

FRUIT CLASSIFICATION USING KERAS WITH TENSOR FLOW AND DEPLOYMENT IN WEB APPLICATION.

Submitted in partial fulfillment of the requirements for
the award of
Master in Engineering in Department of Computer Science and engineering
By

ANNIE VINCY. F (39310002)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA**

**INSTITUTE OF SCIENCE AND
TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade “A” by NAAC**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119. TAMIL NADU
APRIL 2021**



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" Grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119.

Phone: 044 - 2450 3150 / 51 / 52 / 54 / 55 Fax: 044 - 2450 2344

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of Annie vincy. F (Reg no:39310002) who carried out the project entitled **“FRUIT CLASSIFICATION USING KERAS WITH TENSOR FLOW AND DEPLOYMENT IN WEB APPLICATION.”** under my supervision from November 2020 to April 2021.

Internal Guide

DR.S. GOWRI, M.E., Ph.D.,

Head of the Department

Dr. VIGNESHWARI, M.E.,Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **ANNIE VINCY. F(39310002)** hereby declare that the Professional Training Report on “**FRUIT CLASSIFICATION USING KERAS WITH TENSOR FLOW AND DEPLOYMENT IN WEB APPLICATION**” done by me under the guidance of guide **Dr.S.GOWRI M.E.,Ph.D.**, Department of Information Technology at Sathyabama Institute of Science and technology is submitted in partial fulfillment of the requirements for the award of Master in Engineering degree in Computer Science and engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the Board **of Management** of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.Sasikala M.E.,Ph.d, Dean**, School of Computing, **Dr. Vigneshwari M.E.,Ph.D.**, Head of the Department of Computer science and engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S.Gowri M.E.,Ph.D.**, for this valuable guidance, suggestions and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all teaching and Non-teaching staff members of the **Department of Computer Science and engineering** who were helpful in many ways for the completion of the project.

ABSTRACT:

Fruits are very common in today's world – despite the abundance of fast food and refined sugars, fruits remain widely consumed foods. During production of fruits, it might be that they need to be sorted, to give just one example. Traditionally being performed mechanically, today, deep learning-based techniques could augment or even take over this process. Deep Learning models, or to be more specific, Tensorflow use Convolutional Neural Networks (CNNs) models are proposed in this project. A comparison of the proposed and current algorithms reveals that the accuracy fruits types classification based on CNNs is higher than other algorithms. It is predicted that the success of the obtained results will increase if the CNN method is supported by adding extra feature extraction methods and classifying fruits types on image.

LIST OF FIGURES

FIGURE NO.	FIGURE NAME
1.	Down sampling.
2.	RGB representation of an image.
3.	Convolution operation.
4.	Visualization of convolution.
5.	Convolution with Stride 1.
6.	Stride 1 with Padding 1.
7.	After applying padding.
8.	CNN architecture.
9.	Image matrix multiplies kernel or filter matrix.
10.	Image matrix multiplies kernel or filter matrix
11.	3x3 Output matrix.
12.	Some common filters.

13.	Stride of 2 pixels
14.	ReLU operation
15.	Max Pooling
16.	After pooling layer, flattened as FC layer
17.	Complete CNN architecture
18.	Anaconda distribution.
19.	Anaconda navigator.
20.	System architecture.
21.	Use Case diagram.
22.	Sequence diagram.
23.	Activity diagram.
24.	Convolutional neural networks.
25.	Training the system.
26.	Processing through CNN layers.

27.	Calculating model accuracy.
28.	Graph for accuracy and loss.
29.	Open the anaconda navigator.
30.	Launch the jupyter notebook platform.
31.	Open the corresponding result folder.
32.	Run the corresponding IPYNB file.
33.	Training the system with a type of apple.(example).
34.	Accuracy and loss for training and testing.
35.	Graphical representation of accuracy and loss for training and testing.
36.	Implementation using tensorflow and keras.
37.	Accuracy and loss for implementation using keras and tensorflow.
38.	Graphical representation for implementation using keras and tensorflow.
39.	Uploading fruit image.
40.	Uploaded image.

41.	Fruit classified.
-----	-------------------

LIST OF ABBREVIATION

AI	Artificial Intelligence
API	Application programming interface
NN	Neural Network
IDE	Integrated Development Environment
UML	Unified modeling language
CNN	Convolutional neural networks

TABLE OF CONTENT

CHAPTER NO	TITLE NAME	PAGE NO
1.	INTRODUCTION.	12
2.	LITERATURE SURVEY.	27
3.	EXPERIMENTAL METHODS AND ALGORITHMS.	36
4.	RESULTS AND DISCUSSIONS.	58
5.	CONCLUSIONS AND ENHANCEMENTS.	61

CHAPTER 1

INTRODUCTION

1.1 DOMAIN DESCRIPTION.

CONVOLUTIONAL NEURAL NETWORKS.

1.1.1 What is CNN?

Computer vision is evolving rapidly day-by-day. One of the reasons is deep learning. When we talk about computer vision, a term convolutional neural network(abbreviated as CNN) comes in our mind because CNN is heavily used here. Examples of CNN in computer vision are face recognition, image classification etc. It is similar to the basic neural network. CNN also has learnable parameters like neural networks i.e, weights, biases etc.

1.1.2. Why should we use CNN ?

Problem with Feed forward Neural Network,

Suppose you are working with MNIST dataset, you know each image in MNIST is $28 \times 28 \times 1$ (black & white image contains only 1 channel). Total number of neurons in the input layer will be $28 \times 28 = 784$, this can be manageable. What if the size of the image is 1000×1000 which means you need 10^6 neurons in input layer. Oh! This seems a huge number of neurons are required for operation. It is computationally ineffective right. So here comes Convolutional Neural Network or CNN. In simple words what CNN does is, it extracts the feature of the image and converts it into lower dimension without losing its characteristics. In the following example you can see that initially the size of the image is $224 \times 224 \times 3$. If you proceed without convolution then you need $224 \times 224 \times 3 = 150, 336$ numbers of neurons in the input layer but after applying convolution you input tensor dimension is

reduced to $1 \times 1 \times 1000$. It means you only need 1000 neurons in the first layer of the feed forward neural network.

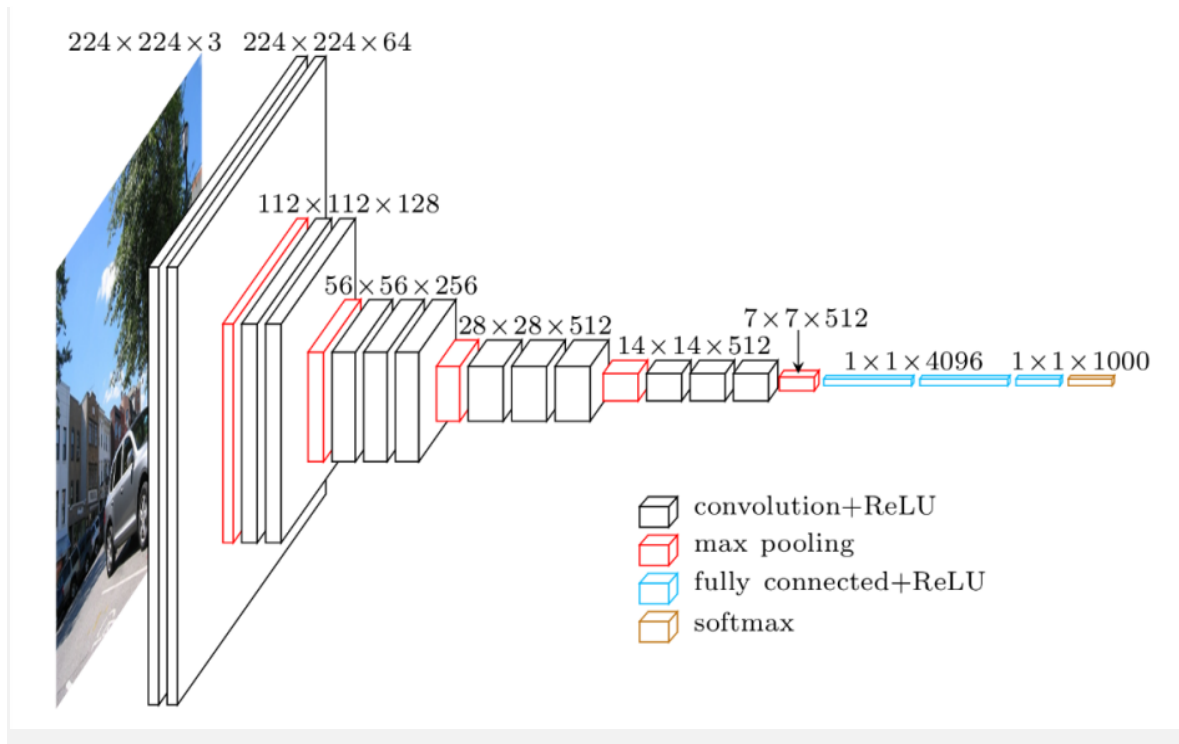


Fig 1: Down sampling.

1.1.3. Few Definitions

There are few definitions you should know before understanding CNN

1.1.3.1 Image Representation

Thinking about images, it's easy to understand that it has a height and width, so it would make sense to represent the information contained in it with a two dimensional structure (a matrix) until you remember that images have colors, and to add information about the colors, we need another dimension, and that is when Tensors become particularly helpful.

Images are encoded into color channels, the image data is represented into each color intensity in a color channel at a given point, the most common one being RGB, which means Red, Blue and Green. The information contained

into an image is the intensity of each channel color into the width and height of the image, just like this



Fig 2: RGB representation of an image.

So the intensity of the red channel at each point with width and height can be represented into a matrix, the same goes for the blue and green channels, so we end up having three matrices, and when these are combined they form a tensor.

1.1.3.2 Edge Detection

Every image has vertical and horizontal edges which actually combine to form an image. Convolution operation is used with some filters for detecting edges. Suppose you have a gray scale image with dimension 6×6 and a filter of dimension 3×3 (say). When a 6×6 grey scale image convolves with 3×3 filter, we get 4×4 image. First of all, the 3×3 filter matrix gets multiplied with the first 3×3 size of our gray scale image, then we shift one column right up to end, after that we shift one row and so on.

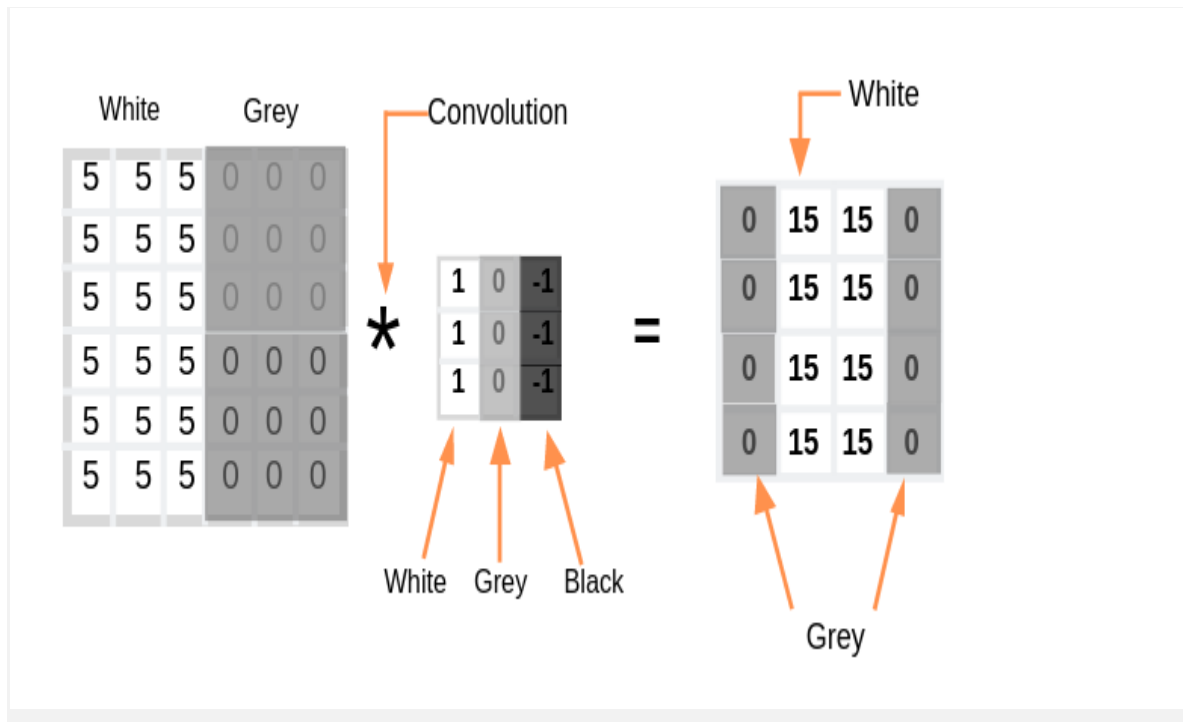


Fig 3: Convolution operation.

The convolution operation can be visualized in the following way. Here our image dimension is 4 x 4 and filter is 3 x 3, hence we are getting output after convolution is 2 x 2.

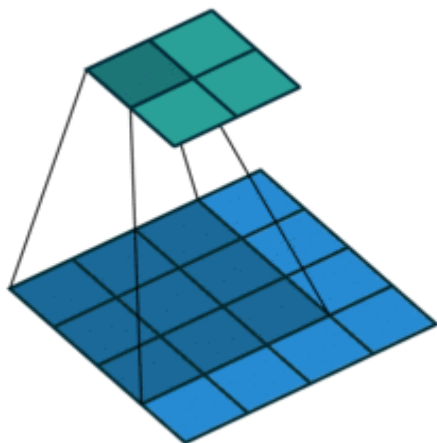


Fig 4: Visualization of convolution.

If we have $N \times N$ image size and $F \times F$ filter size then after convolution result will be

$$(N \times N) * (F \times F) = (N-F+1) \times (N-F+1) \text{ (Apply this for above case)}$$

1.1.3.3 Stride and Padding

Stride denotes how many steps we are moving in each step in convolution. By default it is one.

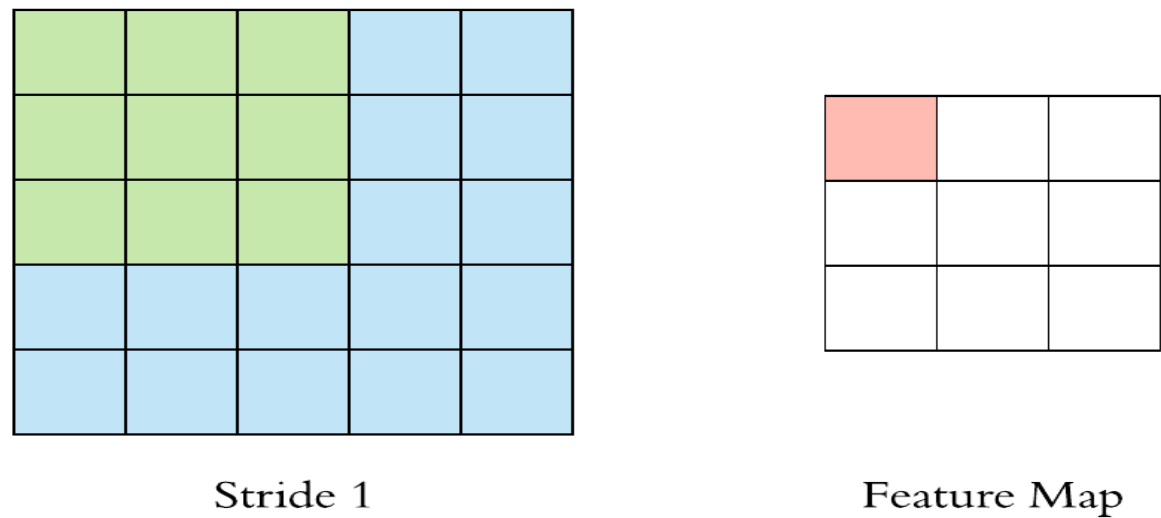


Fig 5: Convolution with Stride 1.

We can observe that the size of output is smaller than input. To maintain the dimension of output as in input, we use padding. Padding is a process of adding zeros to the input matrix symmetrically. In the following example, the extra grey blocks denote the padding. It is used to make the dimension of output same as input.

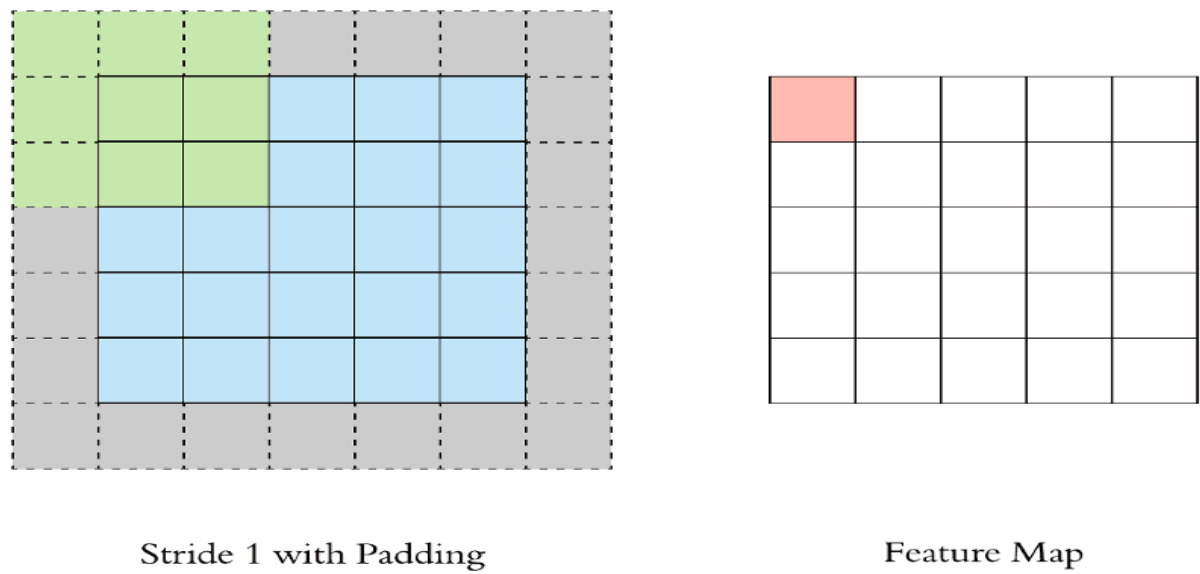


Fig 6: Stride 1 with Padding 1.

Let say 'p' is the padding

Initially(without padding)

$$(N \times N) * (F \times F) = (N-F+1) \times (N-F+1) \dots (1)$$

After applying padding

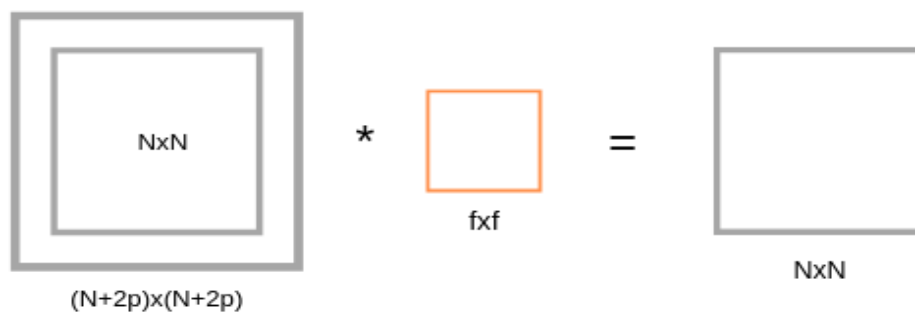


Fig 7: After applying padding.

If we apply filter $F \times F$ in $(N+2p) \times (N+2p)$ input matrix with padding, then we will get output matrix dimension $(N+2p-F+1) \times (N+2p-F+1)$. As we know that after applying padding we will get the same dimension as the original input dimension $(N \times N)$. Hence we have,

$(N+2p-F+1) \times (N+2p-F+1)$ equivalent to $N \times N$

$$N+2p-F+1 = N \text{ ---(2)}$$

$$p = (F-1)/2 \text{ ---(3)}$$

The equation (3) clearly shows that Padding depends on the dimension of the filter.

1.2 CNN LAYERS

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.

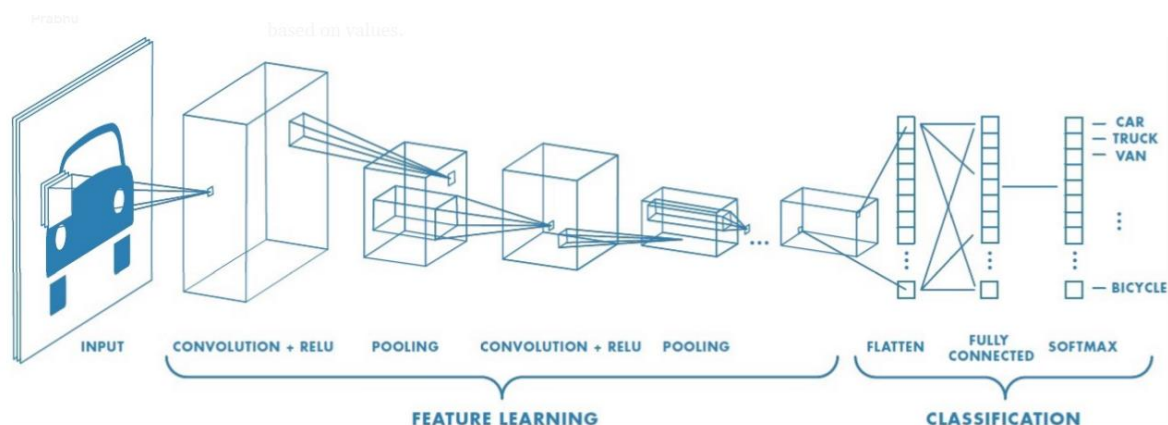


Fig 8: CNN Architecture.

1.2.1 Convolution layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

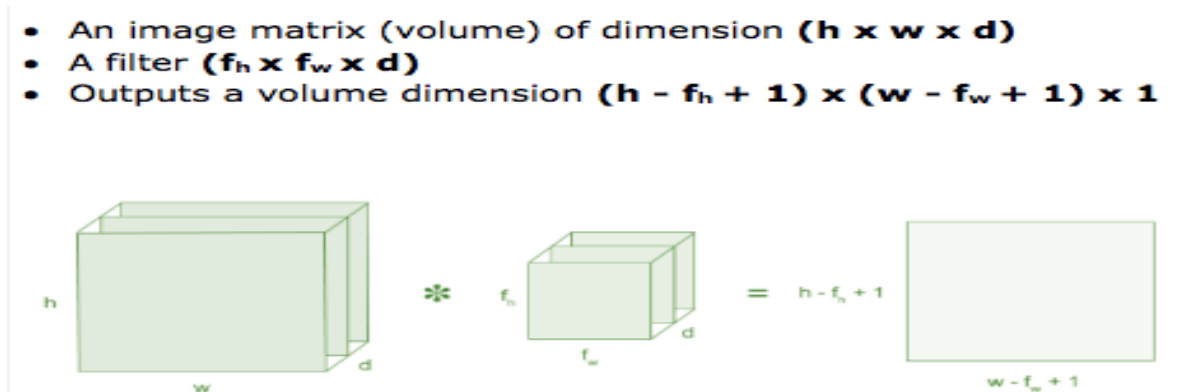


Fig 9: Image matrix multiplies kernel or filter matrix.

Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below



Fig 10: Image matrix multiplies kernel or filter matrix.

Then the convolution of 5 x 5 image matrix multiplied with 3 x 3 filter matrix which is called “**Feature Map**” as output shown in below

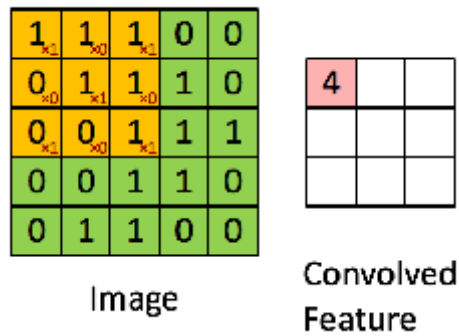


Fig 11: 3x3 Output matrix.

Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters. The below example shows various convolution images after applying different types of filters (Kernels).

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Fig 12 : Some common filters.

1.2.2 Strides

Stride is the number of pixels shifted over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

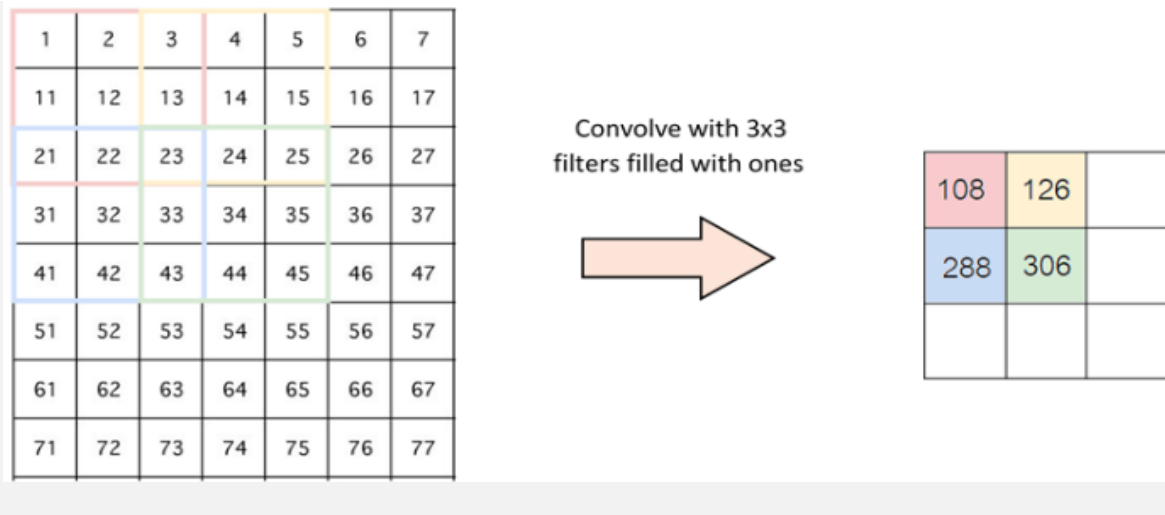


Fig 13 : Stride of 2 pixels.

1.2.3 Padding

Sometimes filter does not fit perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

1.2.4 Non Linearity (ReLU)

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is $f(x) = \max(0, x)$.

Why ReLU is important : ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values.

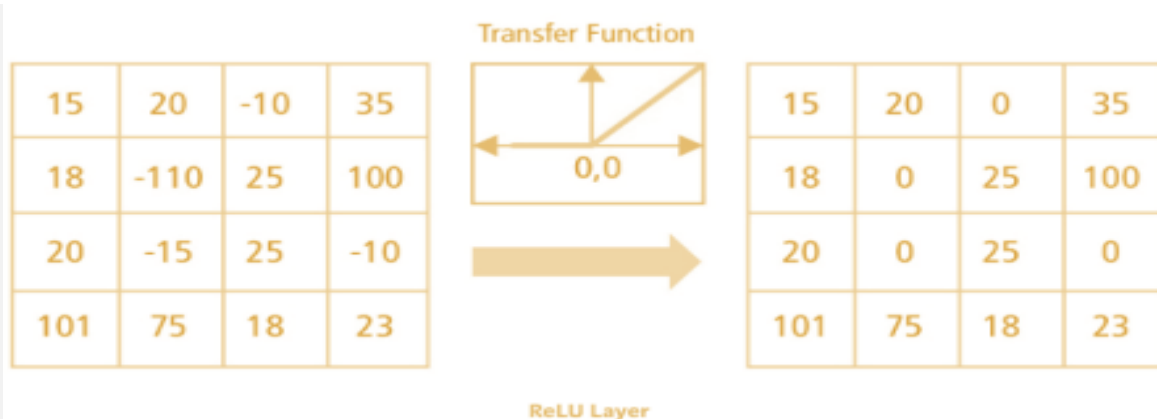


Fig 14: ReLU operation.

There are other nonlinear functions such as tanh or sigmoid that can also be used instead of ReLU. Most data scientists use ReLU since performance wise ReLU is better than the other two

1.2.5 Pooling Layers.

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling is also called subsampling or downsampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling takes the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

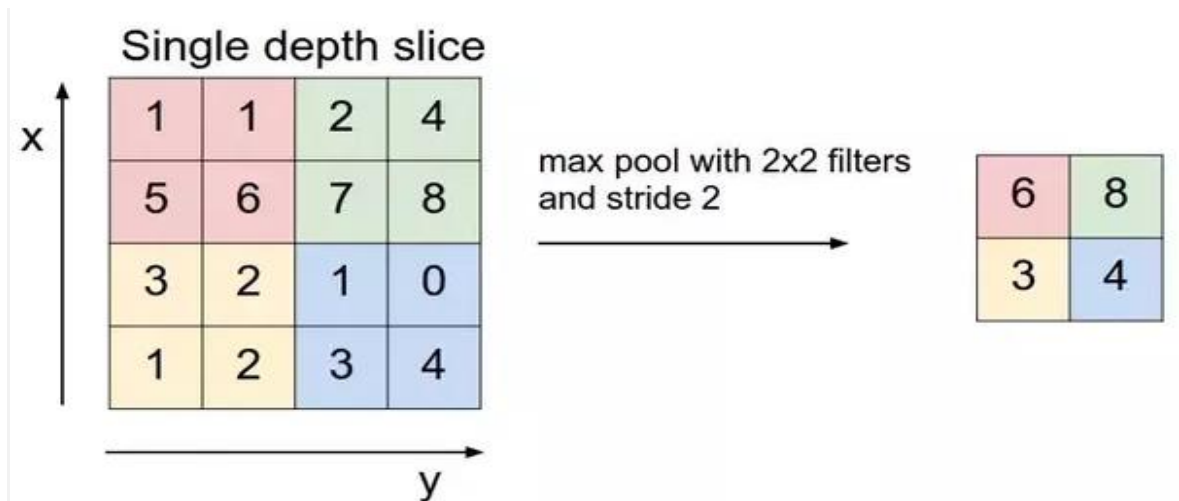


Fig 15: Max Pooling.

1.2.6 Fully Connected Layer

The layer we call the FC layer, we flatten our matrix into vectors and feed it into a fully connected layer like a neural network.

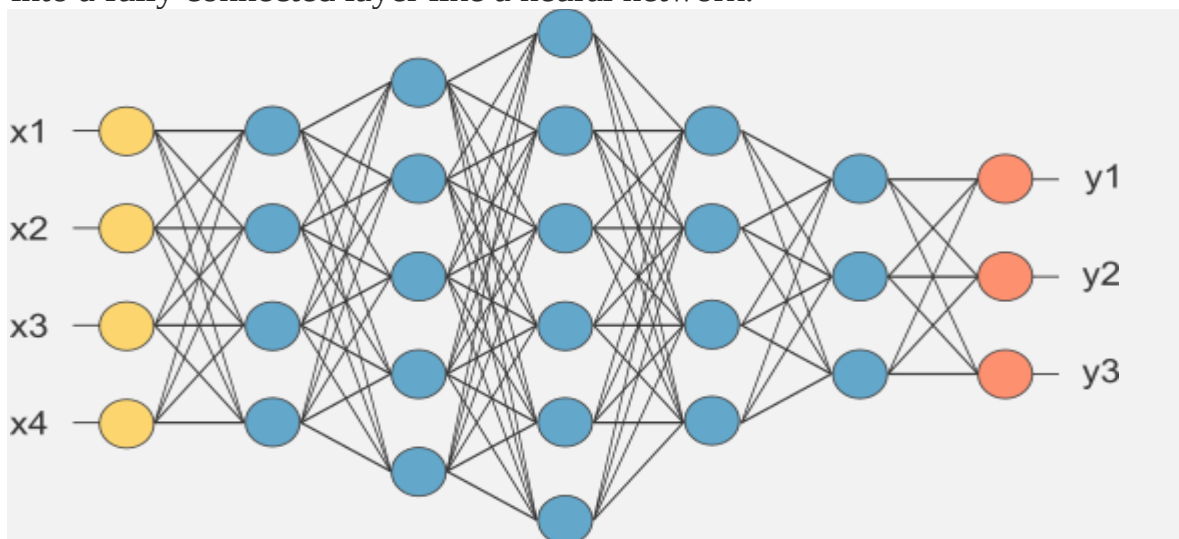


Fig 16: After pooling layer, flattened as FC layer.

In the above diagram, the feature map matrix will be converted as a vector ($x_1, x_2, x_3, \dots$). With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs as cat, dog, car, truck etc.,

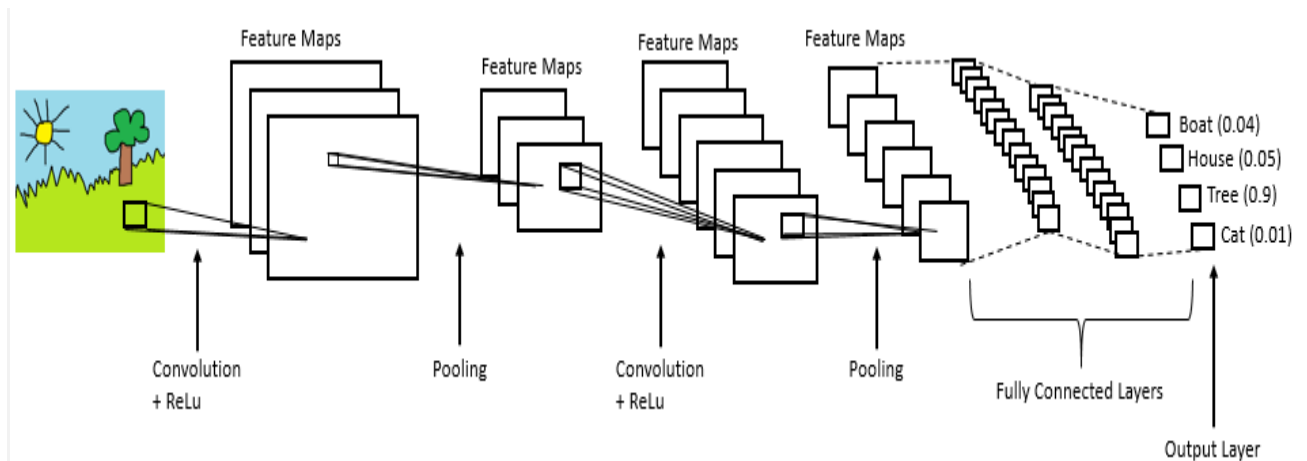


Fig 17: Complete CNN architecture.

1.2.7 Advantages of Convolutional Neural Networks.

The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. For example, given many pictures of cats and dogs, it can learn the key features for each class by itself. The accuracy in image recognition problems.

1.3 Introduction of the project:

In this project, we are introducing a new and robust machine learning model based on CNN. IT will automatically differentiate the single cells in thin blood smears on microscope slides as either infected or not infected. Here the algorithms can also extract complex images of different fruit types and detect fruits in different angles. Whole images are used as inputs, so it is not necessary to perform any preprocessing of fruits images or collection of data about the fruits. To do this fruit classification deep learning methods are used. This will make the fruits classification easier with the inclusion of different deep-learning algorithms. Deep-learning algorithms will also give accurate results in the classification. Tensor-flow will make the classification of fruits easier and faster. Some of the sensor systems will contain five dual-band antennas and a customized circuit to approximate the values. Phase detector module and down-

conversion mixers are the main components of the circuits that are customized for sensor systems. Growing global consumption of thick rind fruits has contributed to their greater economic significance. There are a variety of thick rind fruits, such as pomelo, watermelon, and durian. To command premium prices, quality control of the fruits is paramount. Conventionally, fruit quality assessment relies on human sensory input. However, the practice is subjective, leading to inconsistent fruit quality and reputation damage. Thick rind fruits, e.g., pomelo, watermelon, are typical of spherical shape or, specifically, the concentric dielectric sphere with rind as the outer layer and flesh as the inner layer. These simulations were carried out using a pomelo fruit model whose outer and inner diameters are 16 and 12 cm, which are the typical size of pomelo fruit. The flesh portion of a typical pomelo fruit consists of 15 sections, and the simulations are to differentiate between normal and granulated pomelo fruits. Granulation is a condition in which juice vesicles become dry and tough, which makes the fruit unpalatable. In the simulation, the pomelo that contains one granulated section (out of 15 sections) is regarded as failing. Before the simulation, the relative permittivity of real pomelo fruits (three each for normal and granulated pomelo) was determined using Agilent 85070E dielectric probe connected to N9916A Field Fox microwave vector network analyzer. The relative permittivity of the rind and flesh was measured using the dielectric probe. To obtain accurate relative permittivity, the pomelo fleshes were homogeneously pressed before measurement by the dielectric probe. Simulations were carried out using single- (1 GHz) and dual-frequency (1 GHz/2.3 GHz) sensor systems; and the results indicated that the single frequency scheme achieved lower accuracy in the classification of pomelo fruits. As a result, a prototype of four beams dual-frequency sensor system was fabricated, and experiments were carried out using plastic pomelo models and real pomelo fruits. They further demonstrate The experimental results showed that the average of the normal and granulated pomelo fruits are almost identical,

while the SD of the granulated pomelo is noticeably larger than that of the normal pomelo. As such, SD is used as the determinant of pomelo quality.

1.4 Motivation:

The main motivation behind this project is that there are many applications for differentiating fruits and vegetables. This application is developed to classify the difference between the fruits. Tensor-flow will make the differentiation between the fruits through images.

1.5 Objective:

The main objective is to classify the fruits of the same variety. Identification of fruits even if it is in an uneven background and also classification of fruits based on the image but not based on the physical measurements of the fruits.

CHAPTER -2 LITERATURE SURVEY

2.1 EXISTING SYSTEM

This reports a four-beam 1 GHz/2.3 GHz dual frequency sensor system to assess the quality of thick rind pomelo fruits. The principle of the proposed sensor system is based on phase difference between lower- and higher-frequency reflection coefficients. The sensor system consists of five dual-band antennas and a customized circuit to approximate. The main components of the customized circuit are down conversion mixers and phase detector modules. Growing global consumption of thick rind fruits has contributed to their greater economic significance. There are a variety of thick rind fruits, such as pomelo, watermelon, and durian. To command premium prices, quality control of the fruits is paramount. Conventionally, fruit quality assessment relies on human sensory input. However, the practice is subjective, leading to inconsistent fruit quality and reputation damage. Thick rind fruits, e.g., pomelo, watermelon, are typically of spherical shape or, specifically, concentric dielectric spheres with rind as the outer layer and flesh as the inner layer. In this research, simulations were carried out using a pomelo fruit model whose outer and inner diameters are 16 and 12 cm, which are the typical size of pomelo fruit. The flesh portion of a typical pomelo fruit consists of 15 sections, and the simulations are to differentiate between normal and granulated pomelo fruits. Granulation is a condition in which juice vesicles become dry and tough, which makes the fruit unpalatable. In the simulation, the pomelo that contains one granulated section (out of 15 sections) is regarded as failing. Prior to simulation, the relative permittivity of real pomelo fruits (three each for normal and granulated pomelos) were determined using Agilent 85070E dielectric probe connected to N9916A Field Fox microwave vector network analyzer (Figure 3(a)). The relative permittivity of the rind and flesh was measured using the dielectric probe, To obtain accurate relative permittivity, the pomelo fleshes were homogeneously pressed prior to measurement by the dielectric probe.

Simulations were carried out using single- (1 GHz) and dual-frequency (1 GHz/2.3 GHz) sensor systems; and the results indicated that the single-frequency scheme achieved lower accuracy in classification of pomelo fruits. As a result, a prototype of a four beam dual-frequency sensor system was fabricated, and experiments were carried out using plastic pomelo models and real pomelo fruits. They further demonstrate The experimental results showed that the average of the normal and granulated pomelo fruits are almost identical, while SD of the granulated pomelo is noticeably larger than that of the normal pomelo. As such, SD is used as the determinant of pomelo quality.

2.1.2 Disadvantages:

- It has not focused on identifying Alex Net CNN as a classifier.
- They are not using deep learning technique
- It has not focused on increasing the recognition rate and classification accuracy of severity of fruit types.

2.2 LITERATURE SURVEY.

2.2.1.Title: Machine Learning Driven Approach Towards the Quality Assessment of Fresh Fruits Using Non-invasive Sensing

Author: Aifeng Ren, Adnan Zahid, Ahmed Zoha, Syed Aziz Shah

In agriculture science, accurate information of moisture content (MC) in fruits and vegetables in an automated fashion can be vital for astute quality and grading evaluation. This demands for a viable, feasible and cost-effective technique for the defect recognition using timely detection of MC in fruits and vegetables to maintain a healthy sensory characteristic of fruits. They propose a non-invasive machine learning (ML) driven technique to monitor variations of MC in fruits using the terahertz (THz) waves with Swissto12 material

characterization kit (MCK) in the frequency range of 0.75 THz to 1.1 THz. In this regard, multi-domain features are extracted from time-frequency, and time-frequency domains, and applied three ML algorithms such as support vector machine (SVM), knearest neighbour (KNN) and Decision Tree (D-Tree) for the precise assessment of MC in both apple and mango slices. The results illustrated that the performance of SVM exceeded other classifiers results using 10-fold validation and leave-one-observation-out-cross-validation techniques. The novel and non-invasive sensing technique for the quality assessment of fresh fruits by integrating machine learning ML with terahertz (THz) waves. For this purpose, scattering measurements of both apple and mango slices were obtained using the Swissto12 MCK system. Multiple domain features such as time-, frequency-, and time-frequency domains were extracted and the classification was performed to determine the moisture content (MC) in fruits slices using SVM, KNN and D-tree classifiers. Results showed that by discarding unwanted features and applying comprehensive cross-validation technique, classification accuracy was substantially improved which eventually helped to reduce the computational time. Furthermore, it was perceived that, in most of the cases, the SVM classifier based on RBF kernel outperformed KNN and D-Tree classifiers for both fruit slices. Thus, SVM demonstrated more precise quality assessments of fresh fruits by determining their MC more precisely for four consecutive days.

2.2.2 Title: Weakly Supervised Fruit Counting for Yield Estimation Using Spatial Consistency

Author: Enrico Bellocchio, Thomas A. Ciarfuglia, Gabriele Costante

Fruit counting is a fundamental component for yield estimation applications. Most of the existing approaches address this problem by relying on fruit models. Despite the impressive results achieved by these approaches,

all of them need strong supervision information during the training phase. In agricultural applications, manual labeling may require a huge effort or, in some cases, it could be impossible to acquire fine-grained ground truth labels. In this letter, we tackle this problem by proposing a weakly supervised framework that learns to count fruits without the need for task-specific supervision labels. a novel weakly-supervised framework for fruit counting in agricultural applications. The WSCOUNT strategy is able to learn to count without requiring task-specific supervision labels, such as manually labelled object bounding boxes or total instance count. Instead, it exploits a simple binary presence absence classifier (PAC), trained with only image level labels, and a spatial consistency loss that imposes coherency among counting branches at different scale levels. By removing the necessity for laborious data collection processes, our approach takes an important step towards the complete automation of yield estimation systems. The experiments run on three different fruit species clearly show that our approach guarantees performances that are comparable to those of fully supervised baselines.

2.2.3 Title: Classification of Cape Gooseberry Fruit According to Its Level of Ripeness Using Machine Learning Techniques and Different Color Spaces

Author: WILSON CASTRO , JIMY OBLITAS , MIGUEL DE-LA-TORRE

The classification of fresh fruits according to their visual ripeness is typically a subjective and tedious task; consequently, there is a growing interest in the use of non-contact techniques to automate this process.

Machine learning techniques, such as artificial neural networks, support vector machines (SVMs), decision trees, and K-nearest neighbor algorithms, have been successfully applied for classification problems in the literature, particularly for images of fruit. However, the particularities of each classification problem make it difficult, if not impossible, to select a general technique that is applicable to all types of fruit. In this paper, the

combinations of four machine learning techniques and three color spaces (RGB, HSV, and $L^*a^*b^*$) were evaluated with regard to their ability to classify Cape gooseberry fruits. To this end, 925 Cape gooseberry fruit samples were collected, and each fruit was manually classified into one of seven different classes according to its level of ripeness. Develop a non-intrusive system for the classification of gooseberry fruits according to their degree of ripeness. Twelve classification models were developed by combining four machine learning techniques (the ANN, KNN, DT, and SVM classifiers) with three color spaces (RGB, HSV, and $L^*a^*b^*$). The color space utilized for the classification clearly influenced the accuracy of the sorting system; this dynamic was observed mainly in the models based on the ANN and SVM techniques. Meanwhile, the models based on the KNN and DT techniques yielded good results regardless of the color space used. The $L^*a^*b^*$ color space produced good results regardless of the machine learning technique employed. However, the classifier developed from the combination of the SVM technique and $L^*a^*b^*$ color space gave the best performance in terms of the accuracy and f-measure.

2.2.4 Title: BIFFOA: A Novel Binary Improved Fruit Fly Algorithm for Feature Selection

Author: YUN HOU, JINGYAO LI, HAIHONG YU

The number of attributes of high-dimensional data and an essential preprocess work in classification. It eliminates irrelevant, redundant, and noisy features, improves the performance of the model and reduces the computational burden. Fruit fly optimization algorithm is a new algorithm proposed in recent years, which imitates the foraging behavior of fruit fly. To the best of our knowledge, it has not been systematically applied to feature selection. This paper uses the fruit fly optimization algorithm as a search strategy and designs

a wrapper-based feature selection method, named binary improved fruit fly optimization algorithm (BIFFOA). Besides, four different strategies based on evolutionary population dynamics (EPD) and new mutation operators are employed to enhance the BIFFOA. The extensive experiments on 25 datasets show that the performance of the BIFFOA is better than several state-of-the-art algorithms. a binary IFFOA (BIFFOA) algorithm is proposed to deal with feature selection problems. Four mechanisms based on EPD with new mutation operators were used to improve the exploration and development capabilities of the BIFFOA, and the diversity of the population was improved. Although, in contrast to the feature selection algorithms that have been proposed in recent years, the improved hybrid algorithm, BIFFOA_EPD_Tour has a specific advantage in dealing with feature selection problems. it can guarantee a high classification accuracy and reduce the number of selected features at the same time is the focus of our further research.

2.2.5 Title: Dual-frequency sensor for thick rind fruit quality assessment

Author: Chainarong Kittiyapunya, Chuwong Phongcharoenpanich

A novel multi-beam dual frequency sensor system to assess the quality of thick rind fruits based on phase difference between lower- and higher-frequency reflection coefficients (Φ_{12}). The proposed sensor system consists of five dual-band antennas and a customized circuit that approximates lower- and higher-frequency phase difference. The main components of the customized circuit are down-conversion mixers and phase detector modules. For comparison, simulations were initially carried out using single- (1 GHz) and dual-frequency (1 GHz/2.3 GHz) sensor systems, and results indicated lower accuracy in classification of thick rind fruits of the single frequency scheme. As a result, a dual-frequency sensor prototype was fabricated and

experimented with normal and granulated plastic pomelo models and real pomelo fruits. The experimental results revealed that the average Φ_{12} of the normal and granulated pomelos are almost identical, while the standard deviation (SD) of Φ_{12} of the granulated pomelo is significantly larger than the normal pomelo. SD is thus used as the determinant of pomelo quality. The principle of the proposed sensor system is based on phase difference between lower- and higher-frequency reflection coefficients (Φ_{12}). The sensor system consists of five dual-band antennas and a customized circuit to approximate Φ_{12} . The main components of the customized circuit are down conversion mixers and phase detector modules. Simulations were carried out using single- (1 GHz) and dual-frequency (1 GHz/2.3 GHz) sensor systems; and the results indicated that the single-frequency scheme achieved lower accuracy in classification of pomelo fruits. As a result, a prototype of a four beam dual-frequency sensor system was fabricated, and experiments were carried out using plastic pomelo models and real pomelo fruits.

2.2.6 Title: A SAR Image Classification Algorithm Based on Multi-Feature Polarimetric Parameters Using FOA and LS-SVM

Author: SHIYU LUO, KAMAL SARABANDI, LELAND PIERCE

Synthetic Aperture Radar (SAR) image classification algorithm based on multi-feature using Fruit Fly Optimization Algorithm (FOA) and Least Square Support Vector Machine (LS-SVM). First, pixel-based information derived from three elements of coherency matrix, six parameters obtained by $H/\alpha/A$ decomposition and Freeman decomposition techniques, and three polarimetric parameters including the total receive power (SPAN), pedestal height, and Radar Vegetation Index (RVI), as well as region-based information derived from eight texture parameters obtained by Grey Level Co-occurrence Matrix (GLCM) are combined to use as the features of land cover. Second, Kernel

Principal Component Analysis (KPCA) is used to reduce the dimensionality of the multi-feature data derived from the integration of the pixel-based and region-based information. Third, LS-SVM is used as the classifier in this study due to its fast solving speed and desirable classification capability. Since the input parameters of LS-SVM significantly affect the classification performance, we employ FOA to obtain the optimized input parameters. a classification method based on multi-feature using FOA and LS-SVM. The basic features for a SAR image are polarimetric parameters. Coherency matrix and six parameters obtained by $H/\alpha/A$ decomposition and Freeman decomposition, as well as the total receive power (SPAN), polarimetric pedestal height, and (RVI) as the features of land cover. In addition to pixel-based information from polarimetric parameters, region-based information is another significant feature for an image. GLCM is an efficient tool to obtain region-based information. Thus, eight representative textural parameters obtained by GLCM, which are CON, ASM, COR, ENT, IDM, MAX, DIS, and INV are introduced to obtain texture information. However, while the integration of multi-feature characterizes the ground object better, it results in the curse of dimensionality and feature redundancy.

2.2.7 Title: Classification Of Eligibility Consumption Of Manalagi Apple Fruit Varieties Using Backpropagation

Author: Muladi Muladi, Dyah Lestari, Dedy Tri Prasetyo

The quality of Manalagi apple fruit varieties prior to sorting them automatically. The sorting system used is still manually in fruit sorting with the help of human labor that takes a long time. The sorting process is slow therefore the sorted result is low in number and inconsistent in quality. The parameters used in the Manalagi apple classification are based on the skin color of the fruit. The color of the fruit skin will be taken with color features

with average RGB. In this study, the classification system is automatically developed for the Manalagi apple fruit with the extraction of the average RGB color feature with the Backpropagation algorithm. The purpose of the developed system is to distinguish the Manalagi apple fruit of worth consumption and not worth the consumption. The fruit classification is based on the average RGB color composition of the fruit skin. Systems developed using image processing such as image segmentation, color feature extraction with average RGB models and fruit quality classification decisions with the Backpropagation algorithm. The classification of Manalagi apple fruit using image processing and neural network based on the value of input fruit skin color. Changes in the skin of apple fruit that daily change the level of brightness become important in the classification process. Color changes in the skin of Manalagi apple fruit from the harvest of trees to begin to rot every fruit varies. The characteristic of Manalagi apple fruit skin worth consumption has RGB value Green color is more dominant.

CHAPTER 3

EXPERIMENTAL METHODS AND ALGORITHMS

3.1 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as

hardware. A hardware requirements list is often accompanied by a hardware compatibility list, especially in case of operating systems. The minimal hardware requirements are as follows,

1. PROCESSOR : PENTIUM IV / III.
2. RAM : minimum 2 GB
3. HARD DISK DRIVE : 80 GB.

3.2 SOFTWARE REQUIREMENTS

Software requirements deals with defining resource requirements and prerequisites that need to be installed on a computer to provide functioning of an application. These requirements need to be installed separately before the software is installed. The minimal software requirements are as follows,

1. IDE : ANACONDA WITH JUPYTER NOTEBOOK.
2. IDE : PYCHARM.
3. OPERATING SYSTEM : WINDOWS / LINUX.

3.3 PROPOSED SYSTEM

- The proposes a new and robust machine learning model based on a convolutional neural network (CNN) to automatically classify single cells in thin blood smears on standard microscope slides as either infected or uninfected. there is a trade-off to make: will we create a model that highly specializes in variety, or will we create a generic model that can be capable of recognizing some fruits.
- The proposed algorithms can automatically extract complex image fruits types and successfully detect fruits in different scenes. As input, we used whole images, so it was not necessary to perform any preprocessing or the fruits , samples of more number of images were collected that consisted of

different classes data. a different number of images is collected for each class that was classified into input images.

- We proposed a deep learning (dl) based fruits types classification method to prevent fruits. The deep learning method used in the study is the LeNet convolutional neural network (cnn). It is predicted that the success of the obtained results will increase if the cnn method is supported by adding extra feature extraction methods and classifying successful fruit types.

3.3.1 Advantages:

- To identify the fruits types used on artificial neural networks.
- It is the best model for deep learning technique to easily identify the fruit types.

3.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are

1. Economic feasibility
2. Technical feasibility
3. Social feasibility

3.5 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into

the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.6 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. A feasibility study evaluates the project's potential for success.

3.7 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.8 LANGUAGE DESCRIPTION

3.8.1 About Python Language

Python is an object-oriented programming language created by Guido Rossum in 1989. It is ideally designed for rapid prototyping of complex applications. It

has interfaces to many OS system calls and libraries and is extensible to C or C++. Many large companies use the Python programming language including NASA, Google, YouTube, BitTorrent, etc. Python programming is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science. Python has a deep focus on code readability & this class will teach you python from basics.

3.8.2 Python Programming Characteristics

- It provides rich data types and easier to read syntax than any other programming languages
- It is a platform independent scripted language with full access to operating system API's
- Compared to other programming languages, it allows more run-time flexibility
- It includes the basic text manipulation facilities of Perl and Awk
- A module in Python may have one or more classes and free functions
- Libraries in Python are cross-platform compatible with Linux, Macintosh, and Windows
- For building large applications, Python can be compiled to byte-code
- Python supports functional and structured programming as well as OOP
- It supports interactive mode that allows interacting Testing and debugging of snippets of code
- In Python, since there is no compilation step, editing, debugging and testing is fast.

3.8.3 Applications of Python Programming

3.8.3.1 Web Applications

You can create scalable Web Apps using frameworks and CMS (Content Management System) that are built on Python. Some of the popular platforms for creating Web Apps are: Django, Flask, Pyramid, Plone, Django CMS. Sites like Mozilla, Reddit, Instagram and PBS are written in Python.

3.8.3.2 Scientific and Numeric Computing

There are numerous libraries available in Python for scientific and numeric computing. There are libraries like: SciPy and NumPy that are used in general purpose computing. And, there are specific libraries like: EarthPy for earth science, AstroPy for Astronomy and so on. Also, the language is heavily used in machine learning, data mining and deep learning.

3.8.3.3 Creating software Prototypes

Python is slow compared to compiled languages like C++ and Java. It might not be a good choice if resources are limited and efficiency is a must. However, Python is a great language for creating prototypes. For example: You can use Pygame (library for creating games) to create your game's prototype first. If you like the prototype, you can use language like C++ to create the actual game.

3.8.3.4 Good Language to Teach Programming

Python is used by many companies to teach programming to kids and newbies. It is a good language with a lot of features and capabilities. Yet, it's one of the easiest languages to learn because of its simple easy-to-use syntax.

3.9 Python GUI Tkinter

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with

Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a tkinter app:

- ◆ Importing the module – tkinter
- ◆ Create the main window (container)
- ◆ Add any number of widgets to the main window
- ◆ Apply the event Trigger on the widgets.

Importing tkinter is the same as importing any other module in the Python code. Note that the name of the module in Python 2.x is ‘Tkinter’ and in Python 3.x it is ‘tkinter’. ‘tkinter’ also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes.

- ✓ pack() method: It organizes the widgets in blocks before placing in the parent widget.
- ✓ grid() method: It organizes the widgets in grid (table-like structure) before placing in the parent widget.
- ✓ place() method: It organizes the widgets by placing them on specific positions directed by the programmer.

3.10 About OpenCV Package

Python is a general purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability.

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules.

This gives us two advantages: first, our code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

And the support of Numpy makes the task easier. Numpy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases the number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this.

So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

3.11 FEATURES OF ANACONDA NAVIGATOR

3.11.1 ANACONDA

Anaconda is a free and open source, easy to install distribution of Python and R programming languages. Anaconda provides a working environment which is used for scientific computing, data science, statistical analysis and machine learning.

The latest distribution of Anaconda is Anaconda 5.3 and is released in October, 2018. It has the conda package, environment manager and a collection of 1000+ open source packages long with free community support.

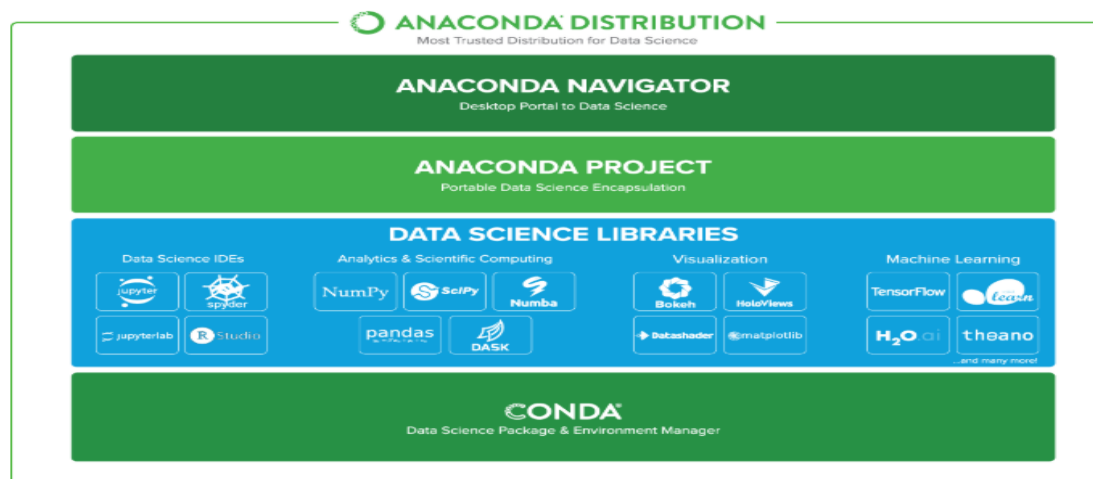


Fig 18: Anaconda distribution.

3.11.1.1 What is Anaconda Navigator?

Anaconda Navigator is a desktop graphical user interface (GUI) included in the Anaconda distribution. It allows us to launch applications provided in the Anaconda distribution and easily manage conda packages, environments and channels without the use of command-line commands. It is available for Windows, macOS and Linux.

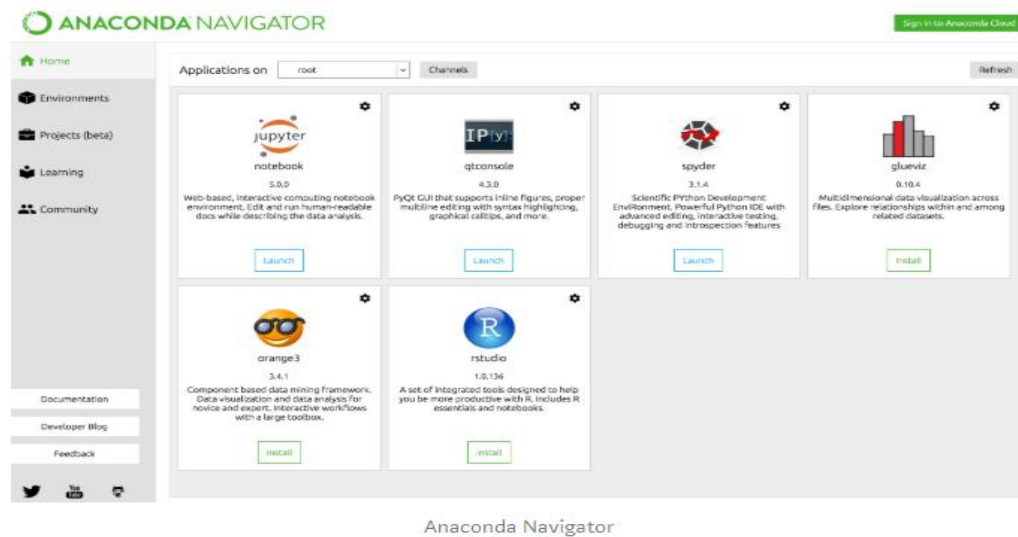


Fig 19:Anaconda navigator.

3.11.1.2 Applications Provided In Anaconda Distribution

The Anaconda distribution comes with the following applications along with Anaconda Navigator.

1. JupyterLab
2. Jupyter Notebook
3. Qt Console
4. Spyder
5. Glueviz
6. Orange3
7. RStudio
8. Visual Studio Code

- > **JupyterLab:** This is an extensible working environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
- > **Jupyter Notebook:** This is a web-based, interactive computing notebook environment. We can edit and run human-readable docs while describing the data analysis.
- > **Qt Console:** It is the PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips and more.
- > **Spyder:** Spyder is a scientific Python Development Environment. It is a powerful Python IDE with advanced editing, interactive testing, debugging and introspection features.
- > **VS Code:** It is a streamlined code editor with support for development operations like debugging, task running and version control.
- > **Glueviz:** This is used for multidimensional data visualization across files. It explores relationships within and among related datasets.
- > **Orange 3:** It is a component-based data mining framework. This can be used for data visualization and data analysis. The workflows in Orange 3 are very interactive and provide a large toolbox.
- > **Rstudio:** It is a set of integrated tools designed to help you be more productive with R. It includes R essentials and notebooks.

3.11.1.3 New Features of Anaconda 5.3

Compiled with Latest Python release: Anaconda 5.3 is compiled with Python 3.7, taking advantage of Python's speed and feature improvements.

- **Better Reliability:** The reliability of Anaconda has been improved in the latest release by capturing and storing the package metadata for installed packages.

- **Enhanced CPU Performance:** The Intel Math Kernel Library 2019 for Deep Neural Networks(MKL 2019) has been introduced in Anaconda 5.3 distribution. Users deploying Tensorflow can make use of MKL 2019 for Deep Neural Networks. These Python binary packages are provided to achieve high CPU performance.

- **New packages are added:** There are over 230 packages which has been updated and added in the new release.

- **Work in Progress:** There is a casting bug in Numpy with Python 3.7 but the team is currently working on patching it until Numpy is updated.

3.12 SYSTEM DESIGN

3.12.1 SYSTEM ARCHITECTURE

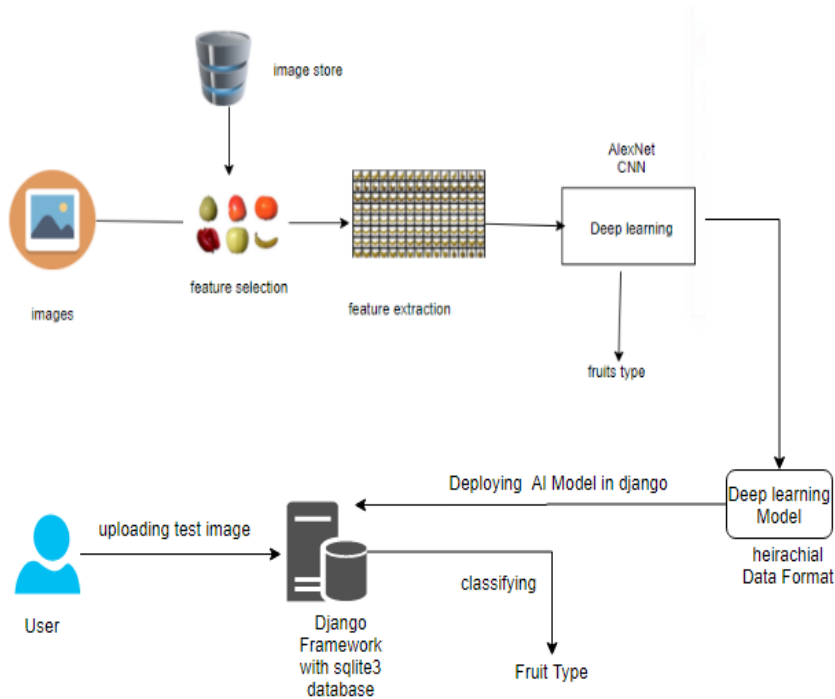


Fig 20: System Architecture.

3.13 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

- The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML consists of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.
- The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of

software system, as well as for business modeling and other non-software systems.

- The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.
- The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

3.13.1 GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

3.13.2 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any

dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

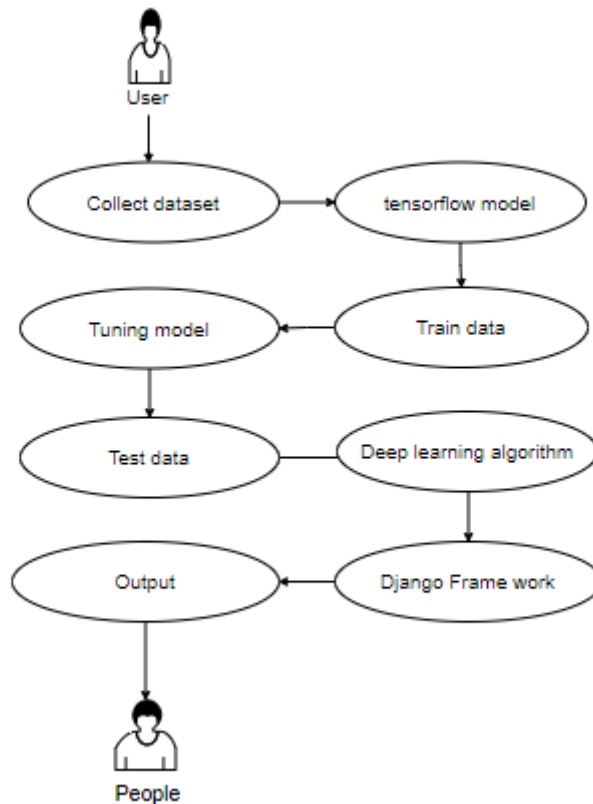


Fig 21: Use case diagram.

3.13.3 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

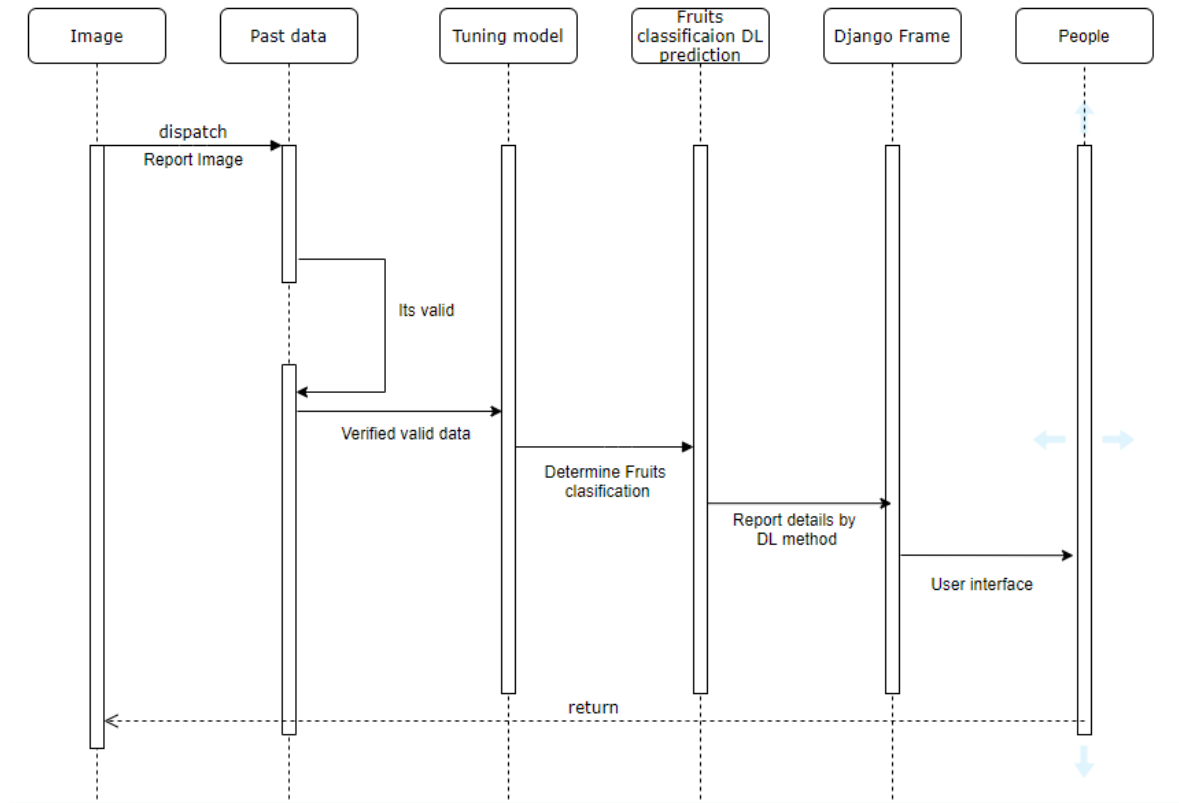


Fig 22: Sequence diagram.

3.13.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

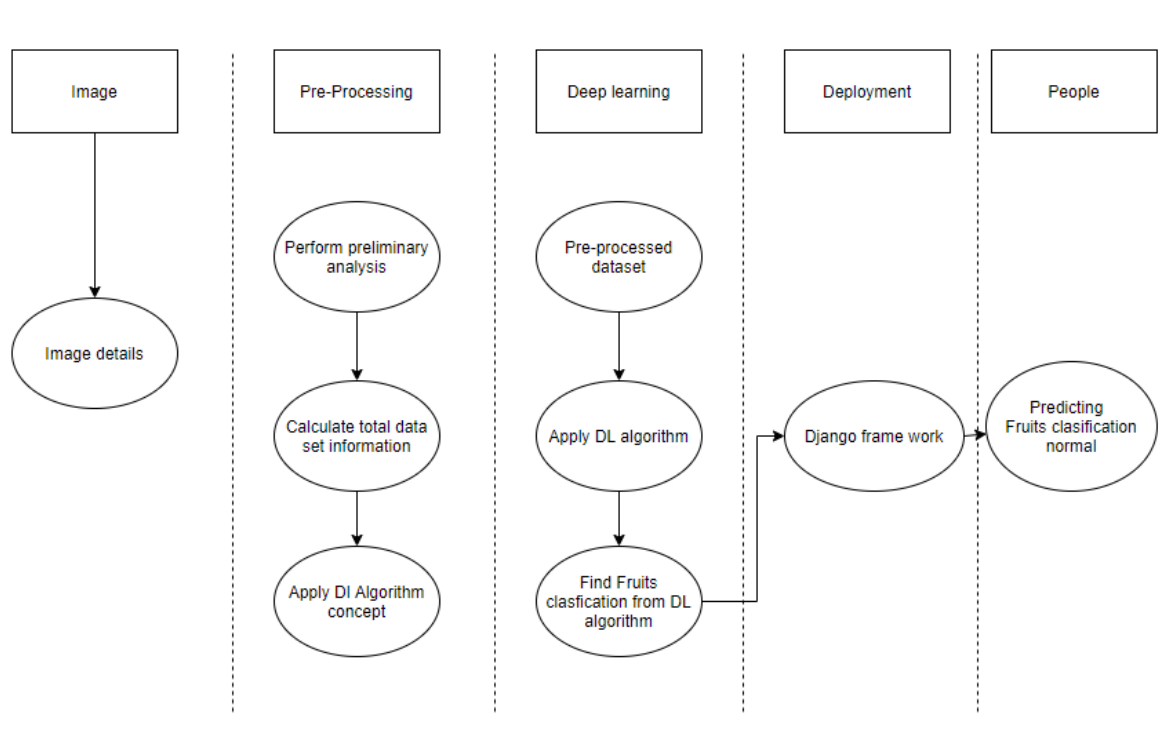


Fig 23: Activity diagram.

3.14 METHODOLOGIES

3.14.1 Training from Scratch

To train a deep network from scratch, you gather a very large labeled data set and design a network architecture that will learn the features and model. This is good for new applications, or applications that will have a large number of output categories. This is a less common approach because with the large amount of data and rate of learning, these networks typically take days or weeks to train.

3.14.2 Transfer Learning

Most deep learning applications use the transfer learning approach, a process that involves fine-tuning a pretrained model. You start with an existing network, such as AlexNet or GoogLeNet, and feed in new data containing

previously unknown classes. After making some tweaks to the network, you can now perform a new task, such as categorizing only dogs or cats instead of 1000 different objects. This also has the advantage of needing much less data (processing thousands of images, rather than millions), so computation time drops to minutes or hours

3.14.3 Feature Extraction

A slightly less common, more specialized approach to deep learning is to use the network as a **feature extractor**. Since all the layers are tasked with learning certain features from images, we can pull these features out of the network at any time during the training process.

3.14.4 Algorithms:

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

In a word, accuracy. Deep learning achieves recognition accuracy at higher levels than ever before. This helps consumer electronics meet user expectations, and it is crucial for safety-critical applications like driverless cars. Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks like classifying objects in images.

1. Deep learning requires large amounts of labeled data. driverless car development requires millions of images and thousands of hours of video.
2. Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

Most deep learning methods use neural network architectures, which is why deep learning models are often referred to as deep neural networks.

The term “deep” usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150.

Deep learning models are trained by using large sets of labeled data and neural network architectures that learn features directly from the data without the need for manual feature extraction

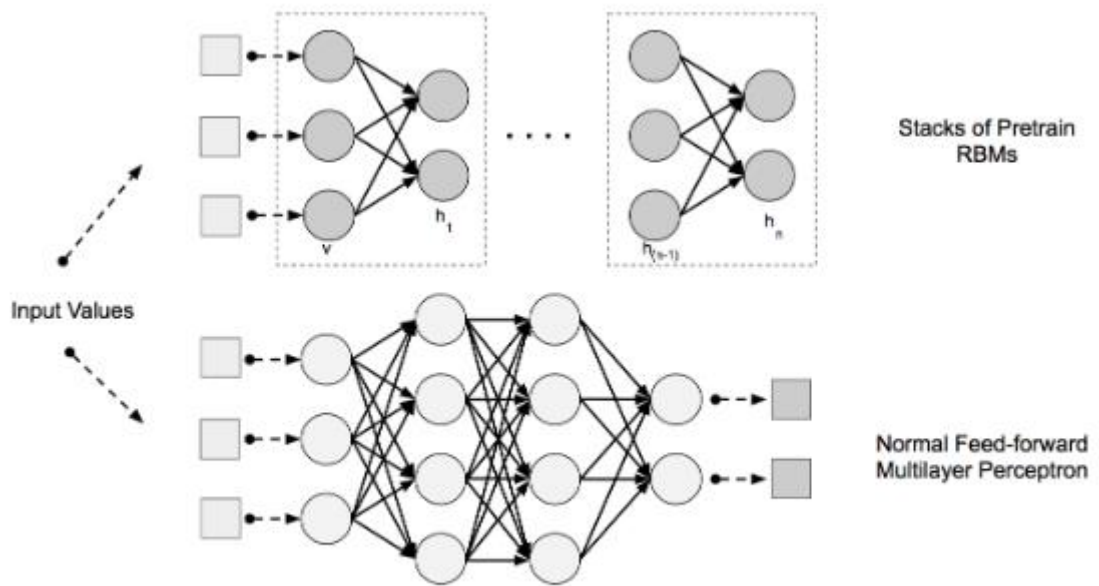


Fig 24:Convolutional neural network.

One of the most popular types of deep neural networks is known as convolutional neural networks (CNN or ConvNet). A CNN convolves learned features with input data, and uses 2D convolutional layers, making this architecture well suited to processing 2D data, such as images.

CNNs eliminate the need for manual feature extraction , so you do not need to identify features used to classify images. CNN works by extracting features directly from images. The relevant features are not retrained; they are learned while the network trains on a collection of images. This automated feature extraction makes deep learning models highly accurate for computer vision tasks such as object classification.

CNNs learn to detect different features of an image using tens or hundreds of hidden layers. Every hidden layer increases the complexity of the learned image features. For example, the first hidden layer could learn how to detect edges, and the last learns how to detect more complex shapes specifically catered to the shape of the object we are trying to recognize.

3.15 TESTING

3.15.1 Software Testing

In a generalized way, we can say that the system testing is a type of testing in which the main aim is to make sure that the system performs efficiently and seamlessly. The process of testing is applied to a program with the main aim to discover an unprecedented error, an error which otherwise could have damaged the future of the software. Test cases which bring up a high possibility of discovery and error are considered successful. This successful test helps to answer the still unknown errors.

3.15.2 TEST CASE

Testing, as already explained earlier, is the process of discovering all possible weak-points in the finalized software product. Testing helps to counter the working of sub-assemblies, components, assembly and the complete result. The software is taken through different exercises with the main aim of making sure that software meets the business requirement and user-expectations and doesn't fail abruptly. Several types of tests are used today. Each test type addresses a specific testing requirement.

3.15.3 Testing Techniques

A test plan is a document which describes approach, its scope, its resources and the schedule of aimed testing exercises. It helps to identify almost other test item, the features which are to be tested, its tasks, how will everyone do each task, how much the tester is independent, the environment in which the test is taking place, its technique of design plus the both the end criteria which is used, also rational of choice of theirs, and whatever kind of risk which requires emergency planning. It can be also referred to as the record of the process of test planning. Test plans are usually prepared with significant input from test engineers.

3.15.3.1 UNIT TESTING

In unit testing, the design of the test cases is involved that helps in the validation of the internal program logic. The validation of all the decision branches and internal code takes place. After the individual unit is completed it takes place. Plus it is taken into account after the individual unit is completed before integration. The unit test thus performs the basic level test at its component stage and test the particular business process, system configurations etc. The unit test ensures that the particular unique path of the process gets performed precisely to the documented specifications and contains clearly defined inputs with the results which are expected.

3.15.3.2 INTEGRATION TESTING

These tests are designed to test the integrated software items to determine whether they really execute as a single program or application. The testing is event driven and thus is concerned with the basic outcome of the field. The Integration tests demonstrate that the components were individually satisfied, as already represented by successful unit testing, the components are apt and fine. This type of testing is specially aimed to expose the issues that come-up by the components combination.

3.15.3.3 FUNCTIONAL TESTING

The functional tests help in providing the systematic representation that functions tested are available and specified by technical requirement, documentation of the system and the user manual.

3.15.3.4 SYSTEM TESTING

System testing, as the name suggests, is the type of testing in which ensures that the software system meets the business requirements and aims. Testing of

the configuration is taken place here to ensure predictable results and thus analysis of it. System testing is relied on the description of process and its flow, stressing on pre driven process and the points of integration.

3.15.3.5 WHITE BOX TESTING

The white box testing is the type of testing in which the internal components of the system software is open and can be processed by the tester. It is therefore a complex type of testing process. All the data structure, components etc. are tested by the tester himself to find out a possible bug or error. It is used in situations in which the black box is incapable of finding out a bug. It is a complex type of testing which takes more time to get applied.

3.15.3.6 BLACK BOX TESTING

The black box testing is the type of testing in which the internal components of the software are hidden and only the input and output of the system is the key for the tester to find out a bug. It is therefore a simple type of testing. A programmer with basic knowledge can also process this type of testing. It is less time consuming as compared to the white box testing. It is very successful for software which are less complex and straight-forward in nature. It is also less costly than white box testing.

3.15.3.7 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

CHAPTER 4

RESULT AND DISCUSSION

In this project finally we are comparing the dataset with accuracy and loss of the various algorithms. With CNN and Keras we get almost accurate image classification. We get the accurate image classification only when the system is trained well. The accuracy of the system will increase when the system is trained by more number of epochs.

```
WARNING:tensorflow:From <ipython-input-25-08889a13dbb5>:5: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
WARNING:tensorflow:sample_weight modes were coerced from
...
to
['...']
WARNING:tensorflow:sample_weight modes were coerced from
...
to
['...']
Train for 144 steps, validate for 144 steps
Epoch 1/4
144/144 [=====] - 183s 1s/step - loss: 3.6886 - accuracy: 0.0491 - val_loss: 3.2910 - val_accuracy: 0.0645
Epoch 2/4
144/144 [=====] - 50s 348ms/step - loss: 3.2891 - accuracy: 0.0651 - val_loss: 3.2864 - val_accuracy: 0.0645
Epoch 3/4
144/144 [=====] - 47s 330ms/step - loss: 3.2852 - accuracy: 0.0646 - val_loss: 3.2831 - val_accuracy: 0.0645
Epoch 4/4
144/144 [=====] - 50s 344ms/step - loss: 3.2822 - accuracy: 0.0651 - val_loss: 3.2804 - val_accuracy: 0.0645
```

Fig 25: Training the system.

In the above image the loss of the system is decreasing for every epoch and the accuracy is increasing for every epoch. This accuracy is evaluated for testing and training modules of the system.

The next module of the project is to implement the model with keras and tensorflow. This uses different algorithms to process the image for further classification. It will reduce the image size by using algorithms and then classify the image.

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 23, 23, 96)	34944
activation (Activation)	(None, 23, 23, 96)	0
max_pooling2d (MaxPooling2D)	(None, 11, 11, 96)	0
flatten (Flatten)	(None, 11616)	0
dense (Dense)	(None, 500)	5808500
activation_1 (Activation)	(None, 500)	0
dropout (Dropout)	(None, 500)	0
dense_1 (Dense)	(None, 250)	125250
activation_2 (Activation)	(None, 250)	0
dropout_1 (Dropout)	(None, 250)	0
dense_2 (Dense)	(None, 112)	28112
activation_3 (Activation)	(None, 112)	0
dropout_2 (Dropout)	(None, 112)	0
dense_3 (Dense)	(None, 27)	3051
activation_4 (Activation)	(None, 27)	0

Total params: 5,999,857		
Trainable params: 5,999,857		
Non-trainable params: 0		

Fig 26: Processing through CNN layers.

In this module the accuracy and loss of the system is almost the same as the training and testing process.

```
[INFO] Calculating model accuracy
WARNING:tensorflow:sample_weight modes were coerced from
...
to
['...']
145/145 [=====] - 12s 84ms/step - loss: 0.2445 - accuracy: 0.9328
Test Accuracy: 93.27840209007263
```

Fig 27: Calculating model accuracy.

4.1 COMPARISON GRAPH:

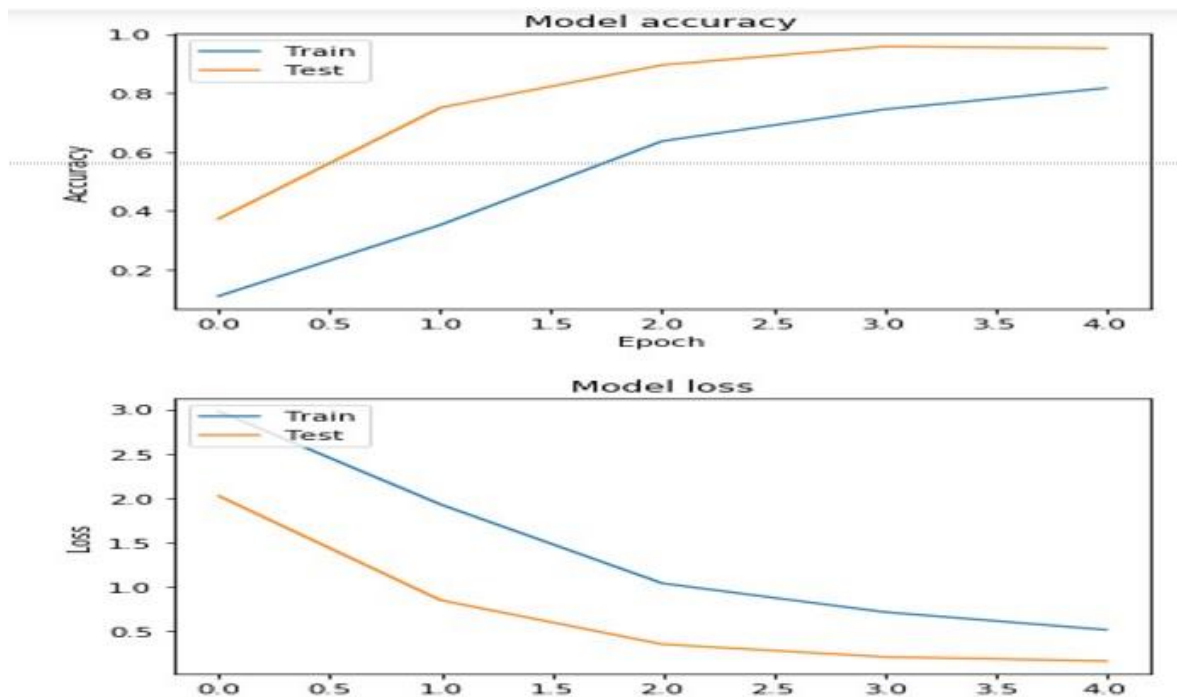


Fig 28:Graph for accuracy and loss.

CHAPTER 5

CONCLUSION AND ENHANCEMENT

5.1 CONCLUSION

Thus the system is developed for classifying the fruits of the same variety. The fruits are being classified manually in many places and this system will be useful for the fruit classification. This system is developed to make the classification process accurate and this system will also make the system to classify the fruits based on the image rather than the exact physical measurements.

5.2 FUTURE WORK

- To deploy real time this process by showing the prediction result in web application or desktop application.
- To optimize the work to implement in the Artificial Intelligence environment.
- To deploy this model to AI on a web application.

5.3 REFERENCES

[1].<https://analyticsindiamag.com/why-convolutional-neural-networks-are-the-go-to-models-in-deep-learning/#:~:text=The%20main%20advantage%20of%20CNN,for%20each%20class%20by%20itself.>

[2].<https://www.quora.com/What-are-the-advantages-of-a-convolutional-neural-network-CNN-compared-to-a-simple-neural-network-from-the-theoretical-and-practical-perspective.>

[3].https://www.researchgate.net/figure/Advantages-and-disadvantages-of-different-CNN-models_tbl1_335036812.

[4].<https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529>.

[5].<https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/>.

[6]. <https://www.kaggle.com/keshavramaiah/fruit-classification-using-cnn>.

[7]. <https://github.com/garodisk/Fruits-classification-120-different-fruits->.

[8]. <https://github.com/ashishT1712/CNN-Fruits-Classifier>.

[9].<https://www.machinecurve.com/index.php/2020/04/08/how-to-perform-fruit-classification-with-deep-learning-in-keras/>.

[10].<https://www.danielpradilla.info/blog/classifying-convolutional-neural-network-keras/>.

[11]. <https://www.kaggle.com/moltean/fruits>.

[12]. Abadi, M., Agarwal, A., Barham, P., Goodfellow, I., et. al.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015).

[13].Bargoti, S., Underwood, J.: Deep fruit detection in orchards. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 3626–3633 (2017)

- [14].Femling, F., Olsson, A., Alonso-Fernandez, F.: Fruit and vegetable identification using machine learning for retail applications. In: 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), pp. 9–15. IEEE (2018)
- [15].Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
- [16].Hossain, M.S., Al-Hammadi, M., Muhammad, G.: Automatic fruit classification using deep learning for industrial applications. IEEE Trans. Ind. Inf. **15**(2), 1027–1034 (2018).

5.4 APPENDICES

5.4.1 SOURCE CODE

```
import os
import numpy as np # linear algebra
import matplotlib.pyplot as plt
# Dl framework - tensorflow, keras a backend
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Input, Dense, Flatten, Dropout,
BatchNormalization
from tensorflow.keras.layers import Conv2D, SeparableConv2D,
MaxPool2D, LeakyReLU, Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint,
ReduceLROnPlateau, EarlyStopping
from IPython.display import display
from os import listdir
from os.path import isfile, join
from PIL import Image
import glob
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
import warnings
```

```

warnings.filterwarnings('ignore')
dir_name_train_Apple_Braeburn= 'fruits-360/train/Apple Braeburn'
dir_name_train_Banana = 'fruits-360/train/Banana'
dir_name_train_Blueberry = 'fruits-360/train/Blueberry'
dir_name_train_Corn = 'fruits-360/train/Corn'
dir_name_train_Dates = 'fruits-360/train/Dates'
dir_name_train_Kiwi = 'fruits-360/train/Kiwi'
dir_name_train_Mango = 'fruits-360/train/Mango'
def plot_images(item_dir, n=6):
    all_item_dir = os.listdir(item_dir)
    item_files = [os.path.join(item_dir, file) for file in all_item_dir][:n]

    plt.figure(figsize=(35, 10))
    for idx, img_path in enumerate(item_files):
        plt.subplot(2, n, idx+1)
        img = plt.imread(img_path)
        plt.imshow(img, cmap='gray')
        plt.axis('off')

    plt.tight_layout()
def Images_details_Print_data(data, path):
    print(" ===== Images in: ", path)
    for k, v in data.items():
        print("%s:\t%s" % (k, v))

def Images_details(path):
    files = [f for f in glob.glob(path + "**/*.*", recursive=True)]
    data = {}
    data['images_count'] = len(files)

```

```

data['min_width'] = 10**100 # No image will be bigger than that
data['max_width'] = 0
data['min_height'] = 10**100 # No image will be bigger than that
data['max_height'] = 0

for f in files:
    im = Image.open(f)
    width, height = im.size
    data['min_width'] = min(width, data['min_width'])
    data['max_width'] = max(width, data['max_width'])
    data['min_height'] = min(height, data['min_height'])
    data['max_height'] = max(height, data['max_height'])

Images_details_Print_data(data, path)
print("")
print("Trained data for Apple Braeburn:")
print("")
Images_details(dir_name_train_Apple_Braeburn)
print("")
plot_images(dir_name_train_Apple_Braeburn, 10)
print("")
print("Trained data for dir_name_train_Banana:")
print("")
Images_details(dir_name_train_Banana)
print("")
plot_images(dir_name_train_Banana, 10)
print("")
print("Trained data for dir_name_train_Blueberry:")

```

```

print("")
Images_details(dir_name_train_Blueberry)
print("")
plot_images(dir_name_train_Blueberry, 10)
print("")
print("Trained data for dir_name_train_Corn:")
print("")
Images_details(dir_name_train_Corn)
print("")
plot_images(dir_name_train_Corn, 10)
Classifier=Sequential()
Classifier.add(Convolution2D(32,(3,3),input_shape=(100,100,3),activation='r
elu'))
Classifier.add(MaxPooling2D(pool_size=(2,2)))
Classifier.add(Flatten())
Classifier.add(Dense(29, activation='relu'))
Classifier.add(Dense(29, activation='softmax'))
Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metric
s=['accuracy'])
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_ra
nge=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
training_set=train_datagen.flow_from_directory('fruits-
360/train',target_size=(100,100),batch_size=32,class_mode='categorical')
test_set=test_datagen.flow_from_directory('fruits-
360/test',target_size=(100,100),batch_size=32,class_mode='categorical')
img_dims = 150
epochs = 4
batch_size = 32

```

```

#### Fitting the model
history = Classifier.fit_generator(
    training_set, steps_per_epoch=training_set.samples // batch_size,
    epochs=epochs,
    validation_data=test_set, validation_steps=test_set.samples //
batch_size)
def graph():
    #Plot training & validation accuracy values
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Model accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

    # Plot training & validation loss values
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

```

Module:2

```

import os
import numpy as np # linear algebra
import matplotlib.pyplot as plt

```

```

# Dl framwork - tensorflow, keras a backend
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Input, Dense, Flatten, Dropout,
BatchNormalization
from tensorflow.keras.layers import Conv2D, SeparableConv2D,
MaxPool2D, LeakyReLU, Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint,
ReduceLROnPlateau, EarlyStopping
from IPython.display import display
from os import listdir
from os.path import isfile, join
from PIL import Image
import glob
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
Classifier.add(Convolution2D(32,3,3,input_shape=(100,100,3),activation='relu'))
Classifier.add(MaxPooling2D(pool_size=(2,2)))
Classifier.add(Convolution2D(128,3,3,activation='relu'))
Classifier.add(MaxPooling2D(pool_size=(2,2)))
Classifier.add(Flatten())
Classifier.add(Dense(256, activation='relu'))

```

```

Classifier.add(Dense(27, activation='softmax'))

Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])
Classifier.summary()
Classifier.add(Dense(29, activation='softmax'))
Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
training_set=train_datagen.flow_from_directory('fruits-360/train',target_size=(100,100),batch_size=32,class_mode='categorical')
test_set=test_datagen.flow_from_directory('fruits-360/test',target_size=(100,100),batch_size=32,class_mode='categorical')
img_dims = 150
epochs = 4
batch_size = 32
#### Fitting the model
history = Classifier.fit_generator(
    training_set, steps_per_epoch=training_set.samples // batch_size,
    epochs=epochs,
    validation_data=test_set,validation_steps=test_set.samples //
batch_size)
def graph():
    #Plot training & validation accuracy values
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Model accuracy')

```

```
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

```
# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

Module:3

```
import os

import numpy as np # linear algebra
import matplotlib.pyplot as plt

# Dl framwork - tensorflow, keras a backend
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Input, Dense, Flatten, Dropout,
BatchNormalization
from tensorflow.keras.layers import Conv2D, SeparableConv2D,
MaxPool2D, LeakyReLU, Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```

from tensorflow.keras.callbacks import ModelCheckpoint,
ReduceLROnPlateau, EarlyStopping
from IPython.display import display
from os import listdir
from os.path import isfile, join
from PIL import Image
import glob
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
model = Sequential()
# 1st Convolutional Layer
model.add(Conv2D(filters=96, input_shape=(100,100,3),
kernel_size=(11,11), strides=(4,4), padding='valid'))
model.add(Activation('relu'))
# Max Pooling
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))

model.add(Flatten())
# 1st Fully Connected Layer
model.add(Dense(500, input_shape=(224*224*3,)))
model.add(Activation('relu'))
# Add Dropout to prevent overfitting
model.add(Dropout(0.4))
# 2nd Fully Connected Layer
model.add(Dense(250))
model.add(Activation('relu'))

```

```

# Add Dropout
model.add(Dropout(0.4))
# 3rd Fully Connected Layer
model.add(Dense(112))
model.add(Activation('relu'))
# Add Dropout
model.add(Dropout(0.4))
# Output Layer
model.add(Dense(29))
model.add(Activation('softmax'))
model.summary()

# Compile the model
model.compile(loss = 'categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metric
s=['accuracy'])
Classifier.summary()
Classifier.add(Dense(29, activation='softmax'))
Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metric
s=['accuracy'])
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_ra
nge=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
training_set=train_datagen.flow_from_directory('fruits-
360/train',target_size=(100,100),batch_size=32,class_mode='categorical')
test_set=test_datagen.flow_from_directory('fruits-
360/test',target_size=(100,100),batch_size=32,class_mode='categorical')
img_dims = 150

```

```

epochs = 4
batch_size = 32
#### Fitting the model
history = Classifier.fit_generator(
    training_set, steps_per_epoch=training_set.samples // batch_size,
    epochs=epochs,
    validation_data=test_set, validation_steps=test_set.samples //
batch_size)
import h5py
import tensorflow as tf
Classifier.save('fruit.h5')
from tensorflow.keras.models import load_model
classifierLoad = tf.keras.models.load_model('fruit.h5')
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing import image
test_image=image.load_img('ba_r.jpg',target_size=(100,100))
import matplotlib.pyplot as plt
img = plt.imshow(test_image)
test_image=image.img_to_array(test_image)
test_image=np.expand_dims(test_image,axis=0)
classes=[
'Apple Braeburn',
'Apple Golden 2',
'Banana',
'Banana Red',
'Beetroot',
'Blueberry',
'Cactus fruit',

```

```

'Cantaloupe 1',
'Cocos',
'Corn',
'Dates',
'Ginger Root',
'Grape White',
'Guava',
'Kiwi',
'Lemon',
'Mango',
'Onion Red',
'Orange',
'Pepper Green',
'Pineapple',
'Pitahaya Red',
'Potato White',
'Rambutan',
'Strawberry',
'Tomato Yellow',
'Watermelon']
output=zip(classes,prediction)
output=dict(output)
output
if output['Apple Braeburn']==1.0 :
    print("Apple Braeburn")
elif output['Apple Golden 2']==1.0:
    print("Apple Golden 2")
elif output['Banana']==1.0:
    print("Banana")

```

```
elif output['Banana Red']==1.0:
    print("Banana Red")
elif output['Beetroot']==1.0:
    print("Beetroot")
elif output['Blueberry']==1.0:
    print("Blueberry")
elif output['Cactus fruit']==1.0:
    print("Cactus fruit")
elif output['Cantaloupe 1']==1.0:
    print("Cantaloupe 1")
elif output['Cocos']==1.0:
    print("Cocos")
elif output['Corn']==1.0:
    print("Corn")
elif output['Dates']==1.0:
    print("Dates")
elif output['Ginger Root']==1.0:
    print("Ginger Root")
elif output['Grape White']==1.0:
    print("Grape White")
elif output['Guava']==1.0:
    print("Guava")
elif output['Kiwi']==1.0:
    print("Kiwi")
elif output['Lemon']==1.0:
    print("Lemon")
elif output['Mango']==1.0:
    print("Mango")
elif output['Onion Red']==1.0:
```

```
    print("Onion Red")
elif output['Orange']==1.0:
    print("Orange")
elif output['Pepper Green']==1.0:
    print("Pepper Green")
elif output['Pineapple']==1.0:
    print("Pineapple")
elif output['Pitahaya Red']==1.0:
    print("Pitahaya Red")
elif output['Potato White']==1.0:
    print("Potato White")
elif output['Rambutan']==1.0:
    print("Rambutan")
elif output['Strawberry']==1.0:
    print("Strawberry")
elif output['Tomato Yellow']==1.0:
    print("Tomato Yellow")
elif output['Watermelon']==1.0:
    print("Watermelon")
```

5.4.2 SCREENSHOTS

5.4.2.1 Software involvement steps:

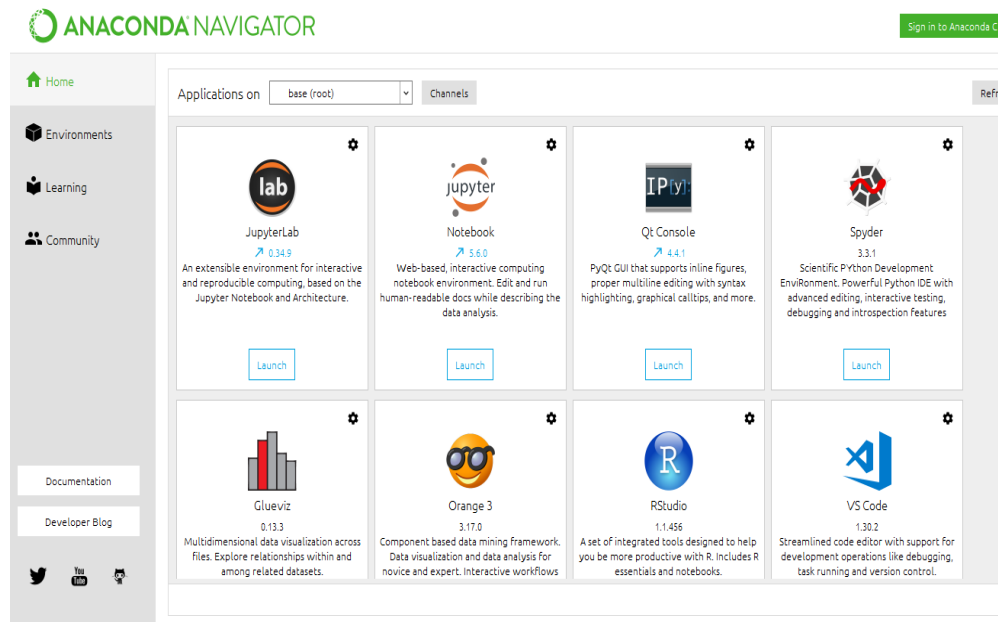


Fig 29: Open the anaconda navigator.

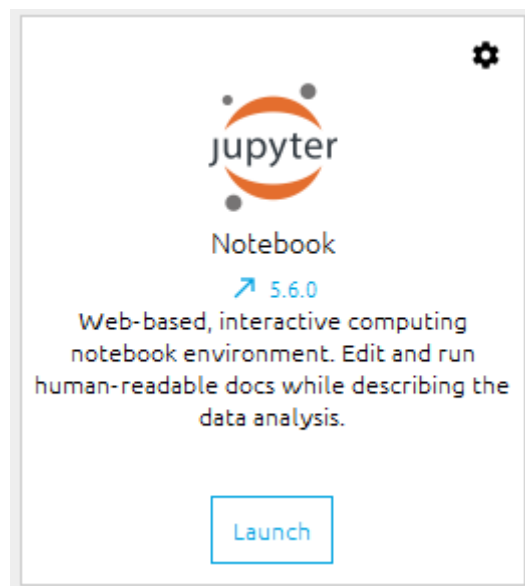


Fig 30: Launch the jupyter notebook platform.



Fig 31: Open the correspondent result folder.

.ipynb_checkpoints	✓	2/26/2021 1:20 PM	File folder	
Deploy	✓	2/26/2021 3:48 PM	File folder	
fruits-360	✓	2/26/2021 1:25 PM	File folder	
apple	✓	5/18/2020 10:39 AM	JPG File	5 KB
ba	✓	5/18/2020 10:39 AM	JPG File	4 KB
fruit.h5	✓	2/5/2021 4:34 PM	H5 File	1,414 KB
fruit.ipynb	✓	3/4/2021 12:01 PM	IPYNB File	1,508 KB
fruit_2.ipynb	✓	3/4/2021 12:17 PM	IPYNB File	42 KB
fruit_3.ipynb	✓	3/4/2021 12:53 PM	IPYNB File	21 KB
fruit1.h5	✓	3/4/2021 12:52 PM	H5 File	1,400 KB

Fig 32: Run the corresponding IPYNB file.

```

Trained data for Apple Braeburn:

===== Images in: fruits-360/train/Apple Braeburn
images_count: 255
min_width: 100
max_width: 100
min_height: 100
max_height: 100

```

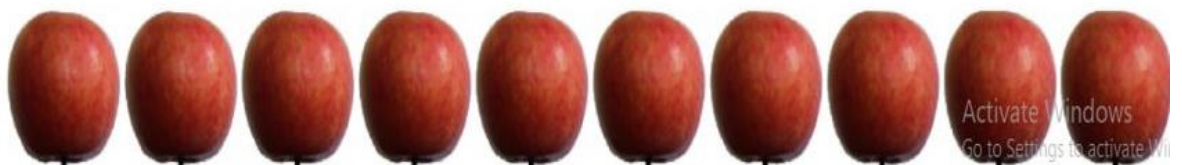


Fig 33: Training the system with a type of apple.(example).

```

WARNING:tensorflow:From <ipython-input-25-08889a13dbb5>:5: Model.fit_generator (from tensorflow.python.keras.engine.training) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
WARNING:tensorflow:sample_weight modes were coerced from
...
to
['...']
WARNING:tensorflow:sample_weight modes were coerced from
...
to
['...']
Train for 144 steps, validate for 144 steps
Epoch 1/4
144/144 [=====] - 183s 1s/step - loss: 3.6886 - accuracy: 0.0491 - val_loss: 3.2910 - val_accuracy: 0.
0645
Epoch 2/4
144/144 [=====] - 50s 348ms/step - loss: 3.2891 - accuracy: 0.0651 - val_loss: 3.2864 - val_accuracy:
0.0645
Epoch 3/4
144/144 [=====] - 47s 330ms/step - loss: 3.2852 - accuracy: 0.0646 - val_loss: 3.2831 - val_accuracy:
0.0645
Epoch 4/4
144/144 [=====] - 50s 344ms/step - loss: 3.2822 - accuracy: 0.0651 - val_loss: 3.2804 - val_accuracy:
0.0645

```

Fig 34: Accuracy and loss for training and testing.

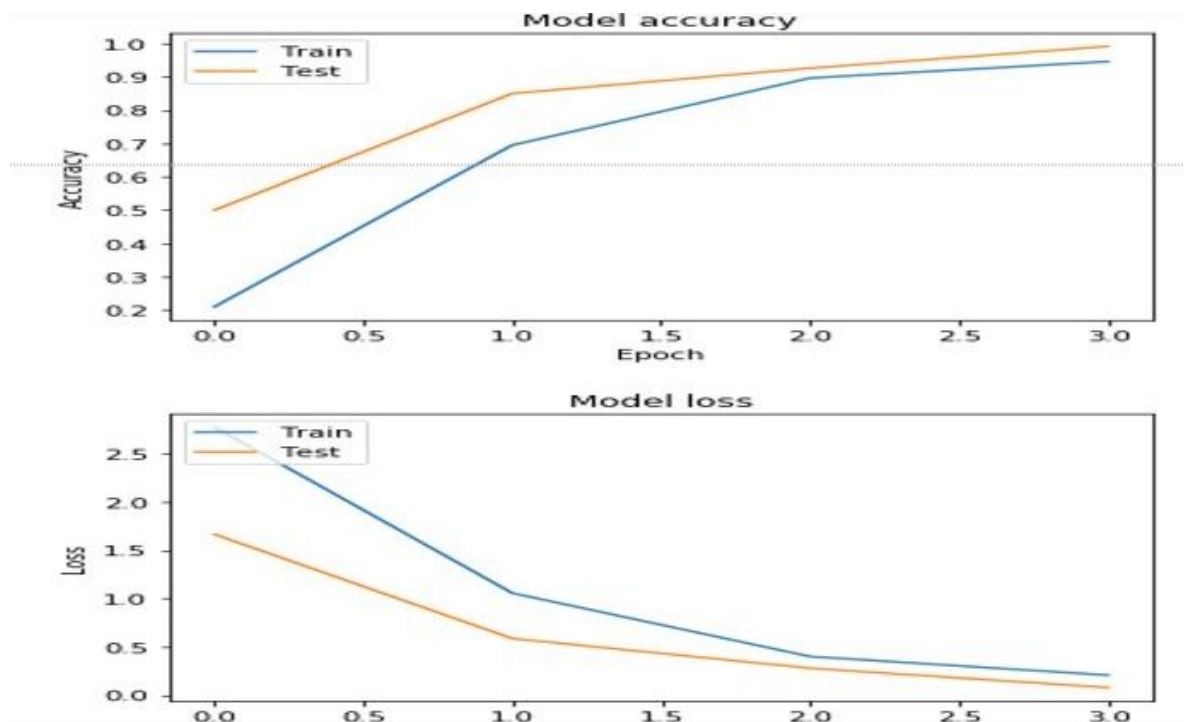


Fig 35: Graphical representation of accuracy and loss for training and testing.

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 23, 23, 96)	34944
activation (Activation)	(None, 23, 23, 96)	0
max_pooling2d (MaxPooling2D)	(None, 11, 11, 96)	0
flatten (Flatten)	(None, 11616)	0
dense (Dense)	(None, 500)	5808500
activation_1 (Activation)	(None, 500)	0
dropout (Dropout)	(None, 500)	0
dense_1 (Dense)	(None, 250)	125250
activation_2 (Activation)	(None, 250)	0
dropout_1 (Dropout)	(None, 250)	0
dense_2 (Dense)	(None, 112)	28112
activation_3 (Activation)	(None, 112)	0
dropout_2 (Dropout)	(None, 112)	0
dense_3 (Dense)	(None, 27)	3051
activation_4 (Activation)	(None, 27)	0

Total params: 5,999,857		
Trainable params: 5,999,857		
Non-trainable params: 0		

Fig 36: Implementation using tensorflow and keras.

```

WARNING:tensorflow:sample_weight modes were coerced from
...
to
['...']
WARNING:tensorflow:sample_weight modes were coerced from
...
to
['...']
Train for 144 steps, validate for 144 steps
Epoch 1/5
144/144 [=====] - 45s 310ms/step - loss: 3.2544 - accuracy: 0.0762 - val_loss: 2.9534 - val_accuracy:
0.1523
Epoch 2/5
144/144 [=====] - 45s 314ms/step - loss: 2.5804 - accuracy: 0.2120 - val_loss: 1.5976 - val_accuracy:
0.4826
Epoch 3/5
144/144 [=====] - 43s 301ms/step - loss: 1.7298 - accuracy: 0.4164 - val_loss: 0.9239 - val_accuracy:
0.7431
Epoch 4/5
144/144 [=====] - 43s 301ms/step - loss: 1.2286 - accuracy: 0.5677 - val_loss: 0.5657 - val_accuracy:
0.8678
Epoch 5/5
144/144 [=====] - 43s 300ms/step - loss: 0.9156 - accuracy: 0.6683 - val_loss: 0.3597 - val_accuracy:
0.8917

```

Fig 37: Accuracy and loss for implementation using keras and tensorflow.

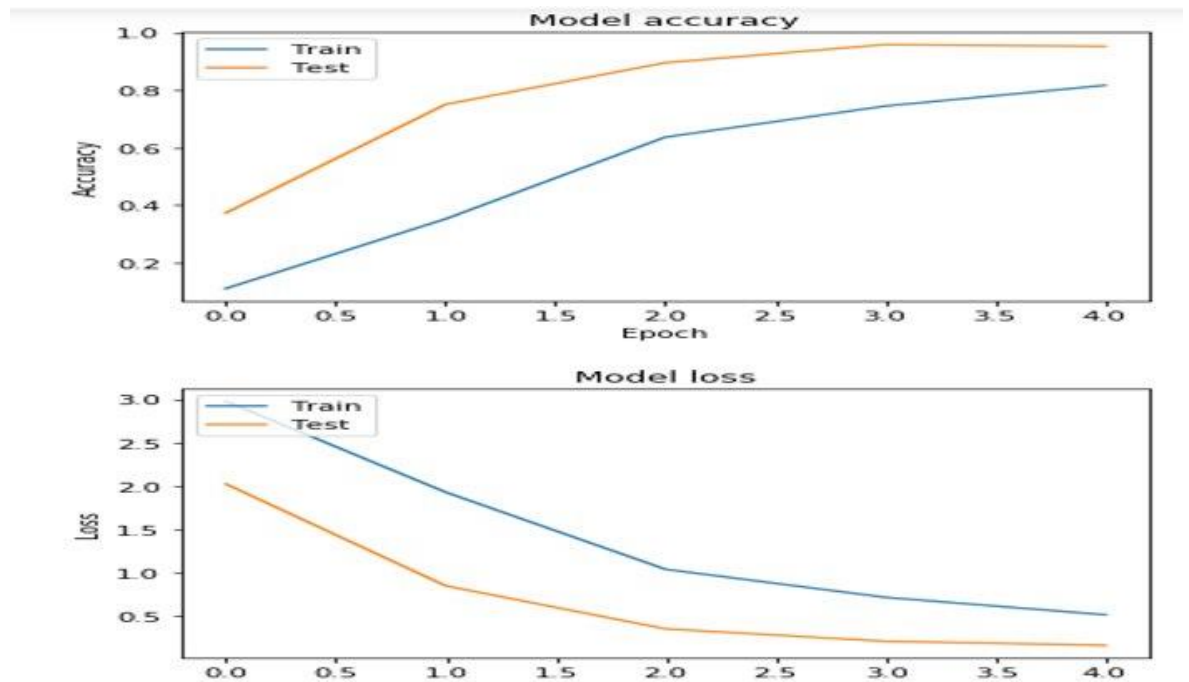


Fig 38: Graphical representation for implementation using keras and tensorflow.

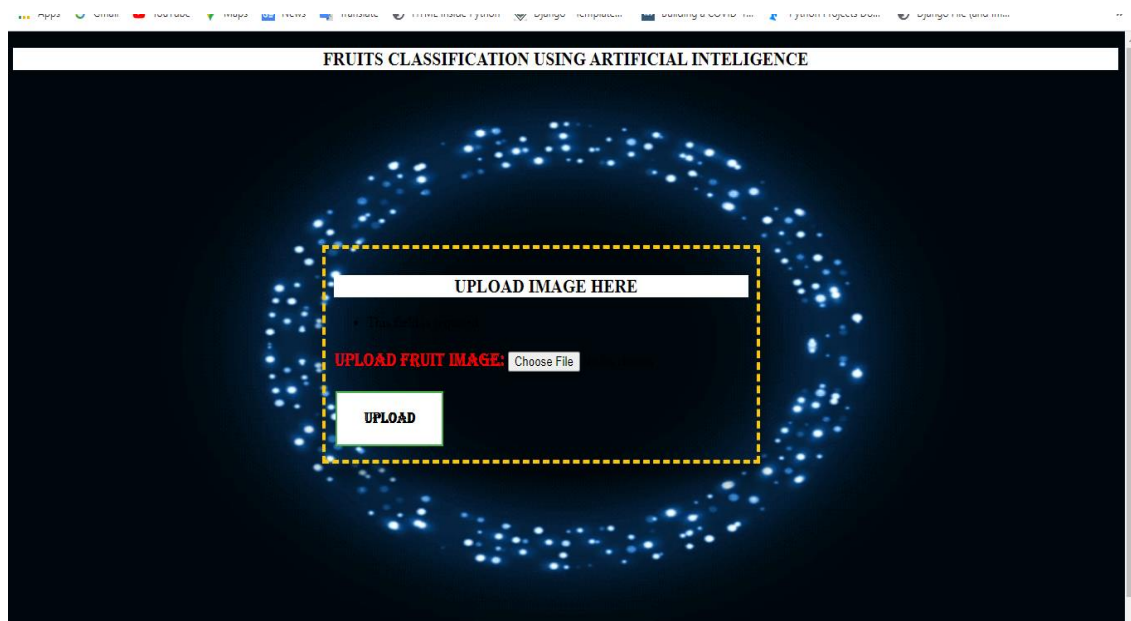


Fig 39: Uploading fruit image.

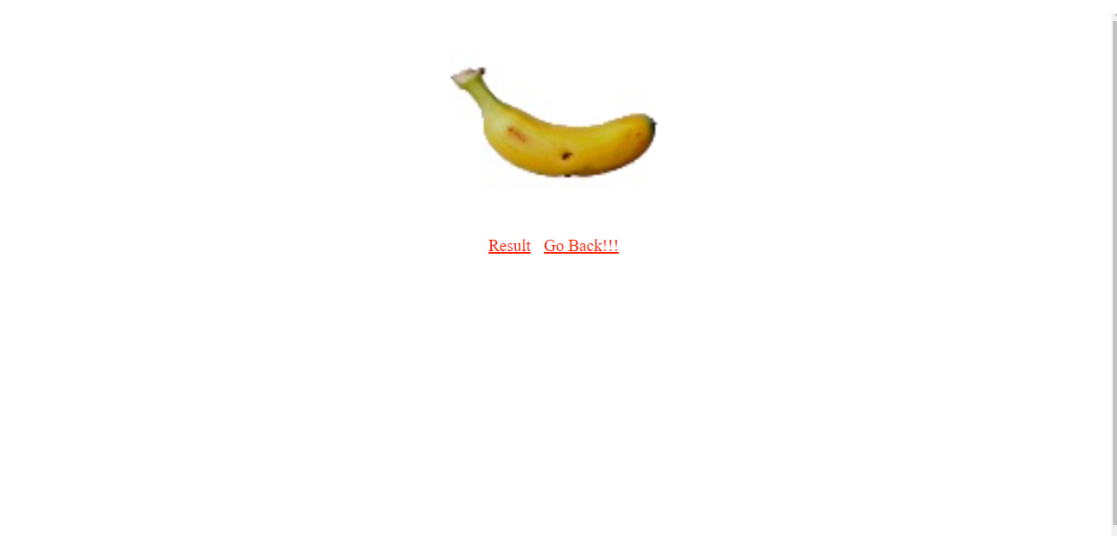


Fig 40: Uploaded image.

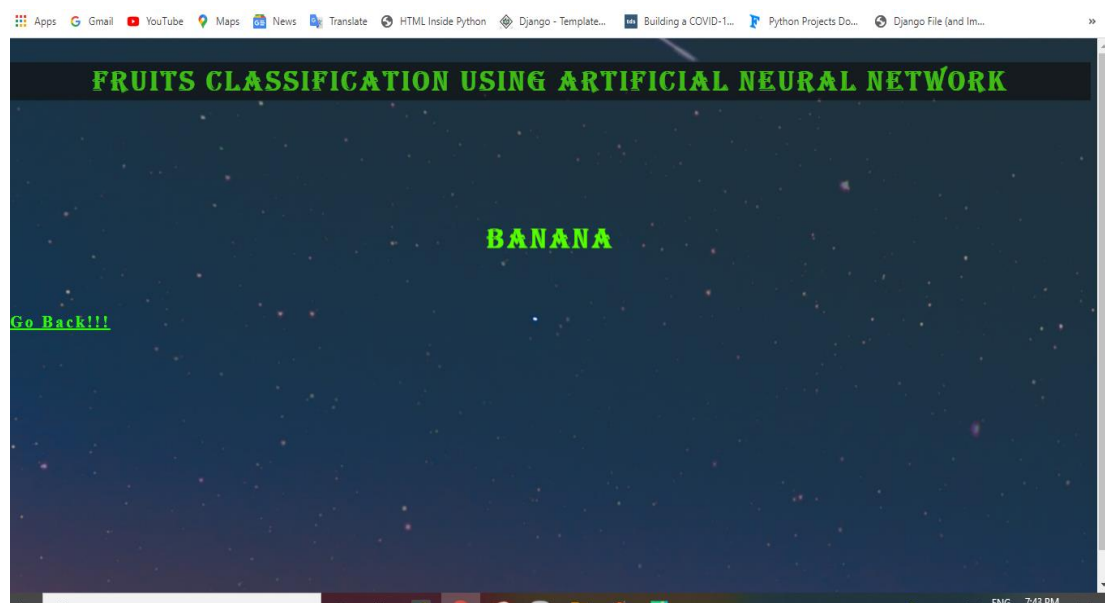


Fig 41: Fruit classified.

5.4.2.2 PLAGIARISM REPORT.

fruit_class-converted report			
ORIGINALITY REPORT			
1 %	1 %	0 %	1 %
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS
PRIMARY SOURCES			
1	Submitted to Indian Institute of Technology, Madras Student Paper		1 %
2	www.i-scholar.in Internet Source		1 %