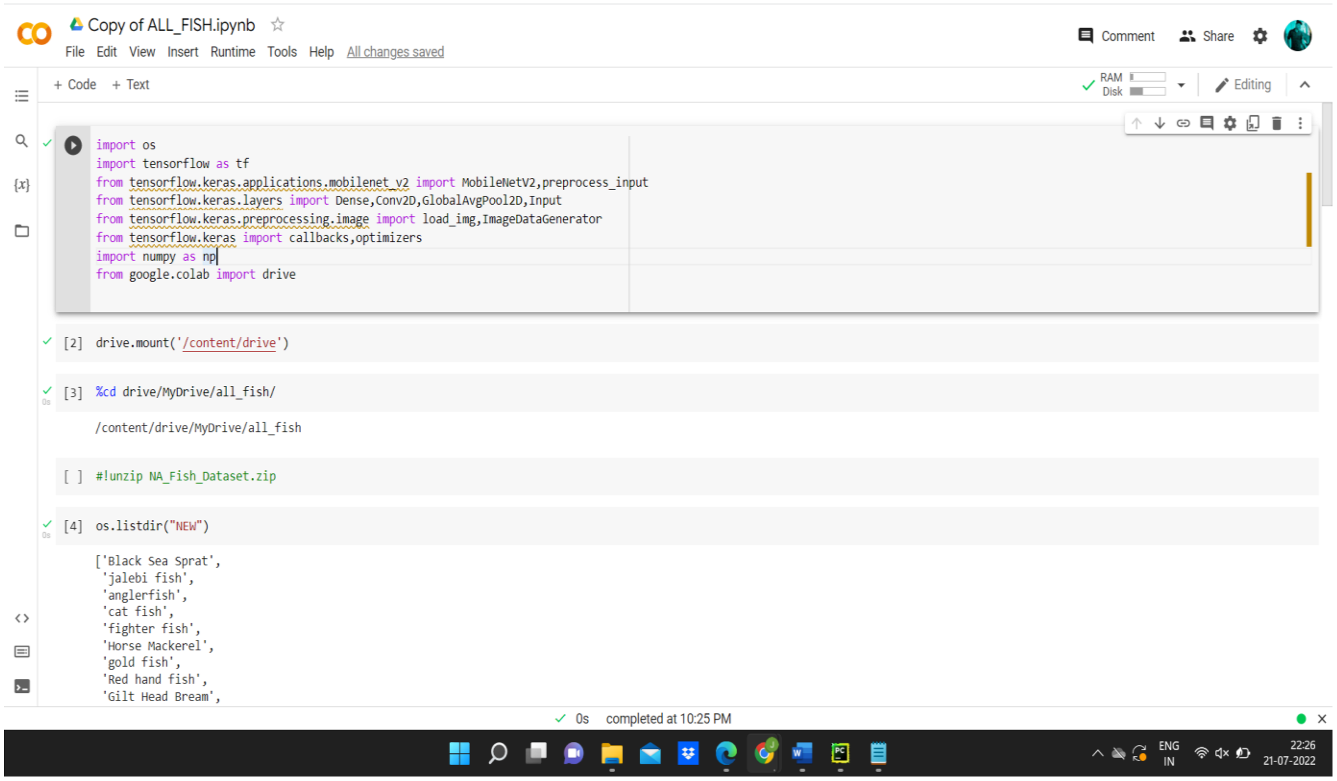
Figure 6.22: BROWSING IMAGE


streamlit_host · Streamlit
localhost:8501
Drag and drop file here
Limit 200MB per file • JPG
Browse files
FIGHTER FISH (10).jpg 8.5KB
Generate Prediction
Error

Choose a image file

Predicted Label for the image is fighter fish

Figure 6.23: PREDICTION PAGE



Copy of ALL_FISH.ipynb

```
import os
import tensorflow as tf
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2,preprocess_input
from tensorflow.keras.layers import Dense,Conv2D,GlobalAvgPool2D,Input
from tensorflow.keras.preprocessing.image import load_img,ImageDataGenerator
from tensorflow.keras import callbacks,optimizers
import numpy as np
from google.colab import drive
```

```
[2] drive.mount('/content/drive')
```

```
[3] %cd drive/MyDrive/all_fish/
```

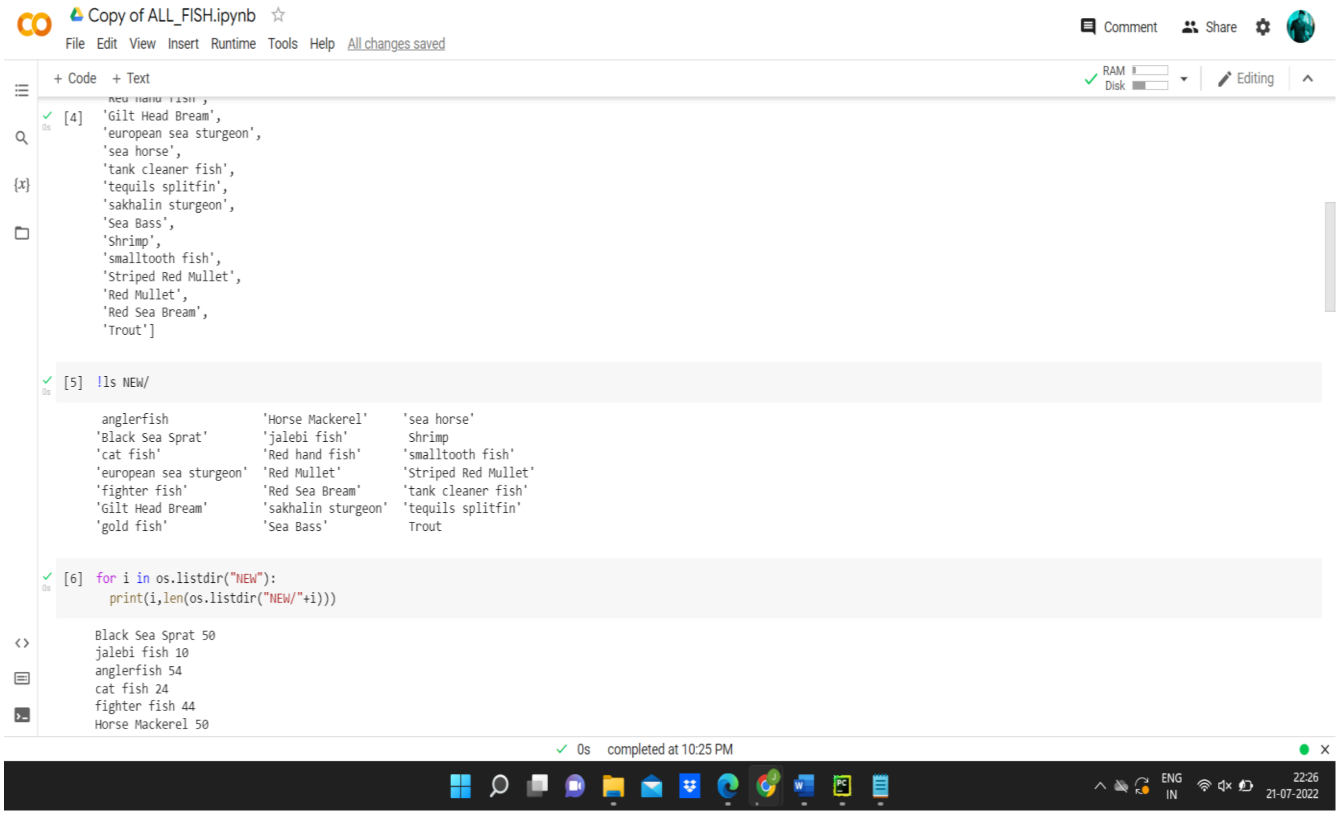
```
/content/drive/MyDrive/all_fish
```

```
[ ] #!unzip NA_Fish_Dataset.zip
```

```
[4] os.listdir("NEW")
```

```
['Black Sea Sprat',
 'jalebi fish',
 'anglerfish',
 'cat fish',
 'fighter fish',
 'Horse Mackerel',
 'gold fish',
 'Red hand fish',
 'Gilt Head Bream',
```

completed at 10:25 PM



Copy of ALL_FISH.ipynb

```
 'Gilt Head Bream',
 'european sea sturgeon',
 'sea horse',
 'tank cleaner fish',
 'tequils splitfin',
 'sakhalin sturgeon',
 'Sea Bass',
 'Shrimp',
 'smalltooth fish',
 'Striped Red Mullet',
 'Red Mullet',
 'Red Sea Bream',
 'Trout']
```

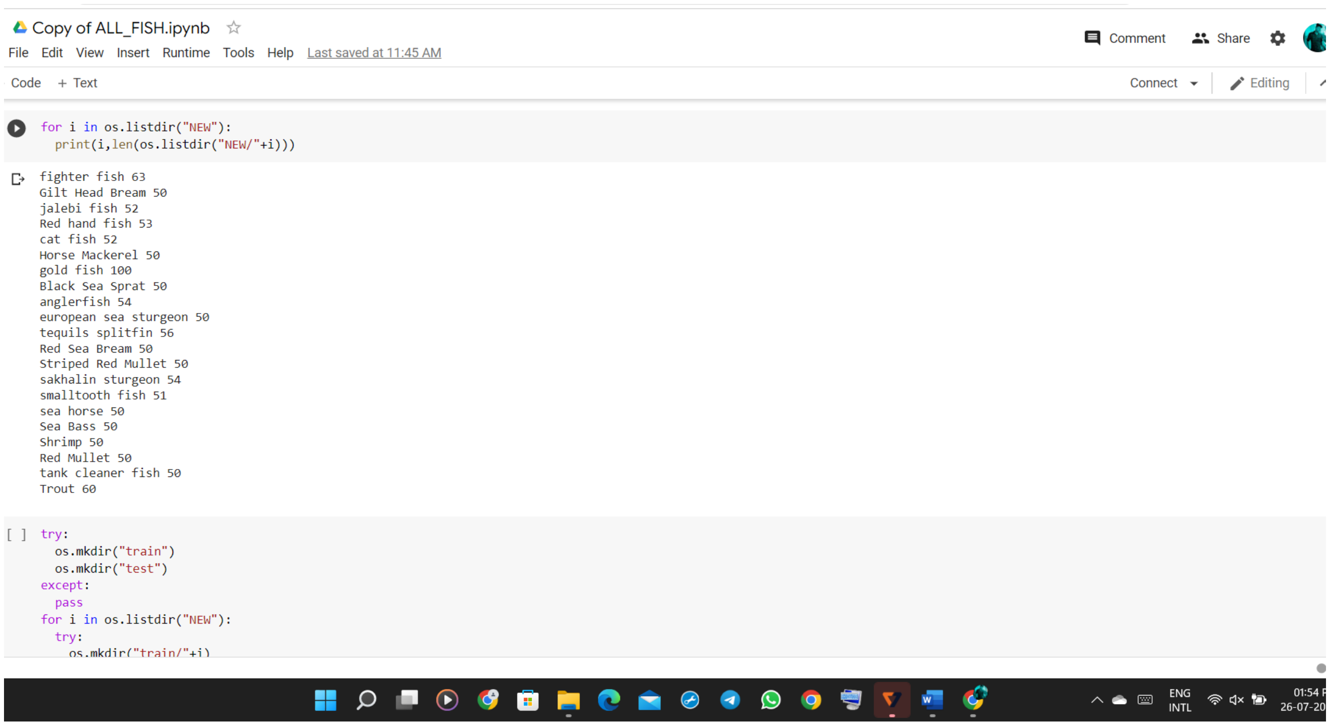
```
[5] !ls NEW/
```

```
 anglerfish              'Horse Mackerel'     'sea horse'
'Black Sea Sprat'        'jalebi fish'         Shrimp
'cat fish'               'Red hand fish'      'smalltooth fish'
'european sea sturgeon'  'Red Mullet'         'Striped Red Mullet'
'fighter fish'           'Red Sea Bream'      'tank cleaner fish'
'Gilt Head Bream'        'sakhalin sturgeon'  'tequils splitfin'
'gold fish'              'Sea Bass'            Trout
```

```
[6] for i in os.listdir("NEW"):
      print(i,len(os.listdir("NEW/"+i)))
```

```
Black Sea Sprat 50
jalebi fish 10
anglerfish 54
cat fish 24
fighter fish 44
Horse Mackerel 50
```

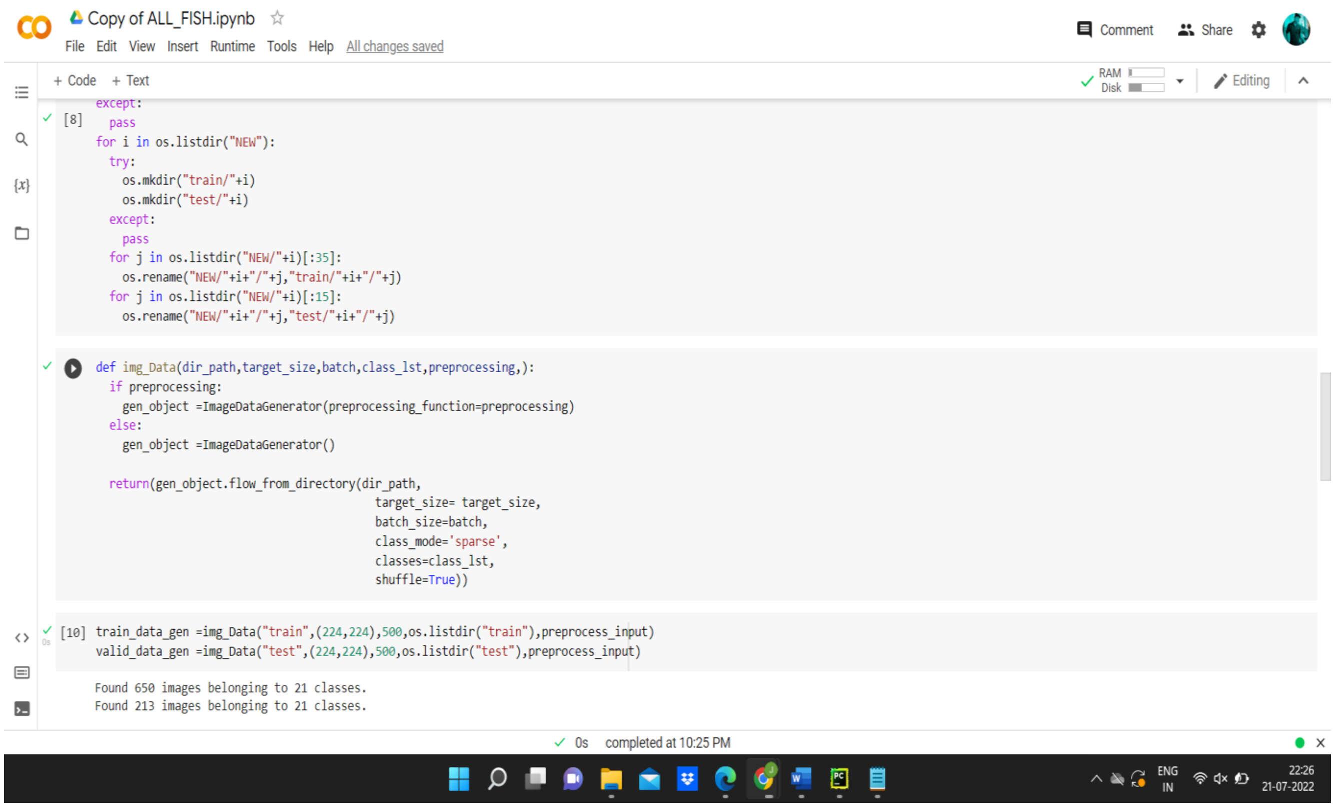
completed at 10:25 PM


Copy of ALL_FISH.ipynb
File Edit View Insert Runtime Tools Help Last saved at 11:45 AM
Comment Share
Code + Text
Connect Editing

```
for i in os.listdir("NEW"):
  print(i,len(os.listdir("NEW/"+i)))
```

```
fighter fish 63
Gilt Head Bream 50
jalebi fish 52
Red hand fish 53
cat fish 52
Horse Mackerel 50
gold fish 100
Black Sea Sprat 50
anglerfish 54
european sea sturgeon 50
tequils splitfin 56
Red Sea Bream 50
Striped Red Mullet 50
sakhalin sturgeon 54
smalltooth fish 51
sea horse 50
Sea Bass 50
Shrimp 50
Red Mullet 50
tank cleaner fish 50
Trout 60
```

```
try:
  os.mkdir("train")
  os.mkdir("test")
except:
  pass
for i in os.listdir("NEW"):
  try:
    os.mkdir("train/"+i)
```

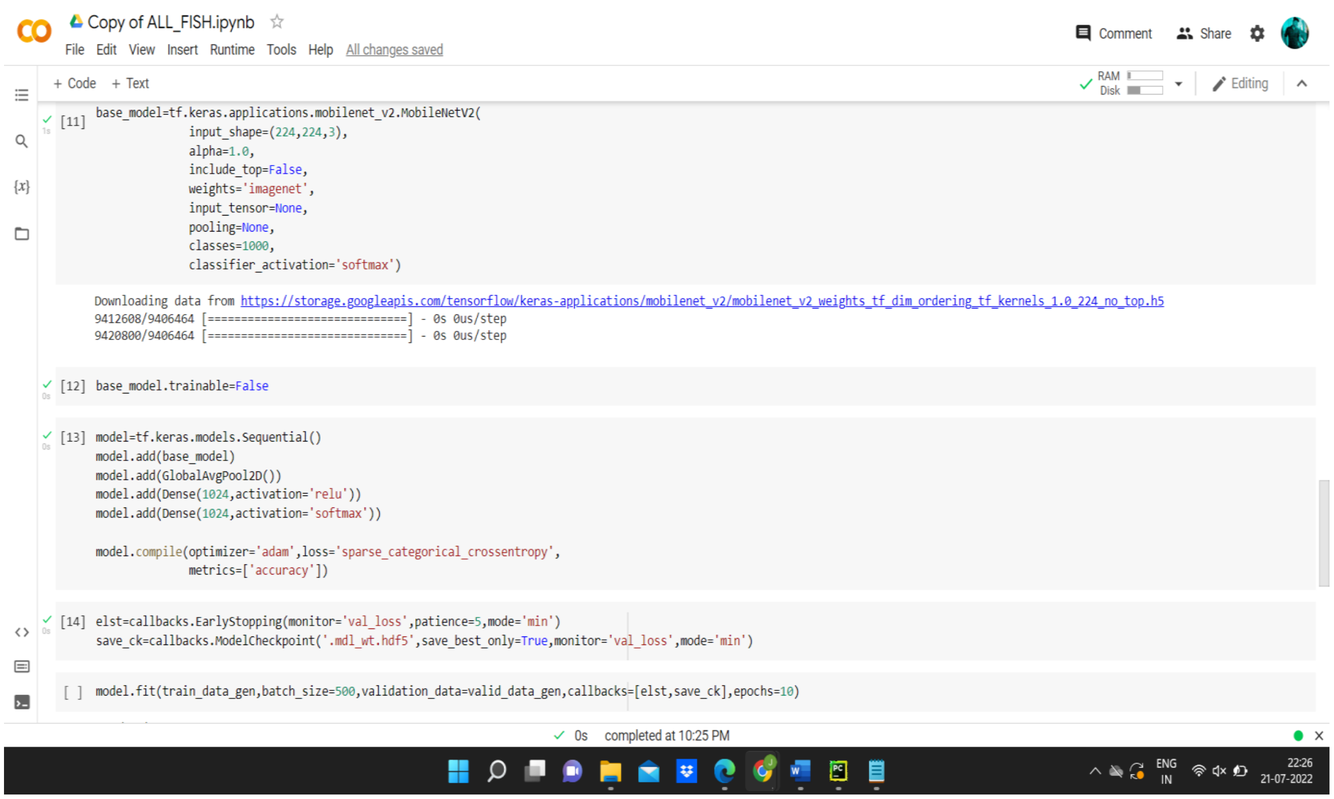


```
except:
    pass
for i in os.listdir("NEW"):
  try:
    os.mkdir("train/"+i)
    os.mkdir("test/"+i)
  except:
    pass
  for j in os.listdir("NEW/"+i)[:35]:
    os.rename("NEW/"+i+"/"+j,"train/"+i+"/"+j)
  for j in os.listdir("NEW/"+i)[:15]:
    os.rename("NEW/"+i+"/"+j,"test/"+i+"/"+j)
```

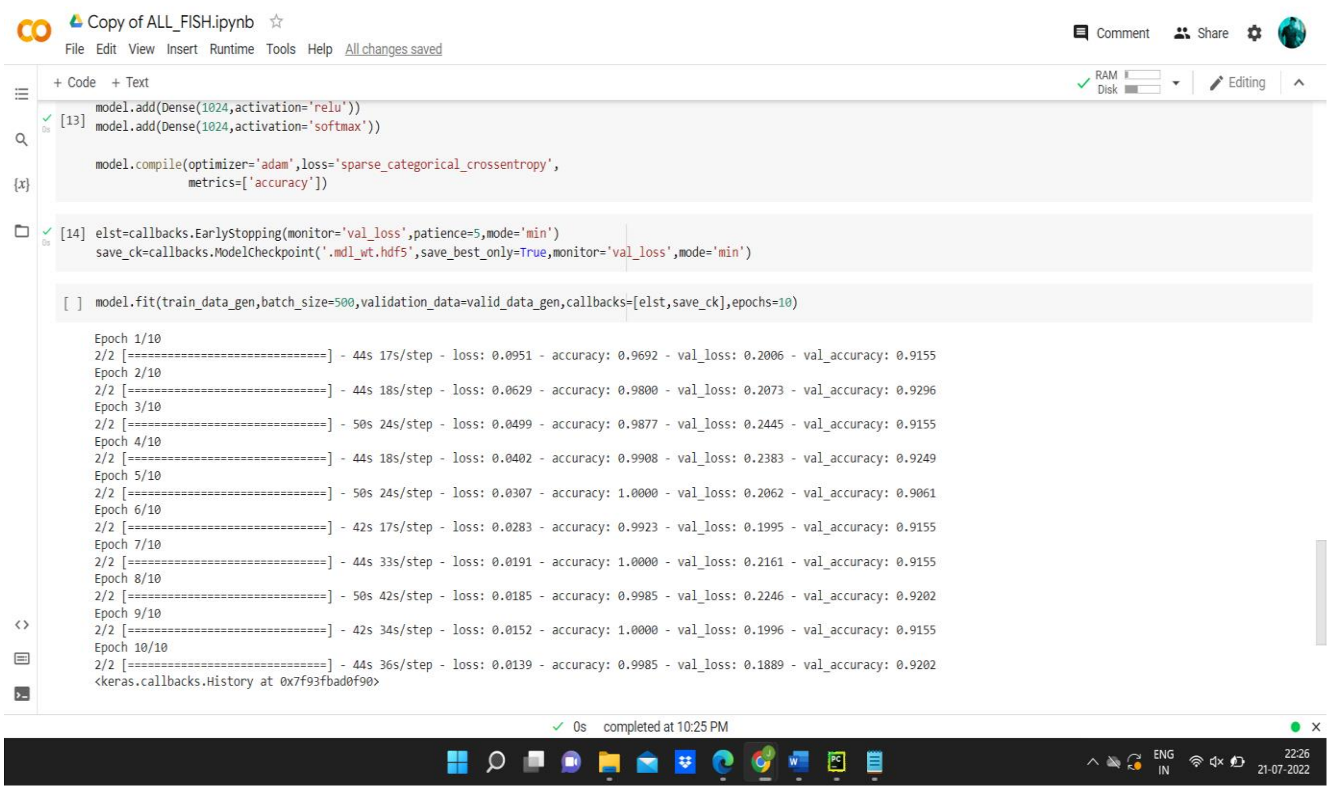
```
def img_Data(dir_path,target_size,batch,class_lst,preprocessing,):
  if preprocessing:
    gen_object =ImageDataGenerator(preprocessing_function=preprocessing)
  else:
    gen_object =ImageDataGenerator()

  return(gen_object.flow_from_directory(dir_path,
                                        target_size= target_size,
                                        batch_size=batch,
                                        class_mode='sparse',
                                        classes=class_lst,
                                        shuffle=True))
```

```
train_data_gen =img_Data("train",(224,224),500,os.listdir("train"),preprocess_input)
valid_data_gen =img_Data("test",(224,224),500,os.listdir("test"),preprocess_input)
```

```
Found 650 images belonging to 21 classes.
Found 213 images belonging to 21 classes.
```

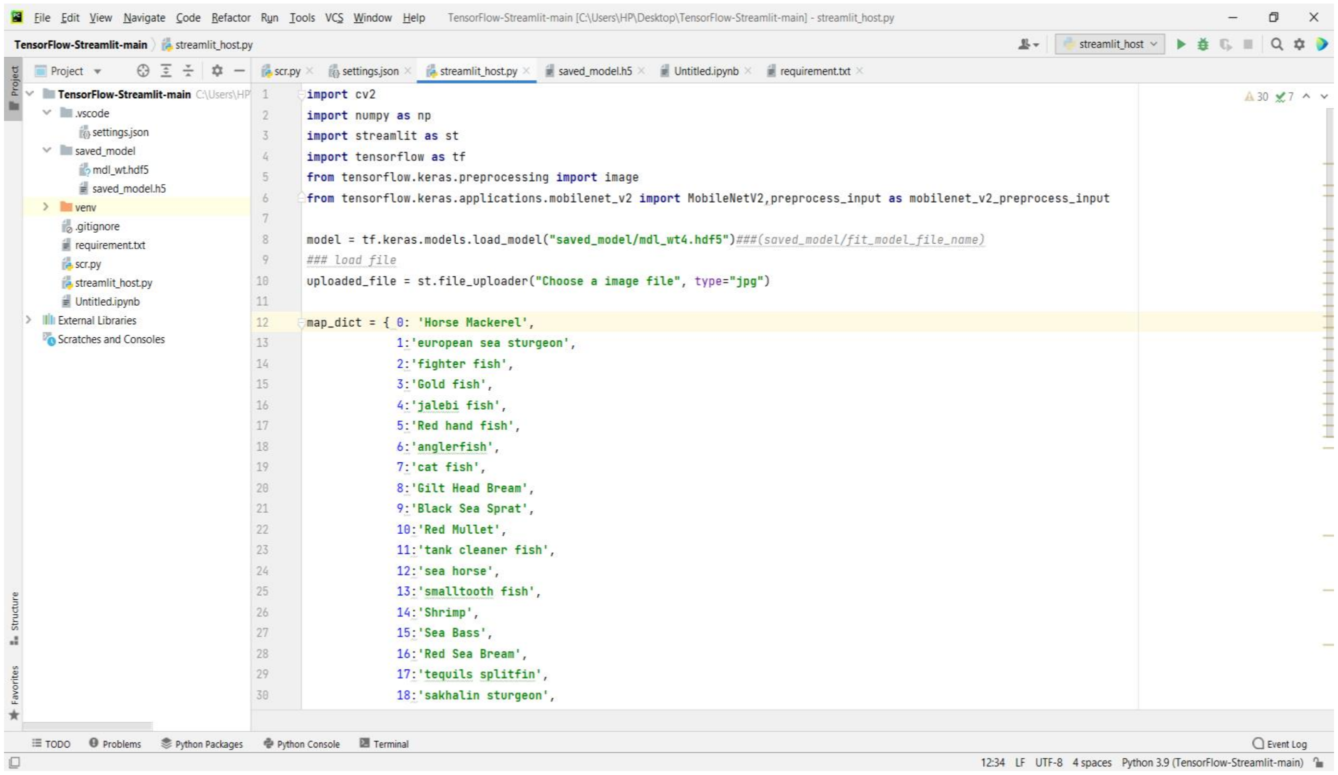


```
[11] base_model=tf.keras.applications.mobilenet_v2.MobileNetV2(
                 input_shape=(224,224,3),
                 alpha=1.0,
                 include_top=False,
                 weights='imagenet',
                 input_tensor=None,
                 pooling=None,
                 classes=1000,
                 classifier_activation='softmax')

     Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5
     9412608/9406464 [==============================] - 0s 0us/step
     9420800/9406464 [==============================] - 0s 0us/step

[12] base_model.trainable=False

[13] model=tf.keras.models.Sequential()
     model.add(base_model)
     model.add(GlobalAvgPool2D())
     model.add(Dense(1024,activation='relu'))
     model.add(Dense(1024,activation='softmax'))

     model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
                 metrics=['accuracy'])

[14] elst=callbacks.EarlyStopping(monitor='val_loss',patience=5,mode='min')
     save_ck=callbacks.ModelCheckpoint('.mdl_wt.hdf5',save_best_only=True,monitor='val_loss',mode='min')

[ ]  model.fit(train_data_gen,batch_size=500,validation_data=valid_data_gen,callbacks=[elst,save_ck],epochs=10)
```

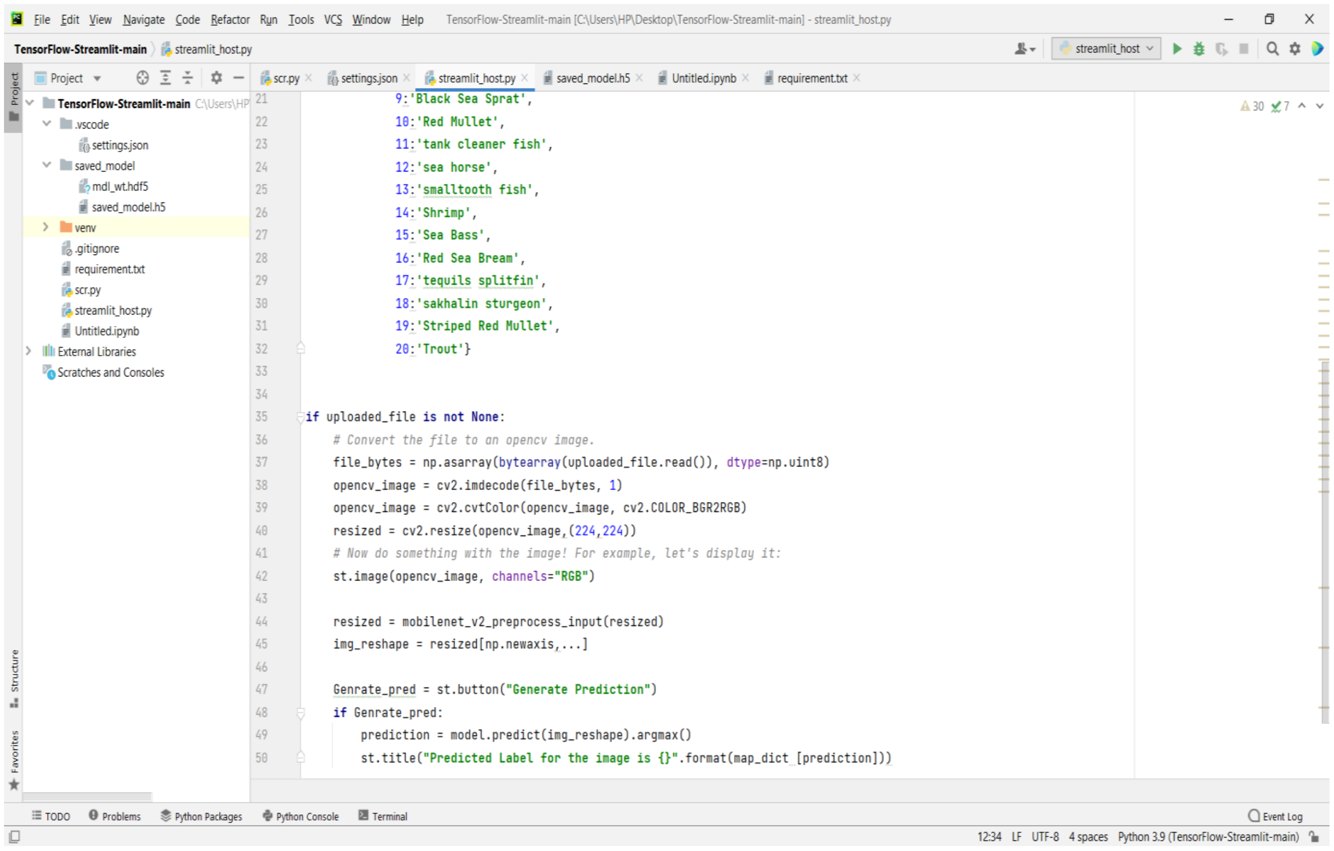


```
     model.add(Dense(1024,activation='relu'))
[13] model.add(Dense(1024,activation='softmax'))

     model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
                 metrics=['accuracy'])

[14] elst=callbacks.EarlyStopping(monitor='val_loss',patience=5,mode='min')
     save_ck=callbacks.ModelCheckpoint('.mdl_wt.hdf5',save_best_only=True,monitor='val_loss',mode='min')

[ ]  model.fit(train_data_gen,batch_size=500,validation_data=valid_data_gen,callbacks=[elst,save_ck],epochs=10)

     Epoch 1/10
     2/2 [==============================] - 44s 17s/step - loss: 0.0951 - accuracy: 0.9692 - val_loss: 0.2006 - val_accuracy: 0.9155
     Epoch 2/10
     2/2 [==============================] - 44s 18s/step - loss: 0.0629 - accuracy: 0.9800 - val_loss: 0.2073 - val_accuracy: 0.9296
     Epoch 3/10
     2/2 [==============================] - 50s 24s/step - loss: 0.0499 - accuracy: 0.9877 - val_loss: 0.2445 - val_accuracy: 0.9155
     Epoch 4/10
     2/2 [==============================] - 44s 18s/step - loss: 0.0402 - accuracy: 0.9908 - val_loss: 0.2383 - val_accuracy: 0.9249
     Epoch 5/10
     2/2 [==============================] - 50s 24s/step - loss: 0.0307 - accuracy: 1.0000 - val_loss: 0.2062 - val_accuracy: 0.9061
     Epoch 6/10
     2/2 [==============================] - 42s 17s/step - loss: 0.0283 - accuracy: 0.9923 - val_loss: 0.1995 - val_accuracy: 0.9155
     Epoch 7/10
     2/2 [==============================] - 44s 33s/step - loss: 0.0191 - accuracy: 1.0000 - val_loss: 0.2161 - val_accuracy: 0.9155
     Epoch 8/10
     2/2 [==============================] - 50s 42s/step - loss: 0.0185 - accuracy: 0.9985 - val_loss: 0.2246 - val_accuracy: 0.9202
     Epoch 9/10
     2/2 [==============================] - 42s 34s/step - loss: 0.0152 - accuracy: 1.0000 - val_loss: 0.1996 - val_accuracy: 0.9155
     Epoch 10/10
     2/2 [==============================] - 44s 36s/step - loss: 0.0139 - accuracy: 0.9985 - val_loss: 0.1889 - val_accuracy: 0.9202
     <keras.callbacks.History at 0x7f93fbad0f90>
```

STREAMLIT IMPLEMENTATION



```python
import cv2
import numpy as np
import streamlit as st
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2,preprocess_input as mobilenet_v2_preprocess_input

model = tf.keras.models.load_model("saved_model/mdl_wt4.hdf5")###(saved_model/fit_model_file_name)
### load file
uploaded_file = st.file_uploader("Choose a image file", type="jpg")

map_dict = { 0: 'Horse Mackerel',
             1:'european sea sturgeon',
             2:'fighter fish',
             3:'Gold fish',
             4:'jalebi fish',
             5:'Red hand fish',
             6:'anglerfish',
             7:'cat fish',
             8:'Gilt Head Bream',
             9:'Black Sea Sprat',
             10:'Red Mullet',
             11:'tank cleaner fish',
             12:'sea horse',
             13:'smalltooth fish',
             14:'Shrimp',
             15:'Sea Bass',
             16:'Red Sea Bream',
             17:'tequils splitfin',
             18:'sakhalin sturgeon',
```



```python
             9:'Black Sea Sprat',
             10:'Red Mullet',
             11:'tank cleaner fish',
             12:'sea horse',
             13:'smalltooth fish',
             14:'Shrimp',
             15:'Sea Bass',
             16:'Red Sea Bream',
             17:'tequils splitfin',
             18:'sakhalin sturgeon',
             19:'Striped Red Mullet',
             20:'Trout'}


if uploaded_file is not None:
    # Convert the file to an opencv image.
    file_bytes = np.asarray(bytearray(uploaded_file.read()), dtype=np.uint8)
    opencv_image = cv2.imdecode(file_bytes, 1)
    opencv_image = cv2.cvtColor(opencv_image, cv2.COLOR_BGR2RGB)
    resized = cv2.resize(opencv_image,(224,224))
    # Now do something with the image! For example, let's display it:
    st.image(opencv_image, channels="RGB")

    resized = mobilenet_v2_preprocess_input(resized)
    img_reshape = resized[np.newaxis,...]

    Genrate_pred = st.button("Generate Prediction")
    if Genrate_pred:
        prediction = model.predict(img_reshape).argmax()
        st.title("Predicted Label for the image is {}".format(map_dict [prediction]))
```

CHAPTER 7

CONCLUSION

I have evaluated two kinds of deep learning-based method for automated Identification of fish species, MobileNet V2. In my project, I have exploited and evaluated the application of Faster CNN approach to Identify the fish species accuratelyin color fish images. I tested 250 images on MobileNet V2 architecture. Experimental result showed that MobileNet V2 offered better performance. The test results obtained by making use of unique data sets with 1000 images demonstrate that this arrangement can be a significant alternative for the computer design aid framework for the large-scale screening programs of Identification of fish species. In the future, we will continue our research to develop new architectures for identify the fish species on large database.

CHAPTER 8

REFERENCE

[1] Ahmad S, Ahsan J, Faisal S, Ajmal M, Mark S, James S, Euan H (2016) Fish species classification in unconstrained underwater environments based on deep learning. Limnol. Oceanogr. Methods, 14, 570-585.

[2] Alcaraz C, Gholami Z, Esmaeili HR, García-Berthou E (2015) Herbivory and seasonal changes in diet of a highly endemic cyprinodontid fish (Aphanius farsicus). Environ. Biol. Fishes, 98, 1541-1554.

[3] Allken V, Handegard NO, Rosen S, Schreyeck T, Mahiout T, Malde K (2019) Fish species identification using a convolutional neural network trained on synthetic data. ICES J. Mar. Sci., 76, 342-349.