A Project Report on

**IMPLEMENTATION OF GREEN LEAF DISEASE DETECTION USING RASPBERRY PI**

*Submitted in partial fulfillment of the requirement for the award of degree of*

**BACHELOR OF TECHNOLOGY**

*In*

**ELECTRONICS AND COMMUNICATION ENGINEERING**

*Submitted by*

**Ms. M.NAGASRI YELESWARI - 18FE1A0471**

**Ms. B. SANDHYA RANI - 19FE5A0404**

**Ms. M.RISHITA KALPANA - 18FE1A0472**

**Mr. B .PAVAN KUMAR - 19FE5A0405**

*Under the esteemed guidance of*

**Mr. RAMA GOPI, M.Tech.**

 Assistant Professor

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**VIGNAN’S LARA INSTITUTE OF TECHNOLOGY &SCIENCE**

(An ISO 9001:2015 Certified, Approved by AICTE, Affiliated to JNTU, KAKINADA)

VADLAMUDI-522213, GUNTUR Dist. ANDHRA PRADESH.

2018-2022

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING VIGNAN’S LARA INSTITUTE OF TECHNOLOGY & SCIENCE**

(An ISO 9001:2015 Certified, Approved by AICTE, Affiliated to JNTUK, KAKINADA)

VADLAMUDI – 522213, GUNTUR Dist. ANDHRA PRADESH.

**CERTIFICATE**

This is to certify that main project work entitled “**IMPLEMENTATION OF GREEN**

**LEAF DISEASE DETECTION USING RASPBERRY PI**” is a bona fide work done by

**M. NAGASRI YELESWARI (18FE1A0471), B. SANDHYA RANI (19FE5A0404), M. RISHITA KALPANA (18FE1A0472)** and **B. PAVAN KUMAR (19FE5A0405),** under my guidance and submitted in partial fulfillment of requirement for the award of degree of **Bachelor of Technology** in **Electronics and Communication Engineering** by **Jawaharlal Nehru Technological University, Kakinada** during the year **2018-2022.**

Project Guide Head of the Department

**Mr. B. RAMA GOPI, M.Tech,**  **Dr. B. Harish (PhD)**

Assistant Professor Professor

**External examiner**

**ACKNOWLEDGEMENT**

We are most thankful to our parents who stood as pillars of motivation and for the way they influenced and molded our lives.

We are more thankful to our chairman **Dr. Lavu Rathaiah** who helped us to have a technical incubation by providing the required infrastructure.

We are very much thankful to our principal **Dr. k.Phaneendra Kumar** who extended a timely help at each and every step of our academic career.

We are very thankful to our Head of the department **Dr.B. Harish** an amicable person who supported us very much and helped to a maximum extent and made this project successful.

We are very thankful to our beloved coordinators **Mr.K. Vijaya Vardhan** and **Dr. Manir Ahmed** for inspiring all the way and arranging all the facilities and resources needed for the project. Their efforts in this aspect are beyond the preview of the acknowledgement.

We are heartily thankful to our project guide **Mr. B. Rama Gopi**, Assistant Professor, the person with vibrant knowledge and amicable by nature who laid a best guide lines and for the efforts made by him to make the project a successful one.

Finally, we are thankful to each and every faculty member technical and non- technical, friends and all the persons who helped us directly or indirectly in making our project a successful one.

**M.NAGARI YELESWARI (18FE1A0471)**

**B.SANDHYA RANI (19FE5A0404)**

**M.RISHITA KALPANA (18FE1A0472)**

**B.PAVAN KUMAR (19FE5A0405)**

**DECLARATION**

We here by declare that the work described in this project work ,entitled “**IMPLEMENTATIONOFGREENLEAFDISEASEDETECTIONUSINGRASPBERRYPI**” which is submitted by us in partial fulfillment for the award of **Bachelor of Technology** (B.Tech)in the **Department of Electronics & communication Engineering** to the **Vignan’s Lara Institute of Technology and science** affiliated to **Jawaharlal Nehru Technological University Kakinada, Andhra Pradesh,** is the result of work done by us under the guidance of **Mr.B.RAMA GOPI**, Assistant Professor ,in the department of **ECE.**

The work is original and has been submitted for any Degree /Diploma of this or any other university.

**M. NAGASRI YELESWARI (18FE1A0471)**

**B. SANDHYA RANI (19FE5A0404)**

**M. RISHITAKALPANA (18FE1A0472)**

**B. PAVAN KUMAR (19FE5A0405)**

Place: Vadlamudi.

Date

**ABSTRACT**

For the detection and prevention of disease of plants from getting spread, this project proposed a system using image processing. For the image analysis, Conventional Neural Networks was used. It has many advantages for the use in big farm of crop and thus it automatically detects signs of is ease whenever they appear on leaves of the plant. In pharmaceutical research of leaf disease detection is necessary and important topic for research because it has advantage in monitoring crops in field at the forma and thus it automatically detects symptoms of disease by image processing by CNN algorithm. The term disease means the type of damage to the plants. This project provides the best method for detection of plant disease using image processing and alerting about the disease caused by sending to IoT Server and displaying the name of the disease and precautions on the mobile application of the owner of the system. It will reduce the cost required for the pesticides and other products. This will lead to increase in productivity of the farming.

**INDEX**

**CONTENTS PAGE NO**

**ABSTARCT v**

**CONTENTS vi-vii**

**CHAPTER 1: INTRODUCTION 1-5**

1.1 Introduction 1

1.2 History 1-2

1.3 Characteristics of Embedded systems 2-3

1.4 Peripherals 3-4

1.5 CNN Algorithm 4-5

**CHAPTER 2: LITERATURE SURVEY 6-9**

**CHAPTER 3: EXSISTING SYSTEM 10-13**

3.1 Description 10

3.2 Block Diagram 10-11

3.3 Algorithm1: Basic steps describing the existing algorithm 12

3.4 Advantages 13

3.5 Disadvantages 13

**CHAPTER 4: PROPOSED SYSTEM 14-17**

4.1 Objective 14

4.2 Diseases 14-15

4.3 CNN Algorithm 16-17

4.4 Advantages 17

4.5 Applications 17

**CHAPTER 5: HARDWARE AND SOFTWARE DESCRIPTION 18-45**

5.1 Hardware Components 18-27

5.1.1 Raspberry Pi 18-20

5.1.2 Camera Module 21-25

5.1.3 Monitor display 25-26

5.1.4 Power Supply 26-27

5.2 Software Components 27-45

5.2.1 Open CV with Python 27-34

5.2.2 CNN Algorithm 34-45

**CHAPTER 6: RESULT ANALYSIS 46**

**CHAPTER 7: CONCLUSION 47**

**REFERENCES 48-49**

**APPENDIX 50-67**

**LIST OF FIGURES**

**Name of the figure Page No.**

Fig 1.1 Analysis of CNN. 4

Fig 3.1 Block Diagram of Existing System. 10

Fig 3.2 The basic procedure of the existing image processing-based

diseased detection solution. 9

Fig 3.3 Architecture of ANN Algorithm. 10

Fig 4.1 Leaf Miner Disease. 11

Fig 4.2 Yellow Spot Disease. 13

Fig 4.3 Basic structure of CNN Algorithm. 15

Fig 5.1 Raspberry Pi. 16

Fig 5.2 Physical specifications. 17

Fig 5.3 Specifications of Raspberry Pi. 18

Fig 5.4 Camera Module. 19

Fig 5.5 Monitor Display. 23

Fig 5.6 Adaptor. 24

Fig 5.7 Filter being applied on 10\*10 image. 29

Fig 5.8 Blurred Image. 33

Fig 5.9 A stride one 3x3 convolutional kernel acting on a 8x8 input

image, outputting an 8x8 filter/channel. 34

Fig 5.10 Kernel scanning over the values in input matrix. 35

Fig 5.11 Visualization of channels/feature maps created from a

layer of convolution Kernel. 36

Fig 5.12 3\*3\*3 convolution kernel acting on a 3-channel input. 37-38

Fig 5.13 VGG-16 network architecture. 42

Fig 6.1 Accessing Images. 46

Fig 6.2 Displaying disease details. 46

Fig 6.3 Displaying solutions. 46

**CHAPTER-1**

**INTRODUCTION**

* + **1.1 Introduction**

Agriculture is the main stay of the Indian economy negative effect on our environment. The use of chemical pesticides has led to enormous levels of chemical build up in our environment, in soil, water, air, in animals and even in our own bodies. Artificial fertilizers give on a short-term effect on productivity but a longer-term negative effect on the environment, where they remain for years after leaching and running off, contaminating groundwater. Another negative effect of this trend has been on the fortunes of the farming communities worldwide. Despite this so-called increased productivity, farmers in practically every country around the world have seen a downturn in their fortunes. This is where organic farming comes in. Organic farming has the capability to take care of each of these problems. The central activity of organic farming relies on fertilization, pest and disease control.

Plant disease detection through naked eye observation of the symptoms on plant leaves, incorporate rapidly increasing of Immense commercialization of an agriculture has created a survey cultivated Crops and their existing psychopathological problems, even experienced agricultural experts and plant pathologists may often fail to successfully diagnose specific diseases, and are consequently led to mistaken conclusions and concern solutions. An automated system designed to help identify plant diseases by the plant’s appearance and visual symptoms could be of great help to amateurs in the agricultural process. This will be proven as useful technique for farmers and will alert the man at the right time before spreading of the disease over large area.

**1.2 History**

* In the earliest years of computers in the 1940s, computers were sometimes dedicated to a single task, but were too large to be considered "embedded". Over time however, the concept of programmable controllers developed from a mix of computer technology, solid state devices, and traditional electromechanical sequences.
* The first recognizably modern Embedded System was the Apollo Guidance Computer, developed by Charles Stark Draper at the MIT Instrumentation Laboratory. At the project's inception, the Apollo guidance computer was considered the riskiest item in the Apollo project. The use of the then new monolithic integrated circuits, to reduce the size and weight, increased this risk.
* The first mass-produced Embedded System was the Automatics D17 guidance computer for the Minuteman (missile), released in 1961. It was built from transistor logic and had a hard disk for main memory.
* When the Minuteman II went into production in 1966, the D-17 was replaced with a new computer that was the first high-volume use of integrated circuits. This program alone reduced prices on quad NAND gate ICs from $1000/each to $3/each, permitting their use in commercial products. Since these early applications in the 1960s, Embedded Systems have come down in price. There has also been an enormous rise in processing power and functionality. For example, the first microprocessor was the Intel 4004, which found its way into calculators and other small systems, but required external memory and support chips.
* In 1978 National Engineering Manufacturers Association released the standard for a programmable microcontroller. The definition was an almost any computer-based controller. They included single board computers, numerical controllers, and sequential controllers in order to perform event-based instructions.
* By the mid-1980s, many of the previously external system components had been integrated into the same chip as the processor, resulting in integrated circuits called microcontrollers, and widespread use of Embedded Systems became feasible. As the cost of a micro controller fell below $1, it became feasible to replace expensive knob based analog components such as potentiometers and variable capacitors with digital electronics controlled by a small micro controller with up/down buttons or knobs.
* By the end of the 80s, Embedded Systems were the norm rather than the exception for almost all electronics devices, a trend which has continued since.
  1. **Characteristics of Embedded system**
* An Embedded System is any computer system hidden inside a product other than a computer.
* They will encounter a number of difficulties when writing Embedded System software in addition to those we encounter when we write applications.
* Throughput – Our system may need to handle a lot of data in a short period of time.
* Response–Our system may need to react to events quickly.
* Testability–Setting up equipment to test embedded software can be difficult.
* Debuggability–Without a screen or a keyboard, finding out what the software is doing wrong (other than not working) is a troublesome problem.
* Reliability – Embedded Systems must be able to handle any situation without human intervention.
* Memory space – Memory is limited on Embedded Systems, and you must make the software and the data fit into whatever memory exists.
* Program installation – You will need special tools to get your software into Embedded Systems.
* Power consumption – Portable systems must run on battery power, and the software in these systems must conserve power.
* Processor hogs – Computing that requires large amounts of CPU time can complicate the response problem.
* Cost – Reducing the cost of the hardware is a concern in many Embedded System projects; software often operates on hardware that is barely adequate for the job.
* Embedded Systems have a microprocessor/ micro controller and a memory. Some have a serial port or a network connection. They usually do not have keyboards, screens or disk drives.
  1. **Peripherals**

Embedded Systems talk with the outside world via peripherals, such as:

* Serial Communication Interfaces (SCI): RS-232, RS-422, RS-485etc.
* Synchronous Serial Communication Interface: I²C, JTAG, SPI, SSC and ESSI.
* Universal Serial Bus (USB).
* Networks: Controller Area Network, Lon Works, etc.
* Timers: PLL(s), Capture/Compare and Time Processing Units.
* Discrete IO: aka General-Purpose Input Output (GPIO).

**1.5 CNN Algorithm**

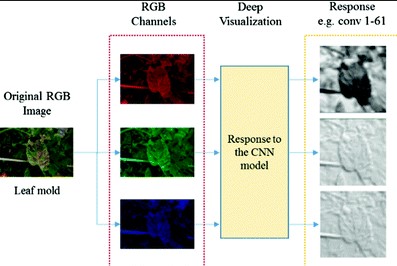


Fig 1.1 Analysis of CCN algorithm.

Deep learning constitutes are cent, modern technique for image processing and data analysis, with accurate results and large potential. As deep learning has been successfully applied in various domains, it has recently entered also the domain of agriculture.

So we will apply deep learning to create an algorithm for automated detection and classification of plant leaf diseases. Nowadays, Convolutional Neural Networks are considered as the leading method for object detection. In this paper, we considered detectors namely Faster Region-Based Convolutional Neural Network (Faster-CNN), Region-based Fully Convolutional Networks(R-FCN) and Single Shot Multi box Detector (SSD). Each of the architecture should be able to be merged with any feature extractor depending on the application or need.

We consider some of the commercial/cash crops, cereal crops, and vegetable crops and fruit plants such as sugarcane, cotton, potato, carrot, chilly, brinjal, rice, wheat, banana and guava, these leaves images are selected for our purpose. Fig 1.1 shows images of the diseased affected leaves on various crops.

The early detection of plant leaf diseases could be a valuable source of information for executing proper diseases detection, plant growth management strategies and disease control measures to prevent the development and the spread of diseases.

**CHAPTER-2**

**LITERATURESURVEY**

In this section describes various approaches for detecting the disease in plant leaf using image processing technique

* Sachin D. Khirade & etal…[1] Identification of the plant diseases is the key to preventing the losses in the yield and quantity of the agricultural product. It requires tremendous amount of work, expertise in the plant diseases, and also require the excessive processing time. Hence, image processing isused for the detection of plant diseases. Disease detection involves the steps like image acquisition, image preprocessing, image segmentation, feature extraction and classification. This paper discussed the methods used for the detection of plant diseases using their leaves images. This paper discussed various techniques to segment the disease part of the plant. This paper also discussed some Feature extraction and classification techniques to extract the features of infected leaf and the classification of plant diseases. The accurately detection and classification of the plant disease is very important for the successful cultivation of crop and this can be done using image processing. This paper discussed various techniques to segment the disease part of the plant. This paper also discussed some Feature extraction and classification techniques to extract the features of infected leaf and the classification of plant diseases. The use of ANN methods for classification of disease in plants such as self-organizing feature map, back propagation algorithm, SVMs etc. can be efficiently used. From these methods, we can accurately identify and classify various plant diseases using image processing technique.
* Prof.Sanjay, B.Dhaygude & etal…[2] The application of texture statistics for detecting the plant leaf disease has been explained Firstly by color transformation structure RGB is converted into HSV single space because HSV is a good color descriptor. Masking and removing of green pixels with pre-computed threshold level. The nineth A segmentation is performed using 32X32 patch size and obtained useful segments. These segments are used for texture analysis by color co-occurrence matrix. Finally, if texture parameters are compared to texture parameters of normal leaf.
* A Mandeep Singh, Maninder Lal Singh & etal…[3] The most significant challenge faced during the work was capturing the quality images with maximum detail of the leaf color. It is very typical task to get the image with all the details within a processable memory. Such images are formed a through high resolution and thus are of 6-10MB of size. This was handled by using a Nikon made D5200 camera which served the task very well. Second challenge faced was to get rid of illumination conditions as from the start to the end of paddy crop season, illumination varies a lot even when the image acquiring time is fixed. However, the solution to this is variable user defined thresholding and making necessary adjustments to the shades of LCC.
* M.Malathi, K.Aruli & etal…[4] They provides survey on plant leaf disease detection using image processing techniques. Disease in crops causes significant reduction in quantity and quality of the agricultural product. Identification of symptoms of disease by naked eye is difficult for farmer. Crop protection especially in large farms is done by using computerized image processing technique that can detect diseased leaf using color information of leaves. Depending on the applications ,many image processing technique has been introduced to solve the problems by pattern recognition and some automatic classification tools. In the next section this papers present a survey of those proposed systems in meaningful way. There are many methods in automated or computer vision for disease detection and classification but still there is lack in this research topic. All the disease cannot be identified using single method.
* Malvika Ranjan, Manasi Rajiv Weginwar & etal…[5] Describes a diagnosis process that is mostly visual and requires precise judgment and also scientific methods. Image of diseased leaf is captured. As the result of segmentation Color HSV features are extracted. Artificial neural network (ANN) is then trained to distinguish the healthy and diseased samples. ANN classification performance is 80% better in accuracy.
* Y.Sanjana, Ashwath Sivasamy & etal…[6] In this it describes the uploaded pictures captured by the mobile phones are processed in the remote server and presented to an expert group for their opinion. Computer vision techniques are used for detection of affected spots from the image and their classification. A simple color difference based approach is followed for segmentation of the disease affected lesions. The system allows the expert to evaluate the analysis results and provide feedbacks to the famers through a notification to their mobile phones. The goal of this research is to develop an image recognition system that can recognize crop diseases. Image processing starts with the digitized color image of disease leaf. A method of mathematics morphology is used to segment these images. Then texture, shape and color features of color image of disease spot on leaf were extracted, and a classification method of membership function was used to discriminate between the three types of diseases.
* Bhumika S.Prajapati, VipulK.Dabhi & etal…[7] In this detection and classification of cotton leaf disease using image processing and machine learning techniques was carried out. Also the survey on back ground removal and segmentation techniques was discussed. Through this survey, we concluded that for back ground removal color single space conversion from RGB to HSV is useful. We also found that thresholding technique gives good result compared to other back ground removal techniques. We performed color segmentation by masking green pixels in the back ground removed image and then applying thresholding on the obtained masked image to get binary image. This is useful to extract accurate features of disease. We found that SVM gives good results, in terms of accuracy, for classification of diseases. There are five major steps in our proposed work, out of which three steps have been implemented: Image Acquisition, Image pre processing, and Image segmentation.
* P.Revathi, M.Hema latha & etal…[8] This proposed work is based on Image Edge detection Segmentation techniques in which, the captured images are processed for enrichment first. Then R, G, B color Feature image segmentation is carried out to get target regions (disease spots). Later, image features such as boundary, shape, color and texture are extracted for the disease spots to recognized is ease s and control the pest recommendation. In this Research work consist three parts of the cotton leaf spot, cotton leaf color segmentation, Edge detection based Image segmentation, analysis and classification of disease.
* Mr.Pramod S.landge, Sushil A.Patil & etal…[9] In this propose and experimentally evaluate a software solution for automatic detection and classification of plant diseases through Image Processing. Farmers in rural India have minimal access to agricultural experts, who can inspect crop images and render advice. Delayed expert responses to queries often reach farmers too late. This paper addresses this problem with the objective of developing image processing algorithms that can recognize problems in crops from images, based on colour, texture and shape to automatically detect diseases or other conditions that might affect crops and give the fast and accurate solutions to the farmer with the help of SMS. The design and implementation of these technologies will greatly aid inselective chemical application, reducing costs and thus leading to improved productivity, as well as improved produce.
* Heeb Al Bashish, Malik Braik & etal…[10] In this paper an image-processing-based approach is proposed and used for leaf and stem disease detection. We test our program on five diseases which effect on the plants; they are: Early scorch, Cottony mold, ashen mold, late scorch, tiny whiteness. The proposed approach is image processing-based. In the first step of the proposed approach, the images at hand are segmented using the K Means technique, in these Cond step the segmented images are passed through a pre-trained neural network. As a test bed we use as ethos leaf images taken from Al-Ghorareain Jordan.
* Satish Madhgoria, Marek Schikora & etal…[11] Proposed automatic pixel based classification method for detecting unhealthy regions in leaf images is presented. The algorithms have been tested extensively. Linear SVM has been used to classify each pixel. We have also shown the results from SVM could be improved remarkably using the neighborhood check technique. The presented algorithm could well extended further detection tasks which also mainly rely on color in formation, but extension to other features is easily possible. The task is performed in three steps. First, we perform segmentation to divide the image into fore ground and background. In these cond step, support vector machines are applied to predict the class of each pixel belonging to the foreground. And finally, we do further refinement by neighborhood-check tomatillo falsely-classified pixels from second step.

**CHAPTER 3**

**EXISTING SYSTEM**

**3.1 Description**

The overall concept that is the frame work for any vision related algorithm of image classification is almost the same. First, the digital images are acquired from the environment using a digital camera. Then image-processing techniques are applied to the acquired images to extract useful features that are necessary for further analysis. After that, several analytical discriminating techniques are used to classify the images according to the specific problem at hand. Figure1 depicts the basic procedure of the proposed vision-based detection algorithm in this research.

**3.2 Block Diagram**

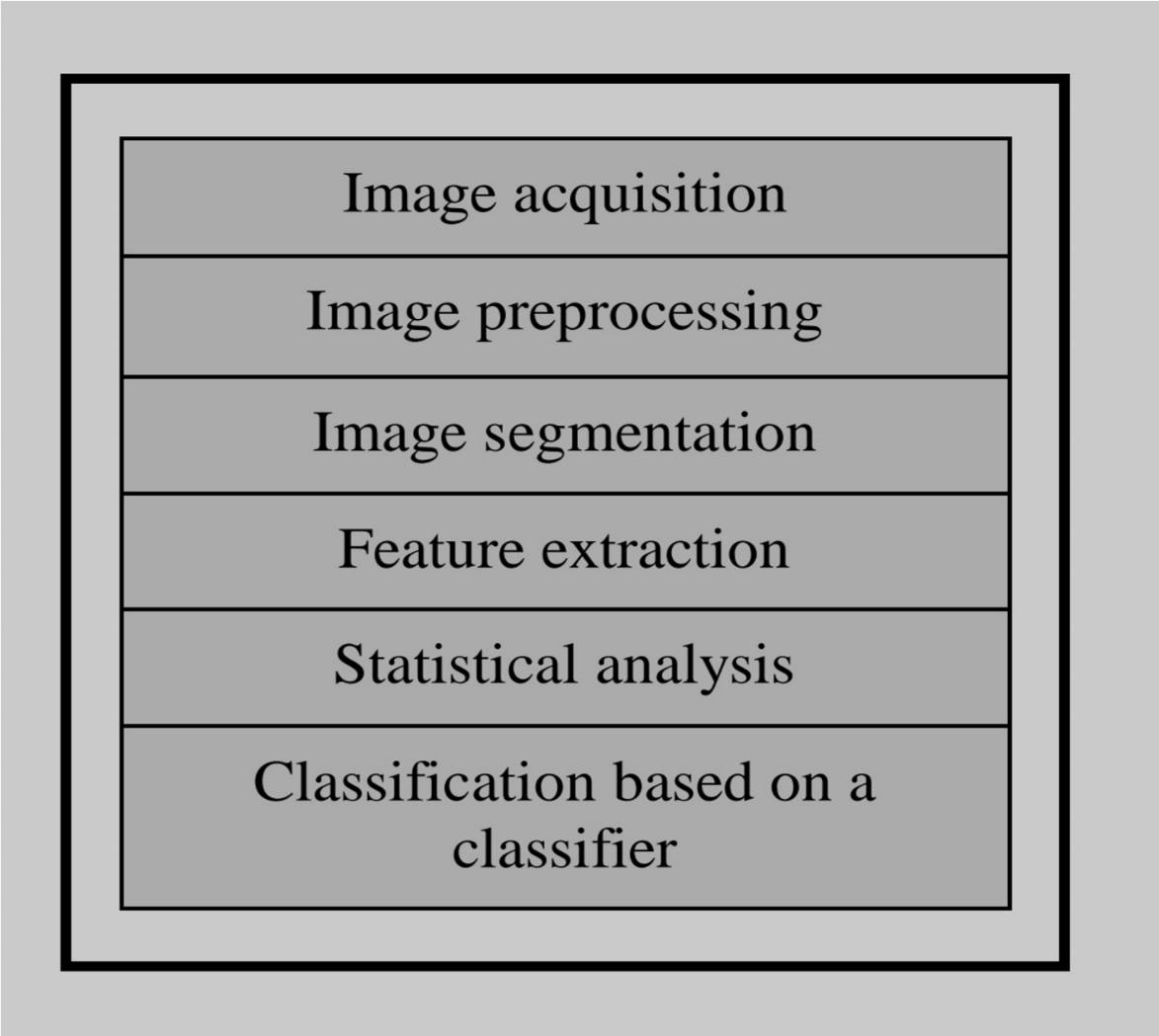
CAMERA

MONITOR DISPLAY

POWER SUPPLY

RASPBERRY PI 3B+MODULE(ANN)

Fig 3.1 Block diagram of existing system.

Fig 3.2 The basic procedure of the existing image processing-based disease detection solution.

The existing approach step-by-step of the image segmentation and recognition process illustrated in Algorithm 1.

In the initial step, the RGB images of all the leaf samples were picked up. Some real samples of those diseases are shown in Fig 3.2.

It is obvious from Fig 3.2 that leaves belonging to early scorch, cottony mold, ashen mold and late scorch have significant differences form grease spot leave sin terms of color and texture.

Also, Fig 3.2. shows two images; the left image is infected with tiny whiteness disease, and the right image is a normal image.

However, the leaves related to these six classes (early scorch, cottony mold, ashen mold, late scorch, tiny whiteness and normal) had very small differences as discernible to the human eye, which may justify them is classifications based on naked eye.

**5.3 Algorithm1: Basic steps describing the existing algorithm**.

1. RGB image acquisition.

2. Create the color transformation structure.

3. Convert the color values in RGB to the single space specified in the color transformation structure.

4. Apply K-means clustering.

5. Masking green-pixels.

6. Remove the masked cells inside the boundaries of the infected clusters.

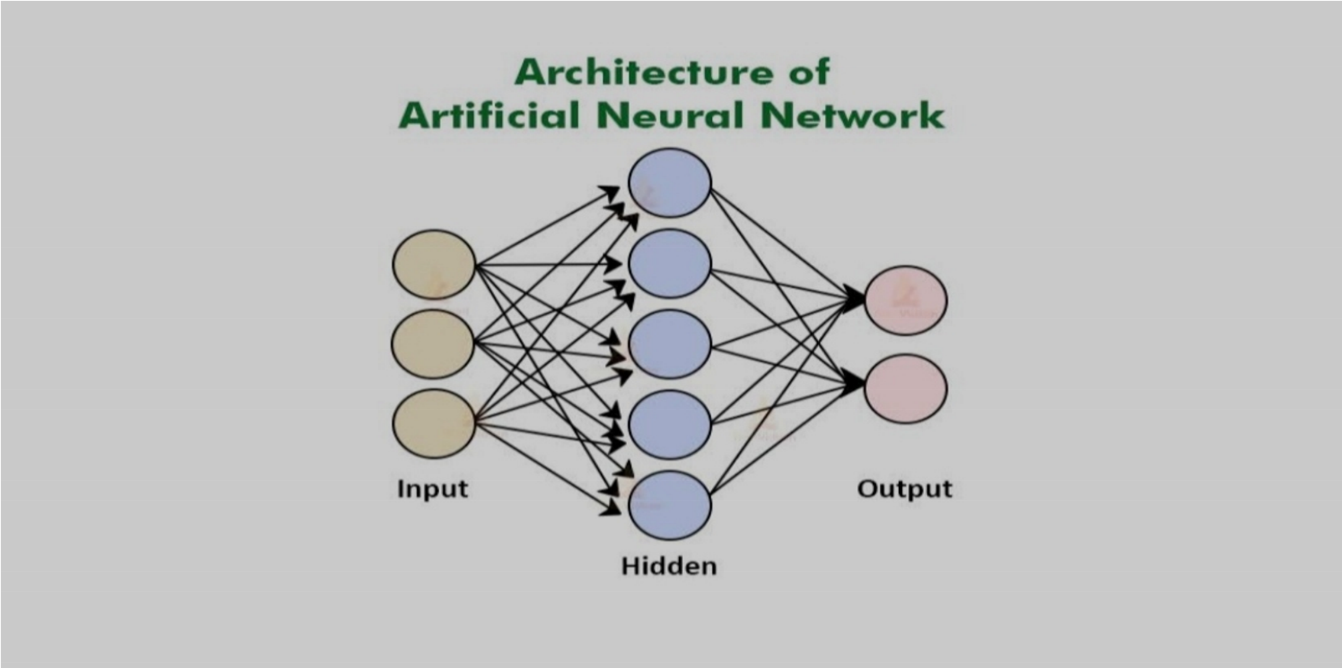
7. Convert the infected (cluster/clusters) from RGB to HIS Translation.

8. SGDM Matrix Generation for H and S.

9. Calling the GLCM function to calculate the features.

10. Texture Statistics Computation.

11. Configuring Neural Networks for Recognition.

 Fig 3.3 Architecture of ANN Algorithm

**5.4 Advantages:**

* It is used to detect plant diseases.
* It gives accurate results.
  1. **Disadvantages:**
* It increases complexity.
* It requires more powerful processor to process.

**CHAPTER-4**

**PROPOSED SYSTEM**

**4.1 Description**

Plants are susceptible to several disorders and attacks caused by diseases. There are several reasons that can be characterizable to the effects on the plants, disorders due to the environmental conditions, such as temperature, humidity, nutritional excess or losses, light and the most common diseases that include bacterial, virus, and fungal diseases. Those diseases along with the plants may shows different physical characteristics on the leaves, such as a change in shapes, colors etc. Due to similar patterns, those above changes are difficult to be distinguished, which makes their recognition a challenge, and an earlier detection and treatment can avoids ever all losses in the whole plant. In this paper, we are discussed to use recent detectors such as Faster Region-Based Convolutional Neural Network (Faster-CNN), Region-based Fully Convolutional Networks (R-FCN) and Single Shot Multi box Detector to detection and classification of plant leaf diseases that affect in various plants. The challenging part of our approach is not only deal with disease detection, and also known the infection status of the disease in leaves and tries to give solution (i.e., name of the suitable organic fertilizers) for those concern diseases.

**4.2 Diseases**

Leaf miners are the insect family at larval stage. They feed between upper and lower part of the leaf.

 Fig 4.1 Leaf miner diseases.

Due to insect on very much amount in plant, it is severely damaged. On a single leaf the number of maggots can be six. Therefore, it can severely damage the leaf of plant. It can restrict plant growth, leads to reduced yields.



Fig 4.2: Yellow spot disease

Hence, we can develop a technique using image processing to detect the disease, to classify it. This ill avoid human interference and hence lead to précised unprejudiced decision. Generally, whatever our observation about the disease is just used for the decision of the disease. A symptom of plant disease is a visible effect of disease on the plant. Symptoms can be change in color, change in the shape or functional changes of the plant as permits response to the pathogens, insects etc. Leaf wilting is a characteristic symptom of verticillium wilt. It is caused due to the fungal plant pathogens V. dahliaeand Verticillium alabastrum. General common bacterial disease symptoms are brown, necrotic lesions which gets surrounded by a bright light-yellow halo at the edge of the leaf of the plant or at inner part of the leaf on the bean plants. You are not actually seeing the disease pathogen, but rather a symptom that is being caused by the pathogen.

**4.3 Image Processing Algorithm**

1. Capture the image in RGB format.

2. Generate color transformation structure.

3. Convert color values from RGB to the space specified in that structure.

4. Apply K means clustering for image segmentation.

5. Masking of green pixels (masking green channel).

6. Eliminate the masked cells present inside the edges of the infected cluster.

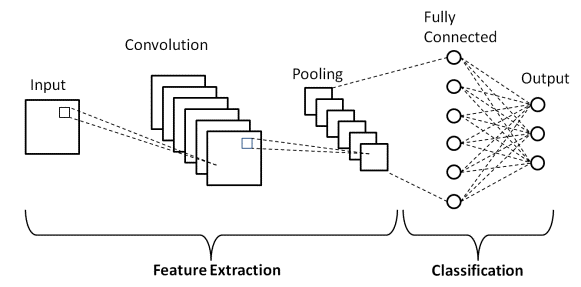
7. Convert the infected cluster from RGB to HIS.

8. Generation of SGDM matrix for H and S.

9. Calling GLCM function in order to calculate the features of it.

10. Computation of texture statistics.

11. Configure CNN (classifier) for recognition.



* Fig 4.3 Basic structure of CNN algorithm.

**4.4 Advantages**

* Increase in form productivity.
* More accuracy.

**4.5 Application**

* It can be used in farming and gardening purposes to detect diseases.

**CHAPTER 5**

**HARDWARE AND SOFTWARE DESCRIPTION**

**5.1 Hardware Components**

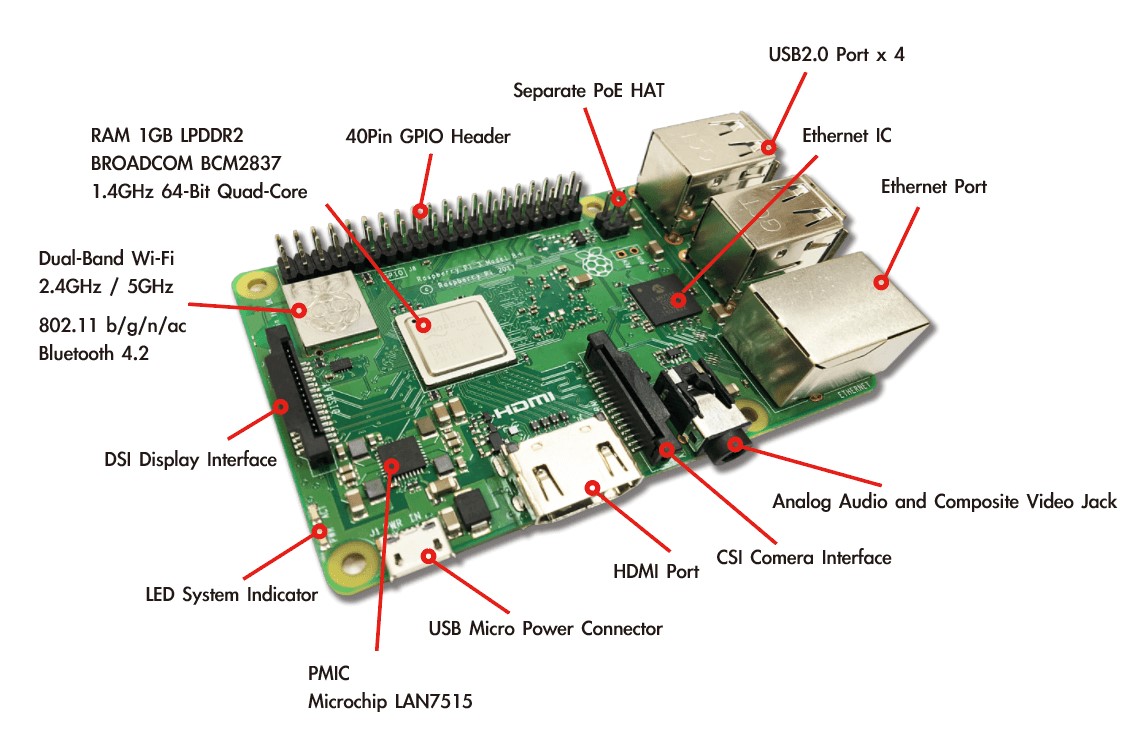
 **5.1.1 Raspberry Pi**

Fig 5.1 Raspberry Pi.

Raspberry Pi3 Mode B+ is the latest product in the popular Raspberry Pi range of computers. It offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+, while retaining backwards compatibility and similar power consumption.

For the end user, Raspberry Pi 3 Model B+ provides desktop performance comparable to entry-level x86 PC systems. This product’s key features include a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI ports, hardware video decode at up to 4Kp60, up to 4GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on).

The dual-band wireless LAN and Bluetooth have modular compliance certification, allowing the board to be designed into end products with significantly reduced compliance testing, improving both cost and time to market.

**Physical Specifications**

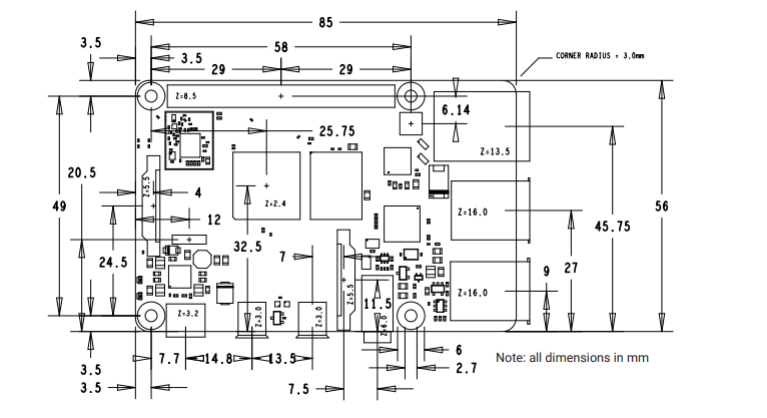


Fig 5.2 Physical specifications.

Processor: Broadcom BCM271 1, quad-core Cortex-A72 (ARM v8)

64-bit SoC @ 1.5GHz

Memory: 1GB, 2GB or 4GB LPDDR4

(depending on model)

Connectivity: 2.4 GHz and 5.0 GHz IEEE 802.1 lb/g/n/ac wireless

LAN, Bluetooth 5.0, BLE

Gigabit Ethernet

2 x USB 3.0 ports

2 x USB 2.0 ports.

GPIO: Standard 40-pin GPIO header

(fully backwards-compatible with previous boards)

Video & sound: 2 x micro HDMI ports (up to 4Kp60 supported)

2-lane MIPI DSI display port

2-lane MIPI CSI camera port

4-pole stereo audio and composite video port

Multimedia: H.265 (4Kp60 decode);

H.264 (1080p60 decode, 1080p30 encode);

OpenGL ES, 3.0 graphics

SD card support: Micro SD card slot for loading operating system

and data storage

Input power: 5V DC via USB-C connector (minimum 3A1)

5V DC via GPIO header (minimum 3A’)

Power over Ethernet (PoE)—enabled

(requires separate PoE HAT)

Environment: Operating temperature 0—50°C

Compliance: For a full list of local and regional product approvals,

please visit

<https://www.raspberrypi.org/documentation/>

hardware/raspberrypilconformity.md

Fig 5.3 Specifications of Raspberry Pi.

**5.1.2 Camera Module**

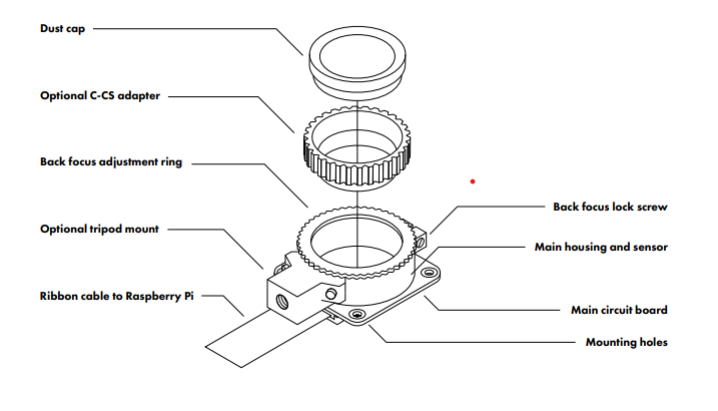


Fig 5.4 Camera module.

* Fitting lenses:

The High-Quality Camera is designed to accept CS-mount lenses. An optional adapter is supplied to extend the back focus by 5 mm, such that the camera is also compatible with Cmount lenses. Please ensure that the dust cap is fitted when there is no lens fitted, because the sensor is sensitive to dust. To fit a lens, unscrew the dust cap and screw the lens into the threads. Remove the C-CS adapter when a CS-mount lens is to be fitted; it is only required when a Count lens is fitted. The CGL 6 mm CS-mount and 16 mm C-mount lenses are examples of third-party products that are compatible with the High-Quality Camera. See step-bystep instructions for fitting the CS-mount and C-mount lenses.

Back focus adjustment: The back focus adjustment mechanism has two purposes:

1. When using a small, low-cost, fixed-focus lens, it allows adjustment of the focus.

2. When using an adjustable-focus lens, it allows adjustment of the focal range.

To adjust the back focus:

1. Ensure the lens is screwed all the way into the back focus adjustment ring.

2. Loosen the back focus lock screw with a small flat screwdriver.

3. Adjust the back focus height by turning the back focus adjustment ring clockwise or anticlockwise relative to the main housing until the camera is in focus.

4. Tighten the back focus lock screw.

Tripod mount the tripod mount is an optional component, and it can be unscrewed when it is not needed. If it is needed, take care not to damage the ribbon when screwing the tripod into the camera. Connecting the High-Quality Camera to a Raspberry Pi computer Ensure your Raspberry Pi is switched off, then carefully release the plastic catch on the Raspberry Pi’s camera connector. Insert the camera ribbon with the contacts facing away from the catch. Once you have pushed the ribbon in as far as it will go, push the catch back in. It is now safe to power up your Raspberry Pi. Operating the camera First, enable the camera in Raspbian: in the Raspbian menu, select Preferences, then Raspberry Pi Configuration. Click the Interfaces tab, find the Camera entry in the list, and select Enabled. Click OK, and reboot your Raspberry Pi when prompted. rasp still is a command line tool for capturing camera images. To check your camera is correctly installed, open a terminal window (Raspbian menu > Accessories > Terminal) and take a test photograph by entering the command: rasp still -o test.jpg When you hit ENTER a live preview image will appear, and after a default period of five seconds, the camera will capture a single still image. This will be saved in your home folder and named test.jpg. To use the camera just as a viewfinder, without saving a photo, use this command:rasp still -t 0 For more detailed information about installing and operating camera software, refer to https://www.raspberrypi.org/documentation/configuration/camera.md. Other features of the camera Rotating the camera It is possible to position the camera circuit board upside down with respect to the tripod mount. In this situation the ribbon will extend out from the top of the unit, rather than from the bottom. To rotate the camera: Work in a clean and dust-free environment, as the sensor will be exposed to the air. Work with the sensor facing downwards throughout the process. Unscrew the two 1.5 mm hex lock keys on the underside of the main circuit board. Be careful not to let the washers roll away. Lift the main housing and rotate it 180 degrees. The gasket will remain on the circuit board, so realign the housing with the gasket. The nylon washer prevents damage to the circuit board; apply this was her first. Next, fit the steel washer, which prevents damage to the nylon washer. Screw down the two hex lock keys. As long as the washers have been fitted in the correct order, they do not need to be screwed very tightly. Infrared (IR) filter The High-Quality Camera contains an IR filter, which is used to reduce the camera’s sensitivity to infrared light. It is possible to remove this filter, but doing so will void the warranty on the product, and is likely to prove irreversible. Regulatory compliance EU The Raspberry Pi High Quality Camera is in conformity with the following applicable community harmonized legislation: Electromagnetic Compatibility Directive (EMC) 2014/30/EU, Restriction of Hazardous Substances (RoHS) Directive 2011/65/EU The following harmonized standards have been used to demonstrate conformity to these standards: EN 55032:2015, EN 55024:2010, IEC60695-2-11, EN60950-1 WEEE Directive Statement for the European Union This marking indicates that this product should not be disposed with other household wastes throughout the EU. To prevent Application of Raspberry-Pi for Plant Disease Detection 19 possible harm to the environment or human health from uncontrolled waste disposal, recycle it responsibly to promote the sustainable reuse of material resources. To return your used device, please use the return and collection systems or contact the retailer where the product was purchased. They can take this product for environmentally safe recycling. FCC The Raspberry Pi High Quality Camera is in conformity with the requirements of the following specifications: FCC 47 CFR Part 15, Subpart B, Class B Digital Device. This device complies with part 15 of the FCC Rules.

Operation is subject to the following two conditions:

(1) This device may not cause harmful interference.

(2) This device must accept any interference received, including interference that may cause undesired operation.

* Safety information

IMPORTANT: PLEASE RETAIN THIS INFORMATION FOR FUTURE REFERENCE

WARNINGS:

•This product should only be connected to and powered by a Raspberry Pi computer. Any external power supply used with the Raspberry Pi should comply with relevant regulations and standards applicable in the country of intended use. This product should be operated in a well-ventilated environment and should not be covered. This product should be placed on a stable, flat, non-conductive surface while it is in use, and it should not be contacted by conductive items.

* INSTRUCTIONS FOR SAFE USE:

To avoid malfunction of or damage to your Raspberry Pi High Quality Camera, please observe the following:

1.Do not expose it to water or moisture, or place it on a conductive surface whilst in operation.

2. Do not expose it to heat from any source; the Raspberry Pi High Quality Camera is designed for reliable operation at normal ambient room temperatures.

3. Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and exposed connectors.

4.Use a tripod with the device to minimize damage to the electronic components.

5. Avoid handling the Raspberry Pi High Quality Camera while it is powered.

6.Handle only by the edges or by the lens mount assembly to minimize the risk of causing damage by electrostatic discharge.

7.Take care not to damage any of the exposed electronics components. These are easily damaged if the unit is dropped, and this is especially the case if a large lens is fitted.

* *Connecting the High-Quality Camera to a Raspberry Pi computer.*

Ensure your Raspberry Pi is switched off, then carefully release the plastic catch on the Raspberry Pi’s camera connector. Insert the camera ribbon with the contacts facing away from the catch. Once you have pushed the ribbon in as far as it will go, push the catch back in. It is now safe to power up your Raspberry Pi. Operating the camera First, enable the camera in Raspbian: in the Raspbian menu, select Preferences, then Raspberry Pi Configuration. Click the Interfaces tab, find the Camera entry in the list, and select Enabled. Click OK, and reboot your Raspberry Pi when prompted. rasp still is a command line tool for capturing camera images. To check your camera is correctly installed, open a terminal window (Raspbian menu > Accessories > Terminal) and take a test photograph by entering the command: rasp still -o test.jpg When you hit ENTER a live preview image will appear, and after a default period of five seconds, the camera will capture a single still image. This will be saved in your home folder and named test.jpg. To use the camera just as a viewfinder, without saving a photo, use this command: Application of Raspberry-Pi for Plant Disease Detection

* *Rotating the camera*

It is possible to position the camera circuit board upside down with respect to the tripod mount. In this situation the ribbon will extend out from the top of the unit, rather than from the bottom. To rotate the camera: Work in a clean and dust-free environment, as the sensor will be exposed to the air. Work with the sensor facing downwards throughout the process. Unscrew the two 1.5 mm hex lock keys on the underside of the main circuit board. Be careful not to let the washers roll away. Lift the main housing and rotate it 180 degrees. The gasket will remain on the circuit board, so realign the housing with the gasket. The nylon washer prevents damage to the circuit board; apply this was her first. Next, fit the steel washer, which prevents damage to the nylon washer. Screw down the two hex lock keys. As long as the washers have been fitted in the correct order, they do not need to be screwed very tightly. Infrared (IR) filter The High-Quality Camera contains an IR filter, which is used to reduce the camera’s sensitivity to infrared light. It is possible to remove this filter, but doing so will void the warranty on the product, and is likely to prove irreversible. Regulatory compliance EU The Raspberry Pi High Quality Camera is in conformity with the following applicable community harmonized legislation: Electromagnetic Compatibility Directive (EMC) 2014/30/EU, Restriction of Hazardous Substances (RoHS) Directive 2011/65/E.

**5.1.3 Monitor display**

Fig 5.5 Monitor display.

A computer monitor is an output device that displays information in pictorial or text form. A monitor usually comprises a visual display, some circuitry, a casing, and a power supply. The display device in modern monitors is typically a thin film transistor liquid crystal display (TFTLCD) with LED back lighting having replaced cold-cathode fluorescent lamp (CCFL)back lighting. Previous monitors used a cathode ray tube (CRT) and some Plasma (also called Gas-Plasma) displays. Monitors are connected to the computer via VGA, Digital Visual Interface (DVI), HDMI, Display Port, USB-C, low-voltage differential signaling (LVDS)or other proprietary connectors and signals

Originally, computer monitors were used for data processing while television sets were used for entertainment. From the 1980s onwards, computers (and their monitors) have been used for both data processing and entertainment, while televisions have implemented some computer functionality. The common aspect ratio of televisions, and computer monitors, has changed from 4:3to16:10, to16:9.

Modern computer monitors are easily interchangeable with conventional television sets and vice versa. However, as computer monitors do not necessarily include integrated speakers nor TV tuners (such as digital television adapters), it may not be possible to use a computer monitor as a TV set without external components.

Laptop displays range from the very small--the10-inch “netbook” computers designed primarily for surfing the Internet and basic word processing—to the substantial--17-inch -or greater screens that are the size of a desktop monitor. There are advantages to both sizes—and the sizes in between—built all depends on what you need the laptop for. Bigger screens are handy for graphics-editing software and video work, whereas smaller displays make your laptop lighter and easier to place in to smaller bags—a plus for traveling.

**5.1.4 Power Supply**



Fig 5.6 Adaptor.

Power Supply (LM7805):

The LM7805 is a voltage regulator that outputs +5volts. Like most other regulators in the market, it is a three-pin IC; input pin for accepting incoming DC voltage, ground pin for establishing ground for the regulator, and output pin that supplies the positive 5 volts.

Product Features:

1. 3-Terminal Regulators Output Current upto1.5A.
2. Internal Thermal-Overload Protection.
3. High Power-Dissipation Capability.
4. Internal Short-Circuit Current Limiting

Output Transistor SAFE-Area Compensation Absolute Maximum

InputVoltage-35v

Recommended Operating Conditions:

Input Voltage: Minimum 7V, Maximum 25V Output Current:1.5A

Operating Virtual Junction Temperature: Minimum 0, Maximum 125°C

Fig:5.6. Power supply Possible High Temperatures If differences between the input and output voltages are not well managed, LM7805 can over heat, which may result in

malfunctioning. Solutions Include: Limiting input voltage to 2-3 volts above the output regulated voltage Placing a heat sink in the circuit to dissipate heat solutions

LM7805ProductApplications:

1. LM7805 is applied in a wide range of circuits Fixed-Output Regulator.

2. Regulator Positive Regulator in Negative Configuration.

3. Adjustable Output Regulator.

4. Current Regulator.

5. Regulated Dual Supply Output.

6. Polarity-Reversal-Protection Circuit.

7. Reverse bias projection Circuit.

8. LM7805 can also be used in building circuits for inductance meter, phone chargers, portable CD player

**5.2 Software Components**

**5.2.1 Open CV with Python**

In this tutorial, we are going to learn how to use OpenCV library in Python. Open CV is an open-source library which is supported by multiple platforms including Windows, Linux, and MacOS, and is available for use in multiple other languages as well; however, it is most commonly used in Python for Machine Learning applications, specifically in the Computer Vision domain.

Apart from its cross-platform support and availability in multiple other computer languages, which allows applications developed in it to be used on different systems, OpenCV is also, in comparison to other such libraries, fairly efficient in terms of computations, as it uses vector operations for most of its functions.

In this tutorial, we'll cover Open CV installation on Mac, Windows, and Linux, image operations, image arithmetic’s, image smoothing, and geometric transformations using Open CV. Show it out further as do, let's start.

**Installation**

Note: Since we are going to use Open CV in the Python language, it is an implicit requirement that you already have Python(version3) installed on your workstation. Depending upon your OS, execute one of the following commands to install OpenCV library on your system:

Windows

$pipinstallopencv-python

MacOS

$brewinstallopencv3--with-contrib--with-python3 Linux

$sudoapt-getinstalllibopencv-devpython-opencv To check if your installation was successful or not, run the following command line ither a Python shell, or your command prompt/terminal:

importcv2

If you do not get an error on importing cv2 then it was installed correctly.

*Basic Image Operations*

Now that we have installed Open CV on our work stations, let's get our hands dirty with some of the functionalities that Open CV offers.

* *Display an Image*

Displaying an image using Open CV is a two-step process; first, we have to load it, and then we can display it. Both operations are done in sequence using different functions.

To display an image, we need to know two things:

1.Image Path (both absolute and relative paths work)

2.Read Mode (read, write, etc.)

The function we'll use for reading/loading an image is cv2.imread(), which has two variations. First one is IMREAD\_ GRAYSCALE, which as the name suggests, converts the image to gray scale be for reading it. These could one is IMREAD \_UNCHANGED, which loads the image without cutting out the alpha channel. The default is IMREAD\_COLOR, which simply reads the colored image using the RGB channel only.

Let's code an example:

importcv2

my\_bike=cv2.imread('bike.png')

This will load the image of a bike from the filesystem and store it in

the my \_bike variable for further operations

Note: If you get an error from the above code, there are only three possible reasons for it. The first one is that the path you specified is in correct, second is that the image file you specified doesn't exist, and the last one is that the image type (jpg/jpeg/png) in the image path is incorrect.

Let's now display the image we just read. It can be done by the cv2.imshow() function. If you have used MATLAB, you may be familiar with these image operations.

cv2.imshow('my\_bike',my\_bike)

The first parameter to the imshow() function is the string name that you want to display on the image window. These Cond parameter is the image handler we created using the cv2.imread() function.

* *Saving an Image*

Saving an image is a very commonly used feature, as we may need to update our image and save the changes to the file system for later use. OpenCV has cv2.imwrite() function to save images.

Here is an example:

cv2.imwrite('bike.png',my\_bike)

Here we specify the name and current location of the image. The resulting image is automatically saved in the current working directory.

* *Arithmetic Operations on Images*

Arithmetic operations on images refer to adding, subtracting, multiplying, or dividing multiple images to generate a new image which is an arithmetic combination of the input images. Image arithmetic’s has a lot of applications, like adding a watermark to an image, creating able ended combination of two images, applying different types of image filters, etc.

While there are many operations you can perform, we will only be showing two examples here, as this will then allow you to apply the concept to other arithmetic operations available in Open CV. The first example will be the addition of two images, and these Cond example will be blending two images.

Let's code these two examples:

* *Adding Images*

importcv2

#Readinthetwoimages

image\_1=cv2.imread('bike.jpg') image\_2=cv2.imread('car.jpg')

#Sumthetwoimagearraysforallchannels result=cv2.add(image\_1,image\_2) cv2.imshow('result',result) cv2.waitKey(0) cv2.destroyAllWindows()

The wait Key command will wait for you to press a key before it moves onto the next command. This is useful so that the program will continue to display your image until a key is pressed, otherwise it will be displayed for a split second and then quickly disappear once the program has stopped executing.

* *Blending Images*

Blending images is similar to image addition, except each image's contribution to the new resulting image can be controlled. Basically, if we want one image to be more focused, and the other one to be more faint when they get merged, we will go with blending, instead of simple addition.

Lets code it to clarify further:

importcv2

#Readinthetwoimages

image\_1=cv2.imread('bike.jpg') image\_2=cv2.imread('car.jpg')

result=cv2.addWeighted(image\_1,0.9,image\_2,0.1)

cv2.imshow('result',result) cv2.waitKey(0)#Waitfortheusertopressakeybeforecontinuing cv2.destroyAllWindows()

The sum of the weights given to the add Weighted function should be equal to1.0. You can also give a scalar value at the end, which would be added to all the pixel values of the result anti image.

Note: The images can be of any type; however, the type should be the same for all images. For instance, if you are using PNG format, all images being used for computation should be in PNG format as well.

* *Image Smoothing*

Image smoothing is a very helpful feature, which is mostly performed before the images are passed onto a machine learning model. It is mostly done to remove noise/high frequency elements from images by passing the image through a low-pass filter. There are many filters, including box filter (averaging filter), median filter, mode filter, Gaussian filter, and many more; however, to understand image smoothing and how to do it using Open CV, we will only cover the box filter.

Let's say you have an image of 10x10, and you want to pass it through a 3x3 box /averaging filter, how would you do it?

You'll start with the top left of the image, place your 3x3 filter there, and replace the central element with the average of all 9 elements. This was the first step, now you will move your filter one step to the right, and repeat the same process until you have covered the whole image. An example of 10x10 image, and 3x3 averaging filter are shown below for your reference:

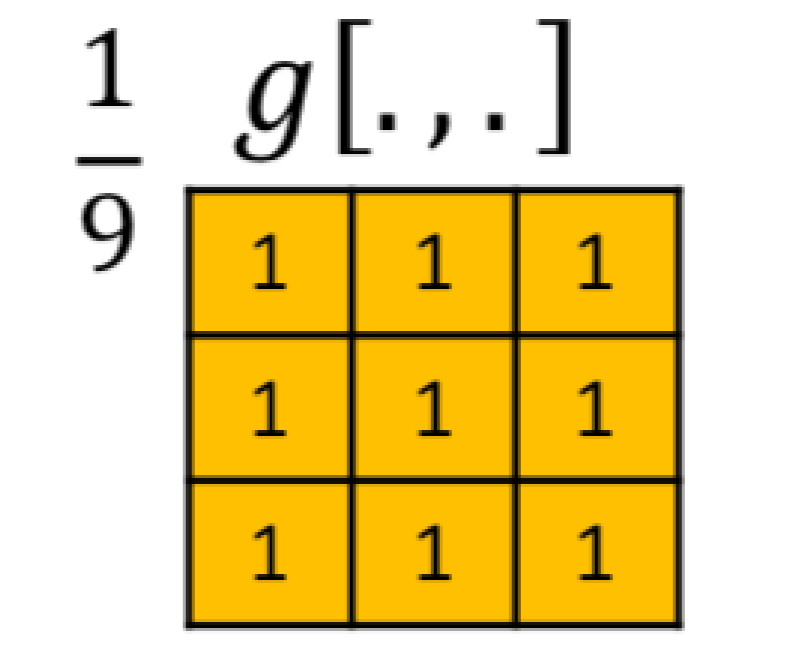
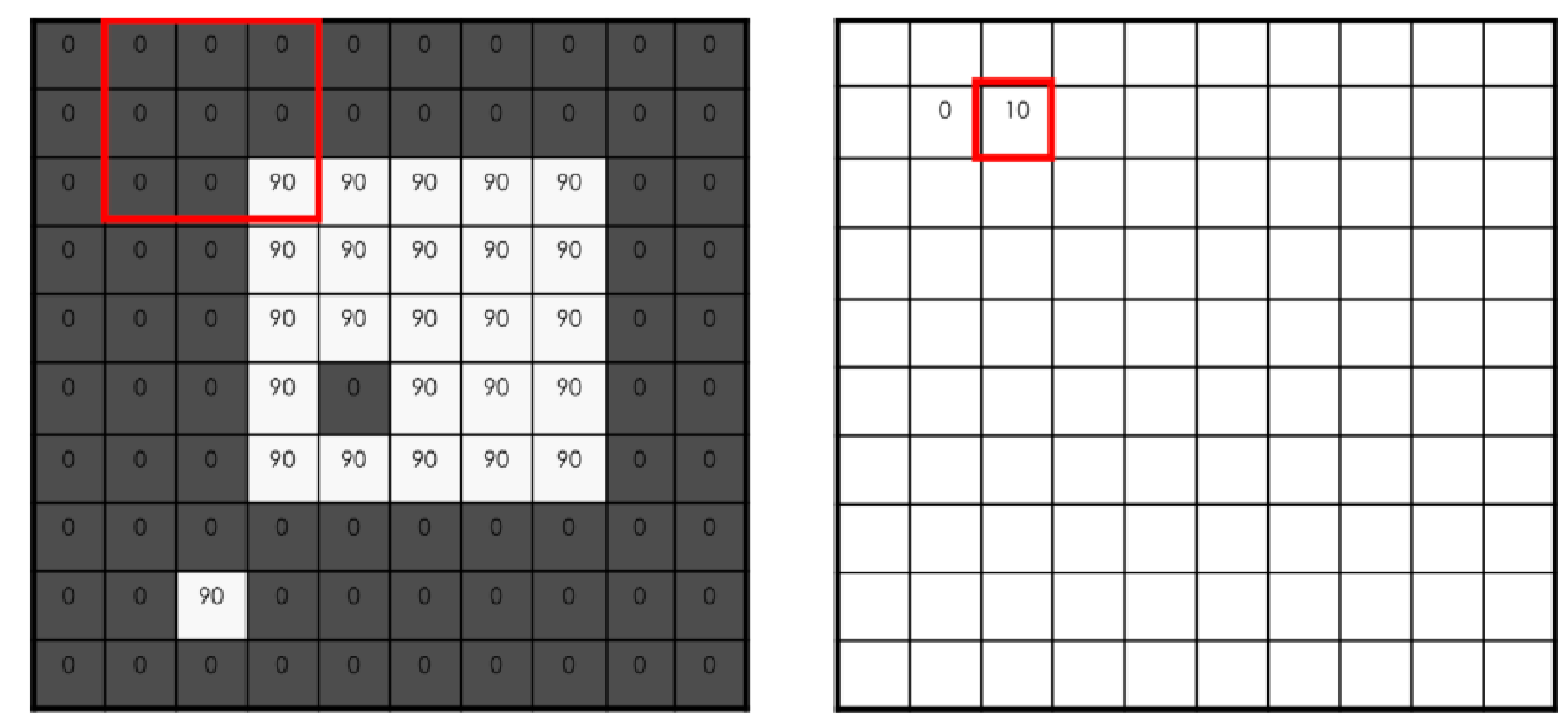
* *Filter/Mask:*

Fig5.7. Filter being applied on 10x10 Image.



Now that we have discussed how it works, let’s try and see how we can apply different filters on our image using Open CV; please read the comments thoroughly to know which line of code is used for which filter: importcv2

#Loadtheoriginalimage original\_image=cv2.imread('my\_bike.png')

#Filterbypassingimagethrough3x3averagingfilter average\_image=cv2.blur(original\_image,(3,3))

#Apply3x3gaussianfilterontheoriginalimage gaussian\_image=cv2.GaussianBlur((original\_image,(3,3),0))

#Apply3x3medianfilterontheoriginalimage median\_image=cv2.medianBlur(original\_image,3)

Note: You can view the resulting images by using the following additional code:

importmatplotlib.pyplotasplt

plt.imshow(average\_image) plt.show()

* *Image Transformations*

Image transformation is the last, but one of the most important topics that we are going to cover with Open CV. It has a lot of applications, but one of the most common ones now a days is in Machine Learning for Data Augmentation, i.e. when you have a shortage of dataset, you augment/ transform the currently available images to make them different from your existing dataset. This effectively increases your data set size and might help in improving your model accuracy.

The list of possible transformations is along one, including scaling, affine, rotation, translation, etc. We will only covert woof the musing Open CV to get a general idea; however, Open CV provides supporting functions for a wide range of them. Let's start with scaling.

* *Scaling*

To put it in simple words, scaling is basically just resizing your image, i.e. either making it bigger or smaller. resize is the function used for scaling the images in Open CV. Resizing has three types: INTER\_CUBIC, INTER\_LINEAR, and INTER\_AREA. Let's code an example using these functions for scaling; please read through the code, comments, anddescriptionscarefullyastheywillexplainwhatexactlyisgoingoninthecode:

importcv2 importnumpyasnp

importmatplotlib.pyplotasplt

image=cv2.imread('my\_bike.jpg')

#Scaleup/expandbothwidthandheightbyfactorof2 result\_1=cv2.resize(image,None,fx=2,fy=2,interpolation=cv2.INTER\_CUBIC)

#Scaledown/shrinkbothwidthandheightbyfactorof2 result\_2=cv2.resize(image,None,fx=2,fy=2,interpolation=cv2.INTER\_AREA)

#Displaytheresultingimages plt.imshow(result\_1) plt.imshow(result\_2) plt.show()

Here in the resize function, the fx parameter in represents the scale factor for width, fy represents the scale factor height, and interpolation specifies the function to be used for scaling (shrinking or expansion).

#### Rotation

Rotation allows us to move an image about the axis for a certain specified angle.

Before we learn how to rotate our images using code, we should know that there is a [rotation matrix](https://en.wikipedia.org/wiki/Rotation_matrix) that is used for performing this transformation; we will not go in details of that, as OpenCV makes it very simple for us to calculate that matrix using a single function call. You will see that in the code below:

import cv2

import matplotlib.pyplot as plt

*# Load the image of a bike*

image = cv2.imread('my\_bike.jpg',0)

*# Rows and columns*

r, c = image.shape

matrix = cv2.getRotationMatrix2D((cols/2,rows/2), 180, 1)

result = cv2.warpAffine(image,matrix,(c,r))

*# Display resulting rotation*

plt.imshow(result)

plt.show()

In the get Rotation Matrix 2D function, 180 specifies the degree by which the image should be rotated, 1 is the scaling factor, the function call would return the rotation matrix in the matrix variable.

The warp Affine function call uses the matrix we calculated from the previous method to rotate the image according to our specifications.

**5.2.2 CNN Algorithm**

A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data.

A convolution is essentially sliding a filter over the input. One helpful way to think about convolutions is this quote from Dr Prasad Samarakoon: “A convolution can be thought as “looking at a function’s surroundings to make better/accurate predictions of its outcome.”

Rather than looking at an entire image at once to find certain features it can be more effective to look at smaller portions of the image.

* *Common uses for CNNs*

The most common use for CNNs is image classification, for example identifying satellite images that contain roads or classifying hand written letters and digits. There are other quite mainstream tasks such as image segmentation and signal processing, for which CNNs perform well at.

CNNs have been used for understanding in Natural Language Processing (NLP) and speech recognition, although often for NLP Recurrent Neural Nets (RNNs) are used.

A CNN can also be implemented as a U-Net architecture, which are essentially two almost mirrored CNNs resulting in a CNN whose architecture can be presented in a U shape. U-nets are used where the output needs to be of similar size to the input such as segmentation and image improvement.

* *Interesting uses for CNNs other than image processing*

More and more diverse and interesting uses are being found for CNN architectures. An example of a non-image-based application is [“The Unreasonable Effectiveness of Convolutional Neural Networks in Population Genetic Inference”](https://www.ncbi.nlm.nih.gov/pubmed/30517664) by Lex Flagel et al. This is used to perform selective sweeps, finding gene flow, inferring population size changes, inferring rate of recombination.

There are researchers such as Professor Gerald Quon at the [Quon-titative biology lab](https://qlab.faculty.ucdavis.edu/publications/" \t "_blank), using CNNs for generative models in single cell genomics for disease identification.

CNNs are also being used in astrophysics to interpret radio telescope data to predict the likely visual image to represent the data.

[Deepmind’s WaveNet](https://deepmind.com/blog/wavenet-generative-model-raw-audio/) is a CNN model for generating synthesized voice, used as the basis for Google’s Assistant’s voice synthesizer.

# *Convolutional kernels*

Each convolutional layer contains a series of filters known as convolutional kernels. The filter is a matrix of integers that are used on a subset of the input pixel values, the same size as the kernel. Each pixel is multiplied by the corresponding value in the kernel, then the result is summed up for a single value for simplicity representing a grid cell, like a pixel, in the output channel/feature map.

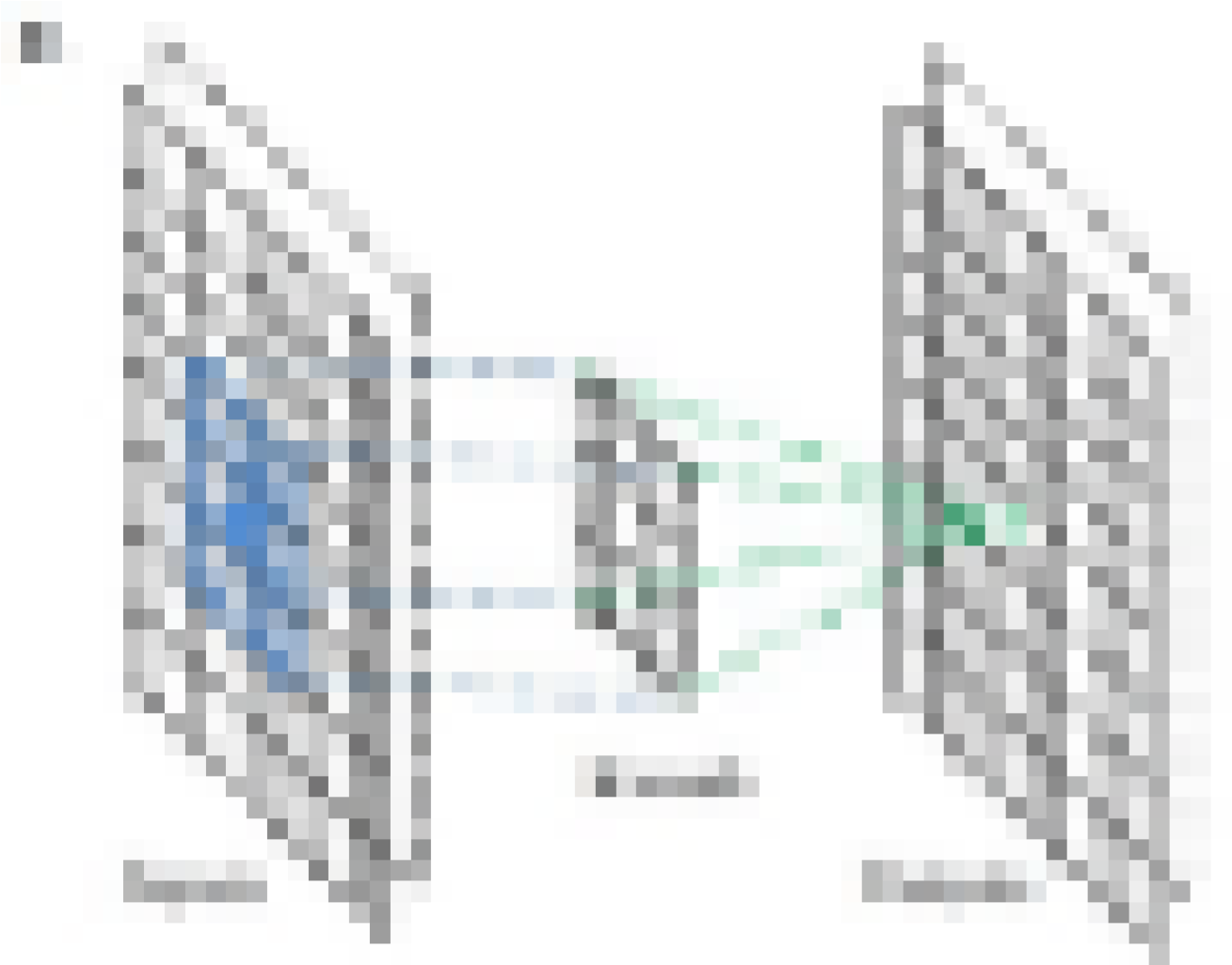
These are linear transformations; each convolution is a type of affine function.

In computer vision the input is often a 3 channel RGB image. For simplicity, if we take a greyscale image that has one channel (a two-dimensional matrix) and a 3x3 convolutional kernel (a two-dimensional matrix). The kernel strides over the input matrix of numbers moving horizontally column by column, sliding/scanning over the first rows in the matrix containing the images pixel values. Then the kernel strides down vertically to subsequent rows. Note, the filter may stride over one or several pixels at a time, this is detailed further below.

In other non-vision applications, one-dimensional convolution may slide vertically over an input matrix.

# *Creating a feature map from a convolutional kernel*

Below is a diagram showing the operation of the convolutional kernel.



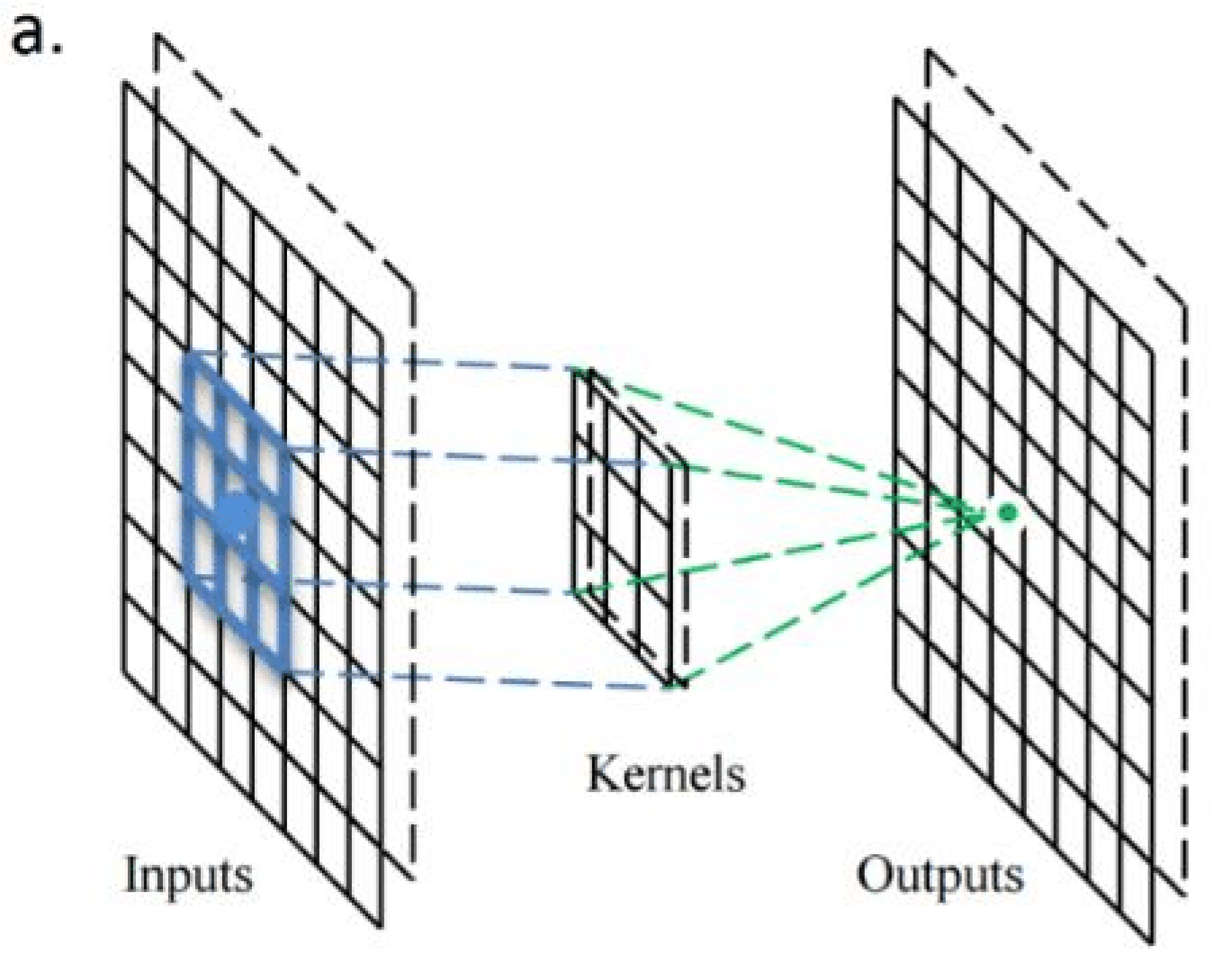
 Fig 5.8 Blurred image.

Fig 5.9 A stride one 3x3 convolutional kernel acting on a 8x8 input image, outputting an 8x8 filter/channel.

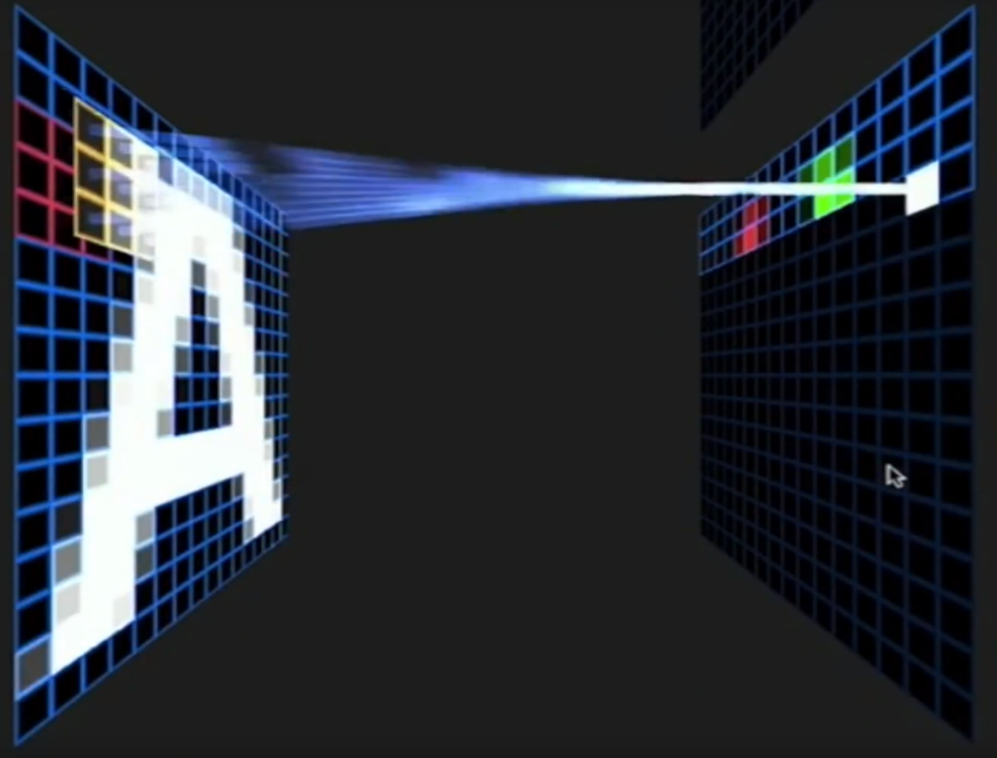
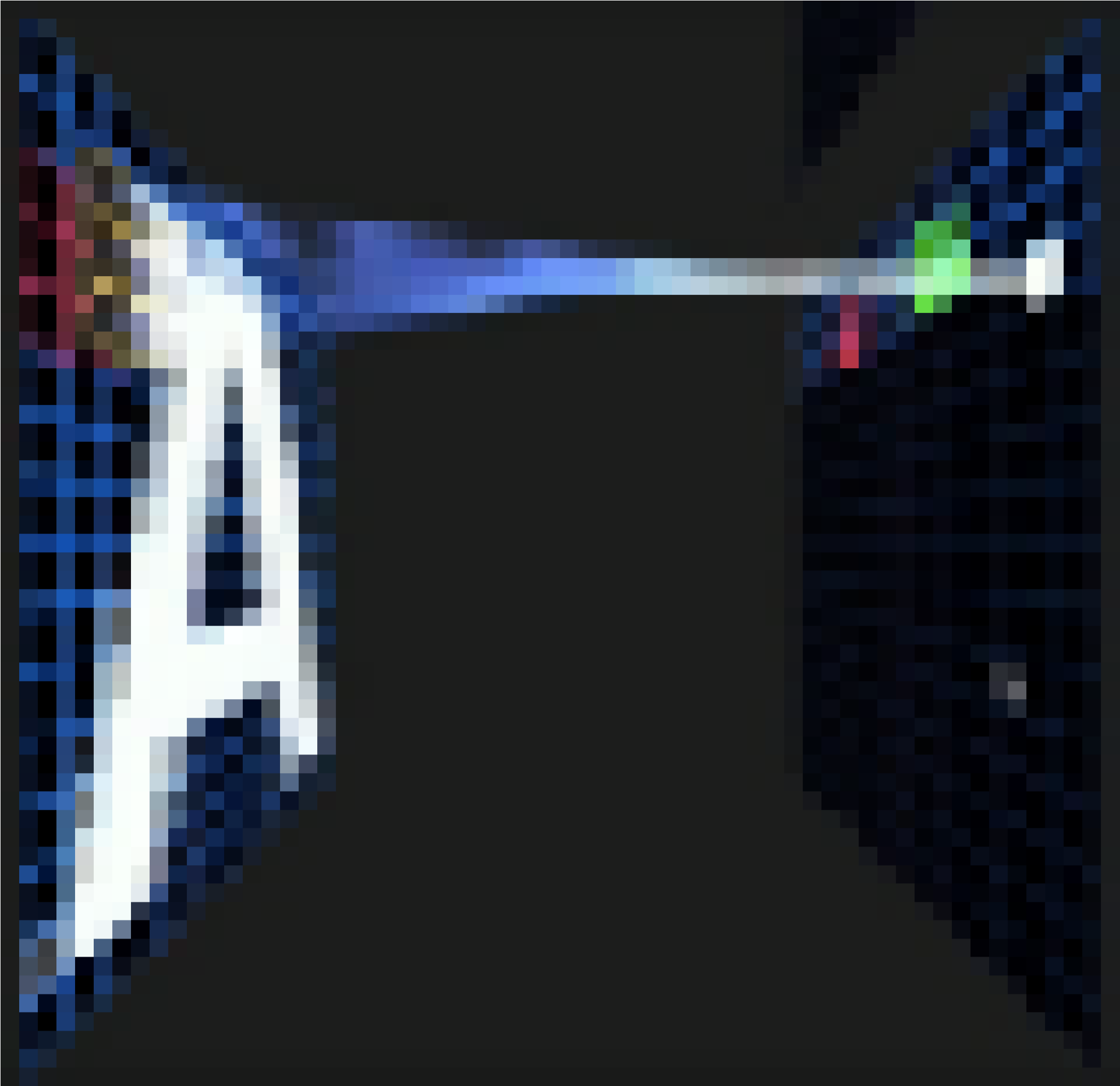
Below is a visualization from an excellent presentation, showing the kernel scanning over the values in the input matrix.

Fig 5.10 Kernel scanning over the values in the input matrix.

* *Padding*

To handle the edge pixels there are several approaches:

* Losing the edge pixels.
* Padding with zero value pixels.
* Reflection padding.

Reflection padding is by far the best approach, where the number of pixels needed for the convolutional kernel to process the edge pixels are added onto the outside copying the pixels from the edge of the image. For a 3x3 kernel, one pixel needs to be added around the outside, for a 7x7 kernel then three pixels would be reflected around the outside. The pixels added around each side is the dimension, halved and rounded down.

Traditionally in many research papers, the edge pixels are just ignored, which loses a small proportion of the data and this gets increasing worse if there are many deep convolutional layers. For this reason, I could not find existing diagrams to easily convey some of the points here without being misleading and confusing stride 1 convolutions with stride 2 convolutions.

With padding, the output from a input of width w and height h would be width w and height h (the same as the input with a single input channel), assuming the kernel takes a stride of one pixel at a time.

# *Creating multiple channels/feature maps with multiple kernels*

When multiple convolutional kernels are applied within a convolutional layer, many channels/feature maps are created, one from each convolutional kernel. Below is a visualization below showing the channels/feature maps being created.



Fig 5.11 Visualization of channels/feature maps created from a layer of convolutional kernels.

# *RGB 3 channel input*

Most image processing needs to operate on RGB images with three channels. A RGB image is a three-dimensional array of numbers otherwise known as a rank three tensor.

When processing a three channel RGB image, a convolutional kernel that is a three-dimensional array/rank 3 tensor of numbers would normally be used. It is very common for the convolutional kernel to be of size 3x3x3 — the convolutional kernel being like a cube.

Usually there is at least three convolutional kernels in order that each can act as a different filter to gain insight from each color channel.

The convolution kernels as a group make a four-dimensional array, otherwise known as a rank four tensor. It is difficult, if not impossible, to visualize dimensions when they are higher than three. In this case imagine it as a list of three-dimensional cubes.

The filter moves across the input data in the same way, sliding or taking strides across the rows then moving down the columns and striding across the rows until it reaches the bottom right corner:



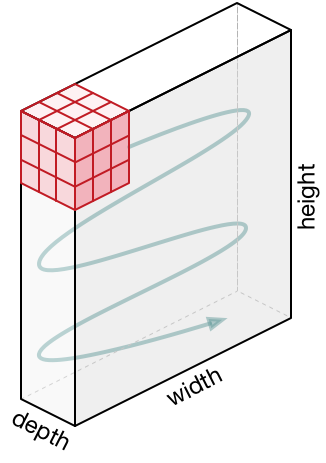


Fig 5.12 3x3x3convolutional kernel acting on a 3-channel input.

With padding and a stride of one, the output from an input of width x, height y and depth 3 would be width x, height y and depth 1, as the cube produces a single summed output value from each stride. For example, with an input of 3x64x64 (say a 64x64 RGB three channel image) then one kernel taking strides of one with padding the edge pixels would output a channel/feature map of 64x64 (one channel).

It is worth noting the input is often normalized, this is detailed further below.

# *Strides*

It is common to use a stride two convolution rather than a stride one convolution, where the convolutional kernel strides over 2 pixels at a time, for example our 3x3 kernel would start at position (1,1), then stride to (1,3), then to 1, 5) and so on, halving the size of the output channel/feature map, compared to the convolutional kernel taking strides of one.

With padding, the output from an input of width w, height h and depth 3 would be the ceiling of width w/2, height h/2 and depth 1, as the kernel outputs a single summed output from each stride.

For example, with an input of 3x64x64 (say a 64x64 RGB three channel image), one kernel taking strides of two with padding the edge pixels, would produce a channel/feature map of 32x32.

# *Many kernels*

In CNN models there are often there are many more than three convolutional kernels, 16 kernels or even 64 kernels in a convolutional layer are common.

These different convolution kernels each act as a different filter creating a channel/feature map representing something different. For example, kernels could be filtering top edges, bottom edges, diagonal lines and so on. In much deeper networks these kernels could be filtering to animal features such as eyes or bird wings.

Having a higher number of convolutional kernels creates a higher number of channels/feature maps and a growing amount of data and this uses more memory. The stride 2 convolution, as per the above example, helps to reduce the memory usage as the output channel of the stride 2 convolution has half the width and height of the input. This assumes reflection padding is being used otherwise it could be slightly smaller.

## An example of several convolutional layers of stride 2

With a 64 pixel square input with three channels and 16 3x3x3 kernels our convolutional layer would have:

Input: 64x64x3  
Convolutional kernels: 16x3x3x3 (a four-dimensional tensor)  
Output/activations of the convolutional kernels: 16x32x32 (16 channels/feature maps of 32x32)

The network could then apply batch normalization to decrease learning time and reduce overfitting, more details below. In addition, a non-linear activation function, such as RELU is usually applied to allow the network to approximate better, more details below.

Often there are several layers of stride 2 convolutions, creating an increasing number of channels/feature maps. Taking the example above a layer deeper:

Input: 16x32x32  
Convolutional kernels: 64x3x3x3  
Output/activations of the convolutional kernels: 64x16x16 (64 channels/feature maps of 16x16)

Then after applying ReLU and batch normalization (see below), another stride 2 convolution is applied:

Input: 64x16x16  
Convolutional kernels: 128x3x3x3  
Output/activations of the convolutional kernels: 128x8x8 (128 channels/feature maps of 8x8).

# *Classification*

If, for example, an image belongs to one of 42 categories and the network’s goal is to predict which category the image belongs to.

Following on from the above example with an output of 128x8x8, first the average pool of the rank 3 tensor is taken. The average pool is the mean average of each channel, in this example each 8x8 matrix is averaged into a single number, with 128 channels/feature maps. This creates 128 numbers, a vector of size 1x128.

The next layer is a matrix or rank 2 tensors of 128x42 weights. The input 1x128 matrix is (dot product) multiplied by the 128x42 matrix producing a 1x42 vector. How activated each of the 42 grid cells/vector elements are, is how much the prediction matches that classification represented by that vector element. SoftMax is applied as an activation function and then argmax to select the element highest value.

# Rectified Linear Unit (ReLU)

A Rectified Linear Unit is used as a non-linear activation function. A ReLU says if the value is less than zero, round it up to zero.

# *Normalization*

Normalization is the process of subtracting the mean and dividing by the standard deviation. It transforms the range of the data to be between -1 and 1 making the data use the same scale, sometimes called Min-Max scaling.

It is common to normalize the input features, standardizing the data by removing the mean and scaling to unit variance. It is often important the input features are centered around zero and have variance in the same order.

With some data, such as images the data is scaled so that its range is between 0 and 1, most simply dividing the pixel values by 255.

This also allows the training process to find the optimal parameters quicker.

# *Batch normalization*

Batch normalization has the benefits of helping to make a network output more stable predictions, reduce overfitting through regularization and speeds up training by an order of magnitude.

Batch normalization is the process of carrying normalization within the scope activation layer of the current batch, subtracting the mean of the batch’s activations and dividing by the standard deviation of the batch’s activations.

This is necessary as even after normalizing the input as some activations can be higher, which can cause the subsequent layers to act abnormally and makes the network less stable.

As batch normalization has scaled and shifted the activation outputs, the weights in the next layer will no longer be optimal. Stochastic gradient descent (SGD) would undo the normalization, as it would minimize the loss function.

To prevent this effect two trainable parameters can be added to each layer to allow SGD to deformalize the output. These parameters are a mean parameter “beta” and a standard deviation parameter “gamma”. Batch normalization sets these two weights for each activation output to allow the normalization to be reversed to get the raw input, this avoids affecting the stability of the network by avoiding having to update the other weights.

# Why CNNs are so powerful

In simple terms a large enough CNN can solve any solvable problem.

Notable CNN architecture’s that perform exceptionally well across many different image processing tasks are the VGG models (K.Simonian and A. Zisserman), the Reset models (Kaiming He et al) and the Google Inception models (Christian Szeged et al). These models have millions of trainable parameters.



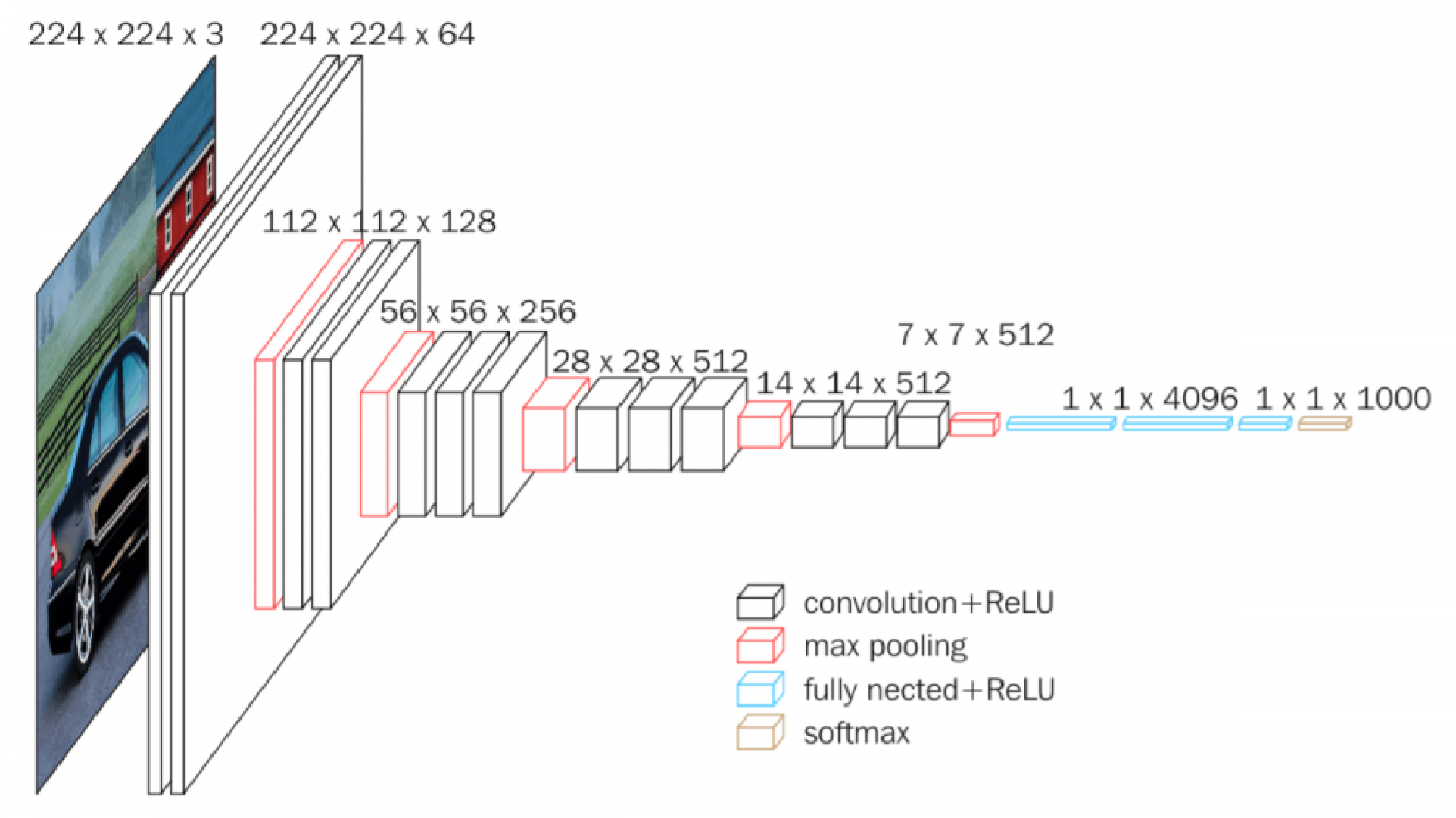


Fig 5.13 VGG-16 Network architecture

## *Universal approximation theorem*

The Universal approximation theorem essentially states if a problem can be solved it can be solved by deep neural networks, given enough layers of affine functions layered with non-linear functions. Essentially a stack of linear functions followed by non-linear functions could solve any problem that is solvable.

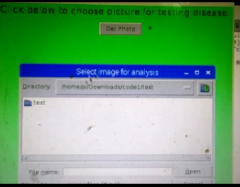
Practically in implementation this can be many matrices multiplication with large enough matrices followed by RELU, stacked together these have a mathematical property resulting in being able to solve any arbitrary complex mathematical function to any arbitrary high level of accuracy assuming you have the time and resource to train it.

Whether this would give the neural network understanding is a debated topic, especially by cognitive scientists. The argument is that no matter how well you approximate the syntax and semantics of a problem, you never understand it. This is basically the foundation of Searle’s Chinese Room Argument. Some would argue that does it matter if you can approximate the solution to the problem well enough that it’s indistinguishable from understanding the problem.

**CHAPTER-6**

**RESULT ANALYSYS**

For the detection, leaves of diseases are selected. The database of healthy leaves and diseased leaves is created at the server. This is necessary to compare the images with diseased and healthy leaves. Hence by comparison, the disease type is classified. Figure 6, figure 7, figure 8 shows the output screenshots of proposed system.



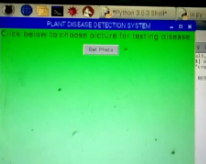


Fig 6.1 Accessing images.

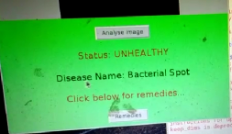
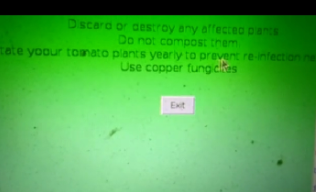


Fig.6.2. Displaying disease details. Fig.6.3. Displaying Solutions.

**CHAPTER-7**

**CONCLUSION**

Basically, there are three main types of Leaf disease, they are Bacterial, Fungal and Viral. It is important in plant disease detection to have the accuracy in the plant disease detection but at the same time the process should be of high speed. Work can be extended by the use of quad copter for the capturing of images of leaves of the different plants in the farm at field level. This system can be connected to the server for further processing. The objective of this work is the detection, classification of leaf diseases using image processing tools and all information about the disease is sent to the farmer’s mobile phone through the internet. To increase the speed and accuracy of detection as well as classification of leaf diseases we using Raspberry pi 3 model B module. One more important benefit of this system is that it gives the name of the pesticide required to use in order to prevent the disease from spreading. It providing exact name of pesticide as per the disease, to save labor price by eliminating need of labor for regular observation of plants to check whether it is affected by any disease or not. This system will largely contribute in growth in the yield of the farms.

**REFERENCES**

[1] Sachin D. Khirade, A.B Patil, “Plant Disease Detection Using Image Processing”, International Conference on Computing Communication Control and Automation”, 2015.

[2] Prof. Sanjay B. Dhaygude, Mr.Nitin P.Kumbhar, “Agricultural plant Leaf Disease Detection Using Image Processing”, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 2, Issue 1, January 2013.

[3] Amandeep Singh ,Maninder Lal Singh, “Automated Color Prediction of Paddy Crop Leaf using Image Processing”, International Conference on Technological Innovations in ICT for Agriculture and Rural Development (TIAR 2015), 2015.

[4] M.Malathi, K.Aruli , S.Mohamed Nizar, A.Sagaya Selvaraj, “A Survey on Plant Leaf Disease Detection Using Image Processing Techniques”,International Research Journal of Engineering and Technology (IRJET),Volume: 02 Issue: 09, Dec 2015.

[5] Malvika Ranjan, Manasi Rajiv Weginwar, NehaJoshi, Prof.A.B. Ingole, “detection and classification of leaf disease using artificial neural network”, International Journal of Technical Research and Applications, 2015.

[6] Y.Sanjana, AshwathSivasamy, SriJayanth, “Plant Disease Detection Using Image Processing Techniques”,International Journal of Innovative Research in Science, Engineering and Technology, Vol. 4, Special Issue 6, May 2015.

[7] Bhumika S.Prajapati, Vipul K.Dabhi Harshadkumar, B.Prajapati, “A Survey on Detection and Classification of Cotton Leaf Diseases”, International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) – 2016.

[8] P.Revathi, M.Hemalatha, “Advance Computing Enrichment Evaluation of Cotton Leaf Spot Disease Detection U sing Image Edge detection”, ICCCNT'12.

[9] Mr. Pramod S. landge, Sushil A. Patil, Dhanashree S. Khot, “Automatic Detection and Classification of Plant Disease through Image Processing”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 7, 2013.

[10] Heeb Al Bashish, Malik Braik, and Sulieman Bani-Ahmad, “A Framework for Detection and Classification of Plant Leaf and Stem Diseases”, IEEE 2010

[11] M. Reyalat,Al Hiary, Bani Ahmad,Braik and Z. Rahamneh, “Fast and Accurate Detection and Classification of Plant Diseases”.

[12] Smith J S., Camargo A., “An imageprocessing based algorithm to automatically identify plant disease visual symptoms”.

[13] Sulaiman, N., Mustapha,“K-means clustering to improve the accuracy”.

[14] David Mount,Tapas kanungo, “An Efficient kmeans Clustering Algorithm: Analysis and Implementation”.

[15] Ford A., Robert, “Colour Space Conversions”,.

[16] C M Procopiuc, P. K. Agarwal, “Exact and Approximation Algorithms for Clustering”.

[17] Devendra P. Marathe, V. A. Patil, V. D. Chaudhari, “Reliable massage transfer to alternate ways i.e. Wi-Fi, Bluetooth, ZigBee, Internet, GPRS/GSM of communication Media”, IJET, vol 5, issue 7. Pp 2015

[18] [Jiang Lu, Jie Hu, Guannan Zhao, Fenghua Mei, Changshui Zhang, Anin-field automatic wheat disease diagnosis system, Computers and](https://www.ijert.org/cfp) [Electronics in Agriculture 142 (2017) 369–379.](https://www.ijert.org/cfp)

[19] [Andreas Kamilaris, Francesc X. Prenafeta-Boldu Deep learning in](https://www.ijert.org/cfp) [agriculture: A survey, Computers and Electronics in Agriculture 147(2018) 70–90.](https://www.ijert.org/cfp)

**APPENDIX**

import tkinter as tk

from tkinter.filedialog import askopenfilename

import shutil

import os

import sys

import PIL

from PIL import Image, ImageTk

window = tk.Tk()

window.title("Dr. Plant")

window.geometry("500x510")

window.configure(background ="lightgreen")

title = tk.Label(text="Click below to choose picture for testing disease....", background = "lightgreen", fg="Brown", font=("", 15))

title.grid()

def bact():

window.destroy()

window1 = tk.Tk()

window1.title("Dr. Plant")

window1.geometry("500x510")

window1.configure(background="lightgreen")

def exit():

window1.destroy()

rem = "The remedies for Bacterial Spot are:\n\n "

remedies = tk.Label(text=rem, background="lightgreen",

fg="Brown", font=("", 15))

remedies.grid(column=0, row=7, padx=10, pady=10)

rem1 = " Discard or destroy any affected plants. \n Do not compost them. \n Rotate yoour tomato plants yearly to prevent re-infection next year. \n Use copper fungicites"

remedies1 = tk.Label(text=rem1, background="lightgreen",

fg="Black", font=("", 12))

remedies1.grid(column=0, row=8, padx=10, pady=10)

button = tk.Button(text="Exit", command=exit)

button.grid(column=0, row=9, padx=20, pady=20)

window1.mainloop()

def vir():

window.destroy()

window1 = tk.Tk()

window1.title("Dr. Plant")

window1.geometry("650x510")

window1.configure(background="lightgreen")

def exit():

window1.destroy()

rem = "The remedies for Yellow leaf curl virus are: "

remedies = tk.Label(text=rem, background="lightgreen",

fg="Brown", font=("", 15))

remedies.grid(column=0, row=7, padx=10, pady=10)

rem1 = " Monitor the field, handpick diseased plants and bury them. \n Use sticky yellow plastic traps. \n Spray insecticides such as organophosphates, carbametes during the seedliing stage. \n Use copper fungicites"

remedies1 = tk.Label(text=rem1, background="lightgreen",

fg="Black", font=("", 12))

remedies1.grid(column=0, row=8, padx=10, pady=10)

button = tk.Button(text="Exit", command=exit)

button.grid(column=0, row=9, padx=20, pady=20)

window1.mainloop()

def latebl():

window.destroy()

window1 = tk.Tk()

window1.title("Dr. Plant")

window1.geometry("520x510")

window1.configure(background="lightgreen")

def exit():

window1.destroy()

rem = "The remedies for Late Blight are: "

remedies = tk.Label(text=rem, background="lightgreen",

fg="Brown", font=("", 15))

remedies.grid(column=0, row=7, padx=10, pady=10)

rem1 = " Monitor the field, remove and destroy infected leaves. \n Treat organically with copper spray. \n Use chemical fungicides,the best of which for tomatoes is chlorothalonil."

remedies1 = tk.Label(text=rem1, background="lightgreen",

fg="Black", font=("", 12))

remedies1.grid(column=0, row=8, padx=10, pady=10)

button = tk.Button(text="Exit", command=exit)

button.grid(column=0, row=9, padx=20, pady=20)

window1.mainloop()

def analysis():

import cv2 # working with, mainly resizing, images

import numpy as np # dealing with arrays

import os # dealing with directories

from random import shuffle # mixing up or currently ordered data that might lead our network astray in training.

from tqdm import \

tqdm # a nice pretty percentage bar for tasks. Thanks to viewer Daniel BA1/4hler for this suggestion

verify\_dir = 'testpicture'

IMG\_SIZE = 50

LR = 1e-3

MODEL\_NAME = 'healthyvsunhealthy-{}-{}.model'.format(LR, '2conv-basic')

def process\_verify\_data():

verifying\_data = []

for img in tqdm(os.listdir(verify\_dir)):

path = os.path.join(verify\_dir, img)

img\_num = img.split('.')[0]

img = cv2.imread(path, cv2.IMREAD\_COLOR)

img = cv2.resize(img, (IMG\_SIZE, IMG\_SIZE))

verifying\_data.append([np.array(img), img\_num])

np.save('verify\_data.npy', verifying\_data)

return verifying\_data

verify\_data = process\_verify\_data()

#verify\_data = np.load('verify\_data.npy')

import tflearn

from tflearn.layers.conv import conv\_2d, max\_pool\_2d

from tflearn.layers.core import input\_data, dropout, fully\_connected

from tflearn.layers.estimator import regression

import tensorflow as tf

tf.reset\_default\_graph()

convnet = input\_data(shape=[None, IMG\_SIZE, IMG\_SIZE, 3], name='input')

convnet = conv\_2d(convnet, 32, 3, activation='relu')

convnet = max\_pool\_2d(convnet, 3)

convnet = conv\_2d(convnet, 64, 3, activation='relu')

convnet = max\_pool\_2d(convnet, 3)

convnet = conv\_2d(convnet, 128, 3, activation='relu')

convnet = max\_pool\_2d(convnet, 3)

convnet = conv\_2d(convnet, 32, 3, activation='relu')

convnet = max\_pool\_2d(convnet, 3)

convnet = conv\_2d(convnet, 64, 3, activation='relu')

convnet = max\_pool\_2d(convnet, 3)

convnet = fully\_connected(convnet, 1024, activation='relu')

convnet = dropout(convnet, 0.8)

convnet = fully\_connected(convnet, 4, activation='softmax')

convnet = regression(convnet, optimizer='adam', learning\_rate=LR, loss='categorical\_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard\_dir='log')

if os.path.exists('{}.meta'.format(MODEL\_NAME)):

model.load(MODEL\_NAME)

print('model loaded!')

import matplotlib.pyplot as plt

fig = plt.figure()

for num, data in enumerate(verify\_data):

img\_num = data[1]

img\_data = data[0]

y = fig.add\_subplot(3, 4, num + 1)

orig = img\_data

data = img\_data.reshape(IMG\_SIZE, IMG\_SIZE, 3)

# model\_out = model.predict([data])[0]

model\_out = model.predict([data])[0]

if np.argmax(model\_out) == 0:

str\_label = 'healthy'

elif np.argmax(model\_out) == 1:

str\_label = 'bacterial'

elif np.argmax(model\_out) == 2:

str\_label = 'viral'

elif np.argmax(model\_out) == 3:

str\_label = 'lateblight'

if str\_label =='healthy':

status ="HEALTHY"

else:

status = "UNHEALTHY"

message = tk.Label(text='Status: '+status, background="lightgreen",

fg="Brown", font=("", 15))

message.grid(column=0, row=3, padx=10, pady=10)

if str\_label == 'bacterial':

diseasename = "Bacterial Spot "

disease = tk.Label(text='Disease Name: ' + diseasename, background="lightgreen",

fg="Black", font=("", 15))

disease.grid(column=0, row=4, padx=10, pady=10)

r = tk.Label(text='Click below for remedies...', background="lightgreen", fg="Brown", font=("", 15))

r.grid(column=0, row=5, padx=10, pady=10)

button3 = tk.Button(text="Remedies", command=bact)

button3.grid(column=0, row=6, padx=10, pady=10)

time.sleep(2)

elif str\_label == 'viral':

diseasename = "Yellow leaf curl virus "

disease = tk.Label(text='Disease Name: ' + diseasename, background="lightgreen",

fg="Black", font=("", 15))

disease.grid(column=0, row=4, padx=10, pady=10)

r = tk.Label(text='Click below for remedies...', background="lightgreen", fg="Brown", font=("", 15))

r.grid(column=0, row=5, padx=10, pady=10)

button3 = tk.Button(text="Remedies", command=vir)

button3.grid(column=0, row=6, padx=10, pady=10

time.sleep(2)

elif str\_label == 'lateblight':

diseasename = "Late Blight "

disease = tk.Label(text='Disease Name: ' + diseasename, background="lightgreen",

fg="Black", font=("", 15))

disease.grid(column=0, row=4, padx=10, pady=10)

r = tk.Label(text='Click below for remedies...', background="lightgreen", fg="Brown", font=("", 15))

r.grid(column=0, row=5, padx=10, pady=10)

button3 = tk.Button(text="Remedies", command=latebl)

button3.grid(column=0, row=6, padx=10, pady=10)

time.sleep(2)

else:

r = tk.Label(text='Plant is healthy', background="lightgreen", fg="Black",

font=("", 15))

r.grid(column=0, row=4, padx=10, pady=10)

button = tk.Button(text="Exit", command=exit)

button.grid(column=0, row=9, padx=20, pady=20)

time.sleep(2)

def openphoto():

dirPath = "testpicture"

fileList = os.listdir(dirPath)

for fileName in fileList:

os.remove(dirPath + "/" + fileName)

# C:/Users/sagpa/Downloads/images is the location of the image which you want to test..... you can change it according to the image location you have

fileName = askopenfilename(initialdir='/home/pi/Desktop/code/test/test/', title='Select image for analysis ',

filetypes=[('image files', '.jpg')])

dst = "/home/pi/Desktop/code/testpicture"

shutil.copy(fileName, dst)

load = Image.open(fileName)

render = ImageTk.PhotoImage(load)

img = tk.Label(image=render, height="250", width="500")

img.image = render

img.place(x=0, y=0)

img.grid(column=0, row=1, padx=10, pady = 10)

title.destroy()

button1.destroy()

button2 = tk.Button(text="Analyse Image", command=analysis)

button2.grid(column=0, row=2, padx=10, pady = 10)

button1 = tk.Button(text="Get Photo", command = openphoto)

button1.grid(column=0, row=1, padx=10, pady = 10)

window.mainloop()

///2nd

import tkinter as tk

from tkinter.filedialog import askopenfilename

import shutil

import os

import sys

import PIL

import http.client, urllib

from PIL import Image, ImageTk

import cv2

window = tk.Tk()

import RPi.GPIO as GPIO

from imutils.video import VideoStream

import time

import urllib.request

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

window.title("Dr. Plant")

video\_capture = cv2.VideoCapture(0)

vs = VideoStream(usePiCamera=True).start() # Raspberry Pi

time.sleep(2)

window.geometry("500x510")

window.configure(background ="lightgreen")

title = tk.Label(text="Click below to choose picture for testing disease....", background = "lightgreen", fg="Brown", font=("", 15))

title.grid()

def bact():

window.destroy()

window1 = tk.Tk()

window1.title("Dr. Plant")

window1.geometry("500x510")

window1.configure(background="lightgreen")

def exit():

window1.destroy()

rem = "The remedies for Bacterial Spot are:\n\n "

remedies = tk.Label(text=rem, background="lightgreen",

fg="Brown", font=("", 15))

remedies.grid(column=0, row=7, padx=10, pady=10)

rem1 = " Discard or destroy any affected plants. \n Do not compost them. \n Rotate yoour tomato plants yearly to prevent re-infection next year. \n Use copper fungicites"

remedies1 = tk.Label(text=rem1, background="lightgreen",

fg="Black", font=("", 12))

remedies1.grid(column=0, row=8, padx=10, pady=10)

button = tk.Button(text="Exit", command=exit)

button.grid(column=0, row=9, padx=20, pady=20)

window1.mainloop()

def vir():

window.destroy()

window1 = tk.Tk()

window1.title("Dr. Plant")

window1.geometry("650x510")

window1.configure(background="lightgreen")

def exit():

window1.destroy()

rem = "The remedies for Yellow leaf curl virus are: "

remedies = tk.Label(text=rem, background="lightgreen",

fg="Brown", font=("", 15))

remedies.grid(column=0, row=7, padx=10, pady=10)

rem1 = " Monitor the field, handpick diseased plants and bury them. \n Use sticky yellow plastic traps. \n Spray insecticides such as organophosphates, carbametes during the seedliing stage. \n Use copper fungicites"

remedies1 = tk.Label(text=rem1, background="lightgreen",

fg="Black", font=("", 12))

remedies1.grid(column=0, row=8, padx=10, pady=10)

button = tk.Button(text="Exit", command=exit)

button.grid(column=0, row=9, padx=20, pady=20)

window1.mainloop()

def latebl():

window.destroy()

window1 = tk.Tk()

window1.title("e-AGROBOT")

window1.geometry("520x510")

window1.configure(background="lightgreen")

def exit():

window1.destroy()

rem = "The remedies for Late Blight are: "

remedies = tk.Label(text=rem, background="lightgreen",

fg="Brown", font=("", 15))

remedies.grid(column=0, row=7, padx=10, pady=10)

rem1 = " Monitor the field, remove and destroy infected leaves. \n Treat organically with copper spray. \n Use chemical fungicides,the best of which for tomatoes is chlorothalonil."

remedies1 = tk.Label(text=rem1, background="lightgreen",

fg="Black", font=("", 12))

remedies1.grid(column=0, row=8, padx=10, pady=10)

button = tk.Button(text="Exit", command=exit)

button.grid(column=0, row=9, padx=20, pady=20)

window1.mainloop()

def analysis():

import cv2 # working with, mainly resizing, images

import numpy as np # dealing with arrays

import os # dealing with directories

from random import shuffle # mixing up or currently ordered data that might lead our network astray in training.

from tqdm import \

tqdm # a nice pretty percentage bar for tasks. Thanks to viewer Daniel BA1/4hler for this suggestion

verify\_dir = 'testpicture'

IMG\_SIZE = 50

LR = 1e-3

MODEL\_NAME = 'healthyvsunhealthy-{}-{}.model'.format(LR, '2conv-basic')

def process\_verify\_data():

verifying\_data = []

for img in tqdm(os.listdir(verify\_dir)):

path = os.path.join(verify\_dir, img)

img\_num = img.split('.')[0]

img = cv2.imread(path, cv2.IMREAD\_COLOR)

img = cv2.resize(img, (IMG\_SIZE, IMG\_SIZE))

verifying\_data.append([np.array(img), img\_num])

np.save('verify\_data.npy', verifying\_data)

return verifying\_data

verify\_data = process\_verify\_data()

#verify\_data = np.load('verify\_data.npy')

import tflearn

from tflearn.layers.conv import conv\_2d, max\_pool\_2d

from tflearn.layers.core import input\_data, dropout, fully\_connected

from tflearn.layers.estimator import regression

import tensorflow as tf

tf.reset\_default\_graph()

convnet = input\_data(shape=[None, IMG\_SIZE, IMG\_SIZE, 3], name='input')

convnet = conv\_2d(convnet, 32, 3, activation='relu')

convnet = max\_pool\_2d(convnet, 3)

convnet = conv\_2d(convnet, 64, 3, activation='relu')

convnet = max\_pool\_2d(convnet, 3)

convnet = conv\_2d(convnet, 128, 3, activation='relu')

convnet = max\_pool\_2d(convnet, 3)

convnet = conv\_2d(convnet, 32, 3, activation='relu')

convnet = max\_pool\_2d(convnet, 3)

convnet = conv\_2d(convnet, 64, 3, activation='relu')

convnet = max\_pool\_2d(convnet, 3)

convnet = fully\_connected(convnet, 1024, activation='relu')

convnet = dropout(convnet, 0.8)

convnet = fully\_connected(convnet, 4, activation='softmax')

convnet = regression(convnet, optimizer='adam', learning\_rate=LR, loss='categorical\_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard\_dir='log')

if os.path.exists('{}.meta'.format(MODEL\_NAME)):

model.load(MODEL\_NAME)

print('model loaded!')

import matplotlib.pyplot as plt

fig = plt.figure()

for num, data in enumerate(verify\_data):

img\_num = data[1]

img\_data = data[0]

y = fig.add\_subplot(3, 4, num + 1)

orig = img\_data

data = img\_data.reshape(IMG\_SIZE, IMG\_SIZE, 3)

# model\_out = model.predict([data])[0]

model\_out = model.predict([data])[0]

if np.argmax(model\_out) == 0:

str\_label = 'healthy'

elif np.argmax(model\_out) == 1:

str\_label = 'bacterial'

elif np.argmax(model\_out) == 2:

str\_label = 'viral'

elif np.argmax(model\_out) == 3:

str\_label = 'lateblight'

if str\_label =='healthy':

status ="HEALTHY"

else:

status = "UNHEALTHY"

message = tk.Label(text='Status: '+status, background="lightgreen",

fg="Brown", font=("", 15))

message.grid(column=0, row=3, padx=10, pady=10)

if str\_label == 'bacterial'

diseasename = "Bacterial Spot "

disease = tk.Label(text='Disease Name: ' + diseasename, background="lightgreen",

fg="Black", font=("", 15))

disease.grid(column=0, row=4, padx=10, pady=10)

r = tk.Label(text='Click below for remedies...', background="lightgreen", fg="Brown", font=("", 15))

r.grid(column=0, row=5, padx=10, pady=10)

button3 = tk.Button(text="Remedies", command=bact)

button3.grid(column=0, row=6, padx=10, pady=10)

elif str\_label == 'viral':

diseasename = "Yellow leaf curl virus "

disease = tk.Label(text='Disease Name: ' + diseasename, background="lightgreen",

fg="Black", font=("", 15))

disease.grid(column=0, row=4, padx=10, pady=10)

r = tk.Label(text='Click below for remedies...', background="lightgreen", fg="Brown", font=("", 15))

r.grid(column=0, row=5, padx=10, pady=10)

button3 = tk.Button(text="Remedies", command=vir)

button3.grid(column=0, row=6, padx=10, pady=10)

elif str\_label == 'lateblight':

diseasename = "Late Blight "

disease = tk.Label(text='Disease Name: ' + diseasename, background="lightgreen",

fg="Black", font=("", 15))

disease.grid(column=0, row=4, padx=10, pady=10)

r = tk.Label(text='Click below for remedies...', background="lightgreen", fg="Brown", font=("", 15))

r.grid(column=0, row=5, padx=10, pady=10)

button3 = tk.Button(text="Remedies", command=latebl)

button3.grid(column=0, row=6, padx=10, pady=10)

else:

r = tk.Label(text='Plant is healthy', background="lightgreen", fg="Black",

font=("", 15))

r.grid(column=0, row=4, padx=10, pady=10)

button = tk.Button(text="Exit", command=exit)

button.grid(column=0, row=9, padx=20, pady=20)

print('Status: ' + str(str\_label))

time.sleep(5)

humidity, temperature = Adafruit\_DHT.read\_retry(sensor, sensor\_pin)

def openphoto():

while(True):

for i in range(100):

print (i)

frame = vs.read()

cv2.imshow('Video1', frame)

key= cv2.waitKey(1)

cv2.imwrite('/home/pi/Desktop/code/testpicture/1.JPG',frame)

fileName='/home/pi/Desktop/code/testpicture/1.JPG'

load = Image.open(fileName)

render = ImageTk.PhotoImage(load)

img = tk.Label(image=render, height="250", width="500")

img.image = render

img.place(x=0, y=0)

img.grid(column=0, row=1, padx=10, pady = 10)

title.destroy()

analysis()

openphoto()

window.mainloop()