# FRUIT RECOGNITION, QUANTITY AND QUALITY ESTIMATION USING A ROBOTIC VISION SYSTEM

**A PROJECT REPORT**

***Submitted by***

**SAI SADHAN S (312416106085)**

**SRIRAM M (312416106104)**

***in partial fulfilment for the award of the degree***

***Of***

**BACHELOR OF ENGINEERING**

***in***

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**St. JOSEPH'S INSTITUTE OF TECHNOLOGY**
**CHENNAI 600 119**

**ANNA UNIVERSITY :: CHENNAI 600 025**
**MARCH 2020**

ANNA UNIVERSITY:CHENNAI 600 025

## BONAFIDE CERTIFICATE

Certified that this project report "**FRUIT RECOGNITION, QUANTITY AND QUALITY ESTIMATION USING A ROBOTIC VISION SYSTEM"** is the bonafide work of **SAI SADHAN.S (312416106085)** and **SRIRAM.M (312416106104)** who carried out the project under my supervision.

| | |
|---|---|
| SIGNATURE | SIGNATURE |
| **Dr.C.GNANA KOUSALYA, M.E., Ph.D.** | **Dr.M.V.KARTHIKEYAN,M.E.,Ph.D.,** |
| PROFESSOR & HEAD OF THE DEPARTMENT | SUPERVISOR & ASSOCIATE PROFESSOR |
| DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING | DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING |
| St. JOSEPH'S INSTITUTE OF TECHNOLOGY, | St. JOSEPH'S INSTITUTE OF TECHNOLOGY |
| OLD MAHABALIPURAM ROAD, | OLD MAHABALIPURAM ROAD, |
| CHENNAI- 600119. | CHENNAI- 600119. |

# CERTIFICATE OF EVALUATION

College : **St. Joseph's Institute of Technology, Chennai-600 119**

Branch : **Electronics and Communication Engineering**

Semester : **VIII**

| NAME OF THE STUDENT | TITLE OF THE PROJECT | NAME OF THE SUPERVISOR WITH DESIGNATION |
|---|---|---|
| SAI SADHAN S (312416106085) SRIRAM M (31241610105) | FRUIT RECOGNITION, QUANTITY AND QUALITY ESTIMATION USING A ROBOTIC VISION SYSTEM | Dr.M.V.KARTHIKEYAN, M.E.,Ph.D. ASSISTANT PROFESSOR |

The report of project work submitted by the above students in partial fulfilment for the award of Bachelor of Engineering degree in Electronics and Communication Engineering branch of Anna University were evaluated and confirmed to be reports of the work done by the above student and then evaluated.

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

# ACKNOWLEDGEMENT

At the outset we would like to express our sincere gratitude to our beloved **Chairman Dr. B. BABU MANOHARAN, M.A., M.B.A., Ph.D.,** for his constant guidance and support.

We would like to express our thanks to our respected **Managing Director**, **Mrs. B.JESSIE PRIYA, M.Com.,** and our **Director Mr. B.SHASHI SEKAR, M.Sc.,** for their kind encouragement and blessings.

We express our sincere gratitude and whole hearted thanks to our **Principal Dr. P.RAVICHANDRAN, M.Tech., Ph.D.,** for his encouragement to make this project a successful one.

We wish to express our sincere thanks and gratitude to our **Head of the Department Dr. C.GNANA KOUSALYA**, **M.E., Ph.D.,** of Electronics and Communication Engineering for leading us towards the completion of this project.

We also wish to express our sincere thanks to our **project guide Dr.M.V.KARTHIKEYAN,M.E.,Ph.D.,** Assistant Professor**,** Department of Electronics and Communication Engineering for his guidance and assistance in solving the various intricacies involved in the project.

Finally, we thank our parents and friends who helped us in the successful completion of this project.

# ABSTRACT

In agriculture, there is a huge demand for smart irrigation. The task of fruit grading plays a vital role in agricultural industry as the consumer's concern on fruit quality is high. Deep Learning helps in fruit recognition, counting, quality and ripeness estimation. The video input is fed to the system via pi-camera and is converted into frames. Using Convolutional Neural Network (CNN), the image is processed after turning it into grayscale image. Tensor flow is used as a machine learning library. CNN and GLCM are used to extract the features of the fruit and estimates the quality by comparing with the trained images. Finally, the type, quantity and quality of the fruit is recognized and sends an email with an image if any disease detected through IoT.

Keywords— Internet of Things (IoT), Deep Learning, Convolutional Neural Network (CNN), GLCM, Tensor Flow.

# TABLE OF CONTENTS

## LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| IoT | Internet of Things |
| IP | Internet Protocol |
| GPIO | General Purpose Input Output |

# CHAPTER 1
# INTRODUCTION

In most developing countries such as India, agriculture constitutes the major part of the economy and it is basic occupation of Indian people. Agronomy, Horticulture, Agricultural Engineering, Agricultural Economics, Animal Science are the five main branches of agriculture. Horticulture is one of the most cost-demanding factors in agriculture industry. Horticulture deals with the study of cultivation of fruits, vegetables, and ornamental crops. Horticulturists apply their knowledge skills, and technology intensively produces fruits for personal or social needs. Their main aim is to improve fruit growth, yield, nutritional value, quality and resistance to insects and diseases. Now-a-days, robotic harvesting has done to help farmers. Reducing input cost, minimizing working time, improving fruits quality and increasing yield of fruits is the main goal of horticulturist. Image processing, computer vision, machine-learning techniques, robotic harvesting playing important role in horticulture sector. Production and consumption of agricultural products, especially fruit and vegetables, in India has shown a marked upward trend over the past few years.

Post-harvest processing of fruits comprises a series of unit operations, including cleaning, waxing, sorting, grading, packing, transport, and storage, while counting without damage to fruit has being considered the most important post-harvest step. Farmers continue to examine and sort/grade harvested fruits by visual appearance. Manually classification and counting of fruits based on geometrical or shape features is laborious, stressful and costly. Thus, there is a critical need to be able to quick, accurate and efficiently evaluate agricultural products without the use of human labor. Therefore, automated classification and sorting systems that use image processing, computer vision, machine-learning techniques to determine geometrical and shape parameters, such as size, shape, color, ripeness, mass, bruising, disease, and rot has being developed in many countries. In addition, an automated detection and counting of on the tree fruit system would free up labor, which is already in short supply and more expensive.

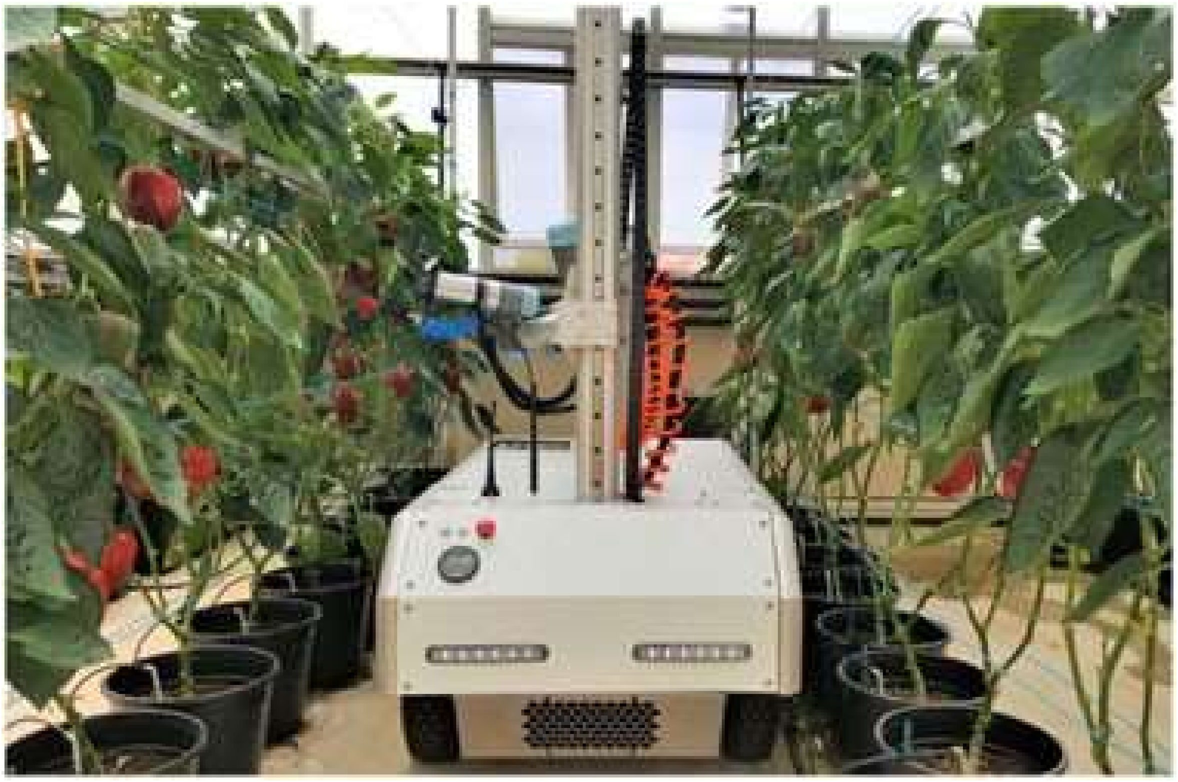


Fig 1.1 An image of Harvey in a protected cropping environment.

Harvesting robots, such as Harvey, detect and segment fruit in complicated scenes which often include high levels of occlusion. Furthermore, the foliage in the scene can share similar colours to the fruit being detected. Such problems require advanced robotic vision techniques to deal with these challenging and unstructured environments.

## 1.1 NEED OF PROJECT

### 1.1.1 Help the farmers in horticulture industry

Generally, in India the quality inspection of fruits has performed by human experts. This manual sorting by visual inspection is labour intensive, time consuming and suffers from the problem of inconsistency and inaccuracy in judgment by different human. Machine learning, computer vision and image processing techniques have found increasingly useful in the fruit industry, especially for applications in quality inspection and defect sorting applications.

### 1.1.2 Robotic Harvesting of on tree fruits

Autonomous robotic harvesting is a rising trend in agricultural applications, like the automated harvesting of fruit. Farmers continuously look for solutions to upgrade their production, at reduced running costs and achieve maximum profit within less amount of time. This is where harvesting robots come into play. One of the challenges of developing a fruit-harvesting robot is fruit detection. To detect the fruit, an image processing approach will be used and has the ability to classify and detect on tree fruits in cluster and partially occluded fruit.

### 1.1.3 Reduce labour cost and labour work

Cut fruit from tree, sort them on basis of ripe condition, count number of fruits requires a lot of labour and it is time consuming. There is need of some automatic machine that cut, sort and count fruits in minimum time without degrading the quality of fruit. Image processing and robotic technology plays important role to solve this problem.

### 1.1.4 Avoid damaging of fruits

During manual counting, fruits are count as well as separate according to their growth conditions. Ripe, unripe, over-ripe fruits has distributed in separate basket. Fruits may get damage by moving fruits manually from one basket to other basket. Image processing, video processing and conveyor belt system will help to avoid damaging of fruits during counting and sorting of fruits.

## 1.2 DEEP LEARNING

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don't need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture. In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousand of their neighbours. The question here is how do we recreate these neurons in a computer. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

### 1.2.1 Working

First, they need to identify the actual problem in order to get the right solution and it should be understood, the feasibility of the Deep Learning should also be checked (whether it should fit Deep Learning or not). Second, they need to identify the relevant data which should correspond to the actual problem and should be prepared accordingly. Third, Choose the Deep Learning Algorithm appropriately. Fourth, Algorithm should be used while training the data set. Fifth, final testing should be done on the data set. Here are some architectures :

1. Deep Neural Network
2. Deep Belief Network
3. Recurrent Neural Network

The working flow diagram of deep learning is shown below.

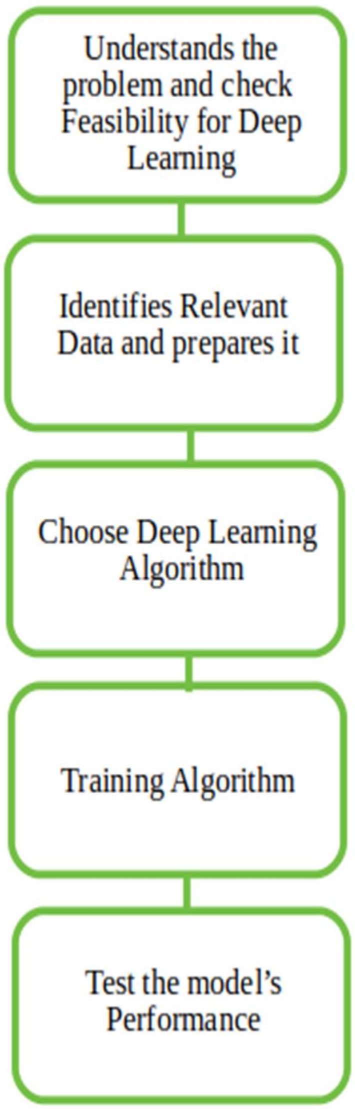

Understands the problem and check Feasibility for Deep Learning
Identifies Relevant Data and prepares it
Choose Deep Learning Algorithm
Training Algorithm
Test the model's Performance

Fig 1.2 Working flow of deep learning

### 1.2.2 Benefits

1. Features are automatically deduced and optimally tuned for desired outcome.
2. Features are not required to be extracted ahead of time. This avoids time consuming machine learning techniques.
3. Robustness to natural variations in the data is automatically learned.
4. The same neural network based approach can be applied to many different applications and data types.

5. Massive parallel computations can be performed using GPUs and are scalable for large volumes of data. Moreover it delivers better performance results when amount of data are huge.
6. The deep learning architecture is flexible to be adapted to new problems in the future.

### 1.2.3 Challenges

1. It requires very large amount of data in order to perform better than other techniques.
2. It is extremely expensive to train due to complex data models. Moreover deep learning requires expensive GPUs and hundreds of machines. This increases cost to the users.
3. There is no standard theory to guide you in selecting right deep learning tools as it requires knowledge of topology, training method and other parameters. As a result it is difficult to be adopted by less skilled people.
4. It is not easy to comprehend output based on mere learning and requires classifiers to do so. Convolutional neural network based algorithms perform such tasks.

## 1.3 OBJECTIVE OF THE WORK

The aim of the project is to use computer vision, machine learning and deep learning techniques to classify, detect and count on tree fruits from images and videos.

### 1.3.1 Objectives of proposed work

The main objectives of this project are:

i. To classify fruit use deep convolution neural network model and machine learning algorithms.
ii. To detect fruit from image remove background such as leaves, branches, sky.
iii. To count number of fruits by using computer vision techniques.

## 1.4 EXISTING SYSTEM

The existing system approach consists of detection system. The detection system takes inspiration from the Deep Fruits technique, but proposes a new network structure which jointly learns detection. Similar to Deep Fruits, they fine tune a pre-trained FRCNN network. It uses VGG-16 architecture to detect the fruit and extract the features of it.

### 1.4.1 Disadvantages

**1.** Results can vary considerably due to human differences in colour perception and human error.
**2.** Available light quantity and quality can influence colour perception.

## 1.5 PROPOSED SYSTEM

The system is divided into the following steps: (1) Image acquisition (2) Image Pre-processing (3) Feature Extraction and (4) Classification. These images are made to undergo pre-processing steps like filtering and segmentation using morphological operations. Then different texture and colour features are extracted from the processed image. Feature extraction is the procedure to opt for the important characteristics of an image. Transforming the input data into the set of features is called feature extraction. The key features which gives specific range for each disease. The feature values are fed as inputs to the classifier to classify the given image. If disease is detected, an email is sent via IoT cloud server.

### 1.5.1 Advantages

1. This method helps in speed up the process of improve accuracy and efficiency and reduce time.
2. Best suited for different illuminant condition.
3. The diseased image is sent to the remote location through e-mail.

## 1.6 HARDWARE REQUIREMENTS

| | |
|---|---|
| ➢ **Controller** | : Raspberry-pi |
| ➢ **Input** | : Pi-Camera |
| ➢ **Storage** | : Memory card |

## 1.7 SOFTWARE REQUIREMENTS

| | |
|---|---|
| ➢ **Toolbox** | : Image Processing |
| ➢ **OS** | : Raspbian |
| ➢ **Language** | : Python |
| ➢ **Training tool** | : Tensorflow keras |

# CHAPTER 2

# LITERATURE SURVEY

## 2.1 INTRODUCTION

In order to acquire wide knowledge about fruit recognition, quantity and quality estimation using a robotic vision system, many research papers of various authors related to this project have been studied thoroughly. The papers listed below will give brief explanation of the whole theme.

## 2.2 Fruit Quantity and Ripeness Estimation Using a Robotic Vision System

In 2018, Michael Halstead, Christopher McCool, Simon Denman, Tristan Perez and Clinton Fookes presented a paper on Fruit Quantity and Ripeness Estimation Using a Robotic Vision System. Accurate localization of crop remains highly challenging in unstructured environments, such as farms. Many developed systems still rely on the use of hand selected features for crop identification and often neglect the estimation of crop quantity and ripeness, which is a key to assigning labour during farming processes. To alleviate these limitations, they present a robotic vision system that can accurately estimate the quantity and ripeness of sweet pepper (Capsicum annuum L), a key horticultural crop.

This system consists of three parts: detection, ripeness estimation, and tracking. Efficient detection is achieved using the Faster RCNN (FRCNN) framework. Ripeness is then estimated in the same framework by learning a parallel layer which they experimentally show results in superior performance than treating ripeness as extra classes in the traditional FRCNN framework. Evaluation of these two techniques outlines the improved performance of the parallel layer, where they achieve an F 1 score of 77.3 for the parallel technique yet only 72.5 for the best scoring (red) of the multiclass implementation. To track the crop, they present a vision only tracking via detection approach, which uses the FRCNN with parallel layers as input. Being a vision only solution, this approach is cheap to implement as it only requires a camera and in experiments, using field data, they show that the proposed system can accurately estimate the number of sweet pepper present, to within 4.1% of the visual ground truth.

Two approaches for joint detection and ripeness estimation are considered. The first approach is to extend the number of classes so that each ripeness value has its own unique class, referred to as Multi Class-RCNN. The second approach is to have two parallel classification layers, referred to as Parallel-RCNN: one to determine if the bounding box corresponds to a fruit, and the second to determine the ripeness of the fruit in the bounding box. Multi Class-RCNN and Parallel-RCNN are both fine-tuned from a pre-trained Faster RCNN network using 152 training images. This was considered an adequate number of images as used 100 images for fine-tuning and showed that from as few as25 training images, a good detector could be trained. From the152 images, there were 253 examples of red fruit, 636 examples of green fruit and 69 examples of mixed fruit. When training, data augmentation consisted of flipping the images horizontally.

## 2.3 Monocular Camera Based Fruit Counting and Mapping with Semantic Data Association

In 2019, Jnaneshwar Das, Camillo J. Taylor and James Underwoodpresented a paper on Monocular Camera Based Fruit Counting and Mapping With Semantic Data Association. It is a cheap, lightweight, and fast fruit counting pipeline. The pipeline relies only on a monocular camera, and achieves counting performance comparable to a state-of-the-art fruit counting system that utilizes an expensive sensor suite including a monocular camera, LiDAR and GPS/INS on a mango dataset.

The pipeline begins with a fruit and tree trunk detection component that uses state-of-the-art convolutional neural networks (CNNs). It then tracks fruits and tree trunks across images, with a Kalman Filter fusing measurements from the CNN detectors and an optical flow estimator. Finally, fruit count and map are estimated by an efficient fruit-as-feature semantic structure from motion algorithm that converts two-dimensional (2-D) tracks of fruits and trunks into 3-D landmarks, and uses these landmarks to identify double counting scenarios. There are many benefits of developing such a low cost and lightweight fruit counting system, including applicability to agriculture in developing countries, where monetary constraints or unstructured environments necessitate cheaper hardware solutions. Fruit detection, segmentation and counting in a single image have seen a revolution from hand-crafted computer vision techniques to data-driven techniques. Traditional hand-engineered techniques for this task are usually based on a combination of shape detection and colour segmentation. Dorj et al. develop a watershed segmentation based method to detect citrus in HSV space. Ramos et al. use contthe analysis on super pixel over-segmentation result to fit ellipses for counting coffee fruits on branches.

The fruit detections in each image frame are used to construct a fruit count for each tree. The challenge in this step is associating detections with each other across all the image frames in the entire dataset, or in other words, identifying double counts. These associated detections then represent a single fruit. The fruit counting and mapping pipeline thus mainly consists of five parts. The first part performs 2D tracking on fruit centres to account for the first source of double count. The second part uses these fruit centres and their associations across frames as feature matches in a semantic SfM reconstruction to estimate 3Dlandmark positions as well as camera poses of each image frame. The third part projects those 3D landmarks back to the image plane of every image in order identify double tracks and address the second source of double count. The fourth part estimates the 3D locations of tree centroids, which are approximated by tree trunks in the implementation. These centroids are used as depth thresholds so only fruits that are closer to the camera than the tree centroids are counted, thus accounting for the third source of double count. Finally, the last part further refines the fruit association across frames, and estimates the final map of fruit landmarks.

## 2.4 Robust Fruit Counting: Combining Deep Learning, Tracking, and Structure from Motion

In 2018, Xu Liu, Steven W. Chen, Shreyas Aditya, Nivedha Sivakumar, Sandeep Dcunha, Chao Qu, Camillo J. Taylor, Jnaneshwar Das, and Vijay Kumar presented a paper on Robust Fruit Counting by Combining Deep Learning, Tracking, and Structure from Motion.This paper presents a novel fruit counting pipeline that combines deep segmentation, frame to frame tracking, and3D localization to accurately count visible fruits across a sequence of images. the pipeline works on image streams from a monocular camera, both in natural light, as well as with controlled illumination at night. They first trained a Fully Convolutional Network (FCN) and segment video frame images into fruit and non-fruit pixels. Then track fruits across frames using the Hungarian Algorithm where the objective cost is determined from a Kalman Filter corrected Kanade-Lucas-Tomasi (KLT) Tracker.

In order to correct the estimated count from tracking process, combine tracking results with a Structure from Motion (SfM) algorithm to calculate relative 3D locations and size estimates to reject outliers and double counted fruit tracks. They evaluate their algorithm by comparing with ground-truth human-annotated visual counts. These results demonstrate that the pipeline is able to accurately and reliably count fruits across image sequences, and the correction step can significantly improve the counting accuracy and robustness. Although discussed in the context of fruit counting, the work can extend to detection, tracking, and counting of a variety of other stationary features of interest such as leaf-spots, wilt, and blossom.

Deep learning techniques have recently replaced traditional hand-engineered computer vision techniques as the state of the art for fruit detection, segmentation and counting in a single image. The Faster Regions with Convolutional Neural Networks (Faster R-CNN) architecture has been used for detection of mangoes, almonds, apples, and a variety of other fruits.

Chen et al. use the Fully Convolutional Network (FCN) to segment the image into candidate regions and CNN to count the fruit within each region. Rahnemoonfar and Sheppard train an Inception style architecture to directly count the number of tomatoes in an image, demonstrating that in some scenarios these deep networks can even be trained using synthetic data. Barthet al. also generate synthetic data to train deep neural networks to segment pepper images. The work differs from these previous works by expanding the counting problem from a single image to an image sequence. This extension is more challenging since it requires tracking of the fruits across image frames. Most approaches that track fruits across image frames use some combination of Structure from Motion (SfM), Hungarian algorithm, Optical Flow, and Kalman Filters to provide the corresponding assignments of fruits across frames. Royand Isler use apple contours in an SfM pipeline to generate dense matches and reconstruct the apples in 3D.

## 2.5 Automatic dragon fruit counting using adaptive thresholds for image segmentation and shape analysis

In 2017, Chi Cuong Tran, Dinh Tu Nguyen, Hoang Dang Le and Quoc Bao Truong presented a paper on Automatic dragon fruit counting using adaptive thresholds for image segmentation and shape analysis. In recent years there have been many proposals for automated applications in vegetable harvesting. There are two major challenges: estimation of yield and harvesting of fruit from trees. This research proposes a new algorithm that allows an accurate counting of the number of fruits on a tree to accurately estimate the output of a dragon fruit.

The algorithm consists of the following main steps: image segmentation, boundary determination, shape analysis, and overlap analysis to count the number of dragon fruits per tree. Experimental results show that the algorithm can solve the problem of counting the correct number of fruits per tree to accurately estimate the yield of dragon fruit in the farm. In agriculture, the farm produce estimation is the important part in the harvest process The disadvantages of the traditional estimation methods are time-consuming and a lot of manpower. Many recently image processing techniques could improve the farm produce estimation accurate and quick. Considering some practical requirements in this area, the authors found that the development of an image processing algorithm to solve the problem of the productivity of agricultural products estimation is very necessary. Because in fact, the process of buying and selling agricultural products is done manually. The trader with the farmer come to the farm to estimate the output of agricultural product and negotiate prices based on experience and manual. This problem sometimes causes disagreement between buyers and sellers. Therefore, an automated estimating system for agricultural products will help solve this problem. To understand the information that exists in the image, the feature extraction and the object identification are the major problems in image processing techniques. This problem is mainly based on the colour and shape of the dragon fruit. Object-recognition methods, including simple algorithms, allow for the identification of discrete, discrete dragons, to complex algorithms for separating overlapping areas. The process of counting the number of dragon fruits in a tree consists of the following steps: collecting, defining images, segmentation, shape analysis, estimating sizes, identifying overlaps and counting the number of fruits per tree.

## 2.6 Automatic Fruit Recognition from Natural Images using Colour and Texture Features

In 2017, Susovan Jana and Ranjan Parekh presented a paper on Automatic Fruit Recognition from Natural Images using colour and texture features. Automated or robot-assisted harvesting is an emerging domain of research that combines the aspects of computer vision and machine intelligence. This research is usable in monitoring, sorting and picking of fruits for ensuring faster production chain. This paper aims to analyze popular methods of auto-harvesting, categorization of fruits and proposes a new approach that overcomes some of the drawbacks of the previous methods. The proposed approach takes into account different types of fruits.

The main goal is to come up with a method for classifying these different types of fruits accurately and efficiently. Images are pre processed in order to separate the fruit in the foreground from the background. Texture features from VGG-16 and statistical colour features are extracted from the segmented image. Two types of features are combined in a single feature descriptor. A Support Vector Machine (SVM) classification model is trained using these feature descriptors extracted from the training dataset. Once trained, the model can be used to predict the category for an unlabeled image from the validation set. The proposed approach also works best for embedded systems and single board computers as it realizes the trade-offs of these devices.

Grab Cut is an efficient and interactive algorithm for foreground/background segmentation from still images proposed by Rother et al. Of Microsoft Research Cambridge, UK. The algorithm makes use of two components: a hard-labelling scheme defined for image pixels and a foreground/background classifier based on Gaussian Mixture Model (GMM) that takes RGB pixel values as input features. Initial user input is a rectangle. Every pixel outside the rectangle is treated as sure background. Everything inside the rectangle is unknown. This rectangle and any subsequent user input, if provided, is considered hard-labelling.

After the initial hard-labelling, Grab Cut proceeds in iterations: (1) Depending on the user defined hard-labelling, a GMM is used to learn and create a new pixel distribution. That is, pixels inside the rectangle are labelled either probable-foreground or probable background depending on their relation with hard-labelled pixels in terms of colour statistics. (2) Using this pixel distribution a graph is generated where each node represents a pixel in the image. Two additional nodes are introduced, i.e. the “source” and the “sink”. All probable foreground pixels are connected with the source and all probable background pixels are connected with the sink. (3) The weights for the edges connecting the pixels with either of the source or the sink is defined by the probability of a pixel being foreground or background. Edge weights connecting the pixels are defined by pixel-similarity. (4) Finally, a mincut algorithm segments the graph in two parts separating source and sink nodes using a minimum cost function. The cost function is the sum of all edge weights that are cut. The cut separates the nodes/pixels connected with the source as foreground and those connected with the sink as background. (5) This process continues until the classification reaches convergence. Grab Cut produces a very accurate segmentation even when initialized with a simple rectangular mask overlapping the foreground providing the initial hard-labelling. This is also how the dataset was prepared for this paper. Region of Interest Images segmented using Grab Cut may have more than one object or region, not connected with each other, marked as foreground. The Grab Cut output is used to generate a binary image replacing foreground pixels with 1's and background pixels with 0's. The white pixel blobs (represented by 1’s) are then sorted according to the area they occupy. The area here is a measure of the actual number of white pixels contained within the blob. The region with the largest area coverage is marked as the region of interest (ROI). This ROI is used to extract a single object from the Grab Cut output.

The gray-level co-occurrence matrix G[i, j] is defined by first specifying a displacement vector d = (dx, dy) and counting all pairs of pixels separated by d and at angle θ having gray levels i and j, provided that the total number of gray levels N is known.

## 2.7 Automatic Fruit Quality Inspection System

In 2016, Manali R. Satpute and Sumati M. Jagdale presented a paper on Automatic Fruit Quality Inspection System. This paper presents the recent development in automatic vision based technology. Use of this technology is increasing in agriculture and fruit industry. An automatic fruit quality inspection system for sorting and grading of tomato fruit and defected tomato detection discussed here. The main aim of this system is to replace the manual inspection system. This helps in speed up the process improve accuracy and efficiency and reduce time. This system collect image from camera which is placed on conveyor belt. Then image processing is done to get required features of fruits such as texture, colour and size. Defected fruit is detected based on blob detection, colour detection is done based on thresholding and size detection is based on binary image of tomato. Sorting is done based on colour and grading is done based on size.

The Infrared sensor is placed on conveyor belt, when fruit is come in front of infrared sensor message will display as fruit detected then conveyor belt moves with small distance and stop when fruit come exactly in front of camera. Camera (High Quality CMOS sensor, 25 MPs, 30fps) always in video mode. When fruit is detected the image processing is done on that image captures and colour is detected. Red, Green, Yellow colour are detected. The system is divided into hardware control and image processing. The image processing results is based on camera image. The results such colour detected. Second part is hardware is controlled based on colour detection. The image processing is done by software OpenCv using a language python. The software is divided into two parts first one is for image analysis and other is for controlling hardware based on image processing results. The system is operated in two different scenarios in first the image is captured with camera the all the image processing is done in the control module. All the process are shown on monitor and then based on decision taken by control module. The conveyor assembly is operated. Tomato is having different defects. Here blob detection technology is used for defect detection. This is the technique by which specific region is detected which is differ in properties compared to surrounding region such as colour or brightness. Colour image, gray image and image showing defect on tomato surface is shown. The defects are highlighted with red circles. The defect which are present on red, green and yellow tomato are shown with green colour.

## 2.8 Apple Fruit Detection and Counting Using Computer Vision Techniques

In 2014, Anisha Syal, Divya Garg and Shanu Sharma presented a paper on Apple Fruit Detection and Counting Using Computer Vision Techniques.In agriculture sector the problem of identification and counting the number of fruits on trees plays an important role in crop estimation work. At present manual counting of fruits and vegetables is carried out at many places. Manual counting has many drawbacks as it is time consuming and requires plenty of labours. The automated fruit counting approach can help crop management system by providing valuable information for forecasting yields or by planning harvesting schedule to attain more productivity. This work presents an automated and efficient fruit counting system using computer vision techniques.

The proposed system uses minimum Euclidean distance based segmentation technique for segmenting the fruit region from the input image. Further circle overlaying is done on the fruit region and in the last fruits are counted on the basis of the centroid of the fruit regions. This proposed system is correctly detecting and counting the apples on the test images. Testing phase consists of following steps:

1) Pre-processing: The same pre-processing steps are then performed as training phase on the test image.

2) Euclidean Distance based Segmentation: The next step of the algorithm is to segment the test image. The pre-processed image is then converted into L*a*b plane. Then the each pixel of the input image is then classified on the basis of minimum Euclidean distance between input a and b values and the values of stored training feature file. Thus the fruit area is segmented on the input image.

3) Circle Fitting: Overlaying of circles was then performed to get the desired boundaries of fruit regions. Circle overlaying basically does the function of superimposing on images. Thus with the help of circle overlaying they get the desired perfectly segmented apple regions.

4) Individual Apple Segmentation: After that individual apples were separated from the occluded apples. And then the automated counting of fruits was performed.

5) Automated Counting: In the last the counting of apples in the image is done. This is the GUI for detection of large apples of the images. These large apples are basically the occluded apples which are divided into individual apples on

their comparison with the perfect apples. Thus this division makes the calculation of number of apples easier.

## 2.9 A Fruit Size Detecting and Grading System Based on Image Processing

In 2010, Hongshe Dang, Jinguo Song and Qin Guo presented a paper on a Fruit Size Detecting and Grading System Based on Image Processing.This paper presents a fruit size detecting and grading system based on image processing. After capturing the fruit side view image, some fruit characters is extracted by using detecting algorithms. According to these characters, grading is realized. Experiments show that this embedded grading system has the advantage of high accuracy of grading, high speed and low cost. It will have a good prospect of application in fruit quality detecting and grading areas.

A. Processing flow

Take apple as the processing example, according to state criterion GB/T 18965-2008, the apple size is its diameter, which is the longest distance in the apple's cross section. So the detecting program is focused on how to calculate the diameter in an apple side-view image.

B. Image Filter

In the process of fruit detecting and grading, the whole algorithm must be efficiency and fast. The pre-processing algorithm is also important. It affects the edge character extraction directly. Pre-process mainly contain image filter. So a good filter algorithm is needed, here a faster median filter is used, which is proposed. It can get the same effect of the normal median filter, but it costs less time to perform that. This algorithm is good way to filter the fruit image.

## C. Edge Detection

The edge extraction is key factor for size detecting. After image gray, the OTSU (maximum classes square error) method is used to get the fruit binary image automatically. And the result of diversion is good. After that, the 8-connected Boundary tracking method is used to get the edge sequences.

## D. Fruit Size detecting algorithm

In order to calculate this diameter, the fruit's natural symmetry is considered, so the fruit size detecting algorithm based on its symmetry mainly contains two parts: finding the centre coordinate of fruit's shape in image and finding fruit's axis in image.

## 2.10 PROPOSED VS EXISTING SYSTEM

| Description | Proposed System | Existing System |
|---|---|---|
| Techniques | The system is divided into the following steps<br>1. Image acquisition<br>2. Image Pre-processing<br>3. Feature Extraction<br>4. Classification.<br>Images undergo pre-processing steps - filtering and segmentation using morphological operations.<br>Texture and color features extracted from the processed image.<br>Feature extraction - procedure to opt for the important characteristics of an image and transforming input data into set of features.<br>Once the fruit is detected with any disease, the mail is sent using IoT cloud server to the respective mail IDs. | The existing system approach consists of two sub-systems:<br>1. Detection ripeness estimation subsystem<br>2. Tracking subsystem.<br>Uses Deep Fruits technique.<br>Pre-trained FRCNN network<br>The tracking sub-system - tracking via detection approach.(VGG-16)<br>Tracking via detection - a simple framework to count the number of fruits using off-the-shelf cameras.<br>**DISADVANTAGES**<br>1. Results can vary considerably due to human differences in color perception and human error.<br>2. Available light quantity and quality can influence color perception.<br>3. VGG-16 architecture it's accuracy is very low. |

Table 2.1 Comparative Study of Proposed System and Existing System

# CHAPTER 3

# PROPOSED SYSTEM

## FRUIT RECOGNITION, QUANTITY AND QUALITY ESTIMATION USING A ROBOTIC VISION SYSTEM

The aim of the project is to recognize the fruit, to estimate the quantity, to detect the disease if any and to closely monitor the environment around the harvesting area.This paper presents a review on methods that use digital image processing techniques to detect, recognize and classify plant diseases from digital image and concludes with discussion of more useful problems in the domain and future direction. For the fruit disease classification problem, precise image segmentation is required; otherwise the features of the non infected region will dominate over the features of the infected region. In this approach image segmentation is used to detect the region of interest which is the infected part only. After segmentation, features are extracted from the segmented image of the fruit and trained based on CNN-Alexnet architecture.

## 3.1 CONCEPTS OF MACHINE LEARNING PROCESS

Classification of fruit is one of the biggest challenge is not achieve by computer vision techniques. Generally, fruits are classified based on their own size, shape, colour, height, diameter and weight but these parameter are not compatible with the image-processing techniques. The steps invloved in machine learning are the following.

● Step 1 – Obtain a labeled data set

● Step 2 – Split dataset into a training portion and validation or test portion.

● Step 3 – Fit model to training dataset

● Step 4 – Evaluate models performance on the test dataset

Image features are the following.

- Edges
- Colors
- Patterns/Shapes

Deep learning is a technique that learns from features provided to it and predict accurate results. In this project, fruits are classified based on the features such as height, diameter, weight and colour code. A dataset with sufficient features and corresponding label is passed to the deep learning algorithms. Further, the dataset is divided into train set and test set. Size of train set is more than test set because more the train sample then machine will learn more features from dataset and give accurate prediction. In this methodology, 70% dataset divided in train set and 30% dataset in test set to learn more features from train set and predict accurate results.

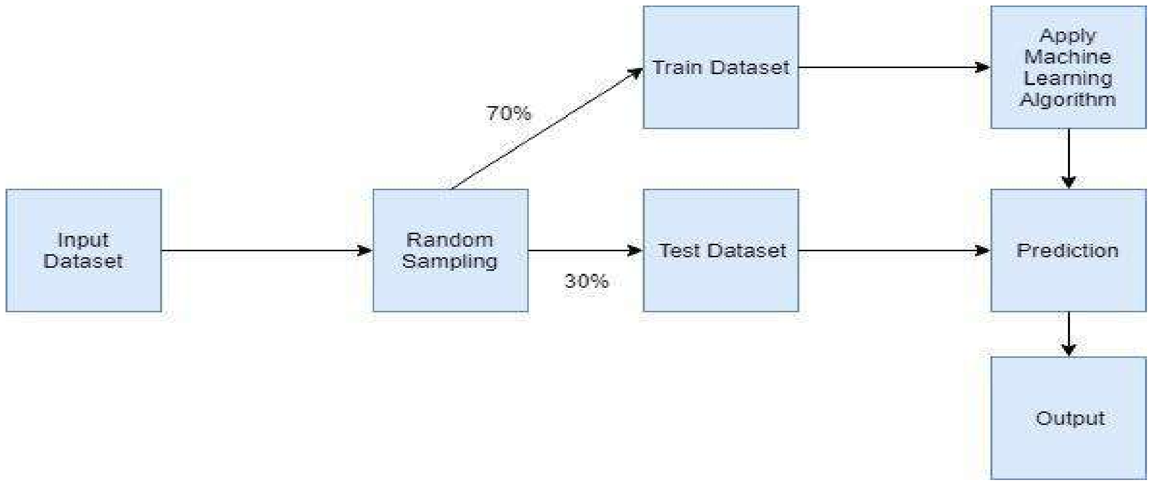


Fig 3.1 Flow chart for classification of Fruits using Deep Learning

### 3.1.1 Convolutions

Convolution is a mathematical term to describe the process of combining two functions to produce a third function. This third function or the output is called a Feature Map. A convolution is the action of using a filter or kernel that is applied to the input. In our case, the input being our input image. Convolutions are executed by sliding the filter or kernel over the input image. This sliding process is a simple matrix multiplication or dot product.

The effects of kernel are :

1. Depending on the values on the kernel matrix, we produce different feature maps. Applying our kernel produces scalar outputs.

2. Convolving with different kernels produces interesting feature maps that can be used to detect different features.

3. Convolution keeps the spatial relationship between pixels by learning image features over the small segments we pass over on the input image.

The Outputs of Convolutions : By applying multiple filters (of size 3x3x3) to an input image (of size 28x28x3), we can produce multiple Feature Maps. These are also called Activation Maps. These outputs are stacked together and treated as another 3D Matrix. This 3D output is the input to our next layer in our CNN.

### 3.1.2 Feature/Activation Maps and Image Features

Each cell in the Activation Map Matrix can be considered a feature extractor or neuron that looks at a specific region of the image. In the first few layers, the neurons activation when edges or other low level features are detected. In Deeper Layers, the neurons will be able to detect high-level or ‘big picture’ features of fruits. Feature Maps can be designed by tweaking the following parameters to control the size of our feature maps.

- Kernel Size (K x K)
- Depth
- Stride
- Padding

### 3.1.2.1 Depth

Depth describes the number of filters used. It does not relate the image depth (3 channels) nor does it describe the number of hidden layers in our CNN. Each filter learns different feature maps that are activated in the presence of different image features (edges, patterns, colors, layouts)

### 3.1.2.2 Stride

Stride simply refers to the step size we take when we slide our kernel the input image. Stride controls the size of the Convolution Layer output. Using a larger Stride produce less overlap in kernels. Stride is one of the methods we can control the spatial input size i.e. the volume of the inputs into the other layers of our CNN.

### 3.1.2.3 Zero-Padding

Zero-padding is a very simple concept that refers to a border we apply to the input volume. We add a border of 0s around our input. Basically this is equivalent of adding a black border around an image.

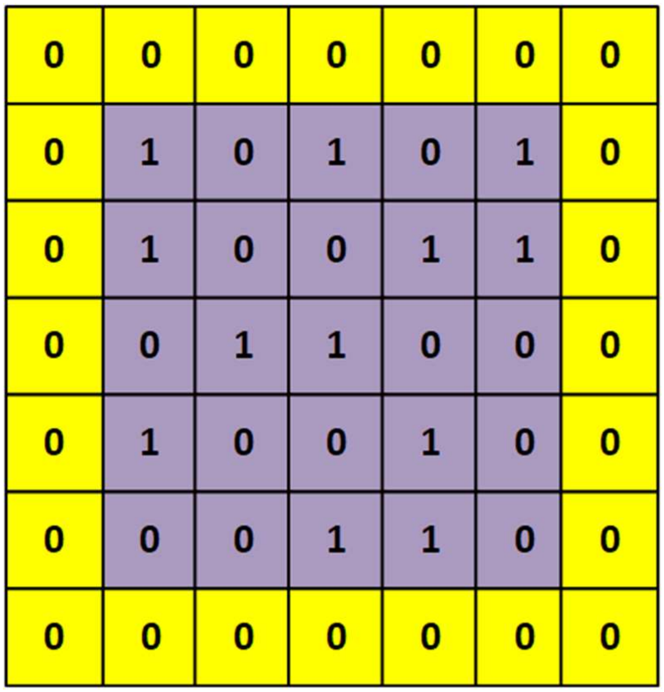


Fig 3.2 Zero Padding

It is possible to set our Padding P to 2 if needed. Zero-padding makes the output larger.

### 3.1.2.4 Calculating Convolution Output

CONV Layer Output size :

○ Kernel/Filter Size – K

○ Depth – D

○ Stride – S

○ Zero Padding – P

○ Input Image Size - I

To ensure the filters cover the full input image symetrically, we used the following equation to do this sanity check. Once the result of this equation is an integer, the settings are valid. The equation is ((I – K + 2P) / S) + 1.

### 3.1.3 ReLU

ReLU is the Activation Function of Choice. ReLU or Rectified Linear Unit simply changes all the negative values to 0 while leaving the positives values unchanged.

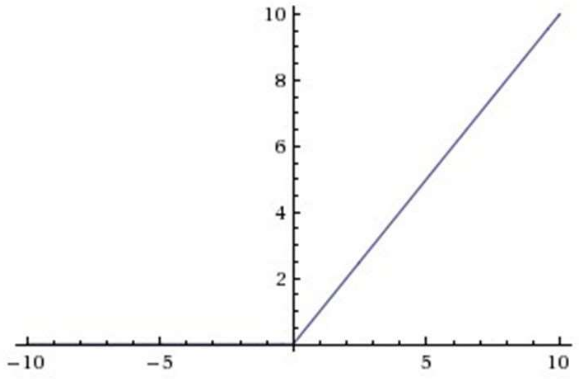


Fig 3.3 ReLU function, f(x) = max(0, x)

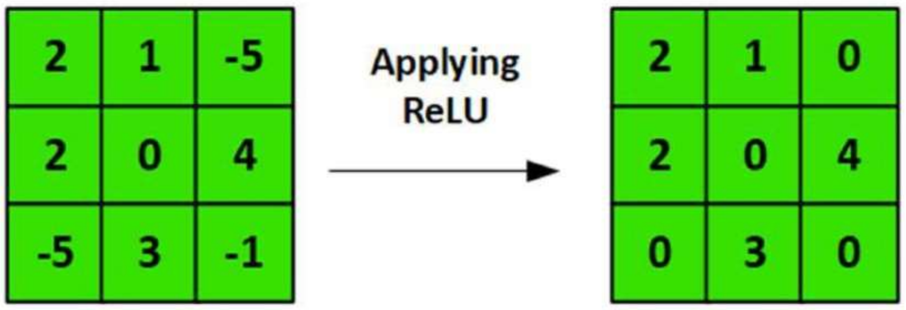


Fig 3.4 Output after ReLU

### 3.1.4 Pooling

Pooling, also known as subsampling or downsampling, is a simple process where we reduce the size or dimensionality of the Feature Map. The purpose of this reduction is to reduce the number of parameters needed to train, whilst retaining the most important features. There are 3 types of Pooling we can apply, Max, Average and Sum. Typically Pooling is down using 2x2 windows with a Stride of 2 and with no padding applied. For smaller input images, for larger images we use larger pool windows of 3x3).

Using the above settings, pooling has the effect of reducing the dimensionality (width and height) of the previous layer by half and thus removing ¾ or 75% of the activations seen in the previous layer.

### 3.1.5 The Fully Connected (FC) Layer

Fully Connected simply means all nodes in one layer are connected to the outputs of the next layer. The FC Layer outputs the class probabilities, where each class is assigned a probability.  All probabilities must sum to 1. The activation function used to produce these probabilities is the Soft Max Function as it turns the outputs of the FC layer (last layer) into probabilities.

### 3.1.6 Training CNNs

Training CNNs is essentially the same once we setup out Network Layers. The flow follows the following steps.

1. Random Weight initialization in the Convolution Kernels.

2. Forward Propagates an image through our network (Input -> CONV -> ReLU -> Pooling -> FC.

3. Calculate the Total Error e.g. say we got an output of [0.2, 0.4, 0.4] while the true probabilities were [0, 1, 0].

4. Use Back Propagation to update our gradients (i.e. the weights in the kernel filters) via gradient descent.

5. Keep propagating all images through our network till an Epoch is complete.

6. Keep completing Epochs till our loss and accuracy are satisfactory.

### 3.1.7 Designing CNNs

Basic CNN's all follow a simple set of rules and layer sequences.

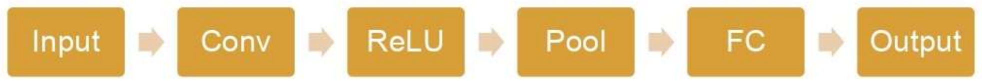


Fig 3.5 Flow of CNN

Basic CNN Design Rules:

● Input Layer typically Square e.g. 28 x 28 x 3, or 64 x 64 x 3 (this isn't necessary but simplifies our design and speeds up our matrix calculations).

● Input should be divisible by at least 4 which allows for downsampling.

● Filters are typically small either 3x3 or 5x5.

● Stride is typically 1 or 2 (if inputs are large).

● Zero padding is used typically to allow the output Conv layer to be the same size as the input.

● Pool kernel size is typically 2x2.

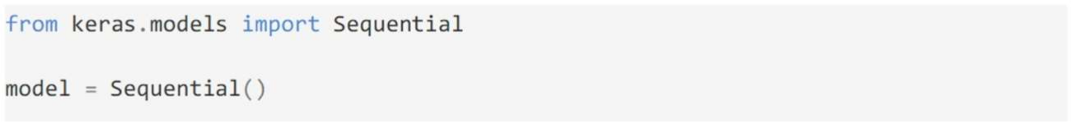
● Dropout is a very useful technique to avoid overfitting in CNNs.

### 3.1.8 Building a CNN in Keras

Keras is a high-level neural network API Keras API for Python. It makes constructing Neural Networks (all types but especially CNNs and recurrent NNs) extremely easy and modular. It has the ability to use TensorFlow, CNTK or Theano back ends. TensorFlow is an open source library created by the Google Brain team in 2015. It is an extremely powerful Machine Learning framework that is used for high performance numerical computation across a variety of platforms such as CPUs, GPUs and TPUs. It too has a python API that makes it very accessible and easy to use. Keras is extremely easy to use extremely easy to use extremely easy to use and follows a pythonic style of coding. Modularity - can easily add and remove Modularity building blocks of Neural Network code, while easily changing cost functions, optimizers, initialization schemes, activation functions, regularization schemes. Allows us to build powerful Neural Networks quickly and efficiently. It works in Python. TensorFlow is not as user friendly and modular as Keras. Unless you're doing academic research or constructing ground breaking Neural Networks, there is no need to use pure TensorFlow.

### 3.1.8.1 Composing a Neural Network in Keras

First we import our Sequential model type. Sequential model is a linear stack of layers.



```
from keras.models import Sequential

model = Sequential()
```

Fig 3.6 Composing a model in Keras

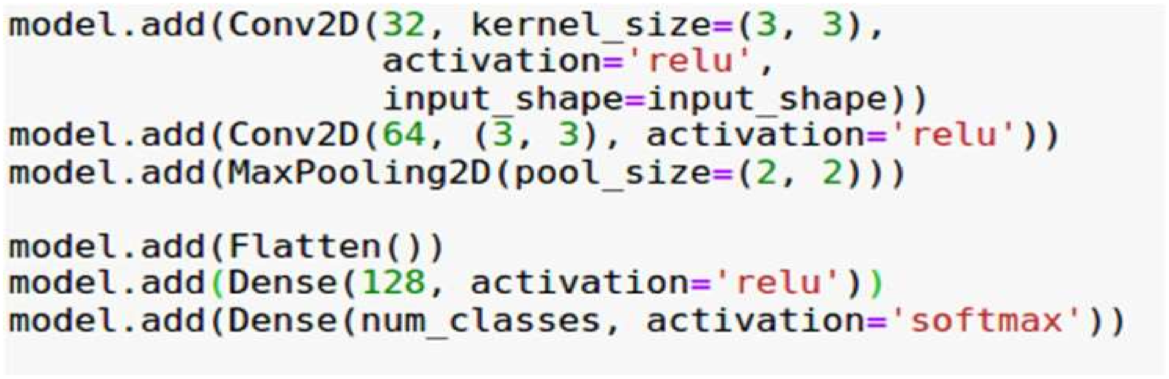
### 3.1.8.2 Creating our Convolution Layer

To add our Conv layer, we use model.add(Conv2D).

○ Specifying firstly the number of kernels or filters

○ The kernel size

○ The input shape



```
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

Fig 3.7 Convolutional Layer

Our model so far:

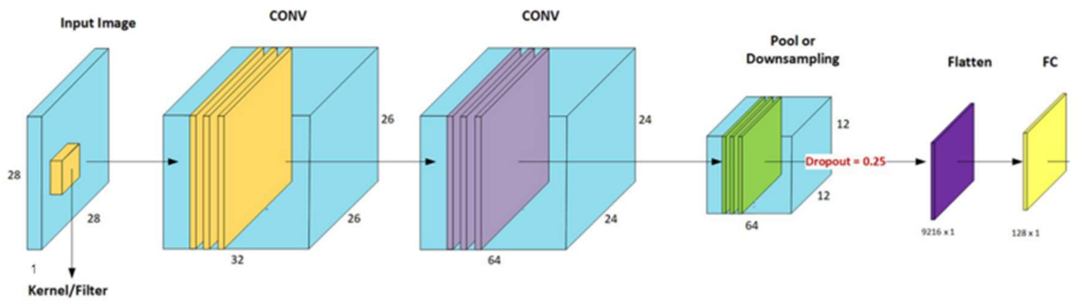
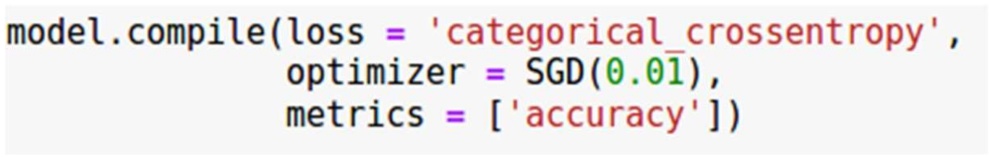


Fig 3.8 CNN Model

### 3.1.8.3 Compilation

Compiling simply creates an object that stores our Compiling model we have just created. We can specify our loss algorithm, optimizer and define our performance metrics. Additionally, we can specify parameters for our optimizer such as learning rates learning rates learning rates and momentum.

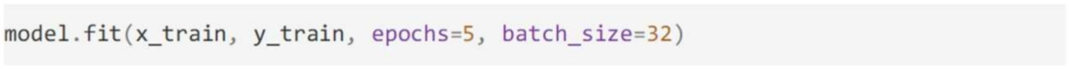


```
model.compile(loss = 'categorical_crossentropy',
              optimizer = SGD(0.01),
              metrics = ['accuracy'])
```

Fig 3.9 Compilation code

### 3.1.8.4 Fit or Train Model

Following the language from the most established Python ML library (Sklearn), we can fit or train our model with the follow code.

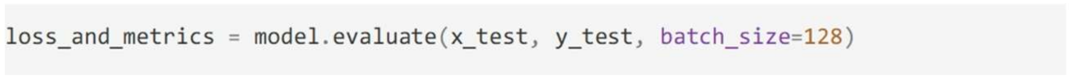


```
model.fit(x_train, y_train, epochs=5, batch_size=32)
```

Fig 3.10 Fitting code

### 3.1.8.5 Evaluate Model and Generate Predictions

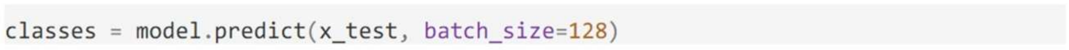
We can now simply evaluate our model's performance with this line of code.



```
loss_and_metrics = model.evaluate(x_test, y_test, batch_size=128)
```

Fig 3.11 Evaluation code

Generate predictions by feeding the Test data to the model.prediction function.



```
classes = model.predict(x_test, batch_size=128)
```

Fig 3.12 Prediction code

### 3.1.8.6 Data Augmentation

The authors used label-preserving transformation to make their data more varied. Specifically, they generated image translations and horizontal reflections, which increased the training set by a factor of 2048. They also performed Principle Component Analysis (PCA) on the RGB pixel values to change the intensities of RGB channels, which reduced the top-1 error rate by more than 1%. The benefits of data augmentation are the following.

● Take a small dataset and make it much larger.

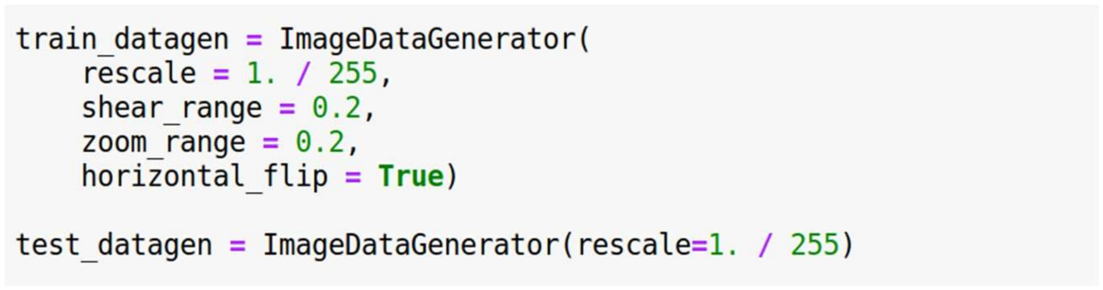
● Require much less effort in creating a dataset.

● Adding variations such as rotations, shifts, zooming etc make our classifier much more invariant to changes in our images. Thus making it far more robust.

● Reduces overfitting due to the increased variety in the training dataset.

● Kera’s Data Augmentation API performs a just-in-time augmented image dataset. This means images aren’t created and dumped to a directory which will be wasteful storage. Instead it generates this dataset during the training process.

### 3.1.8.7 Creating Our Image Generator

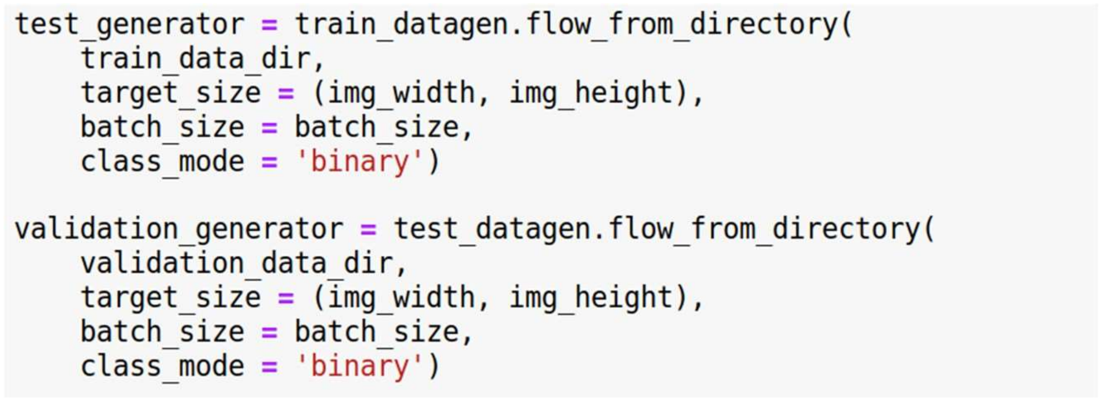


```
train_datagen = ImageDataGenerator(
    rescale = 1. / 255,
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale=1. / 255)
```

Fig 3.13 Data Augmentation

We use the above code to create the generator with types of augmentation to perform specified in the input arguments.

### 3.1.8.8 Configuring Batch Sizes

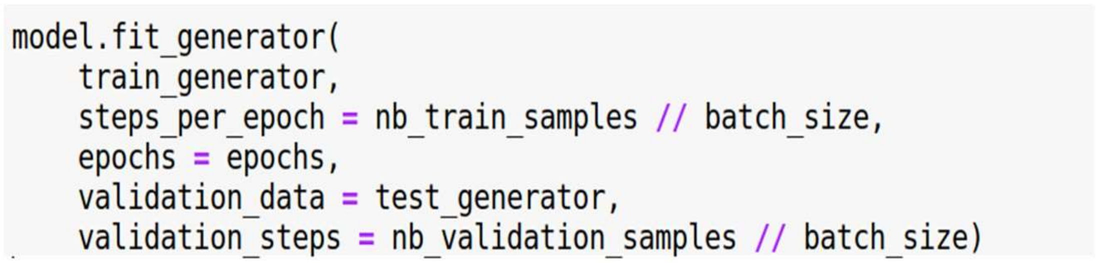


```
test_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size = (img_width, img_height),
    batch_size = batch_size,
    class_mode = 'binary')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size = (img_width, img_height),
    batch_size = batch_size,
    class_mode = 'binary')
```

Fig 3.14 Batch configuration

The flow_from_directory() function takes our image data and creates an iterator (a memory efficient method of returning a sequence of data). We can specify batch sizes, class mode and image size etc.

#### 3.1.8.9 Fitting the Generator



```
model.fit_generator(
    train_generator,
    steps_per_epoch = nb_train_samples // batch_size,
    epochs = epochs,
    validation_data = test_generator,
    validation_steps = nb_validation_samples // batch_size)
```

Fig 3.15 Fitting the Generator function

Lastly, we simply use our model.fit() but with our data augmentation generator, so we using model.fit_generator instead. This starts our training process, but now we're using our augmented dataset.

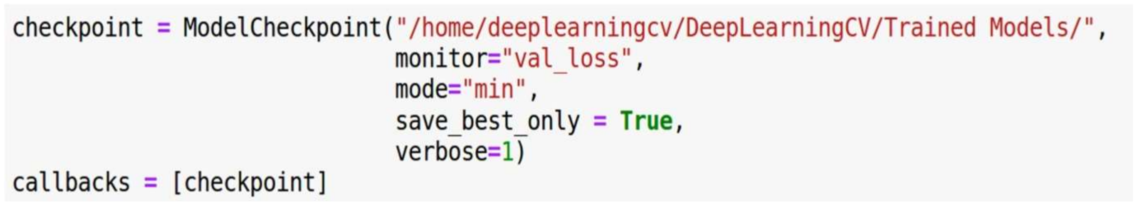
### 3.1.9 Checkpoint Models

● This is a simple but very useful way saving your best model before it starts overfitting (i.e. Loss on our test/validation data starts increases as our Epochs increase)

● Checkpointing allows us keep saving our weights/models after each epoch.

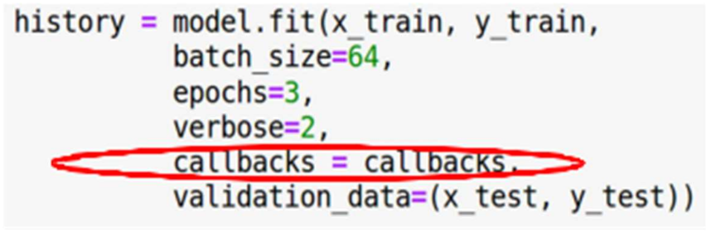
● Keras then allows us to keep saving the 'best' model by monitoring validation loss and accuracy.

### 3.1.9.1 Creating Checkpoint Models



```
checkpoint = ModelCheckpoint("/home/deeplearningcv/DeepLearningCV/Trained Models/",
                             monitor="val_loss",
                             mode="min",
                             save_best_only = True,
                             verbose=1)
callbacks = [checkpoint]
```

Fig 3.16 Checkpoint



```
history = model.fit(x_train, y_train,
          batch_size=64,
          epochs=3,
          verbose=2,
          callbacks = callbacks,
          validation_data=(x_test, y_test))
```

Fig 3.17 Callback

We create a callback that monitors validation loss validation loss validation loss. Here we look at the lowest value and save only the best model.

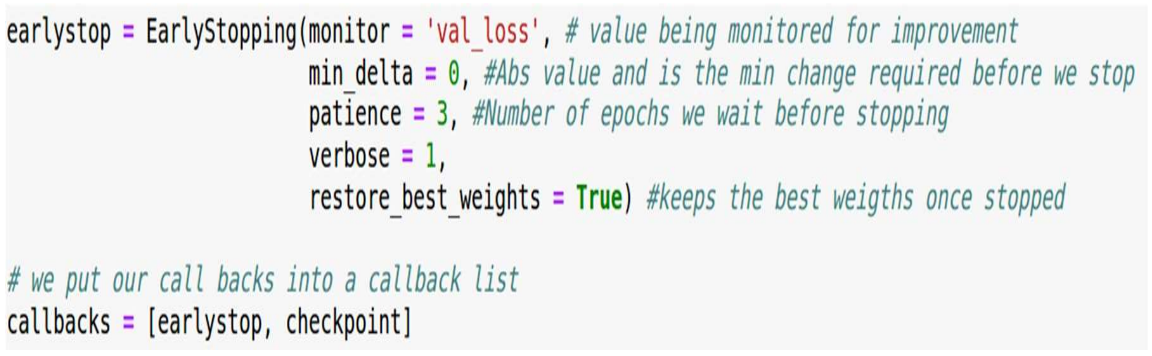
### 3.1.9.2 How to stop Training once Model stops getting better?

● This is another useful feature that can save valuable computing resources.

● Even after executing all 50 epochs, Check Pointing ensures we save the best model, but running a pointless 30 epochs more is wasteful use of time and resources

#### 3.1.9.2.1 Use Early Stopping

Early stopping is Keras Callback that allows us to stop training once the value being monitored (e.g. val_loss) has stopped getting better (decreasing).We use a “patience” parameter to wait X amount of epochs before stopping.

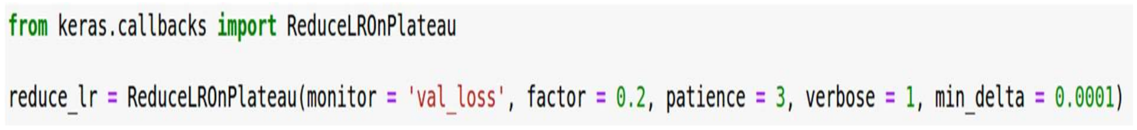


```
earlystop = EarlyStopping(monitor = 'val_loss', # value being monitored for improvement
                          min_delta = 0, #Abs value and is the min change required before we stop
                          patience = 3, #Number of epochs we wait before stopping
                          verbose = 1,
                          restore_best_weights = True) #keeps the best weigths once stopped

# we put our call backs into a callback list
callbacks = [earlystop, checkpoint]
```

Fig 3.18 Early Stop

### 3.1.9.3 Reducing Learning Rate on Plateau

Keras also comes with a Learning Rate adjustment callback. We can avoid having our loss oscillate around the global minimum by attempting to reduce the Learn Rate by a certain fact. If no improvement is seen in our monitored metric (val_loss typically), we wait a certain number of epochs (patience) then this callback reduces the learning rate by a factor.



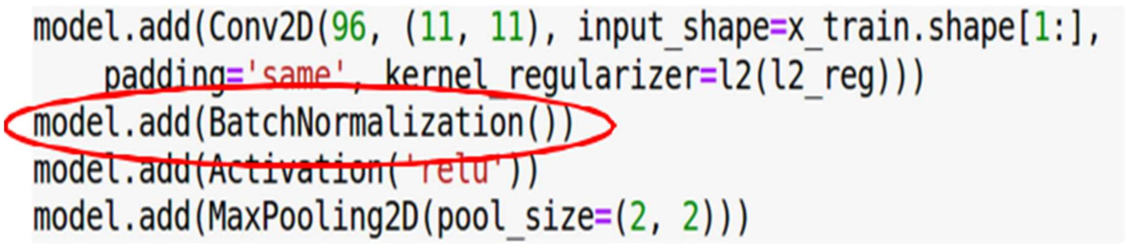
```
from keras.callbacks import ReduceLROnPlateau

reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.2, patience = 3, verbose = 1, min_delta = 0.0001)
```

Fig 3.19 Reducing Learning Rate Function

### 3.1.9.4 Batch Normalization

Batch Normalization is used to normalize the activations of a input tensor before passing it into the next layer in the network. Batch Norm normalizes the output from the activation functions of a selected layer. It then normalizes the output by multiplying it by a parameter and then adding another parameter to this result. The result is that all the activations leaving a batch normalization layer will have approximately zero mean. The weights now don’t become imbalanced with extreme values since normalization is now included in the gradient process.

### 3.1.9.4.1 Batch Normalization Benefits



```
model.add(Conv2D(96, (11, 11), input_shape=x_train.shape[1:],
    padding='same', kernel_regularizer=l2(l2_reg)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

Fig 3.20 Batch Normalization Function

● Batch Normalization reduces the number of Epochs it takes our NN to converge.

● It aids in regularization (reducing overfitting).

● It allows us to improve stability of our training, thus allowing us to use. large learning rates.

● Batch Normalization slows down training.

## 3.2 CLASSIFICATION OF FRUITS USING DEEP CONVOLUTION NEURAL NETWORK

Machine learning requires features and labels to learn from the dataset. Further, these algorithms used to train the model and predict results. Extracting features, applying algorithm, training model and then predicting results is a time consuming process. Machine learning algorithms are unable to classify fruits under shadow, occluded by foliage, branches and during overlap amongst fruits.

To overcome these drawbacks. Deep learning is a new technique use to learn and extract features from images by the help the convolution neural layer (CNN) layer. The time require to classify fruit using CNN is very less as compare to machine learning approach. CNN are complex feed forward neural networks. CNNs is used for image classification and object recognition. A

ConvNet consists of an input and an output layer, as well as multiple hidden layers in between input layer and output layer. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers.

Convolutional layers apply a convolution operation to the input and pass result to the next layer. Pooling means that combines the outputs of neuron clusters at previous layer into a single neuron in the next layer. Generally, max pooling is used to extract maximum value or pixels from each of a cluster of neurons at the prior layer. Fully connected layers connect all neuron in one layer to all neuron in another layer. To construct our own convolution neural network model for our own dataset is time consuming as well as utilize more CPU memory.

To avoid this, in this proposed work Alexnet, a pre-trained model trained on weights of 'ImageNet' is used. This model is used for prediction, feature extraction, and fine-tuning of own dataset which include images of fruits. Fine-tuning is a new approach that replaces and retrain the trained convolution neural network for available dataset.

### 3.2.1 Alexnet Architecture

Alexnet has 8 layers. The first 5 are convolutional and the last 3 are fully connected layers. In between they also have some ‘layers’ called pooling and activation.

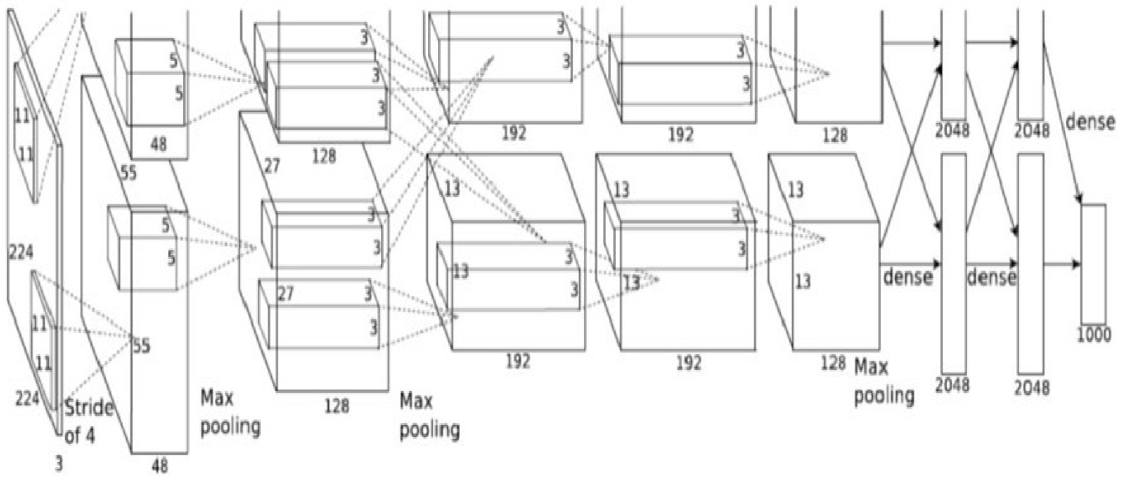


Fig 3.21 Alexnet architecture

The above diagram is the sequence of layers in Alexnet. The problem set is divided into 2 parts, half executing on GPU 1 & another half on GPU 2. The communication overhead is kept low and this helps to achieve good performance overall.

#### 3.2.1.1 Layer 1

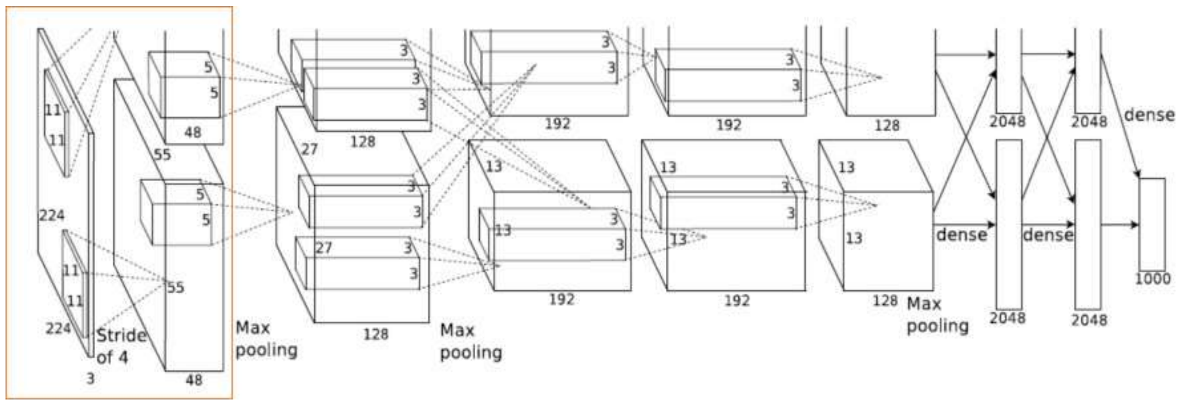


Fig 3.22 Alexnet – layer 1

- Layer 1 is a Convolution Layer,
- Input Image size is – 224 x 224 x 3
- Number of filters – 96
- Filter size – 11 x 11 x 3
- Stride – 4
- Layer 1 Output
  - 224/4 x 224/4 x 96 = 55 x 55 x 96 (because of stride 4)
  - Split across 2 GPUs – So 55 x 55 x 48 for each GPU

There are multiple ways to perform a convolution – it could be a direct convolution – on similar lines to what we've known in the image processing world, a convolution that uses GEMM(General Matrix Multiply) or FFT(Fast Fourier Transform), and other fancy algorithms like Winograd etc

### 3.2.1.2 Layer 2

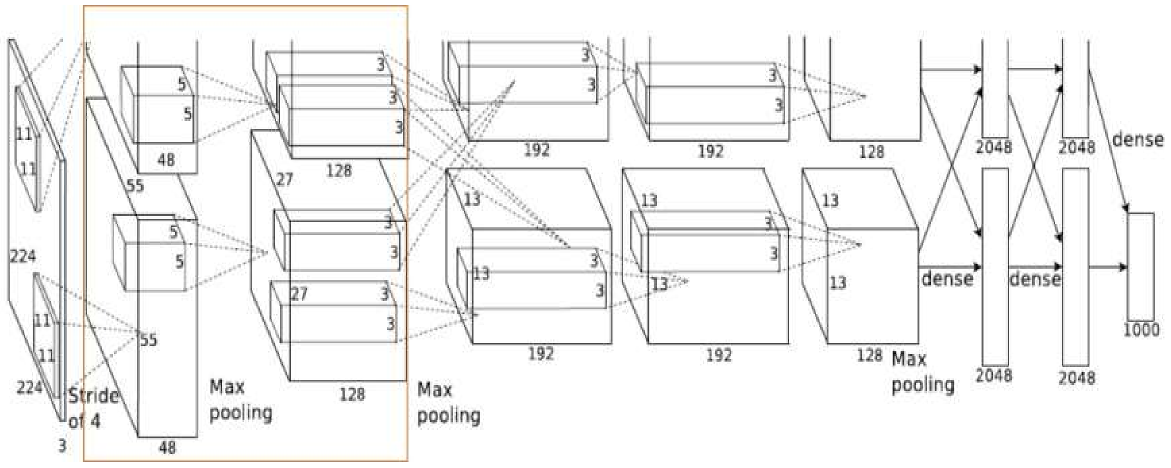


Fig 3.23 Alexnet – layer 2

- Layer 2 is a Max Pooling Followed by Convolution
- Input – 55 x 55 x 96
- Max pooling – 55/2 x 55/2 x 96 = 27 x 27 x 96

- ➢ Number of filters – 256
- ➢ Filter size – 5 x 5 x 48
- ➢ Layer 2 Output
    - 27 x 256
    - Split across 2 GPUs – So 27 x 27 x 128 for each GPU

Pooling is a sub-sampling in a 2×2 window. Max pooling is max of the 4 values in 2×2 window. The intuition behind pooling is that it reduces computation & controls overfitting.

Layers 3, 4 & 5 follow on similar lines.

### 3.2.1.3 Layer 6

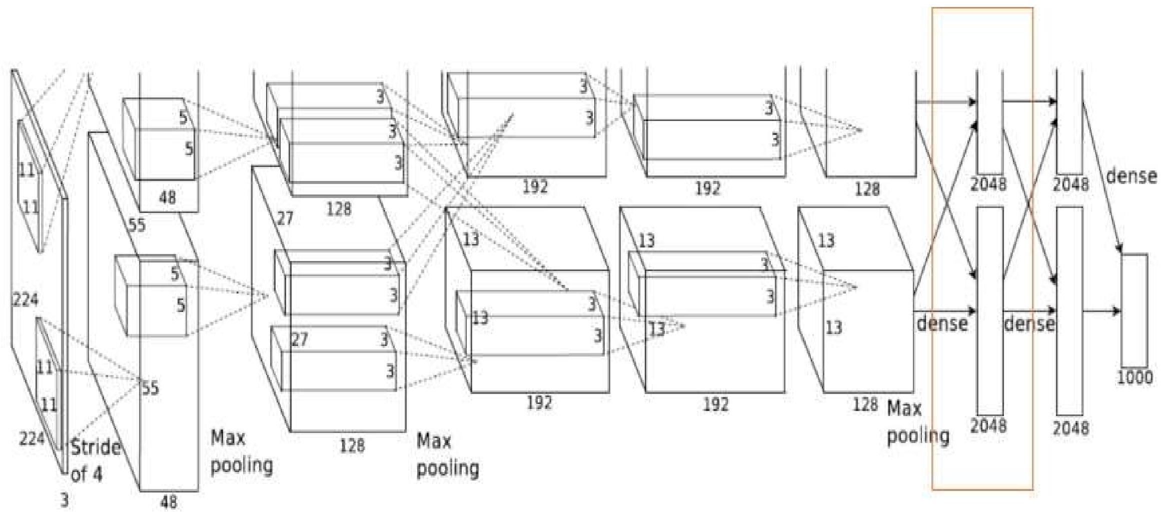


Fig 3.24 Alexnet – layer 6

- ➢ Layer 6 is fully connected
- ➢ Input – 13 x 13 x 128 – > is transformed into a vector
- ➢ And multiplied with a matrix of the following dim – (13x13x128)x2048

GEMV (General Matrix Vector Multiply) is used here:

- Vector X = 1 x (13x13x128)

- Matrix A = (13x13x128) x 2048 – This is an external input to the network

- Output is – 1 x 2048

Layers 7 & 8 follow on similar lines.

### 3.2.2 Uses

The results of AlexNet show that a large, deep convolutional neural network is capable of achieving record-breaking results on a highly challenging dataset using purely supervised learning. Year after the publication of AlexNet we, all the entries in ImageNet competition use the Convolutional Neural Network for the classification task. AlexNet was the pioneer in CNN and open the whole new research era. AlexNet implementation is very easy after the releasing of so many deep learning libraries.

## 3.3 DETECTION AND COUNTING OF FRUITS

The accurate detection of fruits is achieved by using computer vision techniques. First images are taken from dataset is converted into Hue Saturated Value (HSV) image. Further, morphological operations are performed on HSV image to remove noise such as branches, leaves, sky and soil from image. In this proposed work morphological operations such as mask-open and mask-close is performed.

Mask-open operation removes small dots popping randomly on the masked image. Mask-close operation is used to remove the holes present on the output of the mask-open operation. Further, contouring i.e. curve that joins all the continuous point of the same colour is done on previously obtained mask-close image. Last step of the process is to construct a rectangle around contour obtained in the previous step. Count of rectangles in the output image shows count of fruits in the input image. Sequential flow of process is shown below.

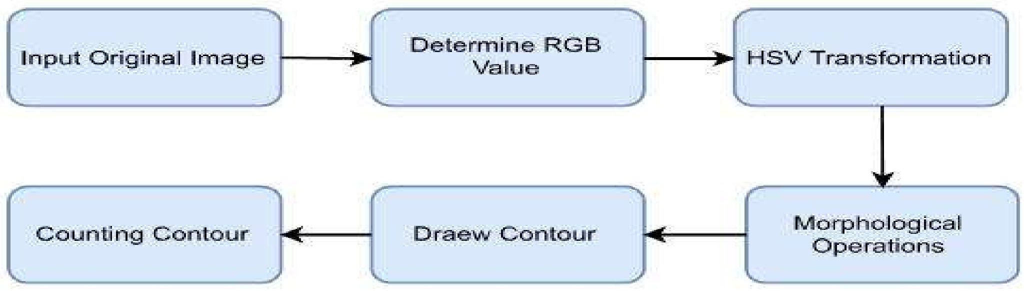


Fig 3.25 Flowchart for Detection and Counting of on tree Fruits.

## 3.4 BLOCK DIAGRAM

Following are the block diagrams of software and hardware.

### 3.4.1 Software

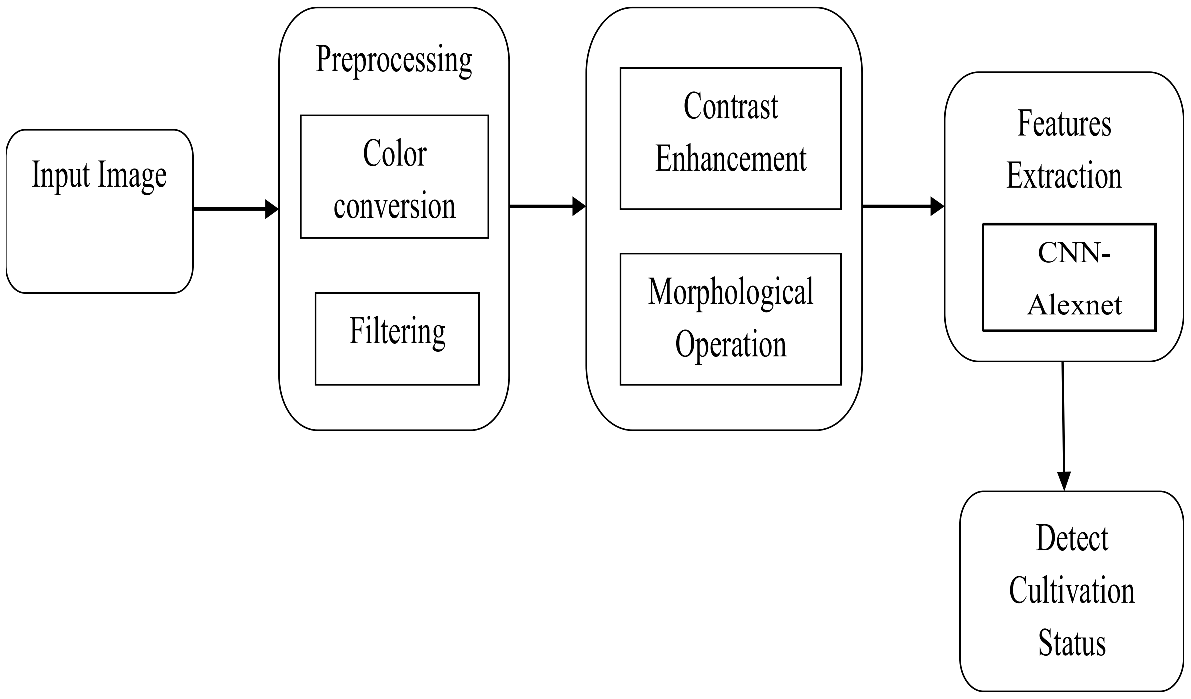


Fig 3.26 Proposed System unit – Software

### 3.4.1.1 Stage I (INPUT IMAGE):

The input image will be feed into the system using pi camera.

### 3.4.1.2 Stage II (PREPROCESSING):

It is the preprocessing stage in which the colour conversion and filtering of the input image is done to get better accuracy.

### 3.4.1.3 Stage III:

In this stage the colour enhancement and morphological operations are done with the input image fed to the system.

### 3.4.1.4 Stage IV:

In this stage using Alexnet the required features are extracted from the input image. Then finally the output will be displayed in the monitor which includes cultivation status and quality and quantity of the fruit.

### 3.4.2 Hardware

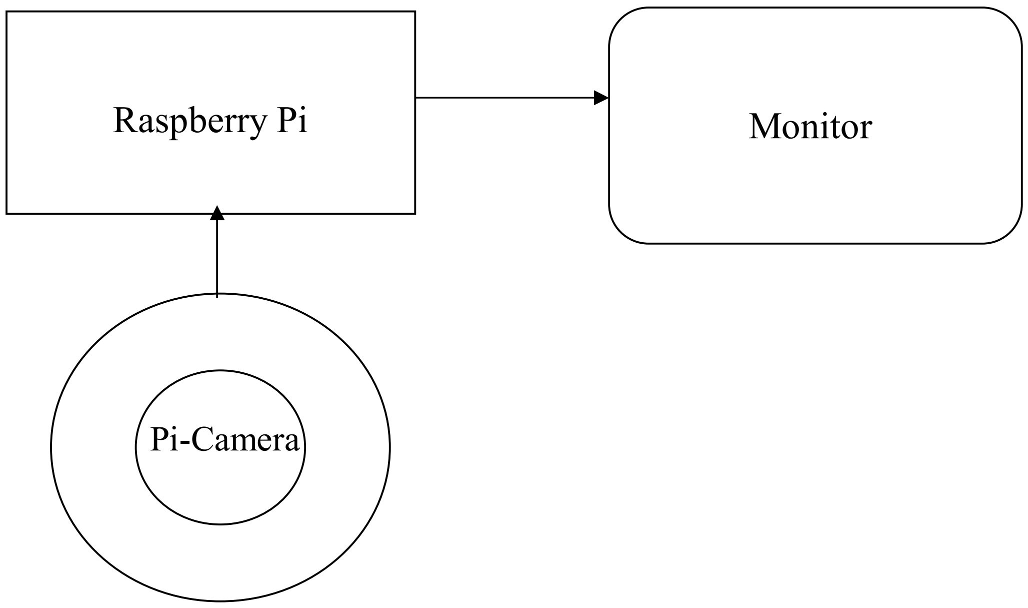


Fig 3.27 Proposed System unit – Hardware

## 3.5 EXPERIMENTATION OF FRUIT CLASSIFICATION,COUNTING AND DISEASE IDENTIFICATION

### 3.5.1 Dataset creation

Machine learning is a technique that learns from features provided to it and predicts accurate results. In this experiment, fruits are classified based on the features such as height, diameter, weight and colour code. A dataset with sufficient features and corresponding label is passed to the machine learning algorithm such as logistic regression, decision tree, k-nearest neighbour, linear discriminant analysis, Gaussian naive bayes and support vector machine. Further, the dataset is divided into train set and test set. Size of train set is more than test set because more the training sample then machine will learn more features from dataset and give accurate prediction. In this methodology, 70% dataset divided in train set and 30% dataset in test set, to learn more features from train set and predict accurate results.

### 3.5.2 Fine-tuning of convolution neural layers

Generally, fine-tuning operation can be performed on model such as Alexnet, ResNet, Inception-v3, AlexNet, GoogleNet and Keras, Tensorflow, Theano is used as backends. In this experiment fine-tuning of convolution neural layers is performed on Alexnet model and Keras is used as backend. Fine-tuning is a process to take a network model that has been already trained for a some particular task, and make use of it to perform a second similar task. This is achieved by replacing the weights of some convolution layers except last output layer of the Alexnet architecture.

Experiment begins with defining fully connected layer and load the ImageNet pre-trained weight to the Alexnet model. Further, truncate the original softmax layer and replace it with own number of class labels for classification task. Further, freeze the weight for the first few layers in network so that they remain intact throughout the fine-tuning process. Perform fine-tuning of layers by minimizing the cross-entropy loss function using stochastic gradient descent algorithm. Next step is to load own dataset, split it into training and testing sets, decide number of batch size, epochs and further start fine-tuning the model. Fine-tuning of convolution layer is trial and error process. Train the model until highest accuracy is achieved. In this experiment model is trained for batch size equal to 110 and 20 epoch. The comparative analysis of obtained results such as training loss, training accuracy, validation loss, validation accuracy obtained during each training phase of experiment.

### 3.5.3 Image pre-processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. In this experiment all input RGB image and videos is converted into HSV image and video. Colour vision can be processed using RGB colour space or HSV colour space. RGB colour space describes colours in terms of the amount of red, green, and blue present. HSV colour space describes colours in terms of the Hue, Saturation, and Value. Colour description plays an integral role, for separation of object from background of the image. The HSV model describes colours similarly, to how the human eye tends to perceive colour. RGB defines colour in terms of a combination of primary colours, whereas, HSV describes colour using more familiar comparisons such as colour, vibrancy and brightness. Hence, the HSV colour model is often preferred over the RGB model. The input image on which pre-processing operation are done to extract required features from the image.

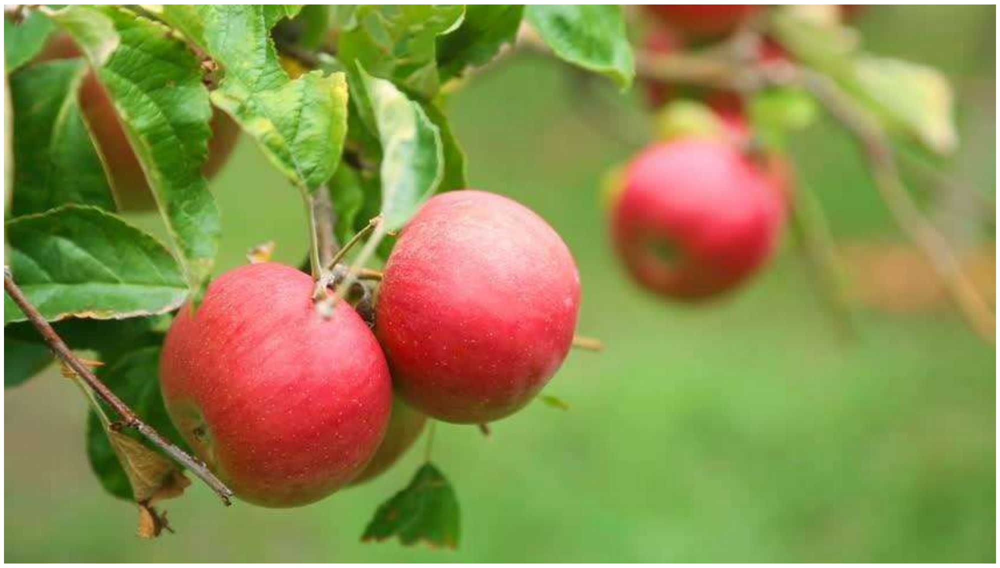


Fig 3.28 Input Image

For object and color based segmentation input RGB image is converted to HSV image. Figure 3.10 shows HSV image obtained after preprocessing operation on RGB image.

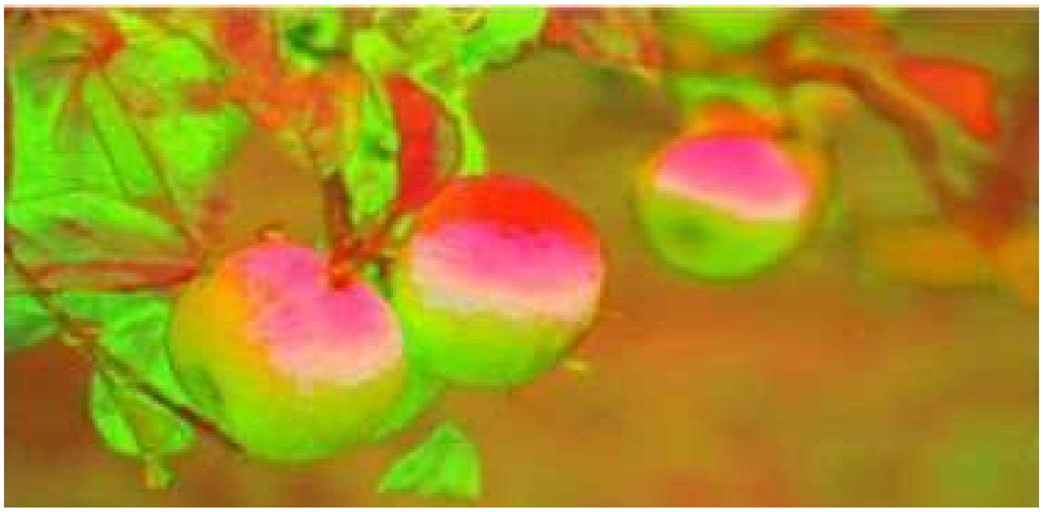


Fig 3.29 HSV Image

### 3.5.4 Morphological operation

Morphological transformations are some simple operations normally performed on binary images. It needs two inputs, one is our original image, second one is called ‘structuring element’ or ‘kernel’ which decides the nature of operation. Two basic morphological operators are Erosion and Dilation. Other variant forms of morphological transforms are ‘opening’ and ‘closing’ which are also perform on image to extract shape of required object from the image. Structuring element used in this experiment is of size 5 X 5 which is smaller than regular input image. Erosion is done to remove background noises from the image. In this experiment a kernel of size 5 X 5 is move across whole image and detect require colour from the whole image.

#### 3.5.4.1 Erosion:

Erosion operation erodes away the boundaries of foreground object. This operation always try to keep foreground in white. In this experiment the kernel slides through the image as in 2D convolution. A pixel in the original image either 1 or 0 will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded and made to zero. All the pixels near boundary will be discarded depending upon the size of kernel. So the thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small white noises so that will detach two connected objects. Following figure 5.4 shows result obtained by performing erosion operation on HSV image.

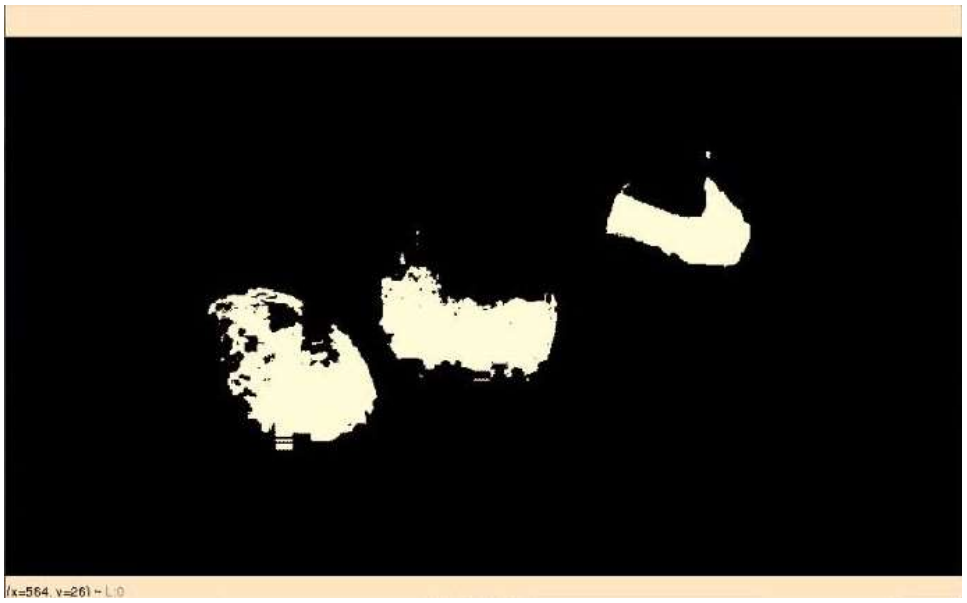


Fig 3.30 Mask image

### 3.5.5 Opening Operations

Opening is just another name of erosion. It is useful in removing noise, In the previous experiment small dots are randomly popping in the image. In this experiment small dots are remove i.e background noise is remove by moving kernel of 5 X 5 across the whole image. Following figure 5.5 shows result obtained by performing opening operation on mask image.

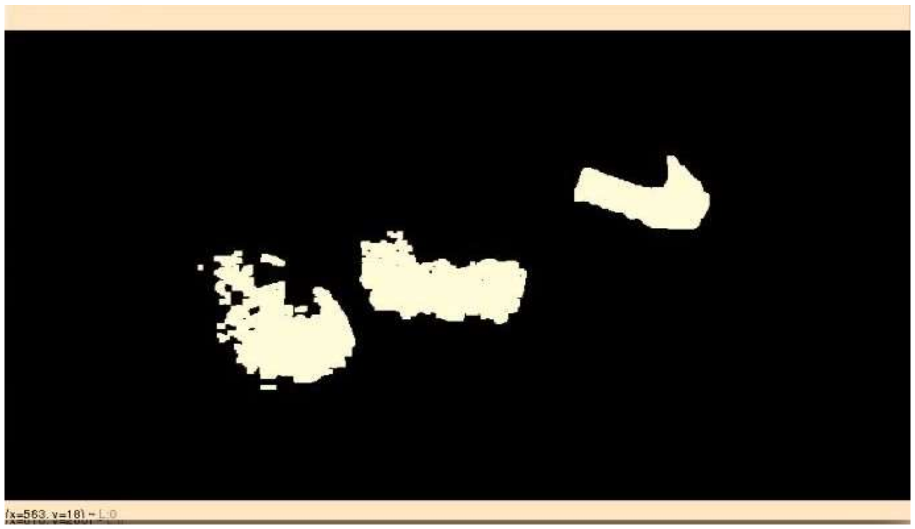


Fig 3.31 Mask open

### 3.5.6 Closing Operation

Closing is reverse of opening. It is useful in closing small holes inside the foreground objects, or small black points on the object. In the previous experiment small holes are present in the actual object which is removed by moving kernel of size 20 X 20 across whole image. Following figure shows result obtained by performing closing operation on mask open image.

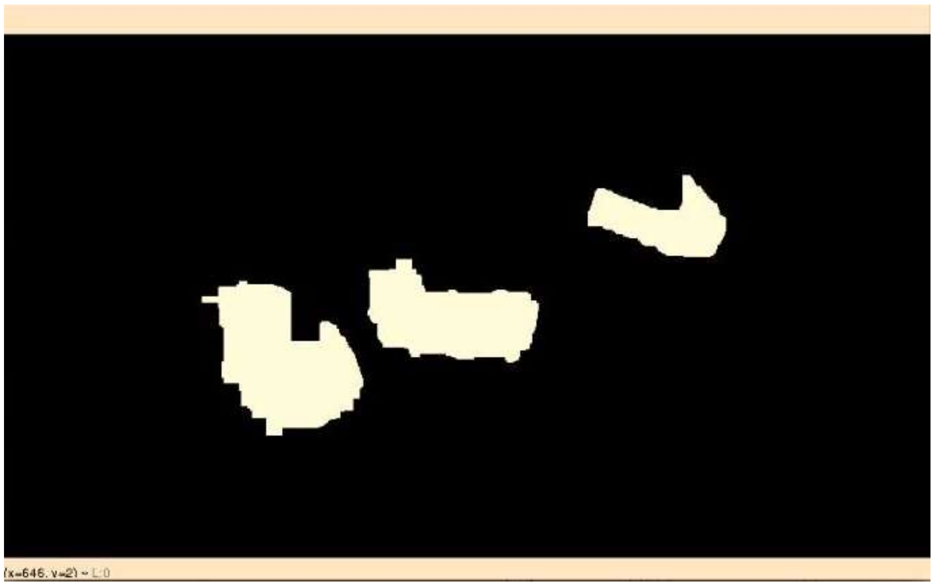


Fig 3.32 Mask close

### 3.5.7 Contouring Operations

Curve that joins all the continuous point of the same color is called as contouring. Contour is used to denote object boundary. Contouring is done on the mask close image to detect accurate object. Following figure shows result obtained by performing contouring operation on mask close image.

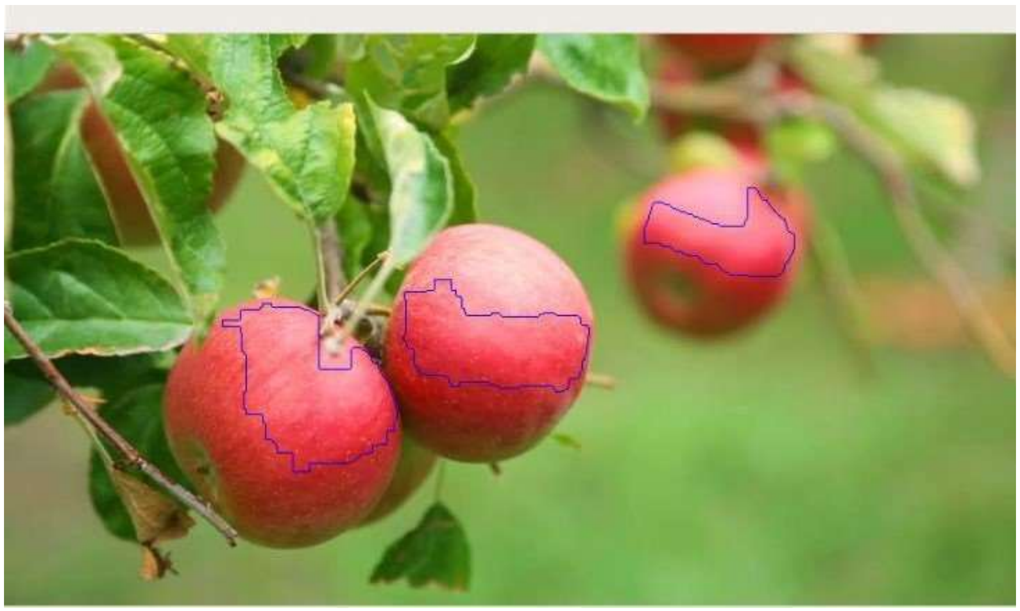


Fig 3.33 Contour operation

### 3.5.8 Fruit Counting and Disease Identification

Last step of the whole experiment is to count fruits present in the image and video. Construct a rectangle across boundary of contour obtained in the contouring experiment. Number of count of rectangles in the output image shows count of fruits in the input image. The features of fruits are extracted using Alexnet and used for disease detection.

All experiments concludes that all the operations such as image pre-processing, noise removal, image segmentation and contouring are performed sequentially with the help of code written in python. The various libraries that are used in python for the experiment are opencv, numpy, and keras. The final output obtained is also an image or video which shows the detected fruit along with the fruit count. The execution time required to perform these experiment is around 7 second. An email is sent to the respective person through IoT cloud server if disease detected.

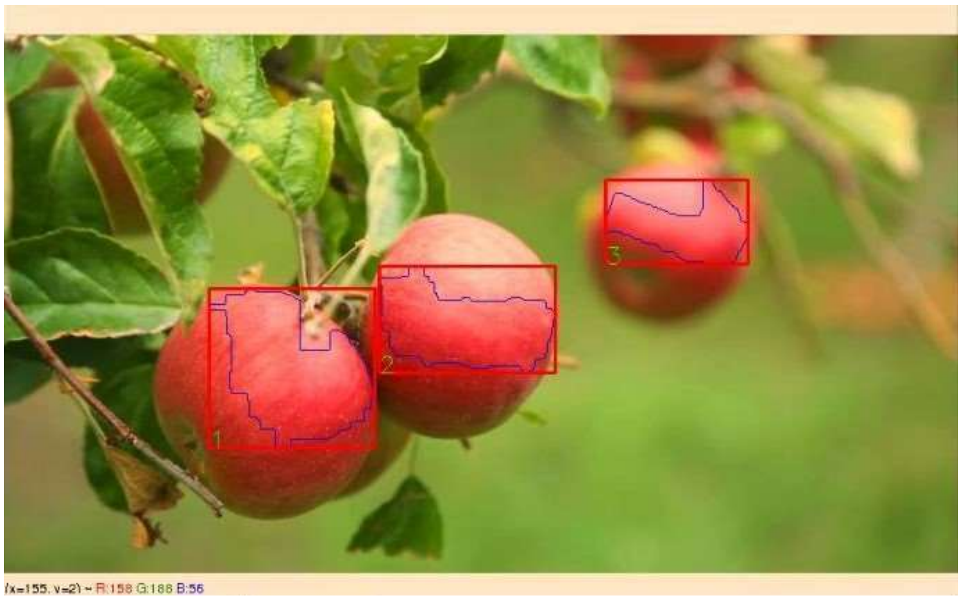


Fig 3.34 Count and disease identification of fruits in an input image

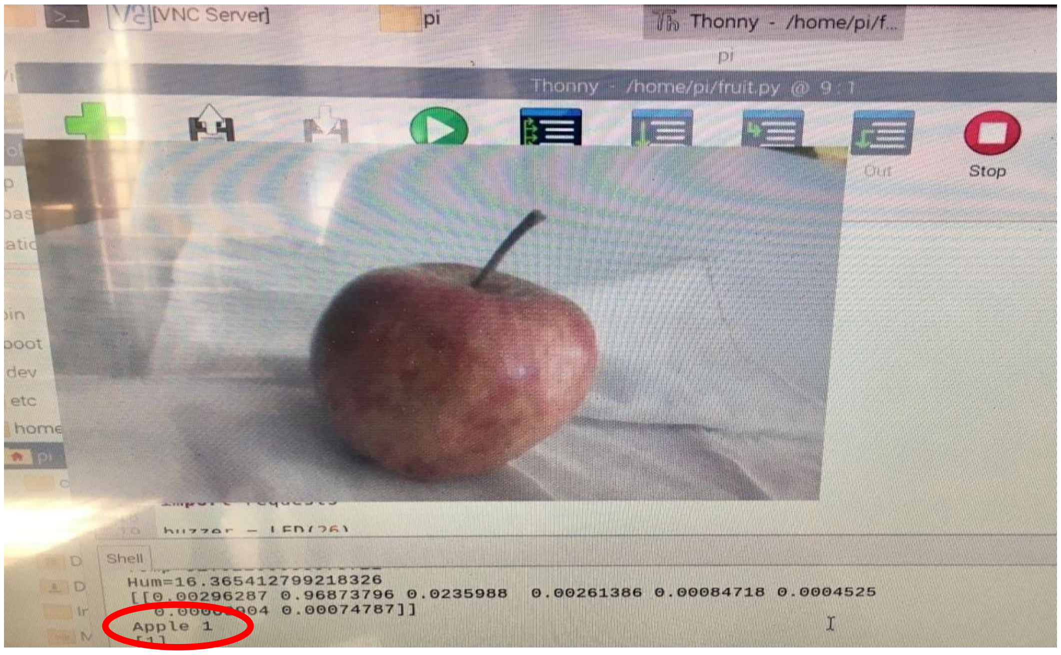
# CHAPTER IV

# RESULTS AND OUTPUTS

Developed methodology consists of three steps. In the first step, dataset of each fruit images is divided into training images and testing images. In the second step, dataset i.e. group of images are passed to the layer of convolution neural network. It is impossible to build own convolution neural network, because it takes a lot of time and consume lot of CPU memory and graphics. Therefore, a pre-trained model called Alexnet is used. A new model is created by fine-tuning of Alexnet model. In the last step, classification of fruit is done based on the new model created during the second phase. Following are the results obtained by using new trained model which classify fruits from image.

## 4.1 DETECTION OF APPLE

The below picture shows the result of detecting one apple.

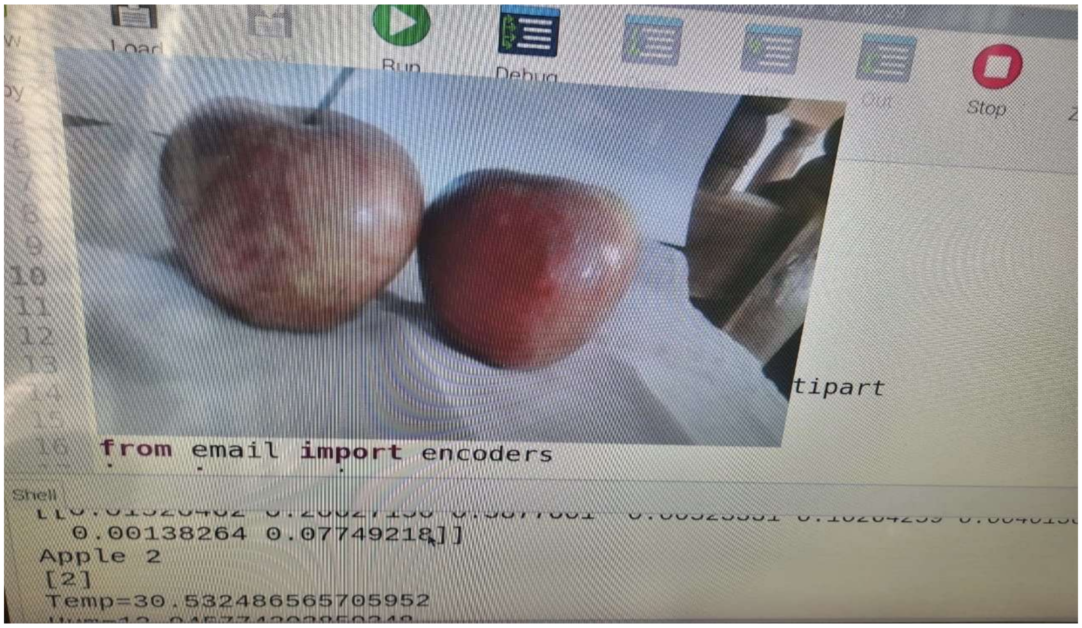


```
buzzer = LED(26)
```

```
Hum=16.365412799218326
[[0.00296287 0.96873796 0.0235988  0.00261386 0.00084718 0.0004525
  0.000[illegible]904 0.00074787]]
Apple 1
[1]
```

Fig 4.1 Detection of 1 Apple

The below picture shows the result of detecting two apples.



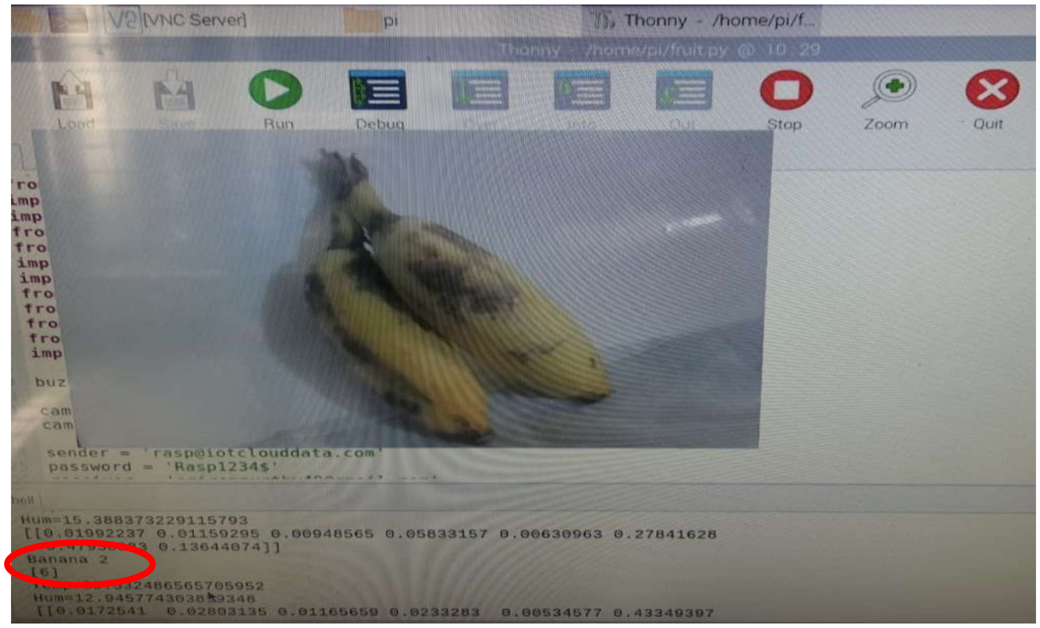
```
from email import encoders
```

```
 0.00138264 0.07749218]]
Apple 2
[2]
Temp=30.532486565705952
```

Fig 4.2 Detection of 2 Apples

## 4.2 DETECTION OF BANANA

The below picture shows the result of detecting bananas.



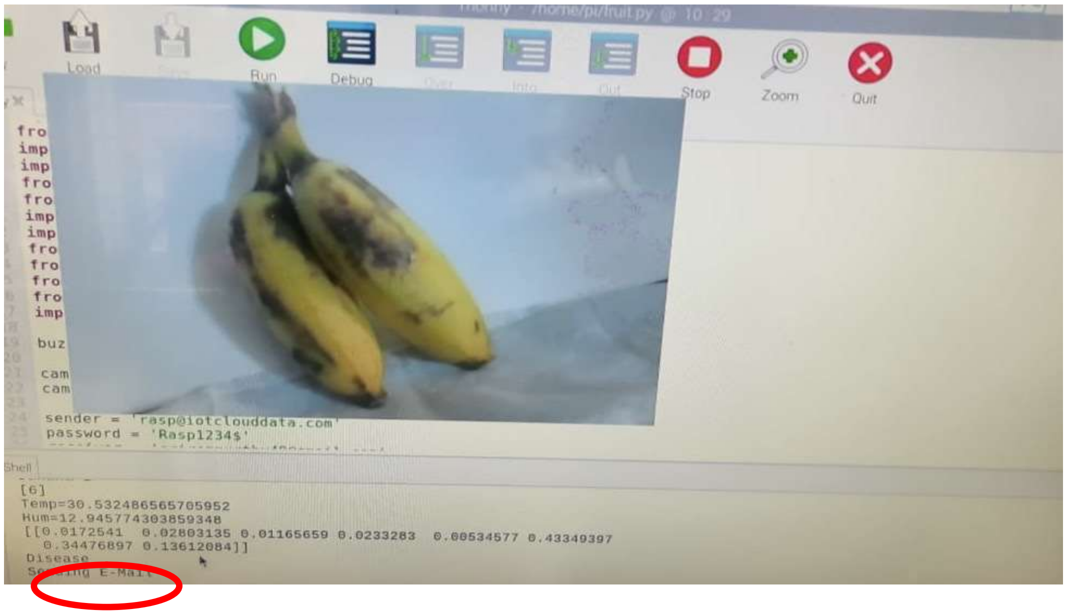
```
sender = 'rasp@iotclouddata.com'
password = 'Rasp1234$'
```

```
Hum=15.388373229115793
[[0.01992237 0.01159295 0.00948565 0.05833157 0.00630963 0.27841628
  0.13644074]]
Banana 2
[6]
 2486565705952
Hum=12.9457743038 9348
[[0.0172541  0.02803135 0.01165659 0.0233283  0.00534577 0.43349397
```

Fig 4.3 Detection of Banana

## 4.3 DETECTION OF DISEASED BANANA

The below picture shows the result of detecting rotten bananas.



```
fro
imp
imp
fro
fro
imp
imp
fro
fro
fro
fro
imp

buz

cam
cam

sender = 'rasp@iotclouddata.com'
password = 'Rasp1234$'
```

```
[6]
Temp=30.532486565705952
Hum=12.945774303859348
[[0.0172541  0.02803135 0.01165659 0.0233283  0.00534577 0.43349397
  0.34476897 0.13612084]]
Disease
Sending E-Mail
```

Fig 4.4 Detection of Diseased Banana

## 4.4 HARDWARE

The below picture represents the hardware of this project.

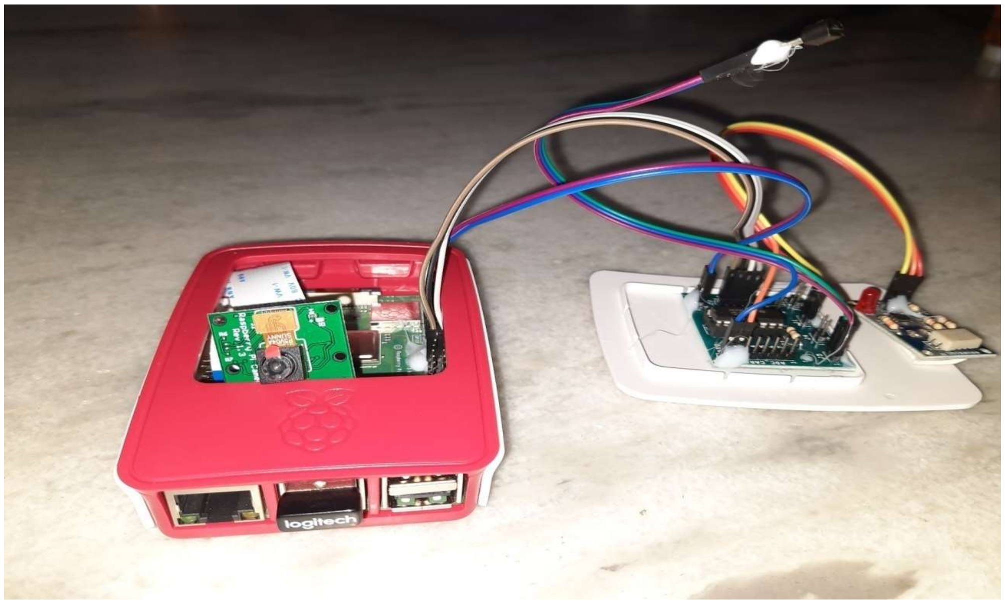


Fig 4.5 Hardware Unit

# CHAPTER V

# CONCLUSION AND FUTURE ENHANCEMENT

This proposed work classifies fruit based on the features such as height, width, diameter and colour code of fruit. All these features are passed to the supervised machine learning algorithms. Machine learning algorithms are unable to classify fruits under shadow, hidden by leaves, branches and during overlap amongst fruits. To overcome these drawbacks ‘Deep Learning’ a new technique has been used to learn and extract features from images by the help of the convolution neural layers. A new fruit classification model is created by the help of fine-tuning of ‘Alexnet’ a pre-trained model. Image pre-processing, noise removal, image segmentation and contouring are used sequentially to detect and count the number of fruits present on the tree from the images as well as videos.

## 5.1 ADVANTAGES

- This method helps in speed up the process and reduces time.
- It improves accuracy and efficiency.
- Best suited for different illuminant condition.

## 5.2 APPLICATIONS

- In the food industry to segregate healthy fruits from diseased fruits and can only the healthy fruits be further processed.
- Harvesting Good and healthy fruits in the farm.
- Segregating fruits based on quality and grade for marketing.
- Grouping fruits according to quality for export.

## 5.3 FUTURE ENHANCEMENT

In future, this proposed system can be further extended to develop a mobile application that helps farmers to classify, detect and count fruits. To extend this proposed work, different categories of fruit images should be used for training purpose. The adoption of this new methodology helps the farmers in robotic harvesting.

# REFERENCES

[1] Michael Halstead , Christopher McCool , Simon Denman , Tristan Perez , and Clinton Fookes “Fruit Quantity and Ripeness Estimation Using a Robotic Vision System” IEEE ROBOTICS AND AUTOMATION LETTERS, VOL. 3, NO. 4, OCTOBER 2018.

[2] Jnaneshwar Das, Camillo J. Taylor, James Underwood “Monocular Camera Based Fruit Counting and Mapping With Semantic Data Association” in IEEE ROBOTICS AND AUTOMATION LETTERS, VOL. 4, NO. 3, JULY 2019.

[3] Xu Liu, Steven W. Chen, Shreyas Aditya, Nivedha Sivakumar, Sandeep Dcunha, Chao Qu, Camillo J. Taylor, Jnaneshwar Das, and Vijay Kumar “Robust Fruit Counting: Combining Deep Learning, Tracking, and Structure from Motion” in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Madrid, Spain, October 1-5, 2018.

[4] Chi Cuong Tran , Dinh Tu Nguyen , Hoang Dang Le , Quoc Bao Truong “Automatic dragon fruit counting using adaptive thresholds for image segmentation and shape analysis” in 2017 4th NAFOSTED Conference on Information and Computer Science.

[5] Susovan Jana, Ranjan Parekh “Automatic Fruit Recognition from Natural Images using Colour and Texture Features “in 2017 Devices for Integrated Circuit (DevIC), 23-24 March, 2017, Kalyani, India.

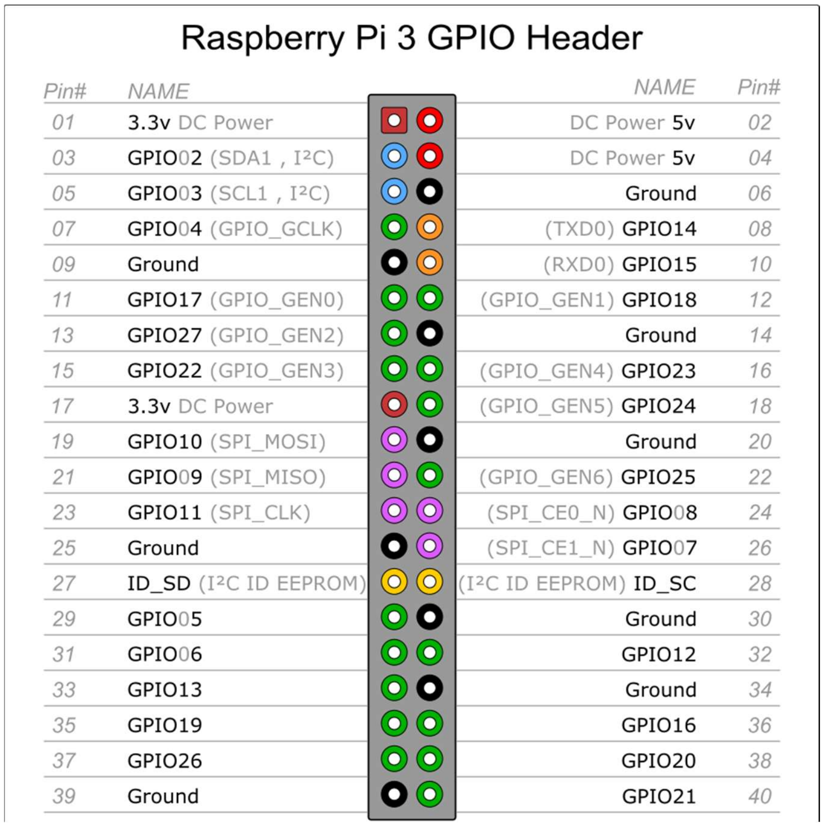
[6] Manali R. Satpute ,Sumati M. Jagdale “Automatic Fruit Quality Inspection System “ in 2016 IEEE conference.

[7] Anisha Syal, DivyaGarg, Shanu Sharma “Apple Fruit Detection and Counting Using Computer Vision Techniques “in 2014 IEEE International Conference on Computational Intelligence and Computing Research.

[8] Hongshe Dang, Jinguo Song, Qin Guo "A Fruit Size Detecting and Grading System Based on Image Processing"in 2010 Second International Conference on Intelligent Human-Machine Systems and Cybernetics.

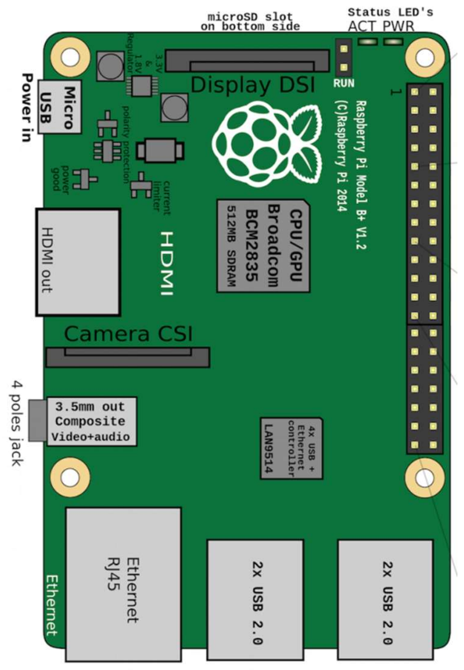
# APPENDIX

## Raspberry pi 3 pin configuration



Raspberry Pi 3 GPIO Header

| Pin# | NAME | NAME | Pin# |
|---|---|---|---|
| 01 | 3.3v DC Power | DC Power 5v | 02 |
| 03 | GPIO02 (SDA1 , I²C) | DC Power 5v | 04 |
| 05 | GPIO03 (SCL1 , I²C) | Ground | 06 |
| 07 | GPIO04 (GPIO_GCLK) | (TXD0) GPIO14 | 08 |
| 09 | Ground | (RXD0) GPIO15 | 10 |
| 11 | GPIO17 (GPIO_GEN0) | (GPIO_GEN1) GPIO18 | 12 |
| 13 | GPIO27 (GPIO_GEN2) | Ground | 14 |
| 15 | GPIO22 (GPIO_GEN3) | (GPIO_GEN4) GPIO23 | 16 |
| 17 | 3.3v DC Power | (GPIO_GEN5) GPIO24 | 18 |
| 19 | GPIO10 (SPI_MOSI) | Ground | 20 |
| 21 | GPIO09 (SPI_MISO) | (GPIO_GEN6) GPIO25 | 22 |
| 23 | GPIO11 (SPI_CLK) | (SPI_CE0_N) GPIO08 | 24 |
| 25 | Ground | (SPI_CE1_N) GPIO07 | 26 |
| 27 | ID_SD (I²C ID EEPROM) | (I²C ID EEPROM) ID_SC | 28 |
| 29 | GPIO05 | Ground | 30 |
| 31 | GPIO06 | GPIO12 | 32 |
| 33 | GPIO13 | Ground | 34 |
| 35 | GPIO19 | GPIO16 | 36 |
| 37 | GPIO26 | GPIO20 | 38 |
| 39 | Ground | GPIO21 | 40 |



## Pi camera module