FRUIT AND VEGETABLE

RECOGNITION

(MINI PROJECT)

Submitted by:

Mr. Harshit Agarwal

Roll. No.:1018493

SECTION : B

Course: B.Tech (CSE)

Submitted To

Mr. Amit Gupta

Table Of Content

ACKNOWLEDGEMENT

I would like to express my gratitude to my parents for their continuous support and encouragement and providing me with the opportunity to reach this far in my studies.

I'd like to express my deepest thanks to my teachers for their patience, support and encouragement throughout the completion of this project and having faith in me.

At last I'd also like to acknowledge the help of all other people who directly or indirectly helped me during this work.

Harshit Agarwal

(Roll No. 1018493)

ABSTRACT

The recognition and classification of fruits play a pivotal role in various fields, including agriculture, food industry automation, and dietary analysis. This project presents a comprehensive approach to fruit recognition utilizing state-of-the-art deep learning techniques. The primary goal of this research is to develop an accurate and efficient system capable of identifying and categorizing a wide range of fruits from images, contributing to improved fruit sorting, quality control, and dietary assessment.

This project contributes to the field of fruit recognition by presenting a deep learning-based approach that has the potential to revolutionize fruit processing and dietary analysis. The developed model showcases promising results and opens up avenues for future research, such as extending the system to recognize fruit ripeness or implementing it in mobile applications for on-the-go fruit identification.

LIST OF FIGURE

1. About Project

1.1 Introduction

In a world where technological advancements continue to reshape various industries, the intersection of computer vision and agriculture has garnered substantial attention. One compelling application within this domain is the recognition and classification of fruits and vegetables through the use of computer vision and machine learning techniques. The ability to accurately and efficiently identify and categorize different types of fruits and vegetables has far-reaching implications, from enhancing agricultural practices and automating food processing to aiding dietary assessment and promoting healthier eating habits.

Historically, the task of recognizing and sorting fruits and vegetables has been predominantly manual, relying on human labor and expertise. However, this manual approach is labor-intensive, time-consuming, and prone to errors. Moreover, as the demand for fresh produce continues to rise, the need for efficient and accurate methods for fruit and vegetable recognition becomes increasingly imperative.

Advancements in computer vision, deep learning, and image processing have paved the way for the development of robust and scalable fruit and vegetable recognition systems. These systems employ algorithms that can analyze images and extract meaningful features, enabling the identification of fruits and vegetables based on their visual characteristics such as shape, color, texture, and size. Machine learning models, particularly convolutional neural networks (CNNs), have demonstrated remarkable success in this regard, offering the potential to automate and optimize the recognition process.

1.2 Objective

Objective: The motive of the project is to recognize different Fruits And Vegetables. In this model, classification machine learning algorithms are trained using a set of image data for different hand Fruits and Vegetables.

1.3 Motivation

The motivation for fruit and vegetable recognition stems from a combination of societal, economic, and technological factors. Here are some key motivations for pursuing research and development in this field:

- **Agricultural Efficiency And Productivity:** Automated recognition of fruits and vegetables can significantly enhance agricultural practices. By accurately identifying and monitoring crops, farmers can optimize irrigation, fertilization, and pest control, leading to increased crop yields and reduced resource wastage.
- **Food Processing Industry:** The food processing industry relies heavily on sorting and grading of fruits and vegetables. Automated recognition systems can improve the efficiency and precision of this process, ensuring consistent product quality and reducing production costs.
- **Time And Labor Saving:** Manual sorting and classification of fruits and vegetables are labor-intensive and time-consuming. Automation can free up human labor for more skilled and value-added tasks while also speeding up the sorting process.

1.4 Literature Survey

AUTHOR	PURPOSE	DATASET	TECHNIQUE USED	ACCURACY
Frida Femling, Adam Olsson, Fernando Alonso-Fernandez	Fruit and Vegetable Identification Using Machine Learning	400 images per class has been extracted from ImageNet	CNN	97% Accuracy
Nur-E-Aznin Mimma, Sumon Ahmed,	Fruits Classification and Detection Application	public dataset named FIDS-30	YOLOv3 deep convolutional neural network	96% Accuracy

Tahsin Rahman, and Riasat Khan	Using Deep Learning			
Swapnil Srivastava, Tripti Singh, Sakshi Sharma, Anil Verma	A Fruit Recognition System based on Modern Deep Learning Technique	fruits 360 dataset, from Kaggle containing 77917 different types of fruits pictures belonging to 103 categories.	CNN Algorithm	93.67 % Accuracy
Feng Xiao , Haibin Wang , Yueqin Xu and Ruiqing Zhang	Fruit Detection and Recognition Based on Deep Learning for Automatic Harvesting:	The Fruits-0360 dataset with 131 categories	CNN Model	92% Accuracy
Arjun Paliwal, Sudhanshu Gaur, Shwetank Tripathi, Utkarsh Kumar Srivastava	Fruit Recognition Using Machine Learning	FRUITS-360 DATA SET	Decision tree K-nearest neighbour Naive bayes Logistic Regression	87% Accuracy

Horea Mureșan , Mihai Oltean	Fruit recognition from images using deep learning	Fruits-360 data set with 103 classes	CNN Model	99.98% Accuracy
Pankaj Shinde , Sakshi Patil , Omkar Ghanwat , Rutuja Palkar , Kunal Mhaske , Dhammjyoti Dhawase	Study on Fruit Recognization Using Image Processing	. FRUITS-360 DATA SET	YOLOv3 model	93% Accuracy
Shubham Kathepuri	Recognition and Classification of Fruits using Deep Learning Techniques	Own Dataset consists of 9 categories of fruits with 155 images.	VGG-16, and ResNet	95.22% Accuracy
Woo Chaw Seng , Seyed Hadi Mirisaee	A new method for fruits recognition system	Fifty fruit images have been collected for Fruits Recognition System	K-Nearest Neighbors Algorithm	90% Accuracy

2.Requirements of Project

2.1 Hardware Requirement

- Processor: 32-bit/ 64-bit, eight-core
- RAM: 4-8GB
- Hard Disk: 2GB or more

2.2 Software Requirement

- Operating System:Windows
- Language : Python
- Libraries:
  - NumPy: library for numerical calculations
  - Pandas: library for data manipulation and analysis
  - Tensorflow: library for large numerical computations without keeping deep learning in mind
  - Keras: neural network library
  - Matplotlib: for creating static, animated, and interactive visualizations
- IDE for workspace : Anaconda , Jupyter Notebook

2.3 Algorithm Used

- Convolutional Neural Networks (CNN)

3.METHODOLOGY

The following steps represents the flow of the working of the project.

- Data Generation
- Data Processing
- Building the model
- Training and Testing the model
- Recognition and Prediction

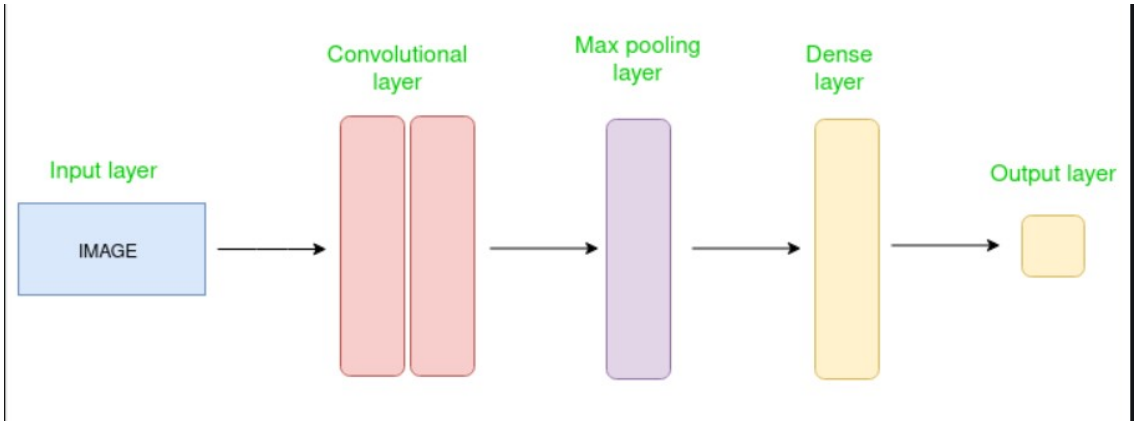
3.1 CONVOLUTIONAL NEURAL NETWORK:

A **Convolutional Neural Network (CNN)** is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

In a Regular Neural Network there are three types of layers:

1. **Input Layers:** It's the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
2. **Hidden Layer:** The input from the Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.

3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.



3.1.1 Convolution Layer : In convolution layer we take a small window size [typically of length 5*5] that extends to the depth of the input matrix. The layer consist of learnable filters of window size. During every iteration we slid the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position. As we continue this process well create a 2-Dimensional activation matrix that gives the response of that matrix at every spatial position. That is, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color

3.1.2 Pooling Layer : We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters. There are two type of pooling :

a) Max Pooling : In max pooling we take a window size [for example

window of size 2*2], and only take the maximum of 4 values. Well lid this window and continue this process, so well finally get a activation matrix half of its original Size.

b) Average Pooling : In average pooling we take average of all values in a window.

3.1.3. Flattening: The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.

3.1.4. Fully Connected Layer : In convolution layer neurons are connected only to a local region, while in a fully connected region, well connect the all the inputs to neurons.

3.1.5. Output Layer : After getting values from fully connected layer, well connect them to final layer of neurons[having count equal to total number of classes], that will predict the probability of each image to be in different classes.

3.2 TENSORFLOW:

TensorFlow is a free and open-source software library for data flow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

Features: TensorFlow provides stable Python (for version 3.7 across all platforms) and C APIs, and without API backwards compatibility guarantee: C++, Go, Java, JavaScript and Swift (early release). Third-party packages are available for C#, Haskell Julia, MATLAB,R, Scala, Rust, OCaml, and Crystal."New language support should be built on top of the C API. However, not all functionality is available in C yet." Some more functionality is provided by the Python API.

Application: Among the applications for which TensorFlow is the foundation, are automated image-captioning software, such as Deep Dream.

3.3 KERAS:

Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. It is made userfriendly, extensible, and

modular for facilitating faster experimentation with 15deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

3.4 NUMPY:

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

4.Dataset

The dataset contains around 100 images of the following 36 different fruits and vegetables.

- fruits- banana, apple, pear, grapes, orange, kiwi, watermelon, pomegranate, pineapple, mango.
- vegetables- cucumber, carrot, capsicum, onion, potato, lemon, tomato, raddish, beetroot, cabbage, lettuce, spinach, soy bean, cauliflower, bell pepper, chilli pepper, turnip, corn, sweetcorn, sweet potato, paprika, jalepeño, ginger, garlic, peas, eggplant.

Dataset Link: https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition

1. PROJECT DESIGN

In the recent years there has been tremendous research done on the fruits and vegetable recognition. With the help of literature survey done we realized the basic steps in recognition are :-

- Data generation
- Data preprocessing
- Building the model
- Training
- Recognition or Testing

5.1 DATA PREPROCESSING:

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

The steps followed are :

1. Rescaling
2. Height and Width shift
3. Resize Image
4. Zooming Images

5.2 BUILDING MODEL

A CNN Model is defined for the classification of different hand sign gestures .

Defined Model:

```
cnn=tf.keras.models.Sequential()

#Adding Layers

cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu',input_shape=[64,64,3]))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))

# adding second layer of the model
cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))

# adding third layer of the model
cnn.add(tf.keras.layers.Dropout(0.5))

# adding the Flatten layer
cnn.add(tf.keras.layers.Flatten())

# adding the Dense Layer
cnn.add(tf.keras.layers.Dense(units=128,activation='relu'))

#Output Layer
cnn.add(tf.keras.layers.Dense(units=36,activation='softmax'))

#Compiling the Model
cnn.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])
```

5.3 TRAINING:

The defined CNN- Model is trained on the self generated dataset.

The dataset contains around 100 images per class.

Total Classes = 36 classes

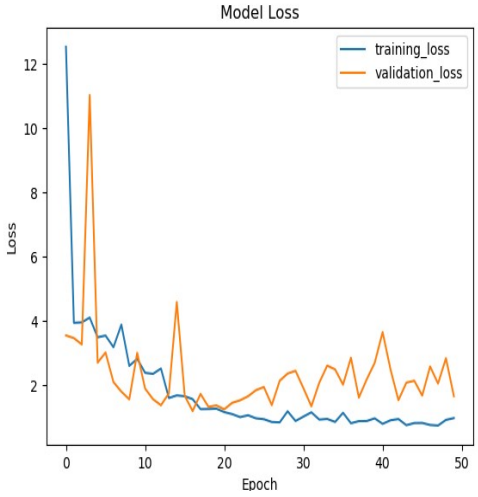
Total Images = 3466 images

Train Set = 3115 images

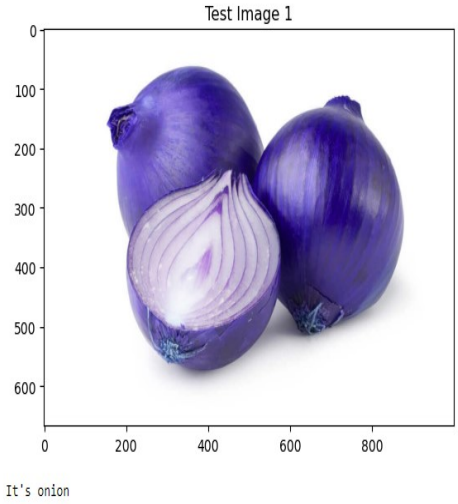
Validation Set = 351 images

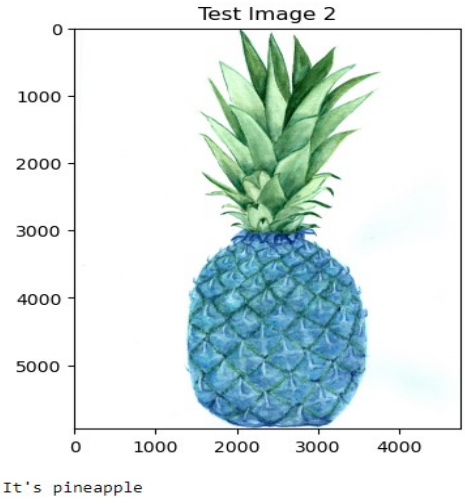
```python
#Training Model
with tf.device('/GPU:0'):
    training_history = cnn.fit(
        training_set,
        verbose=1,
        epochs=10,
        validation_data = validation_set
    )
```

5.4 ACCURACY - LOSS GRAPH



5.5 TESTING





The trained model is successfully able to predict different fruits and vegetables.

6.Applications

- Agriculture and Farming:

a) Crop Monitoring: Automated recognition can help farmers monitor the health and growth of crops, detect diseases, and optimize irrigation and fertilization.
b) Harvesting Robots: Robots equipped with recognition systems can be used for autonomous harvesting of fruits and vegetables.

- Food Processing Industry:

a) Sorting and Grading: Automated systems can sort and grade fruits and vegetables based on quality, size, and ripeness, ensuring consistent product quality.
b) Quality Control: Recognition helps identify defects or foreign objects in food products, reducing waste and improving safety.

- Consumer Convenience:

a) Smart Shopping: Smart shopping carts and apps can simplify the checkout process by recognizing items as they are placed in the cart.

7. Future Work

1. The model can be further trained with variation in dataset.
2. Multiple classes can be added to the dataset.
3. Ripe vs. Unripe Recognition: Developing models capable of distinguishing between ripe and unripe fruits and vegetables is crucial for quality control in agriculture and food processing. Future work can focus on fine-tuning recognition systems for ripeness assessment.
4. Real-time Recognition: Real-time recognition systems, especially for high-speed processing in food production lines, require further optimization and research to reduce latency and improve efficiency.

8.APPENDIX

8.1 Code for FruitAndVegetableRecognitionModel.py

```
import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

import os

print('----Libraries Loaded----')

os.chdir(r'C:\Users\HP\OneDrive\Desktop\Projects\Fruit And Vegetable Recognition Project')

print("----Folder Loaded----")

os.listdir()

training_set= tf.keras.utils.image_dataset_from_directory(

   'C:/Users/HP/OneDrive/Desktop/Projects/Fruit And Vegetable Recognition Project/Dataset/train',

   labels = "inferred",

   label_mode='categorical',

   class_names=None,

   color_mode='rgb',

   batch_size=32,

   image_size=(64,64),

   shuffle=True

)
```

```
validation_set= tf.keras.utils.image_dataset_from_directory(
   'C:/Users/HP/OneDrive/Desktop/Projects/Fruit And Vegetable Recognition Project/Dataset/validation',
   labels='inferred',
   label_mode='categorical',
   class_names=None,
   color_mode='rgb',
   batch_size=32,
   image_size=(64,64),
   shuffle=True
)
print("Dataset Loaded")
cnn=tf.keras.models.Sequential()

#Adding Layers

cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu',input_shape=[64,64,3]))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))

# adding second layer of the model
cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
```

```
# adding third layer of the model

cnn.add(tf.keras.layers.Dropout(0.5))


# adding the Flatten layer

cnn.add(tf.keras.layers.Flatten())


# adding the Dense Layer

cnn.add(tf.keras.layers.Dense(units=128,activation='relu'))

#Output Layer

cnn.add(tf.keras.layers.Dense(units=36,activation='softmax'))


#Compiling the Model

cnn.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])


#Printing Summary Of Model

cnn.summary()

with tf.device('/GPU:0'):

   training_history = cnn.fit(

       x = training_set,

       verbose=1,

       epochs=50,

       validation_data = validation_set

   )
```

```
# Saving Model

cnn.save('fruitAndVegetableRecognitionmodel.h5')

print('----Model Saved----')

#Saving History Of Model

import json

with open ('training_hist.json','w') as f:

  json.dump(training_history.history,f)

print('Validation set Accuracy: {}%'.format(training_history.history['val_accuracy'][-
1]*100))

plt.plot(training_history.history['accuracy'][0:220])

plt.plot(training_history.history['val_accuracy'][0:220])

plt.title('Model Accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['training_accuracy', 'validation_accuracy'])

plt.show()

plt.plot(training_history.history['loss'][0:220])

plt.plot(training_history.history['val_loss'][0:220])

plt.title('Model Loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['training_loss', 'validation_loss'])

plt.show()
```

8.2 Code for FruitAndVegetableRecognitionTesting.py

```
import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

import cv2

print('----Libraries Imported----')

cnn=tf.keras.models.load_model("C:/Users/HP/OneDrive/Desktop/Projects/Fruit And Vegetable Recognition Project/fruitAndVegetableRecognitionmodel.h5")

print('----Model Loaded----')

#Defining Classes for different fruits and vegetables

classes ={0: 'apple',1: 'banana',2: 'beetroot',3: 'bell pepper',4: 'cabbage',5: 'capsicum',

    6: 'carrot',7: 'cauliflower',8: 'chilli pepper',9: 'corn',10: 'cucumber',11: 'eggplant',

    12: 'garlic',13: 'ginger',14: 'grapes',15: 'jalepeno',16: 'kiwi',17: 'lemon',

    18: 'lettuce',19: 'mango',20: 'onion',21: 'orange',22: 'paprika',23:  'pear',

    24: 'peas',25: 'pineapple',26: 'pomegranate',27: 'potato',28: 'raddish',29: 'soy beans',

    30: 'spinach',31: 'sweetcorn',32: 'sweetpotato',33: 'tomato',34: 'turnip',35: 'watermelon'

}

print('----Classes Generated----')

image_path="C:/Users/HP/OneDrive/Desktop/Projects/Fruit And Vegetable Recognition Project/Dataset/test/test166.jpg"

img = cv2.imread(image_path)

plt.imshow(img)

plt.title("Test Image 1")
```

```
plt.show()

#Preprocessing for test image

image=tf.keras.preprocessing.image.load_img(image_path,target_size=(64,64))

input_arr =tf.keras.preprocessing.image.img_to_array(image)

input_arr=np.array([input_arr])


#Predicting Test Image on Trained Model

predictions=cnn.predict(input_arr)

#Getting Index Of Fruit in Test Image

result_index= np.where(predictions[0]==max(predictions[0]))


#Printing Test Image

plt.imshow(img)

plt.title("Test Image 1")

plt.show()

print("It's {}".format(classes[result_index[0][0]]))

image_path2="C:/Users/HP/OneDrive/Desktop/Projects/Fruit And Vegetable
Recognition Project/Dataset/test/test277.jpg"

img = cv2.imread(image_path2)

plt.imshow(img)

plt.title("Test Image 2")

plt.show()

#Preprocessing for test image
```

```
image=tf.keras.preprocessing.image.load_img(image_path2,target_size=(64,64))

input_arr =tf.keras.preprocessing.image.img_to_array(image)

input_arr=np.array([input_arr])


#Predicting Test Image on Trained Model

predictions=cnn.predict(input_arr)

#Getting Index Of Fruit in Test Image

result_index= np.where(predictions[0]==max(predictions[0]))


#Printing Test Image

plt.imshow(img)

plt.title("Test Image 2")

plt.show()

print("It's {}".format(classes[result_index[0][0]]))

image_path3="C:/Users/HP/OneDrive/Desktop/Projects/Fruit And Vegetable
Recognition Project/Dataset/test/test18.jpg"

img = cv2.imread(image_path3)

plt.imshow(img)

plt.title("Test Image 3")

plt.show()

#Preprocessing for test image

image=tf.keras.preprocessing.image.load_img(image_path3,target_size=(64,64))

input_arr =tf.keras.preprocessing.image.img_to_array(image)
```

```
input_arr=np.array([input_arr])


#Predicting Test Image on Trained Model

predictions=cnn.predict(input_arr)

#Getting Index Of Fruit in Test Image

result_index= np.where(predictions[0]==max(predictions[0]))


#Printing Test Image

plt.imshow(img)

plt.title("Test Image 3")

plt.show()

print("It's {}".format(classes[result_index[0][0]]))

image_path4="C:/Users/HP/OneDrive/Desktop/Projects/Fruit And Vegetable
Recognition Project/Dataset/test/test15.jpg"

img = cv2.imread(image_path4)

plt.imshow(img)

plt.title("Test Image 4")

plt.show()

#Preprocessing for test image

image=tf.keras.preprocessing.image.load_img(image_path4,target_size=(64,64))

input_arr =tf.keras.preprocessing.image.img_to_array(image)

input_arr=np.array([input_arr])
```

```
#Predicting Test Image on Trained Model

predictions=cnn.predict(input_arr)

#Getting Index Of Fruit in Test Image

result_index= np.where(predictions[0]==max(predictions[0]))


#Printing Test Image

plt.imshow(img)

plt.title("Test Image 4")

plt.show()

print("It's {}".format(classes[result_index[0][0]]))
```

9.REFERENCES

1. https://www.youtube.com/watch?v=pDXdlXlaCco

2. https://www.youtube.com/watch?v=YjnGou4skGU&t=560s

3. https://www.youtube.com/watch?v=vQZ4IvB07ec&t=487s

4. https://www.youtube.com/watch?v=6Bn0PY_ouBY&t=1s

5. https://www.youtube.com/watch?v=90ElPbbhVWA&list=PLvz5lCwTgdXByZ_z-LFo4vJbbFIMPhkkM&index=2

6. https://www.youtube.com/watch?v=WQeoO7MI0Bs

7. https://www.youtube.com/watch?v=doDUihpj6ro