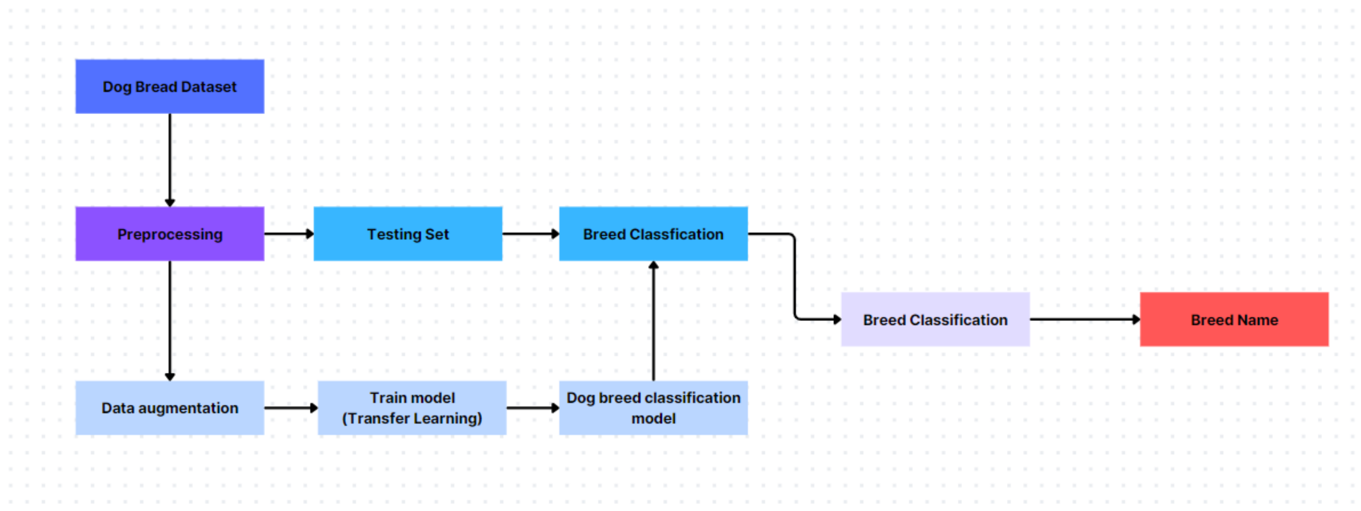
Project ID: SI-GuidedProject-611456-1698332996
Dog Breed Identification using Transfer Learning

Introduction:

By predicting the breed it helps to better care and also prevents unethical breeding practices. Understanding a breed-specific traits and health issues allows the owners and the veterinarians to provide better care. Breed identification can help as an education tool that helps the owners to learn more about their dog, their history and their behavior. This also challenges breed- based stereotypes and discrimination, resulting in more humane treatment of all dog breeds.

Deep learning algorithms can learn to recognize these unique features and patterns in images to make accurate predictions about the breed of a given dog. Here we use Transfer Learning Approach i.e., mobileNetV2 Architecture for dog breed identification, a large dataset of labeled dog images is required.

Technical Architecture:



Prerequisites:

To complete this project we require an platform to train the model:

I used KAGGLE for training the model. Kaggle is an open source platform where we can code and edit it. And we can explore more datasets in the kaggle. Kaggle provides a TPU accelerometer which increases the efficiency in training.

1. **To build Deep learning models you must require the following packages**
   - **Numpy**
     It used to handle multi dimensional arrays and operates large mathematical operations.

- **Pandas**
  It used for handling data frames and csv files.
- **Tensorflow**
  It mainly helps for training for deep learning models.
- **Opencv**
  It helps for the image processing.
- **flask**
  It is a framework for web app.

- **Python packages**
  Install all necessary packages.
- **Deep Learning Concepts**

  o **CNN:** a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery.
  CNN Basic

  o **ImageNetV2:** ImageNetV2 is a deep convolutional neural network architecture for image classification, consisting of 16 layers with small convolution filters. **ImageNetV2**

  o **Flask:** Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications.
  Flask Basics

Project Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques of Convolutional Neural Network.
- Gain a broad understanding of image data.
- Know how to pre-process/clean the data using different data preprocessing techniques.
- know how to build a web application using the Flask framework.

Project Flow:

● The user interacts with the UI (User Interface) to choose the image. ● The chosen image is analyzed by the model which is integrated with flask application. ● CNN Models analyze the image, then prediction is showcased on the Flask

UI. To accomplish this, we must complete all the activities and tasks listed below.

o Data Collection.
  o Collect data into train and test with labels.
  o Now we convert the unique breeds into categorical.
  o Now we glob the path to retrieve all files in specific patterns.
  o Now to merge those files with categorical id.

o Data Preprocessing.
  o Now build a function for imagenetv2.
  o Read the image using opencv and resize it.

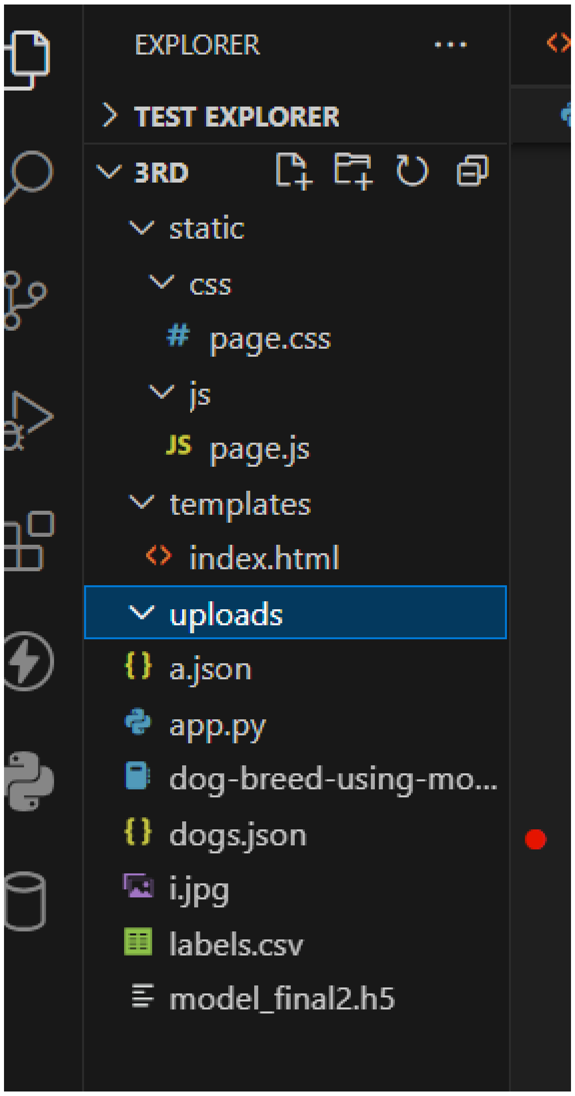
- o Parse functions from tensorflow library to convert unordered into arrays.
- o Then we split files with categorical labels into train and validation.
- o Now we tune the model.
- o Train the model.
- o Save the model.

o Application Building

- o Create an HTML file
- o Build Python Code

Project Structure:

Create a Project folder which contains files as shown below.



```
static
  css
    page.css
  js
    page.js
templates
  index.html
uploads
a.json
app.py
dog-breed-using-mo...
dogs.json
i.jpg
labels.csv
model_final2.h5
```

- We are building a Flask Application that needs HTML pages stored in the **templates.** folder and a python script **app.py** for server-side scripting
- we need the model which is saved and the saved model in this content is a **model_final2.h5**

- **templates** folder contains index.html.
- The **Static** folder contains css file, images.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem.

How might we assure users that the model accurately predicts the breed ?

Activity 2: Business requirements

Here are some potential business requirements for Dog Breed Identification.

a. **Accurate Prediction:**
The predictor must be able to accurately classify the images of different Dog Breeds. So that there will be no misclassification which decreases the accuracy of the model.
b. **Real-time data acquisition:**
The predictor must be able to acquire real-time data from the various sources. The data acquisition must be seamless and efficient to ensure that the predictor is always up-to date with the latest information.

c. **User-friendly interface:**
The predictor must have a user-friendly interface that is easy to navigate and understand. The interface should present the results of the predictor in a clear and concise manner.

d. Report generation:
Generate a report outlining the predicted Dog Breeds. By analyzing data and applying advanced algorithms, the project generates detailed reports that include key findings, Dog Breed classifications, and relevant insights. The report should be presented in a clear and concise manner, with appropriate insights to help patients confirm their results.

Activity 3: Literature Survey (Student Will Write)
The following are existing obstacles to for dog breed identification:
I. Variations in learned features make it challenging to modify pre-trained models to correctly identify dog breeds.
II. Difficulties in achieving the ideal fine-tuning equilibrium and managing overfitting when adjusting The model
III. Unbalanced datasets that favor some breeds over others and ignore others.
IV. Difficulties in correctly recognizing dogs that have features from more than one breed, or mixed breeds.
V. Restraints on the model's applicability to uncommon or uncommon breeds.
These problems lead to less accurate solutions.

Activity 4: Social or Business Impact.

The Dog Breed Identification project can have both social and business impacts.

Social Impact:
By accurately predicting the breed it helps to better care and also prevents unethical breeding

practices. Understanding a breed-specific traits and health issues allows the owners and the veterinarians to provide better care. Breed identification can help as an education tool that helps the owners to learn more about their dog, their history and their behavior. This also challenges breed-based stereotypes and discrimination, resulting in more humane treatment of all dog breeds.

Business Impact:

Dog breed identification can be used by pet retailers and breeders to give accurate information about the breeds they have. This assists prospective buyers in selecting a dog that best suits their needs in terms of compatibility, lifestyle, and preferences. The growth and expansion of pet services and products can be influenced by the identification of dog breeds. Pet service providers, including trainers, groomers, and veterinarians, can customize their services to cater to the specific requirements of various breeds by having a thorough understanding of specific breed characteristics. Additionally, it can guide the creation and promotion of goods like toys, food, and accessories tailored to a particular breed.

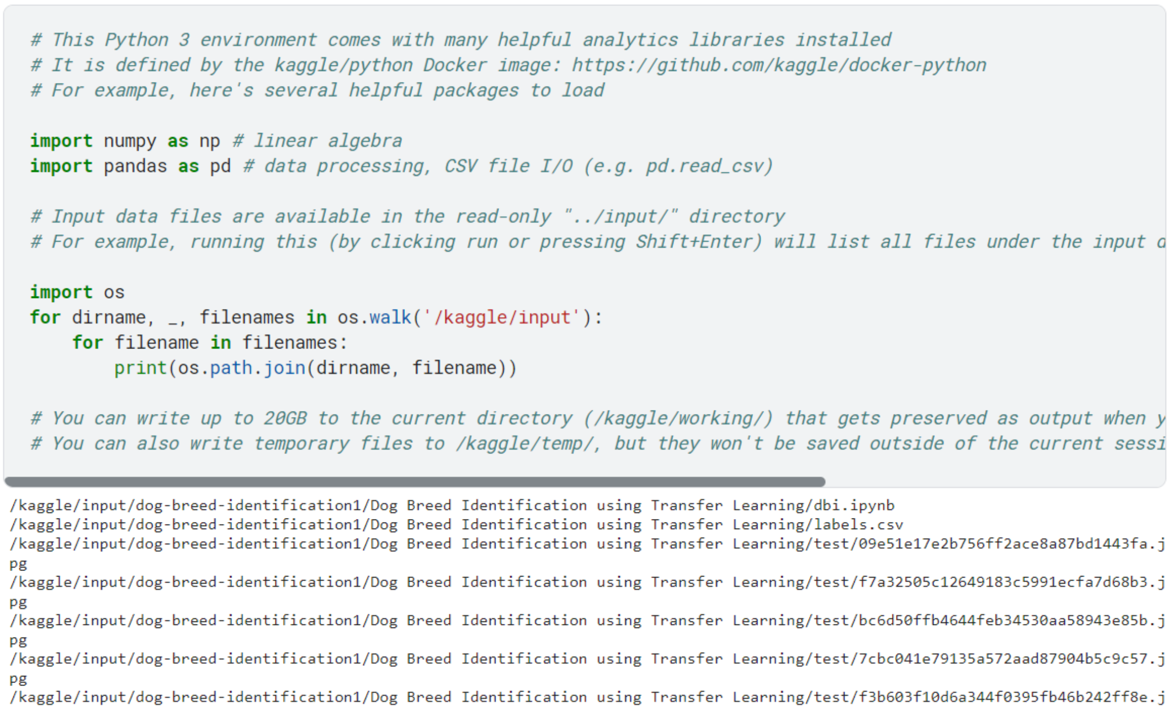
Milestone 2: Data Collection & Image Preprocessing:

Here we have taken 10222 images of 120 breeds. We have labeled images using file id in labels.csv file. By referring to breed names in the dataframe we split data into train and validation.

Download the Dataset -
https://www.kaggle.com/competitions/dog-breed-identification/data?select=train

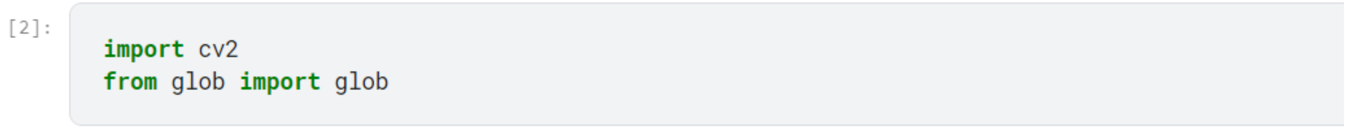
In image processing for every image we rescaled the image using opencv.

Activity 1: Importing necessary libraries.



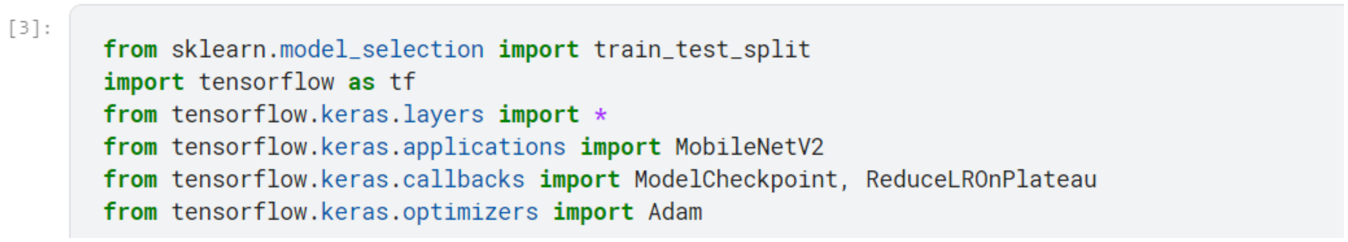
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input d

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when y
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current sessi
```

```
/kaggle/input/dog-breed-identification1/Dog Breed Identification using Transfer Learning/dbi.ipynb
/kaggle/input/dog-breed-identification1/Dog Breed Identification using Transfer Learning/labels.csv
/kaggle/input/dog-breed-identification1/Dog Breed Identification using Transfer Learning/test/09e51e17e2b756ff2ace8a87bd1443fa.j
pg
/kaggle/input/dog-breed-identification1/Dog Breed Identification using Transfer Learning/test/f7a32505c12649183c5991ecfa7d68b3.j
pg
/kaggle/input/dog-breed-identification1/Dog Breed Identification using Transfer Learning/test/bc6d50ffb4644feb34530aa58943e85b.j
pg
/kaggle/input/dog-breed-identification1/Dog Breed Identification using Transfer Learning/test/7cbc041e79135a572aad87904b5c9c57.j
pg
/kaggle/input/dog-breed-identification1/Dog Breed Identification using Transfer Learning/test/f3b603f10d6a344f0395fb46b242ff8e.j
```



[2]:

```
import cv2
from glob import glob
```

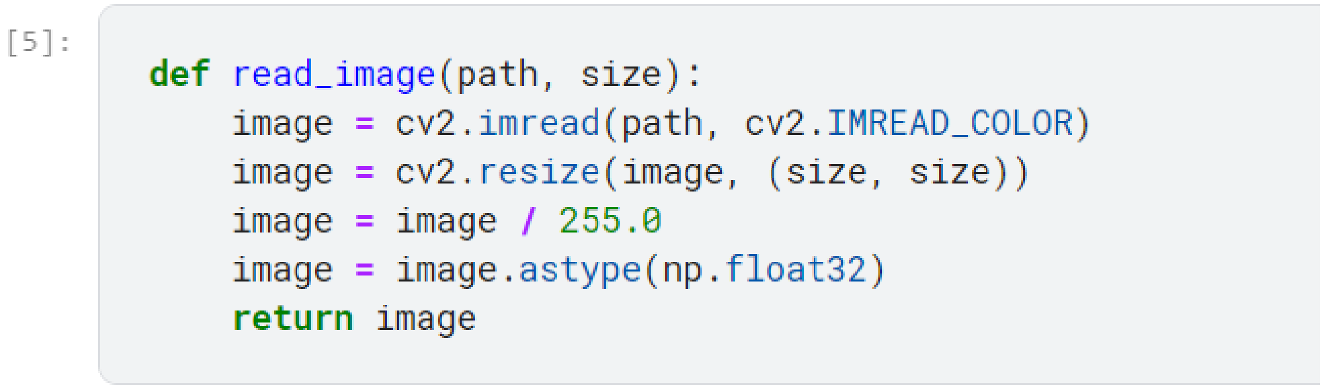


[3]:

```
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.layers import *
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam
```

Activity 2: Defining function for reading image and resizing it using cv2

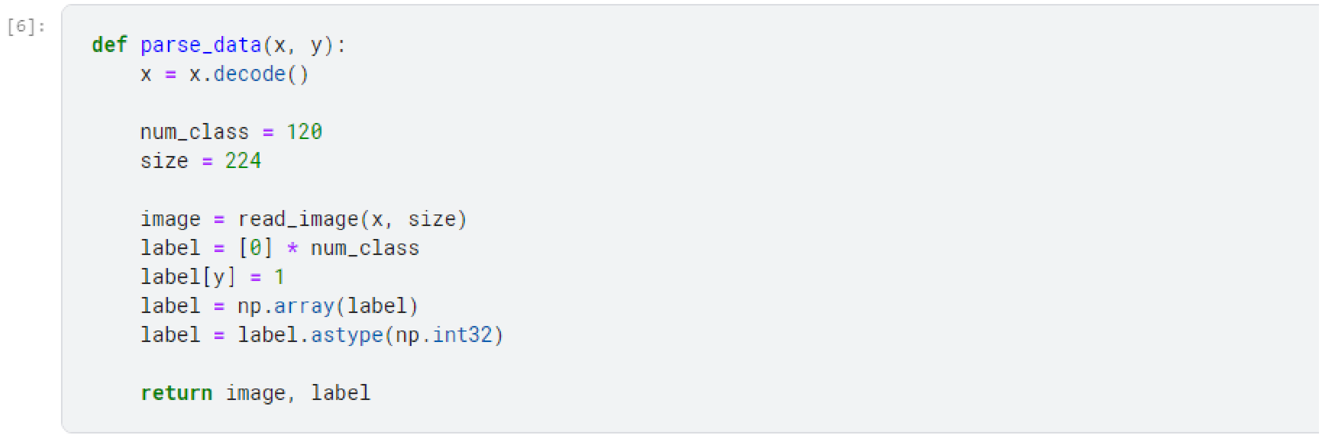
Image reading using cv2



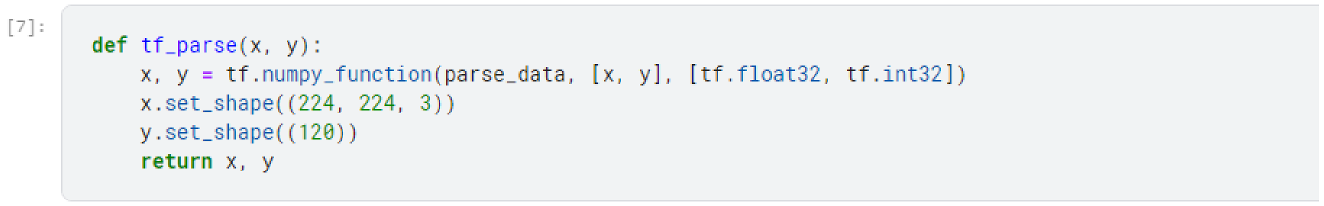
[5]:

```
def read_image(path, size):
    image = cv2.imread(path, cv2.IMREAD_COLOR)
    image = cv2.resize(image, (size, size))
    image = image / 255.0
    image = image.astype(np.float32)
    return image
```

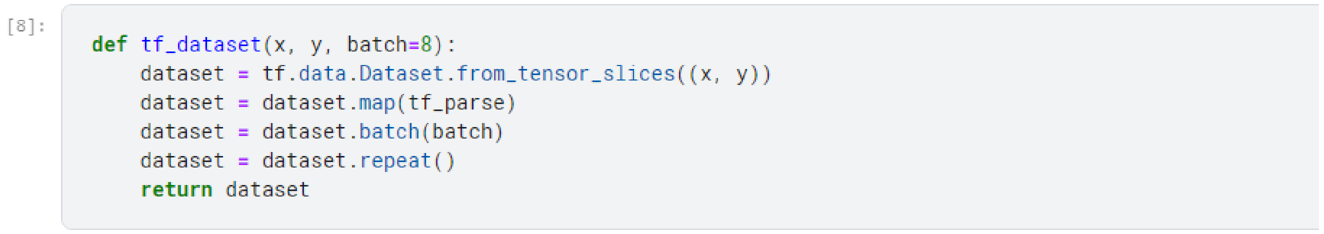
Activity 3: Converting data and resizing and considering file multiple times



```
[6]:
def parse_data(x, y):
    x = x.decode()

    num_class = 120
    size = 224

    image = read_image(x, size)
    label = [0] * num_class
    label[y] = 1
    label = np.array(label)
    label = label.astype(np.int32)

    return image, label
```



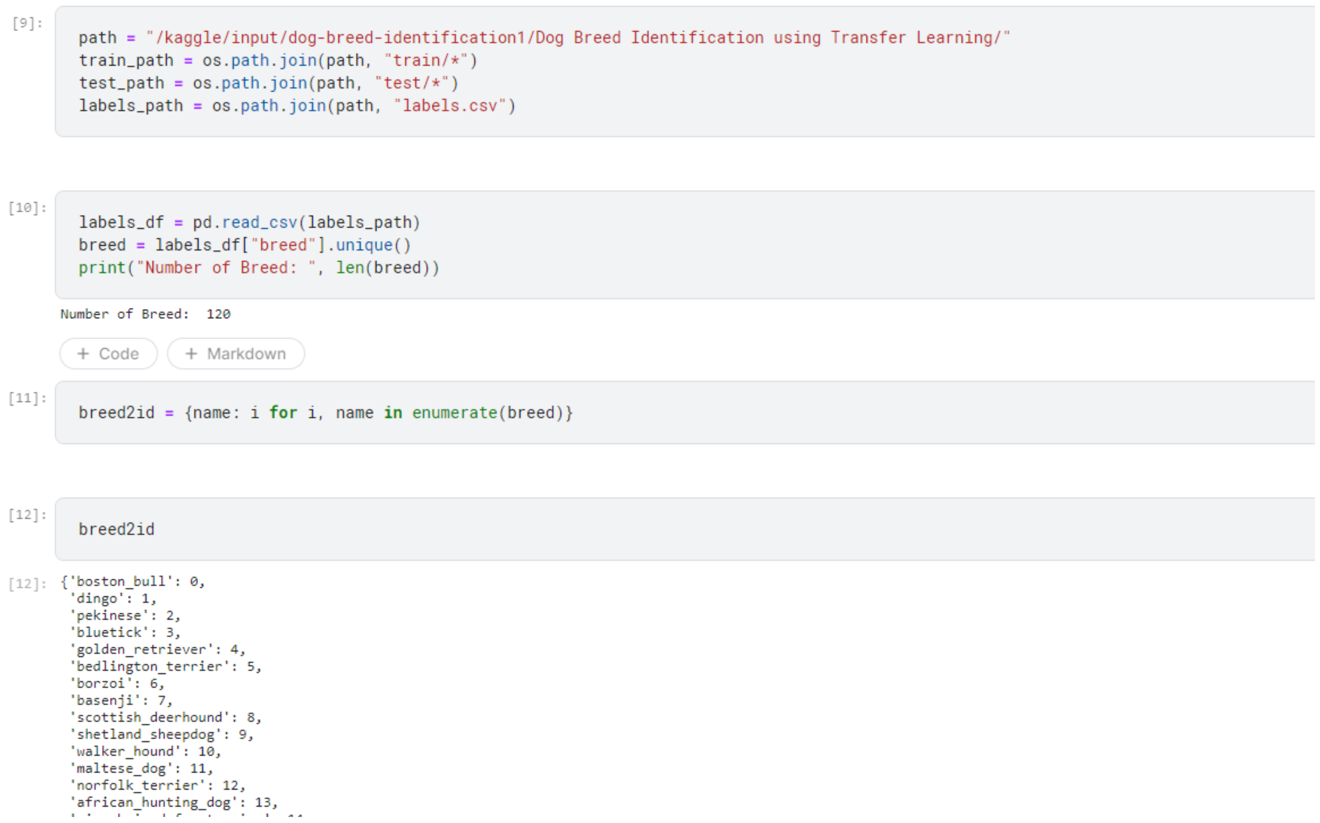
```
[7]:
def tf_parse(x, y):
    x, y = tf.numpy_function(parse_data, [x, y], [tf.float32, tf.int32])
    x.set_shape((224, 224, 3))
    y.set_shape((120))
    return x, y
```



```
[8]:
def tf_dataset(x, y, batch=8):
    dataset = tf.data.Dataset.from_tensor_slices((x, y))
    dataset = dataset.map(tf_parse)
    dataset = dataset.batch(batch)
    dataset = dataset.repeat()
    return dataset
```

parse _datasimply decodes and normalizes images. Tf_dataset it maps tf_parse each and every file or info. Dataset.repeat repeats the dataset indefinitely.

Activity 4: Path setting and getting ids and storing breed names by enumerating as a labelled one



```
[9]:  path = "/kaggle/input/dog-breed-identification1/Dog Breed Identification using Transfer Learning/"
      train_path = os.path.join(path, "train/*")
      test_path = os.path.join(path, "test/*")
      labels_path = os.path.join(path, "labels.csv")
```

```
[10]: labels_df = pd.read_csv(labels_path)
      breed = labels_df["breed"].unique()
      print("Number of Breed: ", len(breed))

      Number of Breed:  120
```

+ Code + Markdown

```
[11]: breed2id = {name: i for i, name in enumerate(breed)}
```

```
[12]: breed2id

[12]: {'boston_bull': 0,
       'dingo': 1,
       'pekinese': 2,
       'bluetick': 3,
       'golden_retriever': 4,
       'bedlington_terrier': 5,
       'borzoi': 6,
       'basenji': 7,
       'scottish_deerhound': 8,
       'shetland_sheepdog': 9,
       'walker_hound': 10,
       'maltese_dog': 11,
       'norfolk_terrier': 12,
       'african_hunting_dog': 13,
```

We labeled breed names by enumerating files. We glob the files of the training folder and stored file id into a list and that labels corresponding to file saved in labels list.

```
[13]:
ids = glob(train_path)
labels = []
```

```
[14]:
for image_id in ids:
    image_id = image_id.split("/")[-1].split(".")[0]
    breed_name = list(labels_df[labels_df.id == image_id]["breed"])[0]
    breed_idx = breed2id[breed_name]
    labels.append(breed_idx)
```

```
[15]:
len(ids)
```

```
[15]: 10222
```

```
[16]:
len(labels)
```

```
[16]: 10222
```

Milestone 3: Model Building

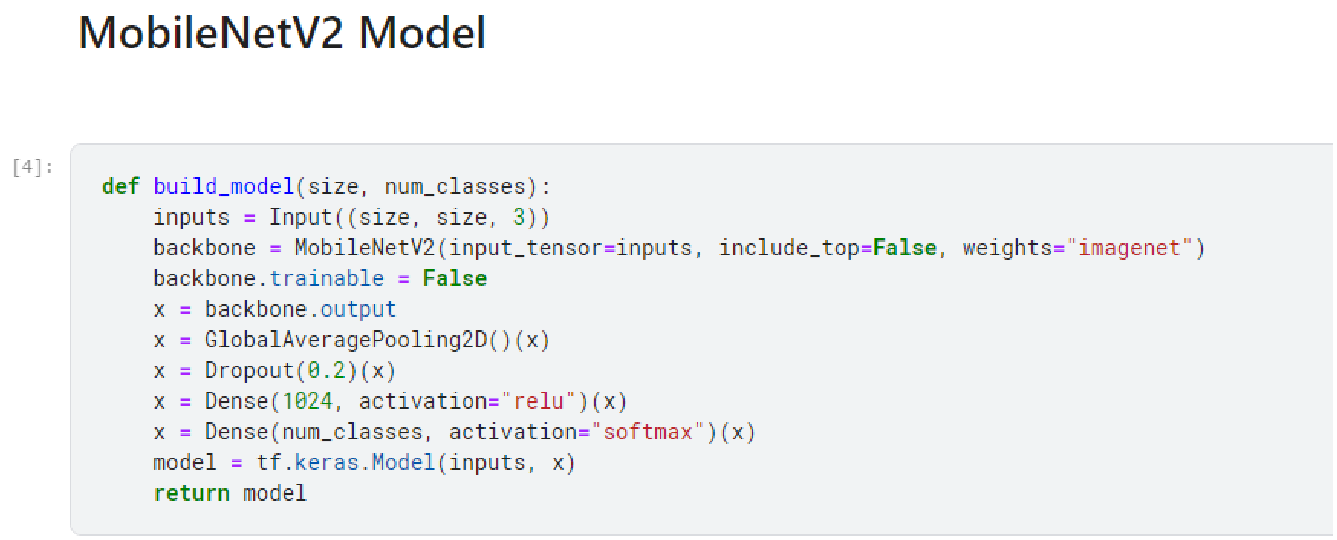
Activity 1: splitting data into train and validation

```
[17]:
train_x, valid_x = train_test_split(ids, test_size=0.2, random_state=42)
train_y, valid_y = train_test_split(labels, test_size=0.2, random_state=42)
```

```
[18]:
## Parameters
size = 224
num_classes = 120
lr = 1e-4
batch = 16
epochs = 10
```

Here we split data into training and validation with files corresponding outputs using random state which generates always ordered files which means it will produce the same splitting datasets. And Batch size and image size declared here.

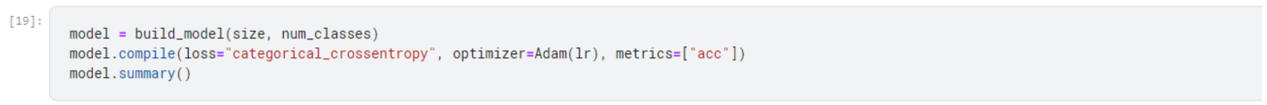
Activity 2: Defining the Imagenetv2 model:



MobileNetV2 Model

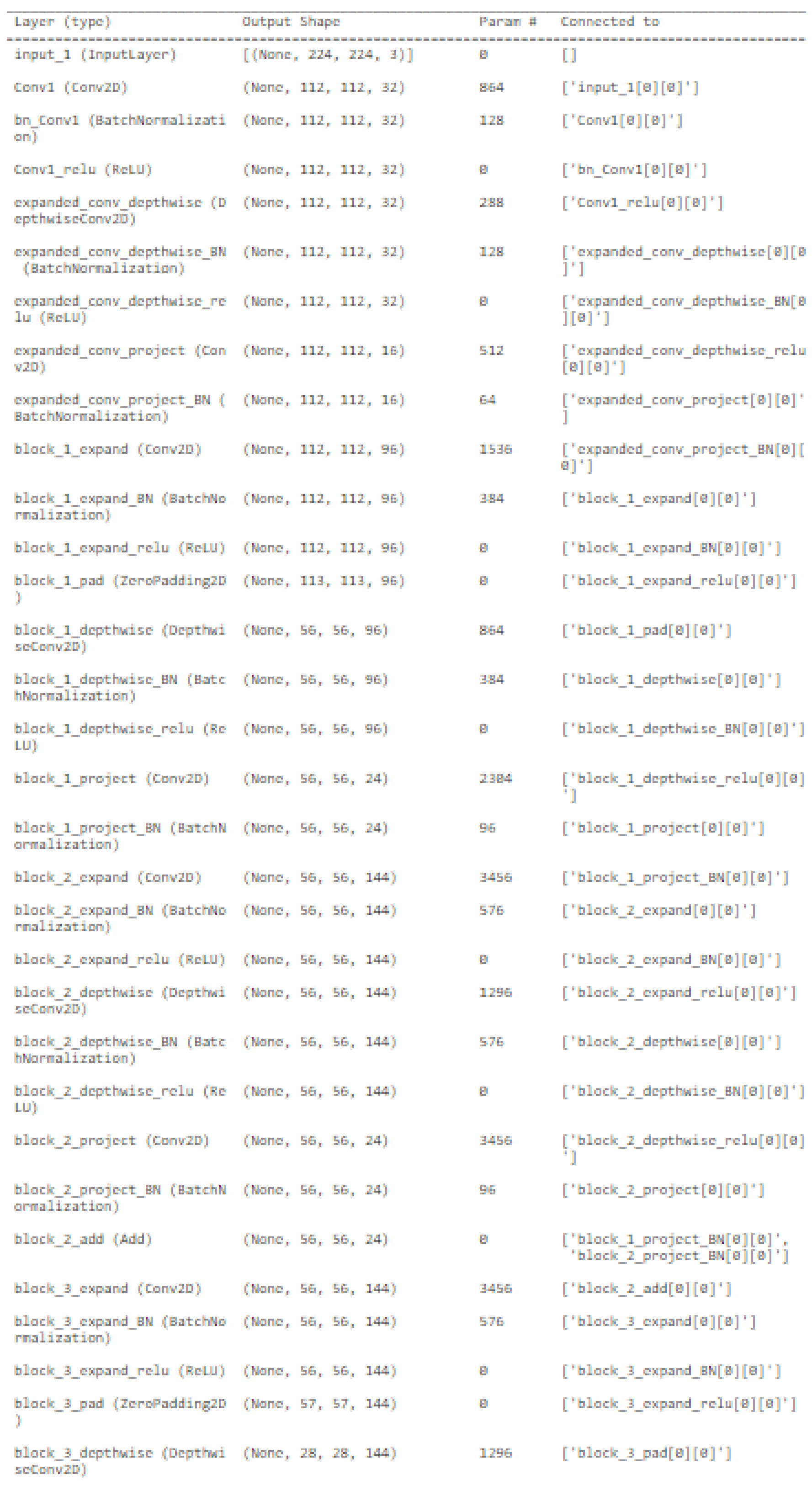
```
[4]:
def build_model(size, num_classes):
    inputs = Input((size, size, 3))
    backbone = MobileNetV2(input_tensor=inputs, include_top=False, weights="imagenet")
    backbone.trainable = False
    x = backbone.output
    x = GlobalAveragePooling2D()(x)
    x = Dropout(0.2)(x)
    x = Dense(1024, activation="relu")(x)
    x = Dense(num_classes, activation="softmax")(x)
    model = tf.keras.Model(inputs, x)
    return model
```

Activity 3: Initializing and configuring The Learning Process



```
[19]:
model = build_model(size, num_classes)
model.compile(loss="categorical_crossentropy", optimizer=Adam(lr), metrics=["acc"])
model.summary()
```

I. Initialize the model.
II. Compile model with selected loss function and learning rate given optimizer.
III. Show the parameters.

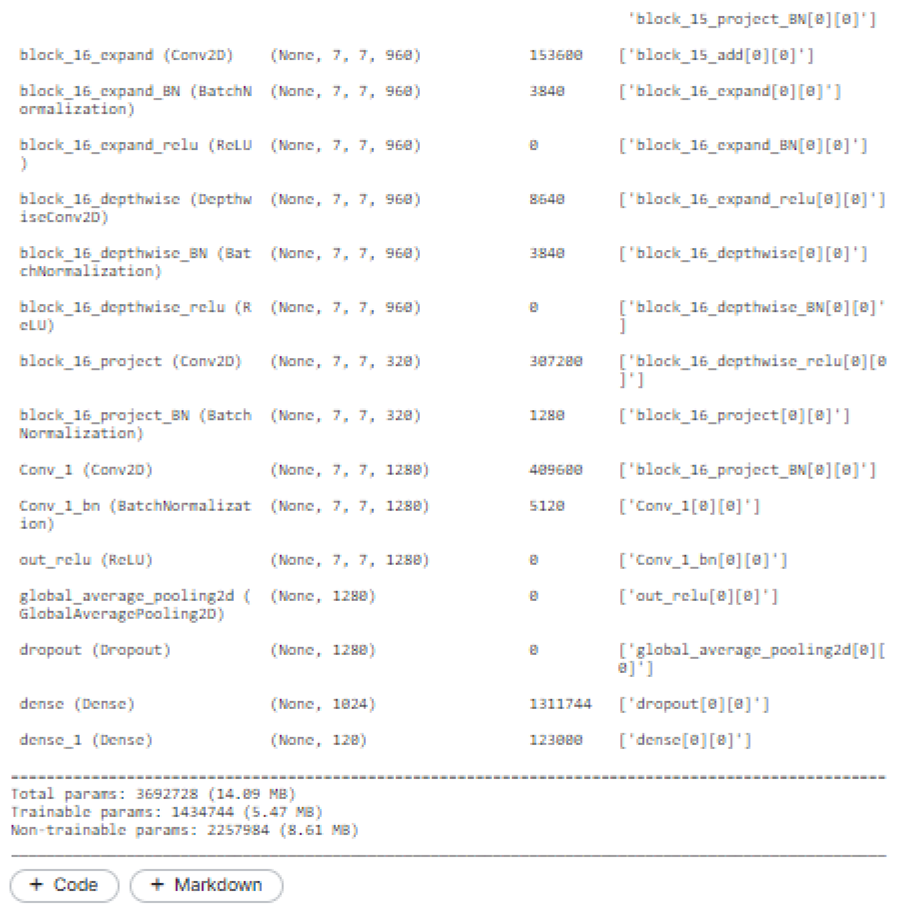


```
__________________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]        0         []

 Conv1 (Conv2D)              (None, 112, 112, 32)         864       ['input_1[0][0]']

 bn_Conv1 (BatchNormalizati  (None, 112, 112, 32)         128       ['Conv1[0][0]']
 on)

 Conv1_relu (ReLU)           (None, 112, 112, 32)         0         ['bn_Conv1[0][0]']

 expanded_conv_depthwise (D  (None, 112, 112, 32)         288       ['Conv1_relu[0][0]']
 epthwiseConv2D)

 expanded_conv_depthwise_BN  (None, 112, 112, 32)         128       ['expanded_conv_depthwise[0][0
  (BatchNormalization)                                              ]']

 expanded_conv_depthwise_re  (None, 112, 112, 32)         0         ['expanded_conv_depthwise_BN[0
 lu (ReLU)                                                          ][0]']

 expanded_conv_project (Con  (None, 112, 112, 16)         512       ['expanded_conv_depthwise_relu
 v2D)                                                               [0][0]']

 expanded_conv_project_BN (  (None, 112, 112, 16)         64        ['expanded_conv_project[0][0]'
 BatchNormalization)                                                ]

 block_1_expand (Conv2D)     (None, 112, 112, 96)         1536      ['expanded_conv_project_BN[0][
                                                                    0]']

 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]']
 seConv2D)

 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_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)

 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]']
 rmalization)

 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_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]']
hNormalization)

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
                                                                    ']

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]']
```

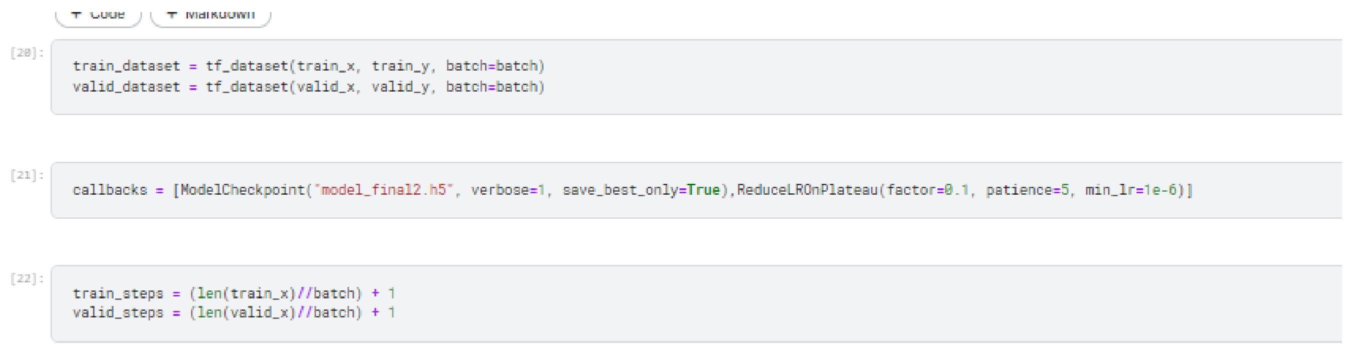
```
block_9_project (Conv2D)    (None, 14, 14, 64)          24576     ['block_9_depthwise_relu[0][0]
                                                                   ']

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]']

block_10_expand (Conv2D)    (None, 14, 14, 384)         24576     ['block_9_add[0][0]']

block_10_expand_BN (BatchN  (None, 14, 14, 384)         1536      ['block_10_expand[0][0]']
ormalization)

block_10_expand_relu (ReLU  (None, 14, 14, 384)         0         ['block_10_expand_BN[0][0]']
)

block_10_depthwise (Depthw  (None, 14, 14, 384)         3456      ['block_10_expand_relu[0][0]']
iseConv2D)

block_10_depthwise_BN (Bat  (None, 14, 14, 384)         1536      ['block_10_depthwise[0][0]']
chNormalization)

block_10_depthwise_relu (R  (None, 14, 14, 384)         0         ['block_10_depthwise_BN[0][0]'
eLU)                                                               ]

block_10_project (Conv2D)   (None, 14, 14, 96)          36864     ['block_10_depthwise_relu[0][0
                                                                   ]']

block_10_project_BN (Batch  (None, 14, 14, 96)          384       ['block_10_project[0][0]']
Normalization)

block_11_expand (Conv2D)    (None, 14, 14, 576)         55296     ['block_10_project_BN[0][0]']

block_11_expand_BN (BatchN  (None, 14, 14, 576)         2304      ['block_11_expand[0][0]']
ormalization)

block_11_expand_relu (ReLU  (None, 14, 14, 576)         0         ['block_11_expand_BN[0][0]']
)

block_11_depthwise (Depthw  (None, 14, 14, 576)         5184      ['block_11_expand_relu[0][0]']
iseConv2D)

block_11_depthwise_BN (Bat  (None, 14, 14, 576)         2304      ['block_11_depthwise[0][0]']
chNormalization)

block_11_depthwise_relu (R  (None, 14, 14, 576)         0         ['block_11_depthwise_BN[0][0]'
eLU)                                                               ]

block_11_project (Conv2D)   (None, 14, 14, 96)          55296     ['block_11_depthwise_relu[0][0
                                                                   ]']

block_11_project_BN (Batch  (None, 14, 14, 96)          384       ['block_11_project[0][0]']
Normalization)

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 (BatchN  (None, 14, 14, 576)         2304      ['block_12_expand[0][0]']
ormalization)

block_12_expand_relu (ReLU  (None, 14, 14, 576)         0         ['block_12_expand_BN[0][0]']
)

block_12_depthwise (Depthw  (None, 14, 14, 576)         5184      ['block_12_expand_relu[0][0]']
iseConv2D)

block_12_depthwise_BN (Bat  (None, 14, 14, 576)         2304      ['block_12_depthwise[0][0]']
chNormalization)

block_12_depthwise_relu (R  (None, 14, 14, 576)         0         ['block_12_depthwise_BN[0][0]'
eLU)                                                               ]

block_12_project (Conv2D)   (None, 14, 14, 96)          55296     ['block_12_depthwise_relu[0][0
                                                                   ]']

block_12_project_BN (Batch  (None, 14, 14, 96)          384       ['block_12_project[0][0]']
Normalization)

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 (BatchN  (None, 14, 14, 576)         2304      ['block_13_expand[0][0]']
ormalization)
```

```
Normalization)

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 (BatchN  (None, 14, 14, 576)          2304      ['block_13_expand[0][0]']
ormalization)

block_13_expand_relu (ReLU  (None, 14, 14, 576)          0         ['block_13_expand_BN[0][0]']
)

block_13_pad (ZeroPadding2  (None, 15, 15, 576)          0         ['block_13_expand_relu[0][0]']
D)

block_13_depthwise (Depthw  (None, 7, 7, 576)            5184      ['block_13_pad[0][0]']
iseConv2D)

block_13_depthwise_BN (Bat  (None, 7, 7, 576)            2304      ['block_13_depthwise[0][0]']
chNormalization)

block_13_depthwise_relu (R  (None, 7, 7, 576)            0         ['block_13_depthwise_BN[0][0]'
eLU)                                                               ]

block_13_project (Conv2D)   (None, 7, 7, 160)            92160     ['block_13_depthwise_relu[0][0
                                                                   ]']

block_13_project_BN (Batch  (None, 7, 7, 160)            640       ['block_13_project[0][0]']
Normalization)

block_14_expand (Conv2D)    (None, 7, 7, 960)            153600    ['block_13_project_BN[0][0]']

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]']
)

block_15_depthwise (Depthw  (None, 7, 7, 960)            8640      ['block_15_expand_relu[0][0]']
iseConv2D)

block_15_depthwise_BN (Bat  (None, 7, 7, 960)            3840      ['block_15_depthwise[0][0]']
chNormalization)

block_15_depthwise_relu (R  (None, 7, 7, 960)            0         ['block_15_depthwise_BN[0][0]'
eLU)                                                               ]

block_15_project (Conv2D)   (None, 7, 7, 160)            153600    ['block_15_depthwise_relu[0][0
                                                                   ]']

block_15_project_BN (Batch  (None, 7, 7, 160)            640       ['block_15_project[0][0]']
Normalization)

block_15_add (Add)          (None, 7, 7, 160)            0         ['block_14_add[0][0]',
                                                                    'block_15_project_BN[0][0]']

block_16_expand (Conv2D)    (None, 7, 7, 960)            153600    ['block_15_add[0][0]']

block_16_expand_BN (BatchN  (None, 7, 7, 960)            3840      ['block_16_expand[0][0]']
ormalization)

block_16_expand_relu (ReLU  (None, 7, 7, 960)            0         ['block_16_expand_BN[0][0]']
```



```
                                                                  'block_15_project_BN[0][0]']
 block_16_expand (Conv2D)      (None, 7, 7, 960)          153600    ['block_15_add[0][0]']

 block_16_expand_BN (BatchN    (None, 7, 7, 960)          3840      ['block_16_expand[0][0]']
 ormalization)

 block_16_expand_relu (ReLU    (None, 7, 7, 960)          0         ['block_16_expand_BN[0][0]']
 )

 block_16_depthwise (Depthw    (None, 7, 7, 960)          8640      ['block_16_expand_relu[0][0]']
 iseConv2D)

 block_16_depthwise_BN (Bat    (None, 7, 7, 960)          3840      ['block_16_depthwise[0][0]']
 chNormalization)

 block_16_depthwise_relu (R    (None, 7, 7, 960)          0         ['block_16_depthwise_BN[0][0]'
 eLU)                                                               ]

 block_16_project (Conv2D)     (None, 7, 7, 320)          307200    ['block_16_depthwise_relu[0][0
                                                                    ]']

 block_16_project_BN (Batch    (None, 7, 7, 320)          1280      ['block_16_project[0][0]']
 Normalization)

 Conv_1 (Conv2D)               (None, 7, 7, 1280)         409600    ['block_16_project_BN[0][0]']

 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)

 dropout (Dropout)             (None, 1280)               0         ['global_average_pooling2d[0][
                                                                    0]']

 dense (Dense)                 (None, 1024)               1311744   ['dropout[0][0]']

 dense_1 (Dense)               (None, 120)                123000    ['dense[0][0]']

==================================================================================================
Total params: 3692728 (14.09 MB)
Trainable params: 1434744 (5.47 MB)
Non-trainable params: 2257984 (8.61 MB)
__________________________________________________________________________________________________
```

Activity 4: Train The model



```
[20]:
train_dataset = tf_dataset(train_x, train_y, batch=batch)
valid_dataset = tf_dataset(valid_x, valid_y, batch=batch)
```

```
[21]:
callbacks = [ModelCheckpoint("model_final2.h5", verbose=1, save_best_only=True),ReduceLROnPlateau(factor=0.1, patience=5, min_lr=1e-6)]
```

```
[22]:
train_steps = (len(train_x)//batch) + 1
valid_steps = (len(valid_x)//batch) + 1
```

Merging the splitted data into a dataset as files with labeling. Callbacks are declared to check for the best learning rate.

New_learning rate=curr_lr*factor It checks for the best learning rate to get a good fit model.

Patience=5 It stops the epochs after there is no loss in 5 continuous epochs.

verbose=1 It shows the progress of the training.

```
[23]:
M=model.fit(train_dataset,
        steps_per_epoch=train_steps,
        validation_steps=valid_steps,
        validation_data=valid_dataset,
        epochs=epochs,
        callbacks=callbacks)
```

```
Epoch 1/10
512/512 [==============================] - ETA: 0s - loss: 3.0769 - acc: 0.3188
Epoch 1: val_loss improved from inf to 1.78527, saving model to model_final2.h5
/opt/conda/lib/python3.10/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
512/512 [==============================] - 140s 246ms/step - loss: 3.0769 - acc: 0.3188 - val_loss: 1.7853 - val_acc: 0.5364 - lr: 1.0000e-04
Epoch 2/10
511/512 [============================>.] - ETA: 0s - loss: 1.4500 - acc: 0.6036
Epoch 2: val_loss improved from 1.78527 to 1.42178, saving model to model_final2.h5
512/512 [==============================] - 59s 115ms/step - loss: 1.4498 - acc: 0.6036 - val_loss: 1.4218 - val_acc: 0.5951 - lr: 1.0000e-04
Epoch 3/10
511/512 [============================>.] - ETA: 0s - loss: 1.0990 - acc: 0.6860
Epoch 3: val_loss improved from 1.42178 to 1.33423, saving model to model_final2.h5
512/512 [==============================] - 61s 119ms/step - loss: 1.0989 - acc: 0.6861 - val_loss: 1.3342 - val_acc: 0.6161 - lr: 1.0000e-04
Epoch 4/10
511/512 [============================>.] - ETA: 0s - loss: 0.8934 - acc: 0.7435
Epoch 4: val_loss improved from 1.33423 to 1.28071, saving model to model_final2.h5
512/512 [==============================] - 58s 113ms/step - loss: 0.8933 - acc: 0.7435 - val_loss: 1.2807 - val_acc: 0.6284 - lr: 1.0000e-04
Epoch 5/10
511/512 [============================>.] - ETA: 0s - loss: 0.7540 - acc: 0.7883
Epoch 5: val_loss improved from 1.28071 to 1.25714, saving model to model_final2.h5
512/512 [==============================] - 58s 114ms/step - loss: 0.7539 - acc: 0.7883 - val_loss: 1.2571 - val_acc: 0.6381 - lr: 1.0000e-04
Epoch 6/10
511/512 [============================>.] - ETA: 0s - loss: 0.6400 - acc: 0.8256
Epoch 6: val_loss did not improve from 1.25714
512/512 [==============================] - 57s 112ms/step - loss: 0.6400 - acc: 0.8256 - val_loss: 1.2611 - val_acc: 0.6347 - lr: 1.0000e-04
Epoch 7/10
511/512 [============================>.] - ETA: 0s - loss: 0.5584 - acc: 0.8467
Epoch 7: val_loss did not improve from 1.25714
512/512 [==============================] - 54s 107ms/step - loss: 0.5583 - acc: 0.8468 - val_loss: 1.2773 - val_acc: 0.6279 - lr: 1.0000e-04
Epoch 8/10
511/512 [============================>.] - ETA: 0s - loss: 0.4802 - acc: 0.8704
Epoch 8: val_loss did not improve from 1.25714
512/512 [==============================] - 56s 110ms/step - loss: 0.4801 - acc: 0.8704 - val_loss: 1.2621 - val_acc: 0.6396 - lr: 1.0000e-04
Epoch 9/10
511/512 [============================>.] - ETA: 0s - loss: 0.4168 - acc: 0.8965
Epoch 9: val_loss did not improve from 1.25714
512/512 [==============================] - 41s 79ms/step - loss: 0.4168 - acc: 0.8965 - val_loss: 1.2808 - val_acc: 0.6455 - lr: 1.0000e-04
Epoch 10/10
511/512 [============================>.] - ETA: 0s - loss: 0.3665 - acc: 0.9068
Epoch 10: val_loss did not improve from 1.25714
512/512 [==============================] - 48s 94ms/step - loss: 0.3665 - acc: 0.9068 - val_loss: 1.2886 - val_acc: 0.6450 - lr: 1.0000e-04
```

+ Code   + Markdown

Activity 7: Save the Model

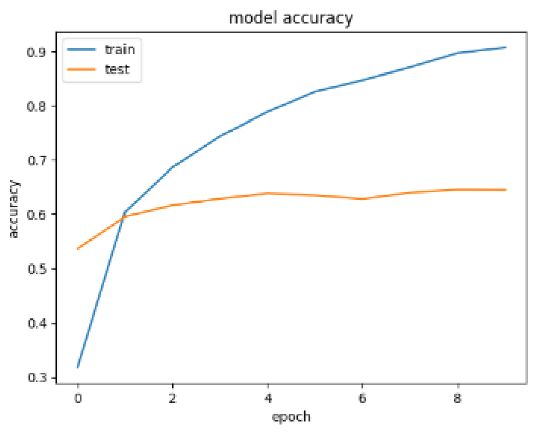
The model is saved with .h5 extension as follows.
An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

```
[24]:
model.save("/kaggle/working/model_final2.h5")
```

Graphs:

```
[26]:
import matplotlib.pyplot as plt
```

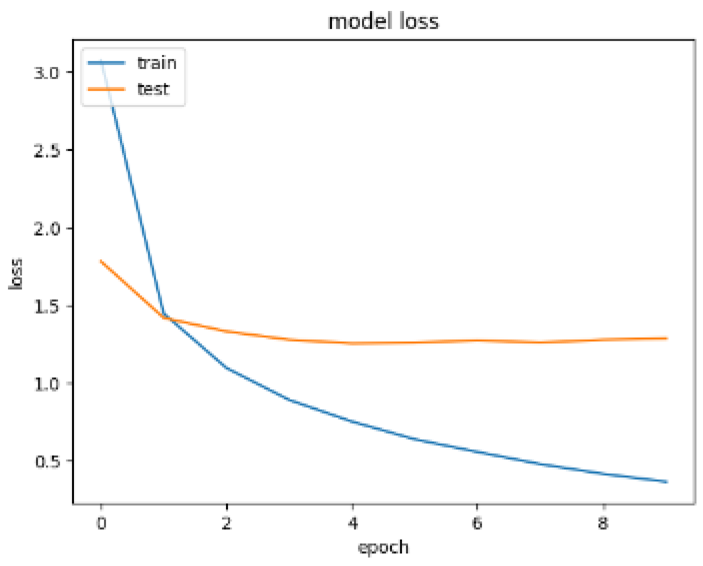
```
[27]:
plt.plot(M.history['acc'])
plt.plot(M.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



+ Code   + Markdown

epoch

```
[28]:
plt.plot(M.history['loss'])
plt.plot(M.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



Activity 8: Test The model

Evaluation is a process during the development of the model to check whether the model is the best fit for the given problem and corresponding data.

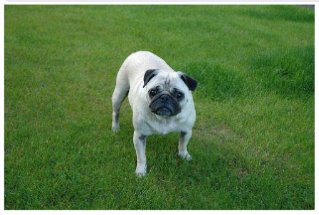
Output-1:

```
[29]:
I="/kaggle/input/dog-breed-identification1/Dog Breed Identification using Transfer Learning/test/0a8d8dda0e354c0571c8d47600ab39a3.jpg"
```

```
[30]:
id2breed = {i: name for i, name in enumerate(breed)}
```

```
[31]:
import PIL
PIL.Image.open(I)
```

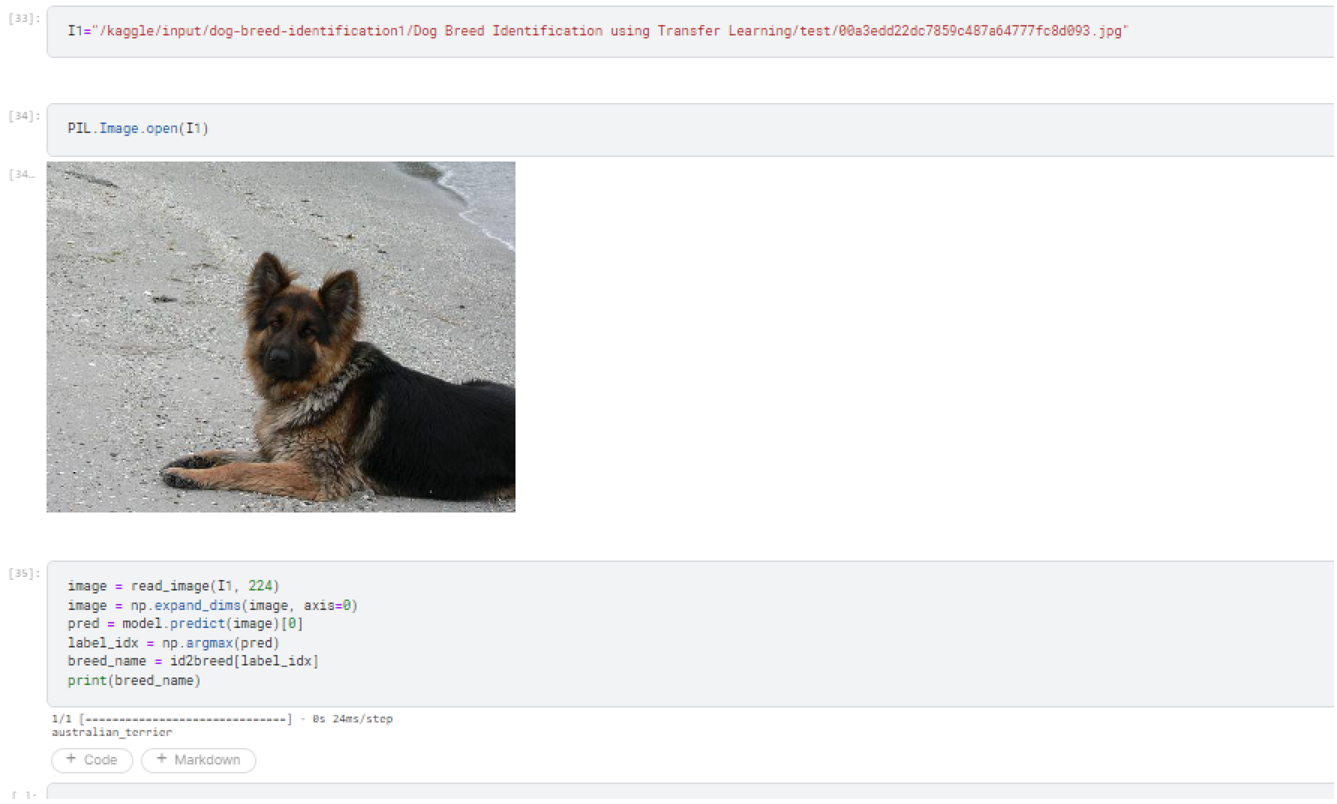
[31…



```
[32]:
image = read_image(I, 224)
image = np.expand_dims(image, axis=0)
pred = model.predict(image)[0]
label_idx = np.argmax(pred)
breed_name = id2breed[label_idx]
print(breed_name)
```

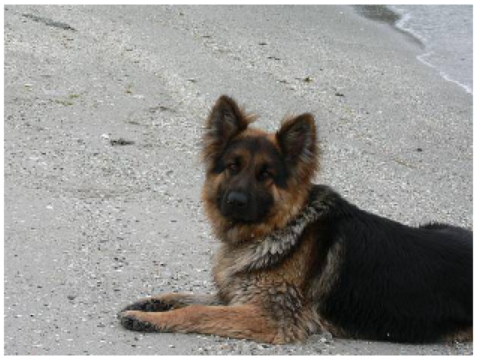
```
1/1 [==============================] - 1s 849ms/step
pug
```

<u>Output-2:</u>



```
[33]: I1="/kaggle/input/dog-breed-identification1/Dog Breed Identification using Transfer Learning/test/00a3edd22dc7859c487a64777fc8d093.jpg"
```

```
[34]: PIL.Image.open(I1)
```



```
[35]: image = read_image(I1, 224)
      image = np.expand_dims(image, axis=0)
      pred = model.predict(image)[0]
      label_idx = np.argmax(pred)
      breed_name = id2breed[label_idx]
      print(breed_name)

1/1 [==============================] - 0s 24ms/step
australian_terrier
```

Taking an image as input and checking the results
By using the model we are predicting the output for the given input image.
The predicted class index name will be printed here.

Milestone 4: Application Building

Now that we have trained our model, let us build our flask application which will be running in our local browser with a user interface.

In the flask application, the input parameters are taken from the HTML page These factors are then given to the model to know to predict the Breed and showcased on the HTML page to notify the user. Whenever the user interacts with the UI and selects the "Predict" button, where the user chooses the image and predicts the output.

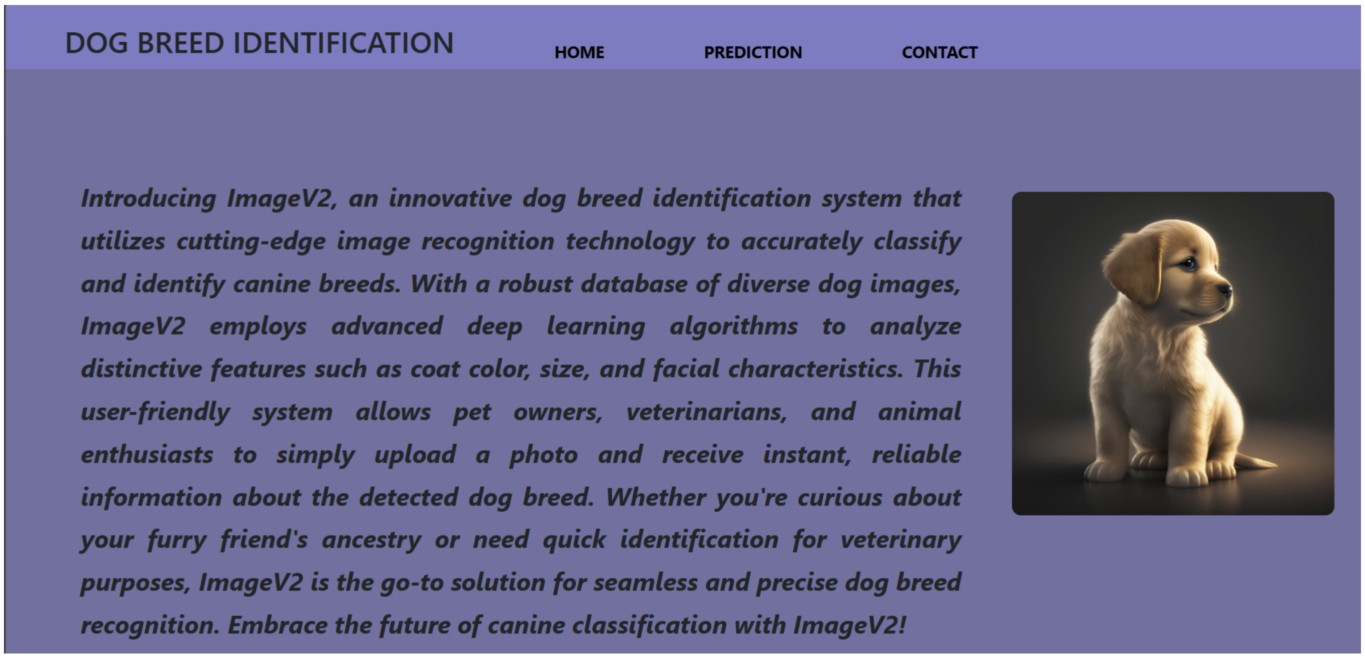
Activity 1 : Create HTML Pages

- o We use HTML to create the front end part of the web page.
- o Here, we have created 1 HTML page- index.html.
- o index.html displays an introduction about the project
- o We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.

○ **Link :**<u>CSS</u> , <u>JS</u>

index.html looks like this

Home

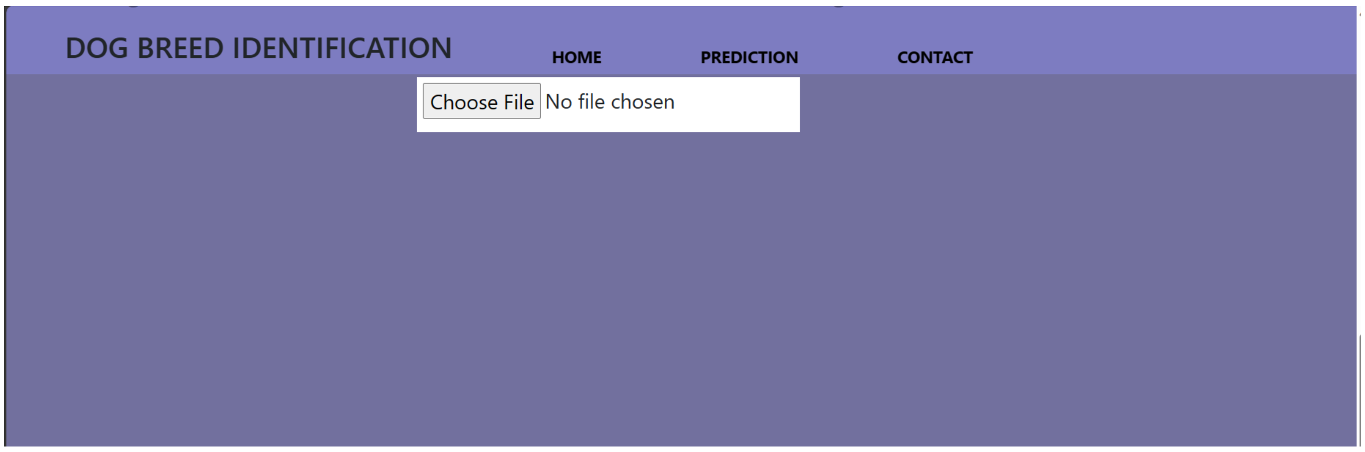


DOG BREED IDENTIFICATION

HOME PREDICTION CONTACT

Introducing ImageV2, an innovative dog breed identification system that utilizes cutting-edge image recognition technology to accurately classify and identify canine breeds. With a robust database of diverse dog images, ImageV2 employs advanced deep learning algorithms to analyze distinctive features such as coat color, size, and facial characteristics. This user-friendly system allows pet owners, veterinarians, and animal enthusiasts to simply upload a photo and receive instant, reliable information about the detected dog breed. Whether you're curious about your furry friend's ancestry or need quick identification for veterinary purposes, ImageV2 is the go-to solution for seamless and precise dog breed recognition. Embrace the future of canine classification with ImageV2!

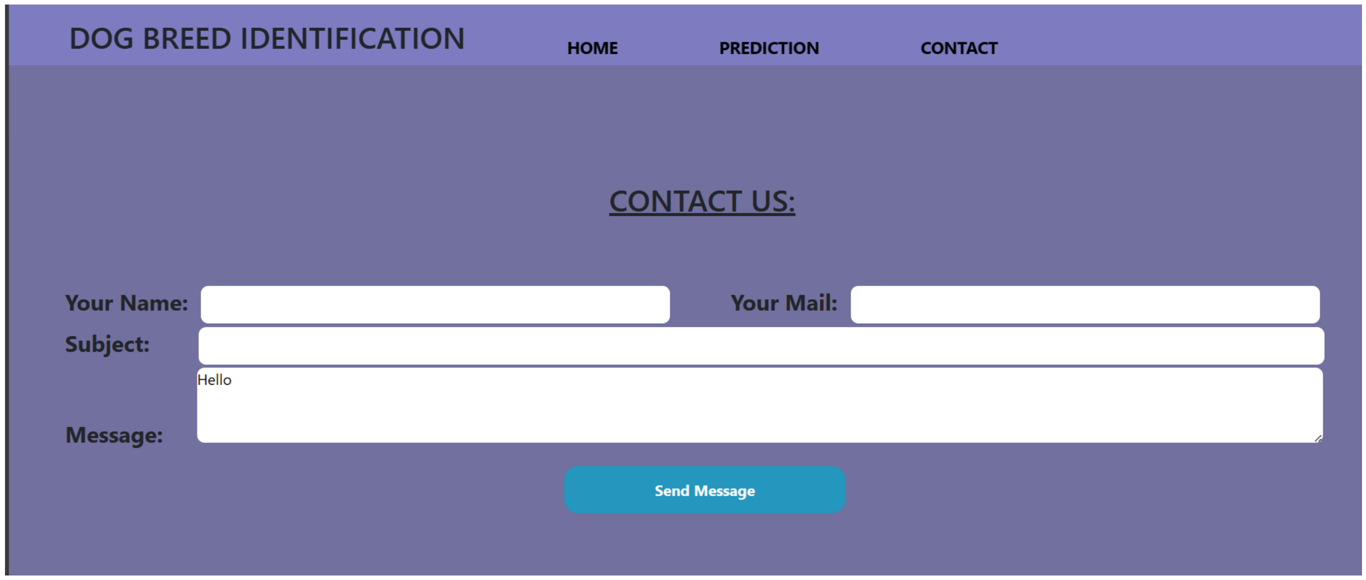
Prediction:



DOG BREED IDENTIFICATION

HOME PREDICTION CONTACT

Contact Us:-



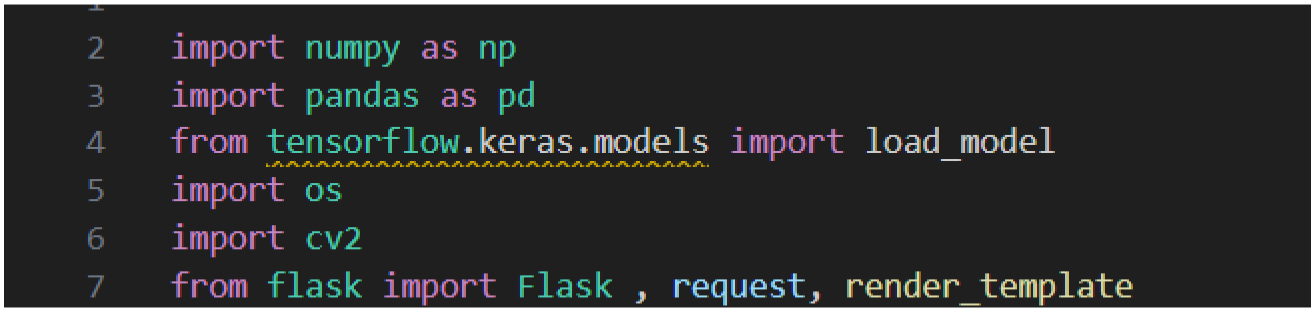
Hello

Activity 2: Build python code

Task 1: Importing Libraries

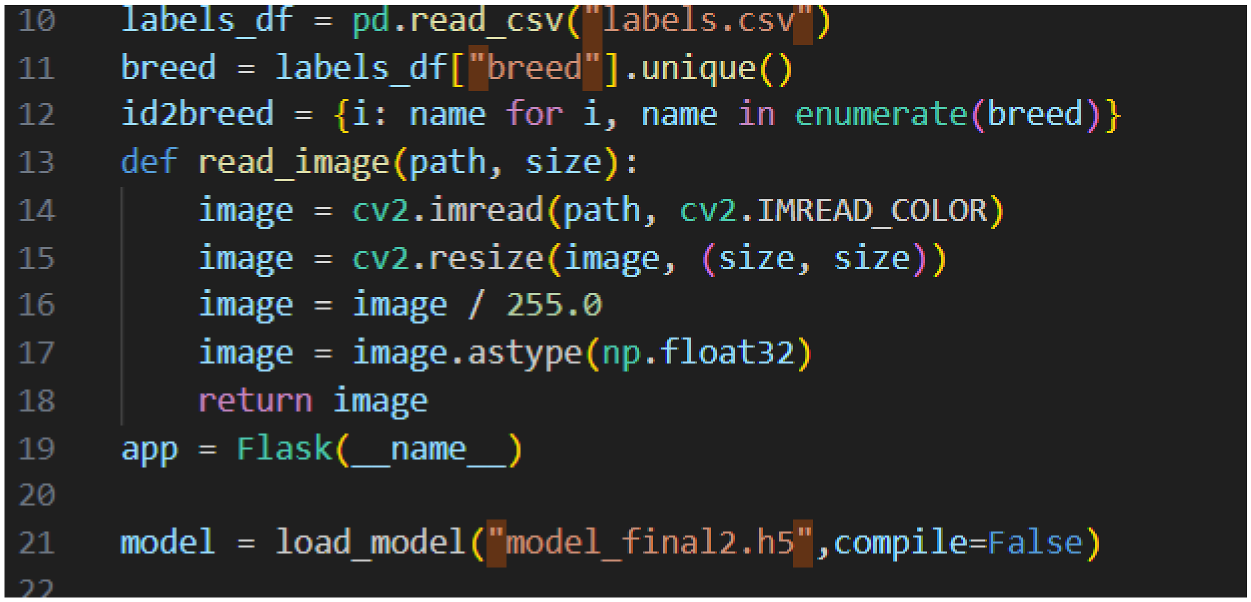
The first step is usually importing the libraries that will be needed in the program.

Importing the flask module in the project is mandatory. An object of the Flask class is our WSGI application. Flask constructor takes the name of the current module (_name_) as argument Pickle library to load the model file.



```python
import numpy as np
import pandas as pd
from tensorflow.keras.models import load_model
import os
import cv2
from flask import Flask , request, render_template
```

Task 2: Creating our flask application and loading our model by using load_model method

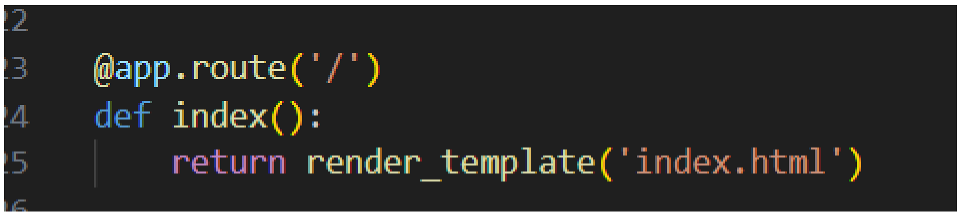


```
labels_df = pd.read_csv("labels.csv")
breed = labels_df["breed"].unique()
id2breed = {i: name for i, name in enumerate(breed)}
def read_image(path, size):
    image = cv2.imread(path, cv2.IMREAD_COLOR)
    image = cv2.resize(image, (size, size))
    image = image / 255.0
    image = image.astype(np.float32)
    return image
app = Flask(__name__)

model = load_model("model_final2.h5",compile=False)
```

Task 3: Routing to the html Page

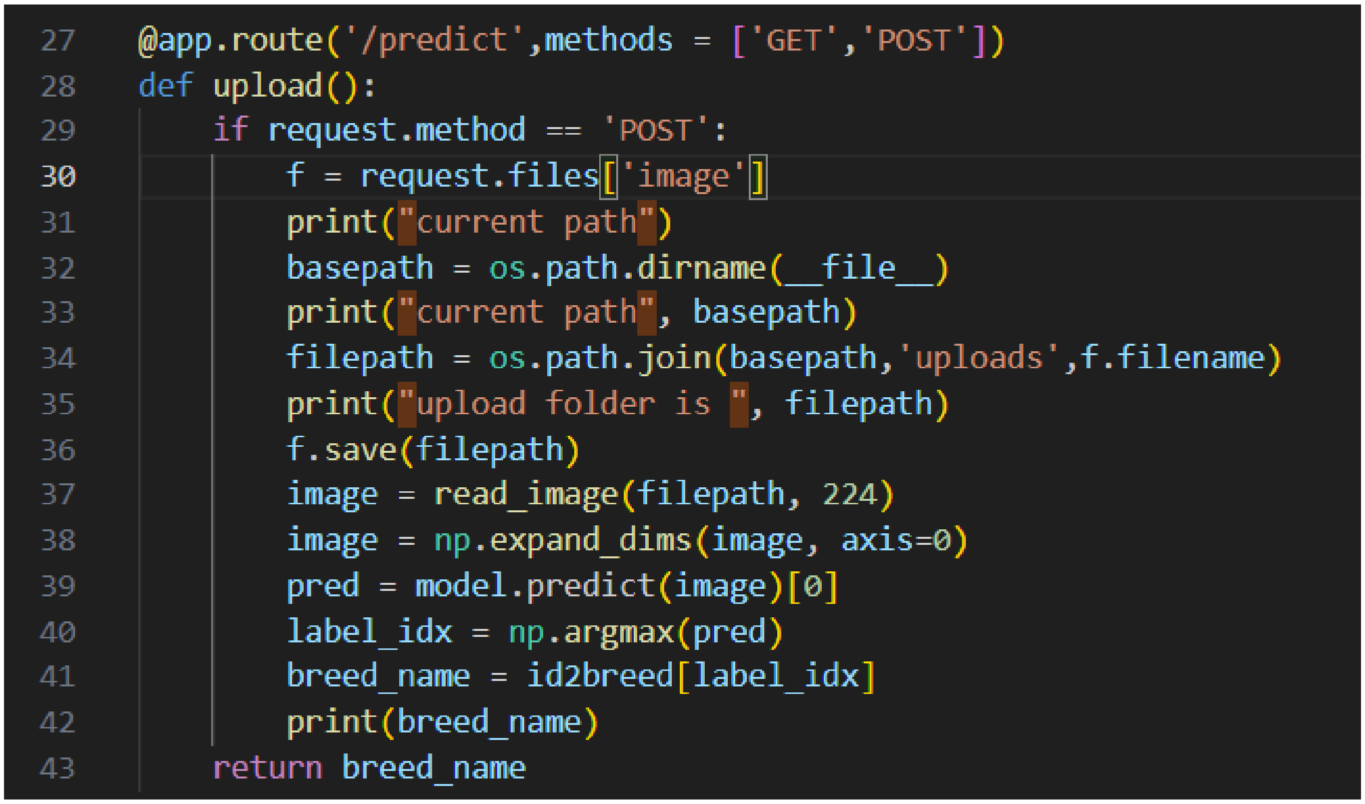
Here, the declared constructor is used to route to the HTML page created earlier.

In the above example, ‘/’ URL is bound with index.html function. Hence, when the home page of a web server is opened in the browser, the html page will be rendered. Whenever you browse an image from the html page this photo can be accessed through POST or GET Method.



```
@app.route('/')
def index():
    return render_template('index.html')
```

Showcasing prediction on UI:

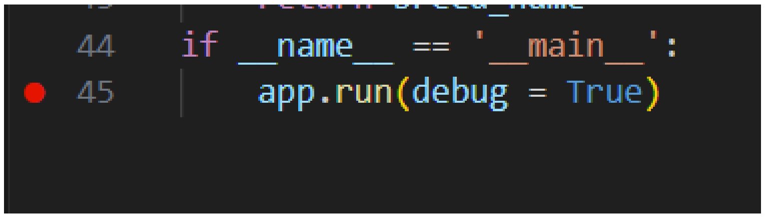


```
@app.route('/predict',methods = ['GET','POST'])
def upload():
    if request.method == 'POST':
        f = request.files['image']
        print("current path")
        basepath = os.path.dirname(__file__)
        print("current path", basepath)
        filepath = os.path.join(basepath,'uploads',f.filename)
        print("upload folder is ", filepath)
        f.save(filepath)
        image = read_image(filepath, 224)
        image = np.expand_dims(image, axis=0)
        pred = model.predict(image)[0]
        label_idx = np.argmax(pred)
        breed_name = id2breed[label_idx]
        print(breed_name)
    return breed_name
```

Here we are defining a function which requests the browsed file from the html page using the post method. The requested picture file is then saved to the uploads folder in this same directory using OS library. Using the load image class from Keras library we are retrieving the saved picture from the path declared. We are applying some image processing techniques and then sending that preprocessed image to the model for predicting the class. This returns the numerical value of a class (like 0 to 119.) which lies in the 0th index of the variable preds. This numerical value is passed to the index variable declared. This returns the name of the class. This name is rendered to the prediction variable used in the html page.

Finally, Run the application

This is used to run the application in a local host.



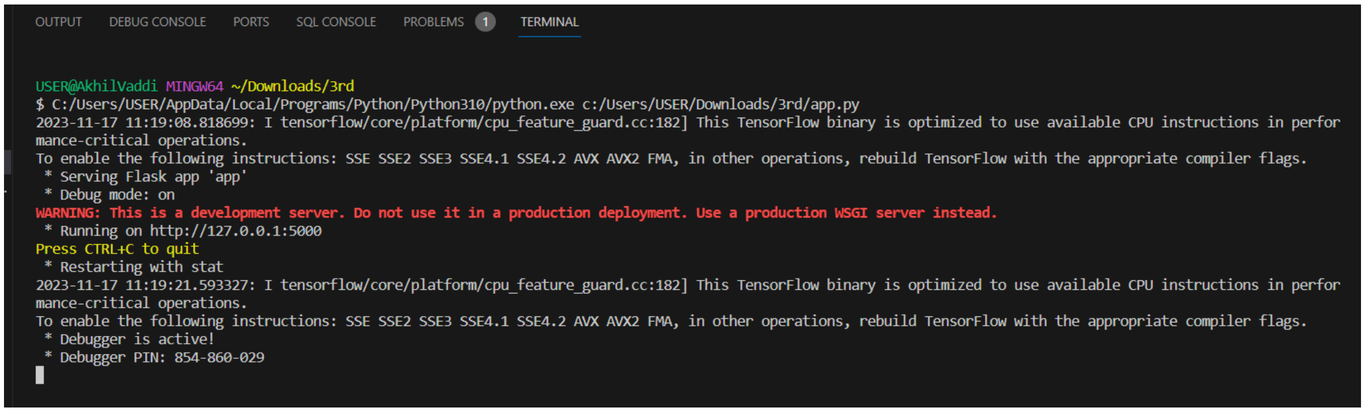
```
if __name__ == '__main__':
    app.run(debug = True)
```

Activity 3:Run the application

- Open vs code or any terminal.
- Navigate to the folder where your app.py resides.

● Now type “python app.py” command.
● It will show the local host where your app is running on **http://127.0.0.1.5000/** ● Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
● Upload the image file, click on the predict button and see the result/prediction on the web page.

Then it will run on localhost: 5000

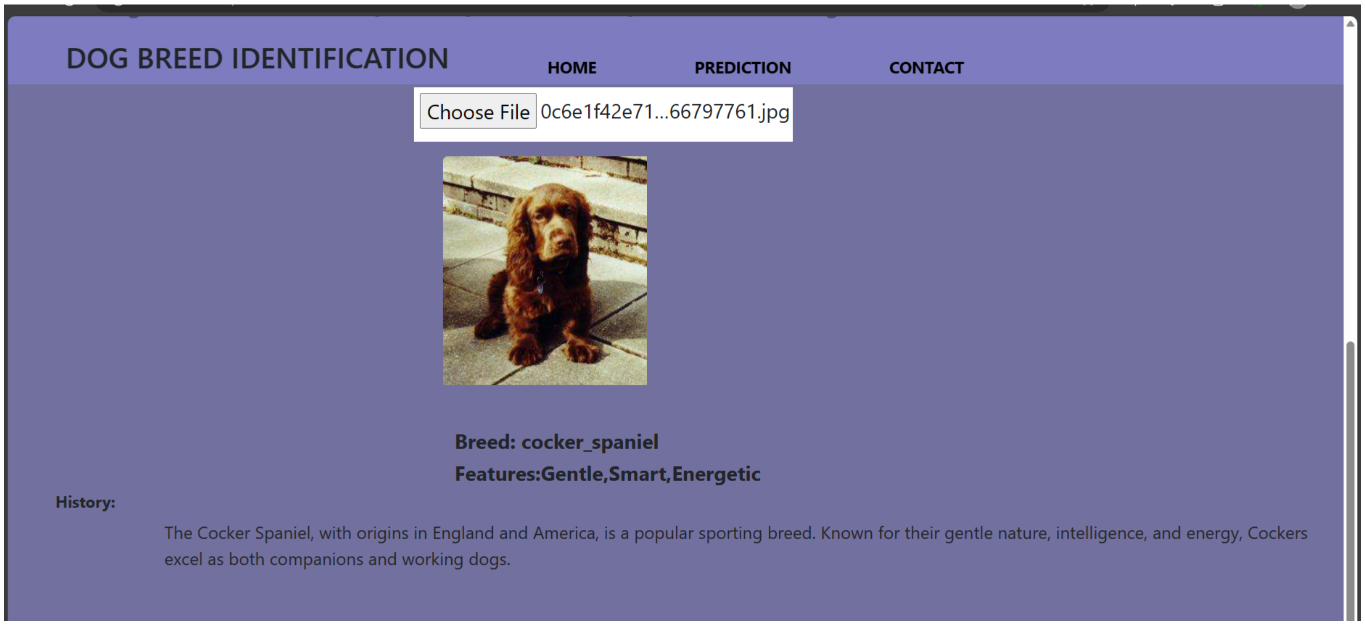


```
USER@AkhilVaddi MINGW64 ~/Downloads/3rd
$ C:/Users/USER/AppData/Local/Programs/Python/Python310/python.exe c:/Users/USER/Downloads/3rd/app.py
2023-11-17 11:19:08.818699: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in perfor
mance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
2023-11-17 11:19:21.593327: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in perfor
mance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Debugger is active!
 * Debugger PIN: 854-860-029
```

Navigate to the localhost (http://127.0.0.1:5000/)where you can view your web page.

FINAL OUTPUT:

Output 1:



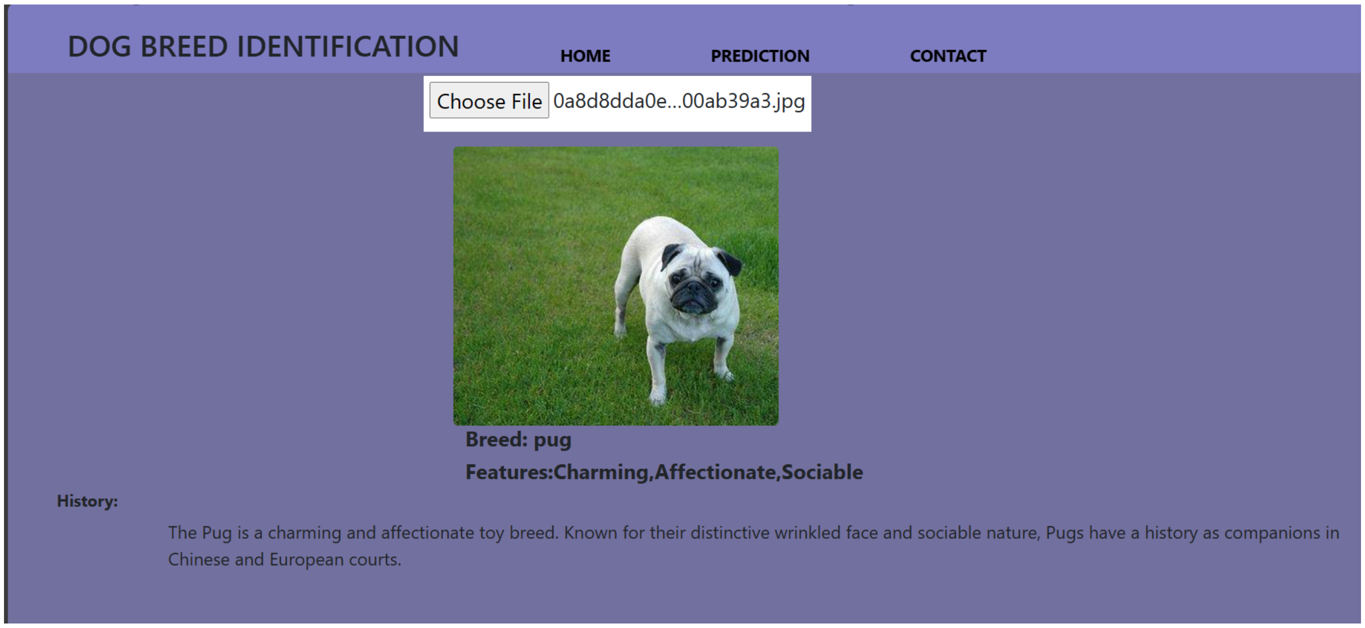
Breed: cocker_spaniel

Features:Gentle,Smart,Energetic

History:

The Cocker Spaniel, with origins in England and America, is a popular sporting breed. Known for their gentle nature, intelligence, and energy, Cockers excel as both companions and working dogs.

Output 2:



Breed: pug

Features:Charming,Affectionate,Sociable

History:

The Pug is a charming and affectionate toy breed. Known for their distinctive wrinkled face and sociable nature, Pugs have a history as companions in Chinese and European courts.