October University for Modern Science & Arts
Faculty of Engineering
Department of Electrical Communication and Electronics Systems

Plant Wheat Detection Using Artificial Intelligence

A Graduation Project
Submitted in Partial Fulfillment of B.Sc. Degree Requirements in Electrical and Computer Engineering
Part II

Prepared By

Marco Ayman Gad	185189
Mahmoud Youssef El Azzab	175133

Supervised By
Dr. Mohamed Gamal

2020-2021

TABLE OF CONTENTS

List of Tables

List of Figures

List of Abbreviations

ANN	**Artificial Neural Network**
BLE	**Bluetooth Low Energy**
BMU	**Best Matching Unit**
CNNs	**Convolutional Neural Networks**
CPU	**Central Processing Unit**
CSI	**Camera Socket Interface**
DBNs	**Deep Belief Networks**
DSI	**Data Socket Interface**
GANs	**Generative Adversarial Networks**
GPU	**Graphics Processing Unit**
LSTMs	**Long Short-Term Memory Networks**
MLPs	**Multilayer Perceptron**
OLS	**Ordinary Least Squares**
RAM	**Random Access Memory**
RBFNs	**Radial Basis Function Networks**
RBMs	**Restricted Boltzmann Machines**
ReLU	**Rectified Linear Unit**
RNNs	**Recurrent Neural Networks**
ROI	**Region of Interest**
RPN	**Region Proposal Network**
SoC	**System on Chip**
SOMs	**Self-Organizing Maps**
USB	**Universal Serial Bus**

Acknowledgment

We want to thank our university MSA for giving us the best high education level that let us choose such that project, and helping us to understand a lot of features behind the major that we are concerned on.

Special thanks to **Dr. Mohammed Gamal**, Electrical communication and electronics Engineering Department, University of MSA, our senior design project advisor for his sincere advises and wonderful contributions throughout the project.

Many thanks go to TA. **Mohammed Sayed Zaky**, Electrical communication and electronics Engineering Department, University of MSA, for his helpful advises and great assistance throughout the project.

Many thanks goes to **TA.Mariam Hamza**, Electrical communication and electronics Engineering Department, University of MSA, for her great assistance in the graphical matters throughout this project, and her announcements about the deadlines of each report.

We are grateful to MSA IT teams for help providing the access to the high-performance computer to perform our project.

Finally, our endless thanks go to our parents for their support towards the accomplishment of this project and during our stay on campus throughout the bachelor studies.

Abstract

In the 20th century, there is a threefold increase in the population of the globe. It is very difficult for us to meet the food demand of the rising population. The development in agriculture is supported by advanced technology. The higher authorities or government's all over the globe square measure according to rising the assembly of some important cereals including wheat, rice, and maize, defrayal massive number of cases to confirm the food demand of the nation. Rapidly developing countries like Egypt also highly active and they have advance planning for agriculture development in the countries.

Agriculture is a most important and ancient occupation in Egypt. As economy of Egypt is based on agricultural production, utmost care of food production is necessary. Pests like virus, fungus and bacteria causes infection to plants with loss in quality and quantity production. There is large amount of loss of farmer in production. Hence proper care of plants is necessary for same.

This project analyzes the detection and the growth stages of the wheat crop by capturing a digital image of the crop from time to time. Later on, the collected image data is transferred to a computer. Further analysis of the images was done by using OpenCV and Raspberry pi. The Wheat crop's green pigment decreases with age. In the early growth stages, wheat crop is having the maximum green pigments which become minimum at the maturity stage. Therefore, the maturity of the wheat crop analyzed by measuring the percentage of green colour pigment that present in the wheat crop. This measurement of the percentage of green colour that present in the wheat crop is done by using the Digital Image Processing technique.

Chapter I
INTRODUCTION

1.1 Problem Definition

Egypt is a cultivated country and about 70% of the population depends on agriculture. Farmers have large range of diversity for selecting various suitable crops and finding the suitable pesticides for plant. Disease on plant leads to the significant reduction in both the quality and quantity of agricultural products[1]. The studies of plant disease refer to the studies of visually observable patterns on the plants. Monitoring of health and disease on plant plays an important role in successful cultivation of crops in the farm. In early days, the expertise person in that field did the monitoring and analysis of plant diseases manually. This requires tremendous amount of work and requires excessive processing time. The image processing techniques can be used in the plant disease detection [2]. In most of the cases disease symptoms are seen on the leaves, stem and fruit. The plant leaf for the detection of disease is considered which shows the disease symptoms. This paper gives the introduction to image processing technique used for plant disease detection.

Wheat is the most cultivated cereal crop in the world, along with rice and maize. Wheat breeding progress in the 1950s was vital for food security of emerging countries when Norman Borlaug developed semi-dwarf kinds of wheat and a complementary agronomy system (the Doubly Green Revolution), saving 300 million people from starvation [1]. However, after increasing rapidly for decades, the rate of increase in wheat yields has slowed down since the early 1990s [2], [3]. Traditional breeding still relies to a large degree on manual observation. Innovations that increase genetic gain may come from genomic selection, new high-throughput phenotyping techniques or a combination of both [4]–[7]. These techniques are essential to select important wheat traits linked to yield potential, disease resistance or adaptation to abiotic stress. Even though high throughput phenotypic data acquisition is already a reality, developing efficient and robust models to extract traits from raw data remains a significant challenge. Among all traits, wheat head density (the number of wheat heads per unit ground area) is a major yield component and is still manually evaluated in breeding trials, which is labour intensive and leads to measurement errors of around

10% [8], [9]. Thus, developing image-based methods to increase the throughput and accuracy of counting wheat heads in the field is needed to help breeders manipulate the balance between yield components (plant number, head density, grains per head, grain weight) in their breeding selections.

1.2 Applied System

Wheat growth can be broadly divided into different stages. Several machine learning applications have been developed to identify wheat leaf diseases, weed detection in wheat, soil management, water stress management for wheat and may more areas. Best of my knowledge there is no effective and efficient work done to identify the growth stages of wheat using machine learning applications. In this related work I have been cover last three decades work for wheat crop. In very first, Zhang, N. et al. 1995 [10] used shape and geometrical features for weed detection in wheat crop. Several image features like compactness, eight the invariant moments and eccentricity for batter result. In another paper Moshou, D. et al. 2004 [11] developed a system with image spectral reflectance features to detection of yellow rust infected and healthy winter wheat canopies and artificial neural network was used to detection application. Result accuracy of healthy wheat were 98.9% and accuracy of infected yellow rust was 99.4%. Moshou, D. et al. 2005 [12] developed a system to identify yellow rust infected and healthy winter wheat under field circumstances. And artificial neural network is used for detecting yellow. Result accuracy achieved for yellow rust infected wheat 99.4% and accuracy for healthy wheat were 98.7%. In another paper Moshou, D. et al. 2006 [13] developed a system to identify field condition for healthy winter wheat and discrimination of nitrogen, yellow rust infected and also stressed. Artificial neural network and SOM were used in detection system.

Result accuracy for nitrogen stressed were 100% while accuracy for yellow rust infected wheat were 99.92% and accuracy for healthy wheat were 99.39%.In paper Guevara-Hernandez, F. et al. 2011[14] on the basis of external characteristics authors reorganized two grain types using different image features like colour, shape and textural features. The overall result accuracy using selected features property like shape, colour and texture were 99%. Tian, Y. et al. 2012 [15] reorganized leaf rust Pucciniatriticina, pucciniastriformis and leaf blight, powdery mildew leaf diseases in wheat crop. Identification of these leaf diseases authors use wheat image colour, texture and shape features. SVM based multiple classifier system were used for detection, accuracy rate of result 95.16%. Moshou, D. et al. 2014 [16] developed an application to identify water stress detection in wheat crop based on optical multisensory fusion with a least square support vector machine classifier. Accuracy in result, achieved in four categories. First category was control treatment healthy and well supplied with water and its accuracy was 100%. Second category was healthy treatment and deficient water supply and its accuracy was also 100%.Third category was Inoculated treatment with sectorial trifici and well supplied with water and its accuracy was 98.75%. Last category was inoculated treatment and deficient water supply and its accuracy was 98.7%. In another paper Majumdar, D. et al. 2015 [17] developed a detection system for 4 different types of leaves disease using fuzzy clustering algorithms and result accuracy for classification of different diseases were 56%. Olgun, M. et al. 2016 [18] developed an application using K-Means clustering algorithm for grain grading of wheat overall result accuracy was 88.33%. Jiang, G. et al. 2016 [19] developed an application to reorganised weed row detection in wheat crop. This application used k-means clustering algorithm. Weed detection accuracy rate was up to 90%. Mondal, D. et al. 2016 [20] developed an identification application for disease detection on wheat leaves. Author used wheat image features for disease detection on wheat leaves and its accuracy rate was 94% for non-diseased wheat images and 95% for diseased wheat images. Pantazi, X.-E.et al. 2016 [21] developed a system for wheat yield prediction within field variation using artificial neural networks. It is very effective for wheat yield prediction. Yield prediction accuracy rate was 81.65%. the author [22] used EM remotely sensed Red, Green, Blue or RGB Images captured by UAV to identified tomatoes optimal clusters for the Image for determine using Bayesian information criterion. Expectation maximization, K-means, and Self-Organizing Map algorithm were used to categorize pixels into two group Non-tomatoes and tomatoes. UAV stand for unmanned aerial vehicle. Performance of purposed method demonstrated by two representative unmanned aerial vehicle images clicked at different-different height, Author found EM gives batter results then K-means and SOM. Zhang, J. et al. 2017a [23] done work for disease and pest detection. There were several image features used like image colour, image shape and texture. These image features of wheat image were used to prepare a system to reorganize diseases and pest detection in wheat crop production. Its result accuracy rate was 77%. At last Shi, Y. et al. 2017 [24] used spectral features of image for reorganization of diseases and pest detection using spectral image features like MSR, NRI, SIPI, NPCI and many more. Result classification accuracy was 82.9%, 87.9%, 89.2% for three occurrence levels such that slight, severe and moderate. In this paper, In this study author [25] developed a method to determine stage prediction for rice development. This method is based on basic geographic information (obtain from china weather station) and support vector machine. Such kind of study has the potential for feature integration to improve the prediction accuracy and practicability with extensive social and economic factor. In the same year in his first study author [26] provided a tool to discriminate infected by smut fungus Microbotyum Silybum during vegetative growth and healthy Silybum marianum plants. In their second study author present a new system to identify yellow rust infected and healthy, as well as detection of nitrogen, stressed winter wheat. This study helps to accurate detection of fertilizers and fungicides according to the needs of the plant. In same year author [27] presented a system for weed detection which is based on counter propagation artificial neural network (CP-ANN) and as well as unmanned aircraft system for multispectral images captured for the identification of Silybum marianum which is difficult to eliminate and causes major loss on crop yield production. In this same year work also done for disease detection according to author [28] developed a system for the classification of parasites which was based on image processing procedure and the automatic identification of trips in strawberry fruit greenhouse environment, for control in real-time. In this study, work is done to enhance the crop quality according to the author [29] present a method for segmenting the diseased part in the cucumber leaves this system was based on the k-means clustering algorithm. They use five multiclass classification for training to different among seven diseases. The result classification accuracy range was 91.25%.

1.3 Main Objectives

1. Determining the wheat plant Feature Extraction techniques.
2. Applying the detection and classification of Plant Diseases.
3. Building efficient and user-friendly system.
4. Train an initial model with labelled images from previous step.
5. Apply the initial model on original images.
6. Improving accuracy up to 86 % with the help of K mean clustering Algorithm and Deep learning algorithm.

7. Increasing layers of deep learning algorithm to get most accurate and appropriate result.
8. Increasing the response speed of the system to be around 300 ms.
9. Feed the generated labels and the image into the image annotator app for validating the bounding box locations and corrections by a human annotator.
10. - Labelling work for randomly selected original images.
11. Add corrected labels and the image to data pool.
12. Implementing the hardware and software of the project.

1.4Methodology

Figure 1.1 illustrates the flow chart of the project, in which the project starts by inputting the image to the microprocessor unit, and applying preprocessing techniques over the image, then performing segmentation for the input image, then performing feature extraction after training the system by using multiple wheat plant image. Operating convolution neural network algorithm to detect the wheat plant. Increase the probabilities of classification depending on knowledge base. Building CNN classifier to have a high accuracy detection. Getting the results of wheat plant detection in real time.

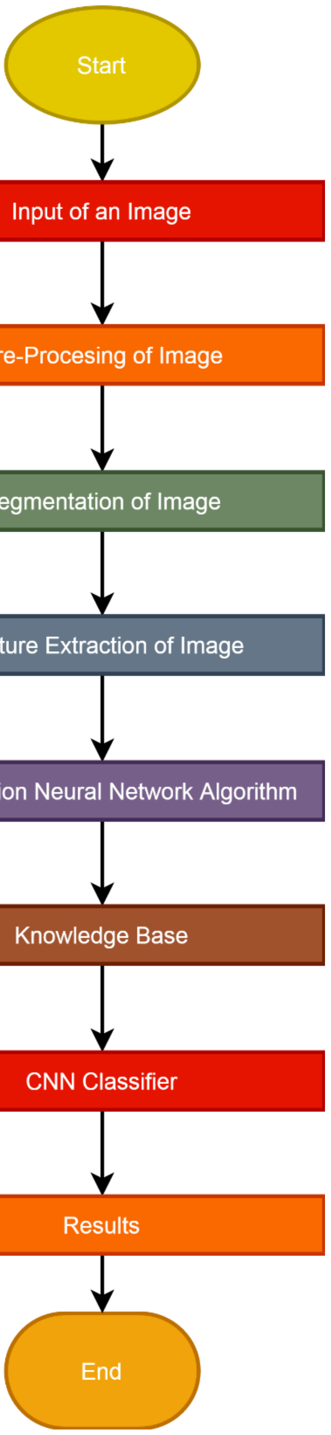


Fig. 1.1: Project Flow chart steps

1.5 Software

1- Python C

2- Open CV

3- Tensor Flow

4- Matlab tools

5- C programming

1.6 Report organization

Chapter I: first it begins by defining the problem definition of the project, second listing the applied system in wheat plant detection, third introducing main objectives of the project, fourth the methodology of the project steps, finally the software that can be used in the project.

Chapter II: first it begins with the survey literature over the traditional and AI techniques in image processing. Second the types of object detection will be presented. Finally, the agriculture deep learning applications.

Chapter III: proposed system block diagram, and the hardware are presented. Second the boards specifications that can be used in the project as well as the camera modules are listed.

Chapter IV: introduce circuit diagram, algorithms for the project and the summary

Chapter V: introduce hardware and software implementation

Chapter VI: Cost analysis of the project

Chapter VII: Time plan of the project

Chapter VIII: Project conclusion as well as the future time plan are presented.

Chapter II

Literature Survey

2.1 Traditional Vs Deep Learning

Deep learning has certainly revolutionized traditional image processing. It has pushed the boundaries of Artificial Intelligence to unlock potential opportunities across industry verticals.

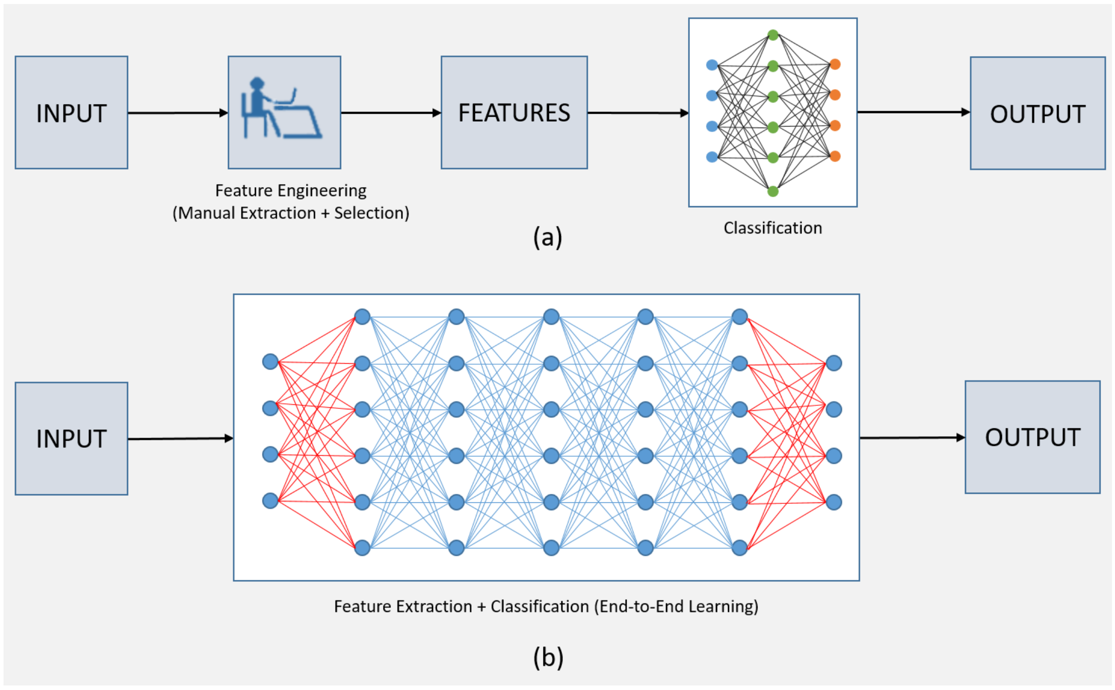

INPUT
Feature Engineering
(Manual Extraction + Selection)
FEATURES
Classification
OUTPUT
(a)
INPUT
Feature Extraction + Classification (End-to-End Learning)
OUTPUT
(b)

Fig. 2.1: (a) Traditional Image Processing workflow vs. (b) Deep Learning workflow.

Several challenges that once seemed impossible to solve, are now solved to a point where machines are performing better than humans. However, that does not mean that the traditional image processing techniques that have advanced in the years before the rise of DL have been made obsolete.

This chapter will analyze the benefits and drawbacks of each approach. This chapter aims to provide better clarity on the subject which can help data scientists/ industries choose the most suitable method depending on the task at hand.

2.2 Comparison of Deep Learning and Traditional Image processing

2.2.1 Advantages of DL

Rapid advancement in DL and enhancements in device capabilities including memory capacity, computing power, power consumption, optics, and image sensor resolution have accelerated the spread of vision-based applications along with improved performance and cost-effectiveness.

Since neural networks used in DL are trained rather than programmed, applications following this approach often require less fine-tuning and expert analysis. The availability of a humongous amount of video data in today’s system supports this cause. While CV algorithms tend to be more domain-specific, DL, on the other hand, provides superior flexibility because CNN models and frameworks can be re-trained using a custom dataset for any use case.

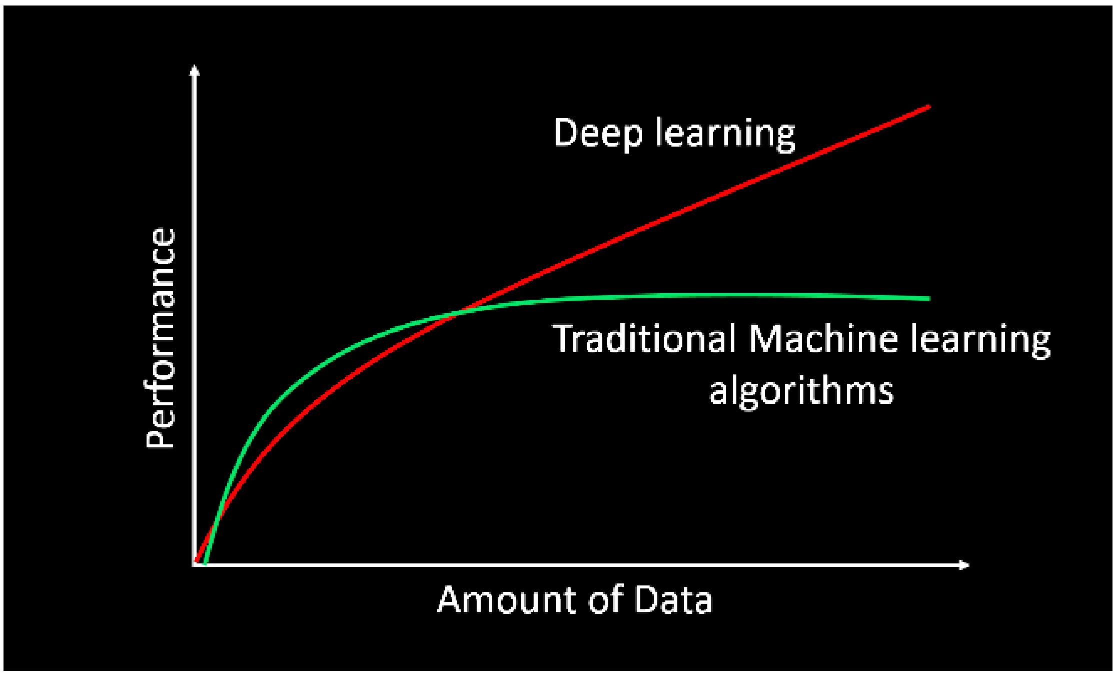


Fig. 2.2: Data vs Performance Comparison

2.2.2 Advantages of traditional Image processing

At times, deep learning is overkill as traditional image processing can often solve a given problem with greater accuracy and in fewer lines of code than DL. The features learned from a deep neural net are specific to the training dataset which if not well constructed, probably won’t perform well for images different from the training set. In contrast, algorithms like SIFT and even simple color thresholding and pixel counting algorithms are not class-specific, that is, they are very general and perform the same for any image.

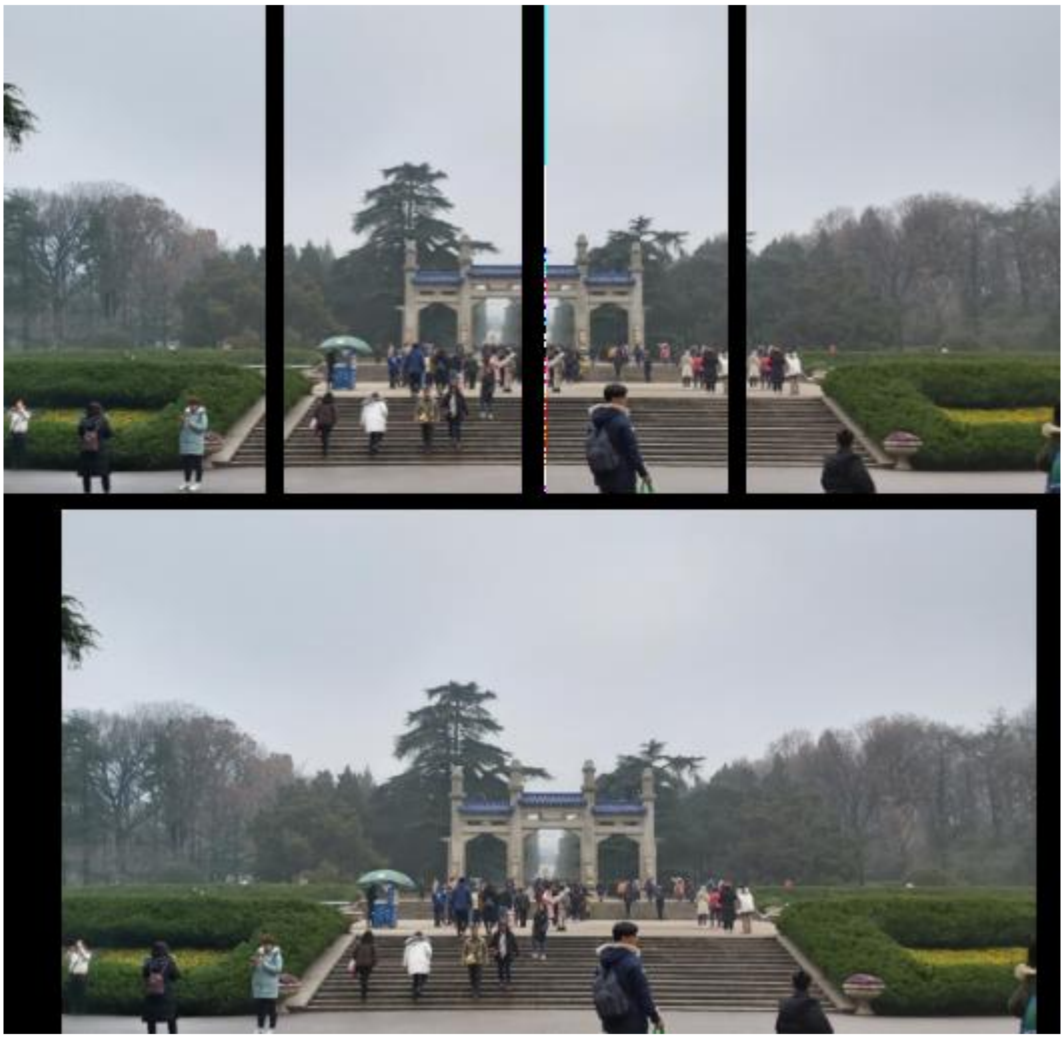


Fig. 2.3: Example of image stitching

Therefore, SIFT and other algorithms are often preferred for applications such as 3D mesh reconstruction/ image-stitching which do not need specific class knowledge. While the solutions to these tasks can be attained by training huge datasets, the vast research effort required for it is not feasible for a closed application. Summarily, deciding on the most suitable approach for a given computer vision problem, one should consider practical feasibility.

A product classification problem can be considered as an example. Supposing the problem aims to classify cans of food on a conveyor belt into either vegetarian or non-vegetarian distinguished by can color – green for veg. and red for non-veg. While the problem can be solved using accurate DL models generated by collecting sufficient training data, the traditional image processing is a much-preferred alternative in this scenario with its simple color thresholding technique. This example also highlights the fact that DL often fails to generalize the task at hand in the event of limited training dataset leading to over-fitting.

Manual tweaking of parameters of a model is a daunting task since a DNN consists of parameters in the order of millions inside it, each with complex interrelationships. As a result, DL models are censured to be a black-box. On the contrary, traditional image processing offers complete transparency and allows one a good estimate of how his/ her techniques will behave outside the training environment. It also offers flexibility to CV engineers to tweak their parameters to either improve their algorithm to achieve better accuracy and performance or investigate their mistakes when the algorithm fails. Traditional image processing is preferred for edge computing too, owing to its delivery of high performance with lower resource usage. This also makes traditional image processing more popular for cloud-based applications where high-powered resources that are required for deep learning applications are expensive.

2.2.3 Hybrid Approaches

Hybrid approaches are an amalgamation of traditional image processing and deep learning that present the best of both. They are gaining importance owing to their ability to maintain the right balance between mature and proven traditional image processing algorithms and versatile and accurate deep learning techniques.

Hybrid approaches have witnessed resounding success in medical image processing. Doctors can generally diagnose if a tumor is benign or malignant through mammal review, but hybridizing DL and CV capabilities allows us to automate this process and reduce the possibility of human error.

They are notably efficient in high-performance systems that require quick development. For example, an image processing algorithm can competently perform face detection over the live feed from a security camera. These detections can then be relayed to a DNN as the next stage for face recognition.

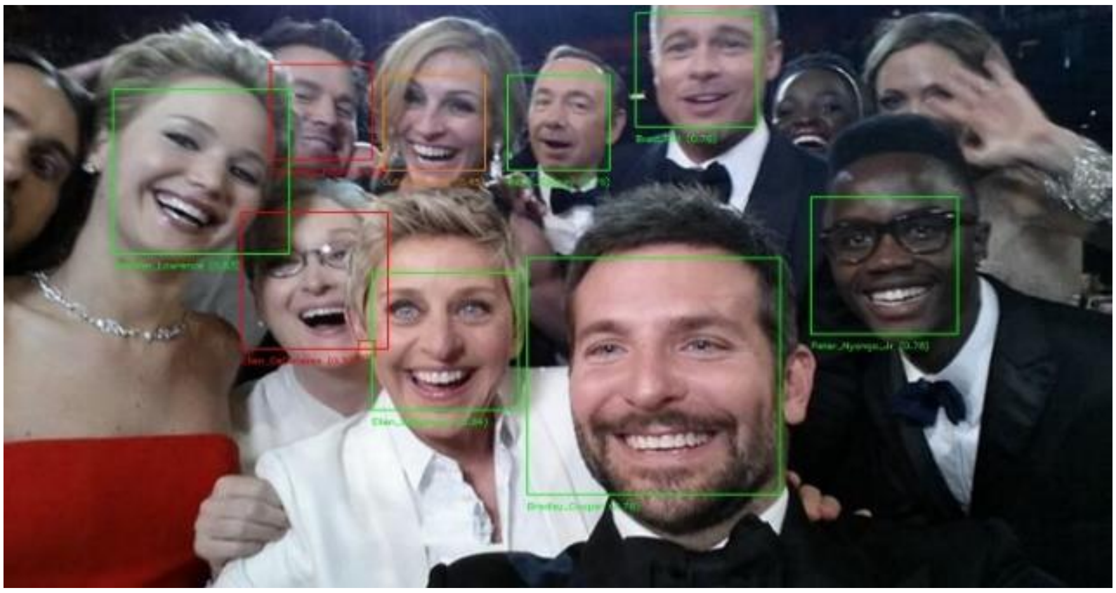


Fig. 2.4: Example of hybrid approach –face recognition

This helps the DNN to work only on a small patch of the image thereby, reducing the considerable amount of computing resources and training effort that would otherwise have been required to process the entire frame.

Fusion can also help achieve better accuracy. One such classic example is document processing where traditional image processing techniques are used for pre-processing tasks like noise reduction, skew detection/ correction, and localization of lines and words. This, when followed OCR using deep techniques, yield better accuracy.

The blend of machine learning metrics and deep networks has gained significance over the years, owing to the evidence that it results in better models. Hybrid vision processing implementations have proved a performance advantage while providing a 130x-1,000x reduction in multiply-accumulate operations and about 10x improvement in frame rates compared to a pure DL solution. Furthermore, the hybrid implementation requires significantly lower CPU resources and about half of the memory bandwidth.

2.3 Artificial Intelligence

To most people, the terms deep learning and machine learning seem like interchangeable buzzwords of the AI world. However, that’s not true. Hence, everyone who seeks to better understand the field of artificial intelligence should begin by understanding the terms and its differences. The good news: It’s not as difficult as some articles on the topic suggest.

To break it down in a single sentence: Deep learning is a specialized subset of machine learning which, in turn, is a subset of artificial intelligence. In other words, deep learning is machine learning.

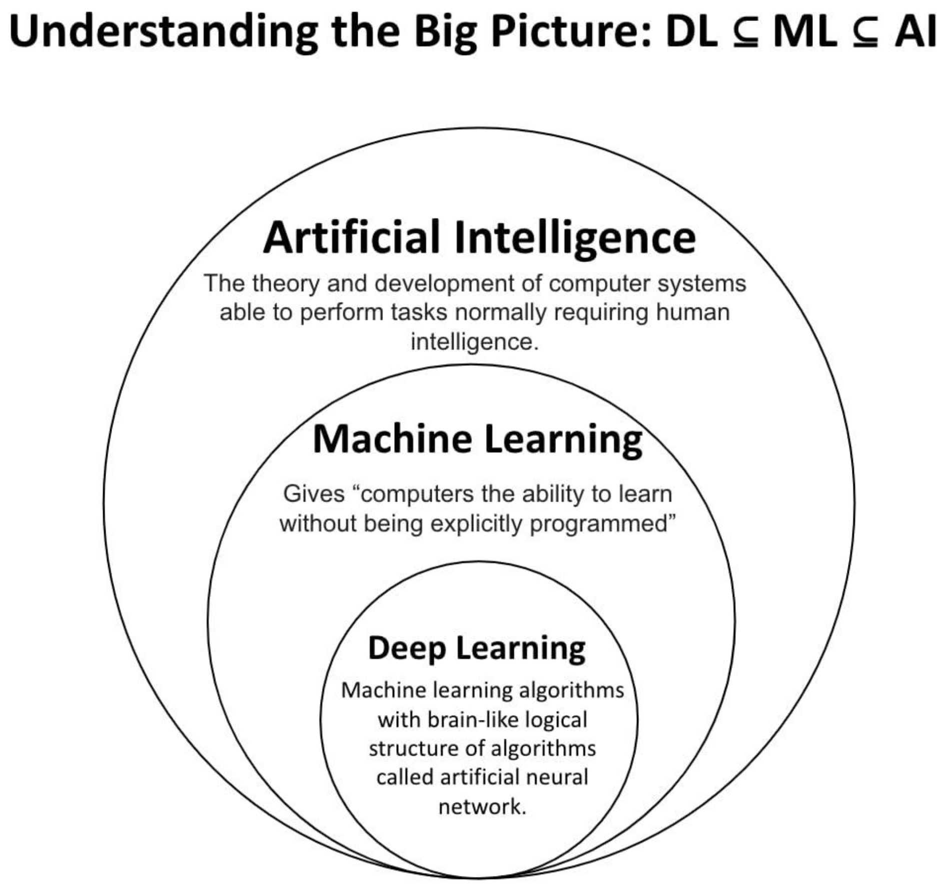

Understanding the Big Picture: DL ⊆ ML ⊆ AI
Artificial Intelligence
The theory and development of computer systems able to perform tasks normally requiring human intelligence.
Machine Learning
Gives "computers the ability to learn without being explicitly programmed"
Deep Learning
Machine learning algorithms with brain-like logical structure of algorithms called artificial neural network.

Fig. 2.5: AI Architecture

2.3.1 Machine Learning

Machine learning is the general term for when computers learn from data. It describes the intersect of computer science and statistics where algorithms are used to perform a specific task without being explicitly programmed; instead, they recognize patterns in the data and make predictions once new data arrives.

In general, the learning process of these algorithms can either be supervised or unsupervised, depending on the data being used to feed the algorithms. If you want to dive in a little bit deeper into the differences between supervised and unsupervised learning have a read through this article.

A traditional machine learning algorithm can be something as simple as linear regression. For instance, imagine you want to predict your income given your years of higher education. In a first step, you have to define a function, e.g. income = y + x * years of education. Then, give your algorithm a set of training data. This could be a simple table with data on some people's years of higher education and their associated income. Next, let your algorithm draw the line, e.g. through an ordinary least squares (OLS) regression. Now, you can give the algorithm some test data, e.g. your personal years of higher education, and let it predict your income.

While this example sounds simple it does count as machine learning – and yes, the driving force behind machine learning is ordinary statistics. The algorithm learned to make a prediction without being explicitly programmed, only based on patterns and inference.

So much about machine learning in general – to summarize:

- Machine learning is at the intersection of computer science and statistics through which computers receive the ability to learn without being explicitly programmed.
- There are two broad categories of machine learning problems: supervised and unsupervised learning.
- A machine learning algorithm can be something as simple as an OLS regression.

2.3.2 Deep Learning

Deep learning algorithms can be regarded both as a sophisticated and mathematically complex evolution of machine learning algorithms. The field has been getting lots of attention lately and for good reason: Recent developments have led to results that were not thought to be possible before.

Deep learning describes algorithms that analyze data with a logic structure similar to how a human would draw conclusions. Note that this can happen both through supervised and unsupervised learning. To achieve this, deep learning applications use a layered structure of algorithms called an artificial neural network (ANN). The design of such an ANN is inspired by the biological neural network of the human brain, leading to a process of learning that's far more capable than that of standard machine learning models.

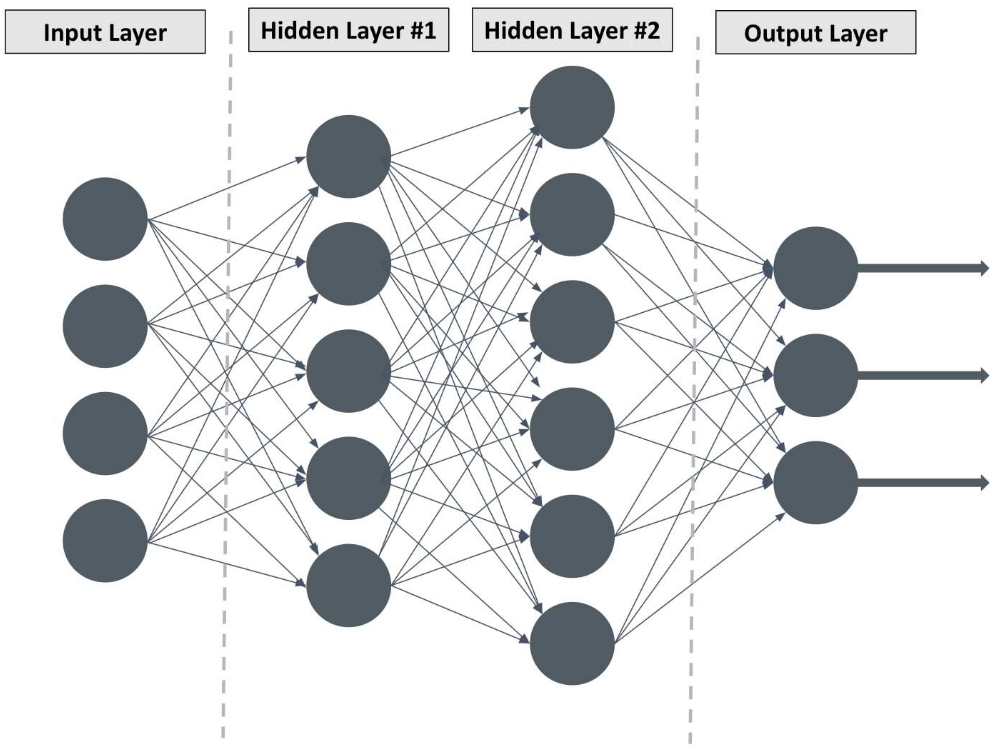


Fig. 2.6: A simple artificial neural network

Consider the example ANN in the image above. The leftmost layer is called the input layer, the rightmost layer of the output layer. The middle layers are called hidden layers because their values aren't observable in the training set. In simple terms, hidden layers are calculated values used by the network to do its "magic". The more hidden layers a network has between the input and output layer, the *deeper* it is. In general, any ANN with two or more hidden layers is referred to as a deep neural network.

Today, deep learning is used in many fields. In automated driving, for instance, deep learning is used to detect objects, such as STOP signs or pedestrians. The military uses deep learning to identify objects from satellites, e.g. to discover safe or unsafe zones for its troops. Of course, the consumer electronics industry is full of deep learning, too. Home assistance devices such as Amazon Alexa, for example, rely on deep learning algorithms to respond to your voice and know your preferences.

How about a more concrete example? Imagine the company Tesla using a deep learning algorithm for its cars to recognize STOP signs. In the first step, the ANN would identify the relevant properties of the STOP sign, also called *features*. Features may be specific structures in the inputted image, such as points, edges, or objects. While a software engineer would have to select the relevant features in a more traditional machine learning algorithm, the ANN is capable of *automatic feature engineering*. The first hidden layer might learn how to detect edges, the next how to differentiate colors, and the last learn how to detect more complex shapes catered specifically to the shape of the object we are trying to recognize. When fed with training data, the deep learning algorithms would eventually learn from their own errors whether the prediction was good, or whether it needs to adjust.

- Deep learning is a specialized subset of machine learning.
- Deep learning relies on a layered structure of algorithms called an artificial neural network.
- Deep learning has huge data needs but requires little human intervention to function properly.
- Transfer learning is a cure for the needs of large training datasets.

2.4 Deep Learning Algorithms

Types of Deep Learning Algorithms

1. Convolutional Neural Networks (CNNs)
2. Recurrent Neural Networks (RNNs)
3. Generative Adversarial Networks (GANs)
4. Self Organizing Maps (SOMs)
5. Deep Belief Networks (DBNs)

Deep learning algorithms work with almost any kind of data and require large amounts of computing power and information to solve complicated issues.

2.4.1 Convolutional Neural Networks (CNNs)

CNN's, also known as ConvNets, consist of multiple layers and are mainly used for image processing and object detection. Yann LeCun developed the first CNN in 1988 when it was called LeNet. It was used for recognizing characters like ZIP codes and digits.

CNN's are widely used to identify satellite images, process medical images, forecast time series, and detect anomalies.

How Do CNNs Work?

CNN's have multiple layers that process and extract features from data:

Convolution Layer

- CNN has a convolution layer that has several filters to perform the convolution operation.

Rectified Linear Unit (ReLU)

- CNN's have a ReLU layer to perform operations on elements. The output is a rectified feature map.

Pooling Layer

- The rectified feature map next feeds into a pooling layer. Pooling is a down-sampling operation that reduces the dimensions of the feature map.
- The pooling layer then converts the resulting two-dimensional arrays from the pooled feature map into a single, long, continuous, linear vector by flattening it.

Fully Connected Layer

- A fully connected layer forms when the flattened matrix from the pooling layer is fed as an input, which classifies and identifies the images.

Below is an example of an image processed via CNN.

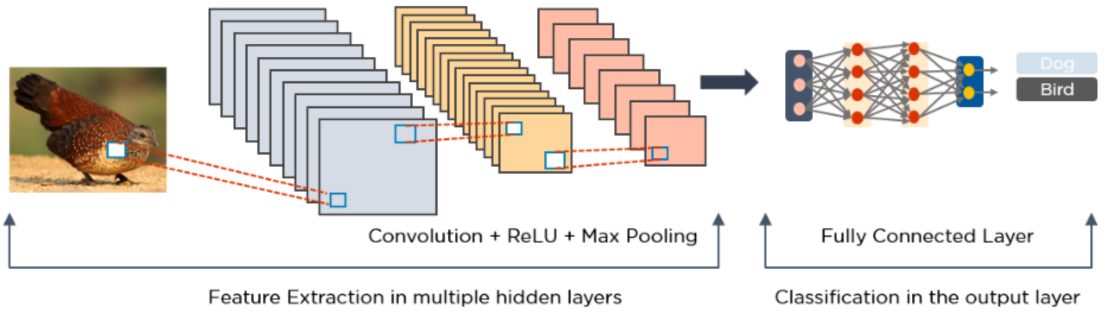

Dog
Bird
Convolution + ReLU + Max Pooling
Fully Connected Layer
Feature Extraction in multiple hidden layers
Classification in the output layer

Fig. 2.7: Convolutional Neural Networks

2.4.2 Recurrent Neural Networks (RNNs)

RNNs have connections that form directed cycles, which allow the outputs from the LSTM to be fed as inputs to the current phase.

The output from the LSTM becomes an input to the current phase and can memorize previous inputs due to its internal memory. RNNs are commonly used for image captioning, time-series analysis, natural-language processing, handwriting recognition, and machine translation.

An unfolded RNN looks like this:

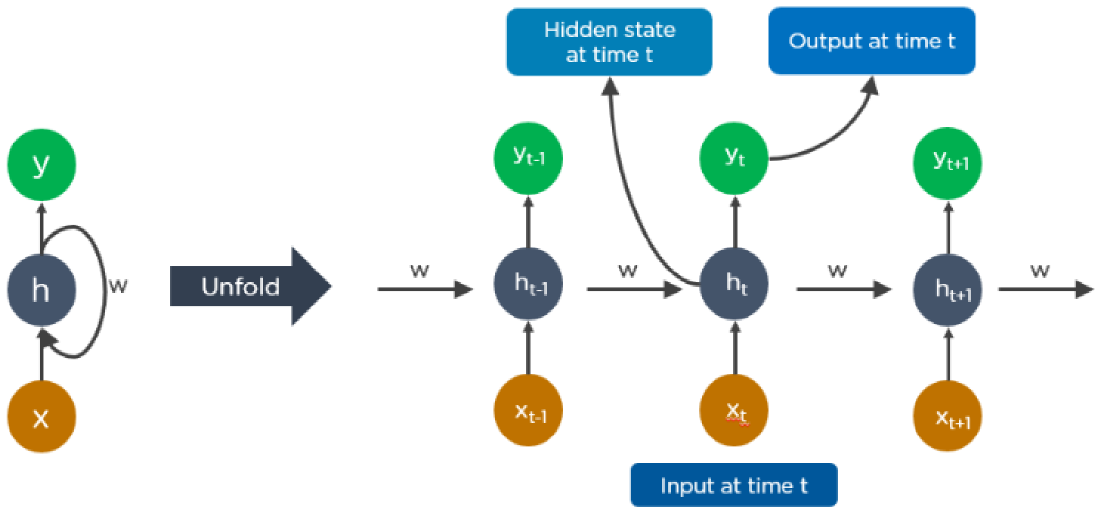


Fig. 2.8: Recurrent Neural Networks

How Do RNNs work?

- The output at time t-1 feeds into the input at time t.
- Similarly, the output at time t feeds into the input at time t+1.
- RNNs can process inputs of any length.
- The computation accounts for historical information, and the model size does not increase with the input size.

2.4.3 Generative Adversarial Networks (GANs)

GANs are generative deep learning algorithms that create new data instances that resemble the training data. GANs have two components: a generator, which learns to generate fake data, and a discriminator, which learns from that false information.

The usage of GANs has increased over a period of time. They can be used to improve astronomical images and simulate gravitational lensing for dark-matter research. Video game developers use GANs to upscale low-resolution, 2D textures in old video games by recreating them in 4K or higher resolutions via image training.

GANs help generates realistic images and cartoon characters, create photographs of human faces, and render 3D objects.

How Do GANs work?

- The discriminator learns to distinguish between the generator’s fake data and the real sample data.
- During the initial training, the generator produces fake data, and the discriminator quickly learns to tell that it's false.

- The GAN sends the results to the generator and the discriminator to update the model.

Below is a diagram of how GANs operate:

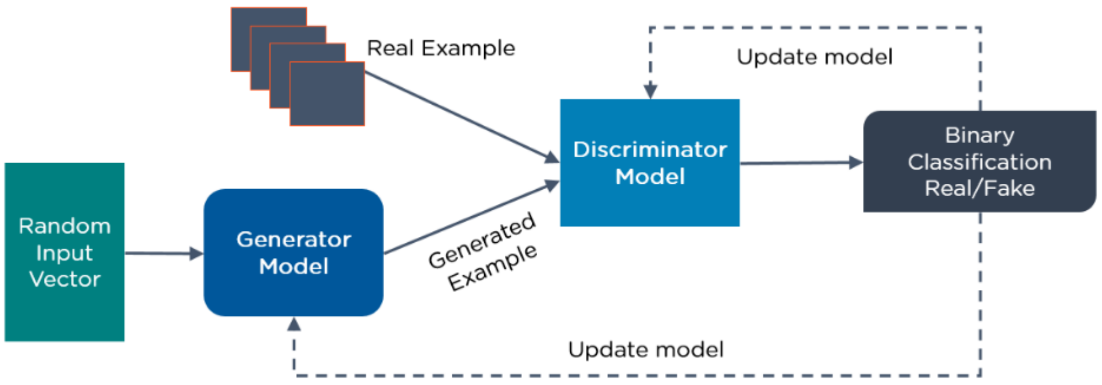


Fig. 2.9: Generative Adversarial Networks

Master deep learning concepts and the TensorFlow open-source framework with the Deep Learning Training Course. Get skilled today!

2.4.4 Self Organizing Maps (SOMs)

Professor Teuvo Kohonen invented SOMs, which enable data visualization to reduce the dimensions of data through self-organizing artificial neural networks.

Data visualization attempts to solve the problem that humans cannot easily visualize high-dimensional data. SOMs are created to help users understand this high-dimensional information.

How Do SOMs Work?

- SOMs initialize weights for each node and choose a vector at random from the training data.
- SOMs examine every node to find which weights are the most likely input vector. The winning node is called the Best Matching Unit (BMU).
- SOMs discover the BMU's neighborhood, and the amount of neighbors lessens over time.
- SOMs award a winning weight to the sample vector. The closer a node is to a BMU, the more its weight changes..
- The further the neighbor is from the BMU, the less it learns. SOMs repeat step two for N iterations.

Below, see a diagram of an input vector of different colors. This data feeds to a SOM, which then converts the data into 2D RGB values. Finally, it separates and categorizes the different colors.

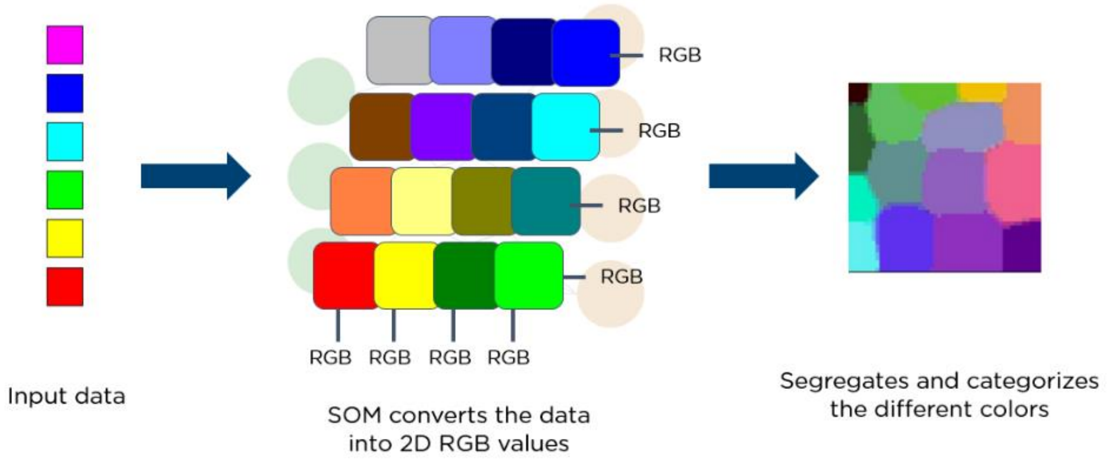

RGB
RGB
RGB
RGB
RGB RGB RGB RGB
Input data
SOM converts the data into 2D RGB values
Segregates and categorizes the different colors

Fig. 2.10: Self Organizing Maps

2.4.5 Deep Belief Networks (DBNs)

DBNs are generative models that consist of multiple layers of stochastic, latent variables. The latent variables have binary values and are often called hidden units.

DBNs are a stack of Boltzmann Machines with connections between the layers, and each RBM layer communicates with both the previous and subsequent layers. DBNs are used for image-recognition, video-recognition, and motion-capture data.

How Do DBNs Work?

- Greedy learning algorithms train DBNs. The greedy learning algorithm uses a layer-by-layer approach for learning the top-down, generative weights.
- DBNs run the steps of Gibbs sampling on the top two hidden layers. This stage draws a sample from the RBM defined by the top two hidden layers.
- DBNs draw a sample from the visible units using a single pass of ancestral sampling through the rest of the model.
- DBNs learn that the values of the latent variables in every layer can be inferred by a single, bottom-up pass.

Below is an example of DBN architecture:

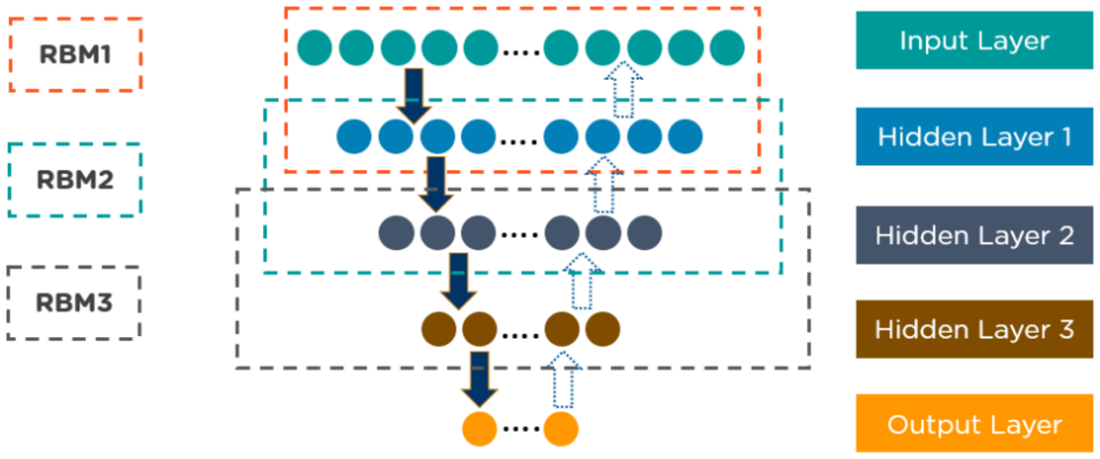


Fig. 2.11: Deep Belief Networks

Build deep learning models in TensorFlow and learn the TensorFlow open-source framework with the Deep Learning Course (with Keras &TensorFlow). Enroll now!

Chapter III

Plant Wheat Detection Using AI

3.1 Introduction

Figure 3.1 illustrates the block diagram of training the wheat plant convolution neural network data set. The block diagram starts with image acquisition from the camera unit and perform preprocessing block to change level of lights. Entering multiple images that contain plant wheat to build the wheat data set block. Then building the CNN network for recognizing the wheat plant. Then training the model using the data set to finalize the wheat CNN model.

Figure 3.2, illustrates the block diagram of the detection system for the wheat plant. The first method is to upload the training image that have been performed in figure 1. Then initialization the wheat plant convolution neural network to classification for the probabilities for the image from data acquisition. Then counting detection for the wheat plant. After that the bounding boxes will be used to draw a bounding box for the detected wheat plant, and export the output image to the screen.

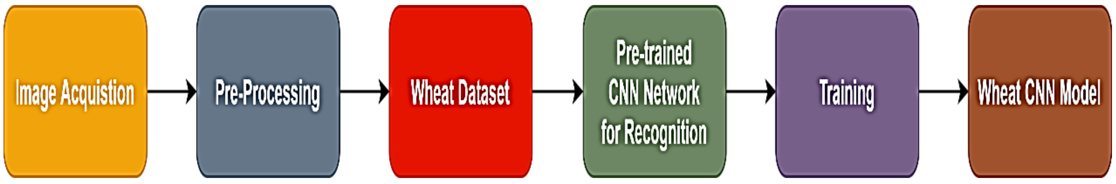


Fig. 3.1: Training block diagram

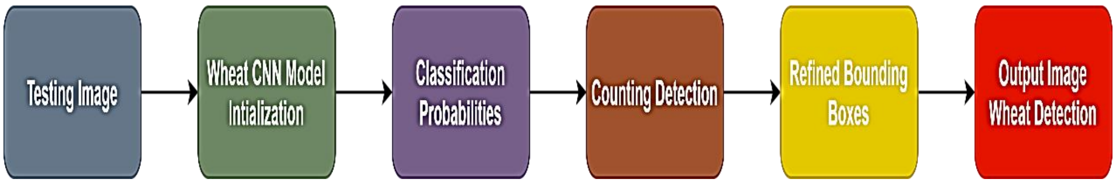

Testing Image
Wheat CNN Model Intialization
Classification Probabilities
Counting Detection
Refined Bounding Boxes
Output Image Wheat Detection

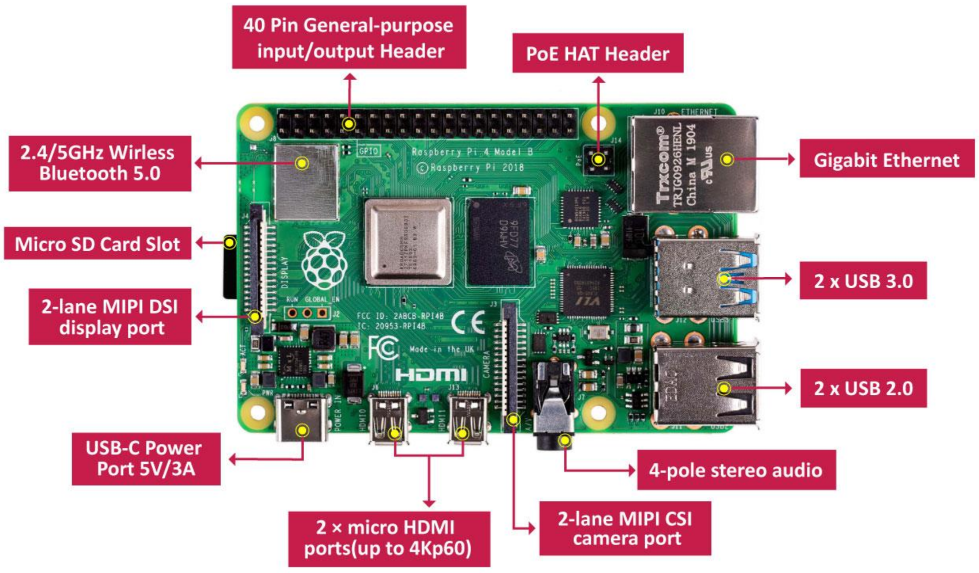
Fig. 3.2: Detection Blok diagram

3.2 Hardware Description

The project will need a microcontroller development board that capable to implement image processing. This project will implement the usage of raspberry pi 4 model B and Raspberry pi camera module v2.

3.2.1 Raspberry pi 4

Raspberry Pi as shown in Figure 3.3 is a card sized single-board computer developed in the UK by the raspberry pi foundation with the intention of promoting the teaching of basic computer science in schools. The raspberry pi manufactured in three board configuration through licensed manufacturing agreements with network element14 (Premier Farnell),RS Component and Egoman. These companies sell the Raspberry Pi online .Egoman produce a version for distribution soley in China and Taiwan, which can be distinguished from other Pis by their red colouring and lack of FCC/CE marks. The hardware is the same across all manufactures .



40 Pin General-purpose input/output Header
PoE HAT Header
2.4/5GHz Wirless Bluetooth 5.0
Gigabit Ethernet
Micro SD Card Slot
2 x USB 3.0
2-lane MIPI DSI display port
2 x USB 2.0
USB-C Power Port 5V/3A
4-pole stereo audio
2 × micro HDMI ports(up to 4Kp60)
2-lane MIPI CSI camera port

Fig. 3.3: Raspberry Pi 4 Model B Microcontroller development board

The Raspberry Pi 4 Model B is the third generation Raspberry Pi. This powerful credit-card sized single board computer can be used for many applications and supersedes the original Raspberry Pi Model B+ and Raspberry Pi 2 Model B. Whilst maintaining the popular board format the Raspberry Pi 4 Model B brings you a more powerful processer, 20x faster than the first generation Raspberry Pi. Additionally it adds wireless LAN & Bluetooth connectivity making it the ideal solution for powerful connected designs as shown in Figure 3.3.

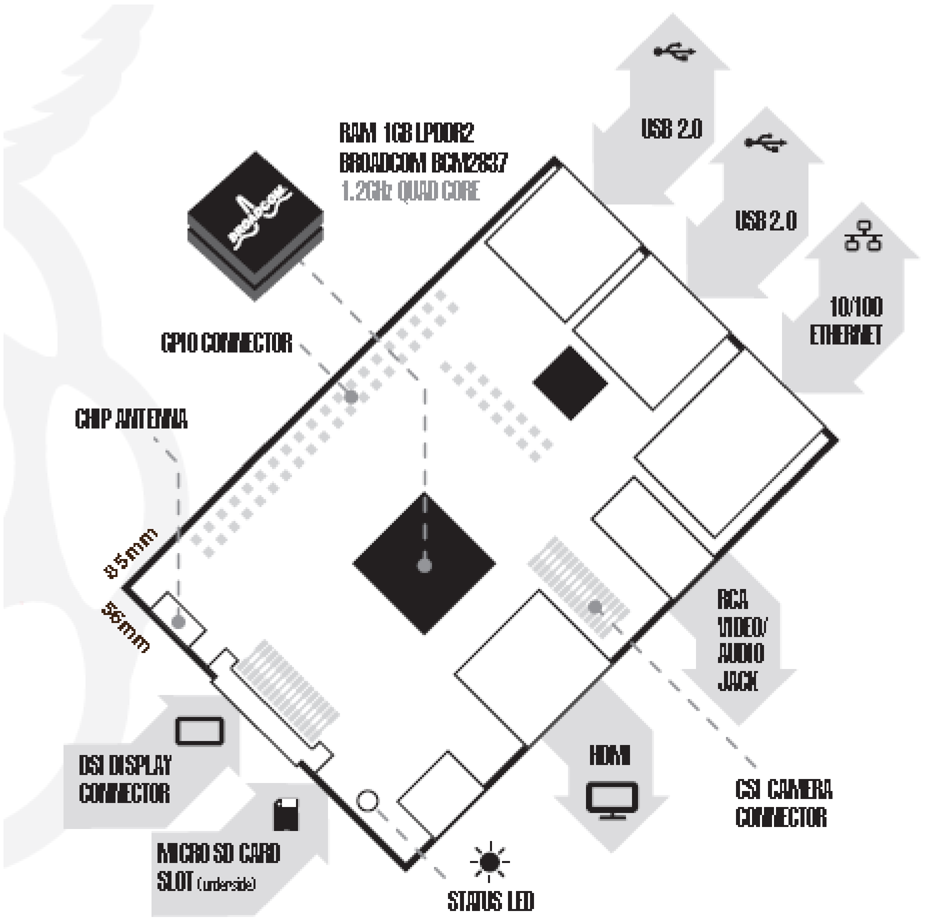
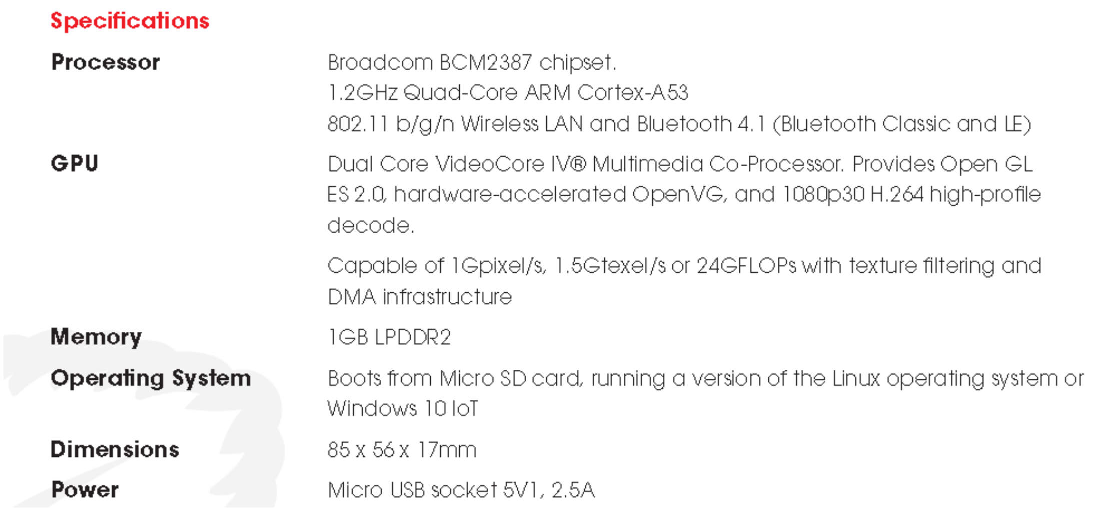


Fig. 3.4: Raspberry pi 3 terminals illustration



Specifications

Processor	Broadcom BCM2387 chipset. 1.2GHz Quad-Core ARM Cortex-A53 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)
GPU	Dual Core VideoCore IV® Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode. Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
Memory	1GB LPDDR2
Operating System	Boots from Micro SD card, running a version of the Linux operating system or Windows 10 IoT
Dimensions	85 x 56 x 17mm
Power	Micro USB socket 5V1, 2.5A

The physical specification in Figure 3.4 of the raspberry pi 4 model b can be presented

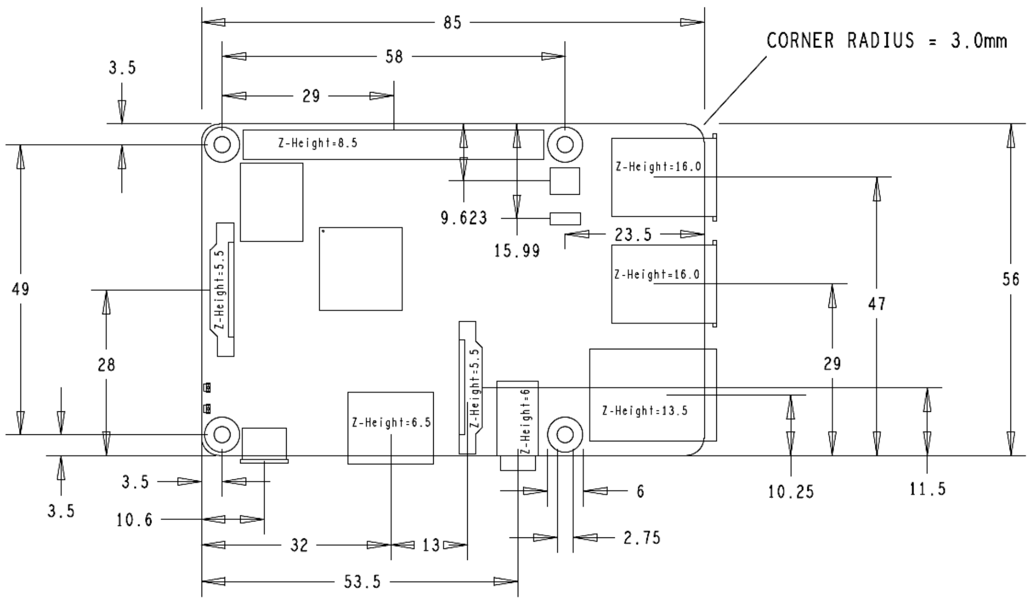


Fig. 3.5: Raspberry pi Layout

In 2014 Raspberry pi Foundation launched the compute module ,which packages a Raspberry Pi Model B into a SODIMM 200-pin module, to encourage its use in embedded systems .The Raspberry Pi is based on the Broadcom BCM2835 system on chip(SoC),which includes an ARM1176JZF-S 700 MHz processor, videocore IV GPU and was originally shipped with 256 MB of RAM ,later upgraded (Model B & Model B+) to 512 MB. The system has secure SD or MicroSD socket for boot media and persistent storage .The foundation provide Debian and Arch Linux ARM distribution for download. Tools are available for Python as the main programming language, with support for BBC BASIC(via the RISC OS image or the Brandy Basic clone for Linux),C,C++,Java, Perl and Ruby.

I it provides hardware specifications as follows

- Broadcom BCM2837 64bit ARMv7 Quad Core Processor powered Single Board Computer running at
- 1.2GHz
- 1GB RAM
- BCM43143 WiFi on board
- Bluetooth Low Energy (BLE) on board
- 40pin extended GPIO
- 4 x USB 2 ports
- 4 pole Stereo output and Composite video port
- Full size HDMI

- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source (now supports up to 2.4 Amps)
- Expected to have the same form factor has the Pi 2 Model B, however the LEDs will change position

GPIO Pinout

The Raspberry Pi offers up as shown in Figure 3.6 its GPIO over a standard male header on the board. Over the years the header has expanded from 26 pins to 40 pins while maintaining the original pinout.

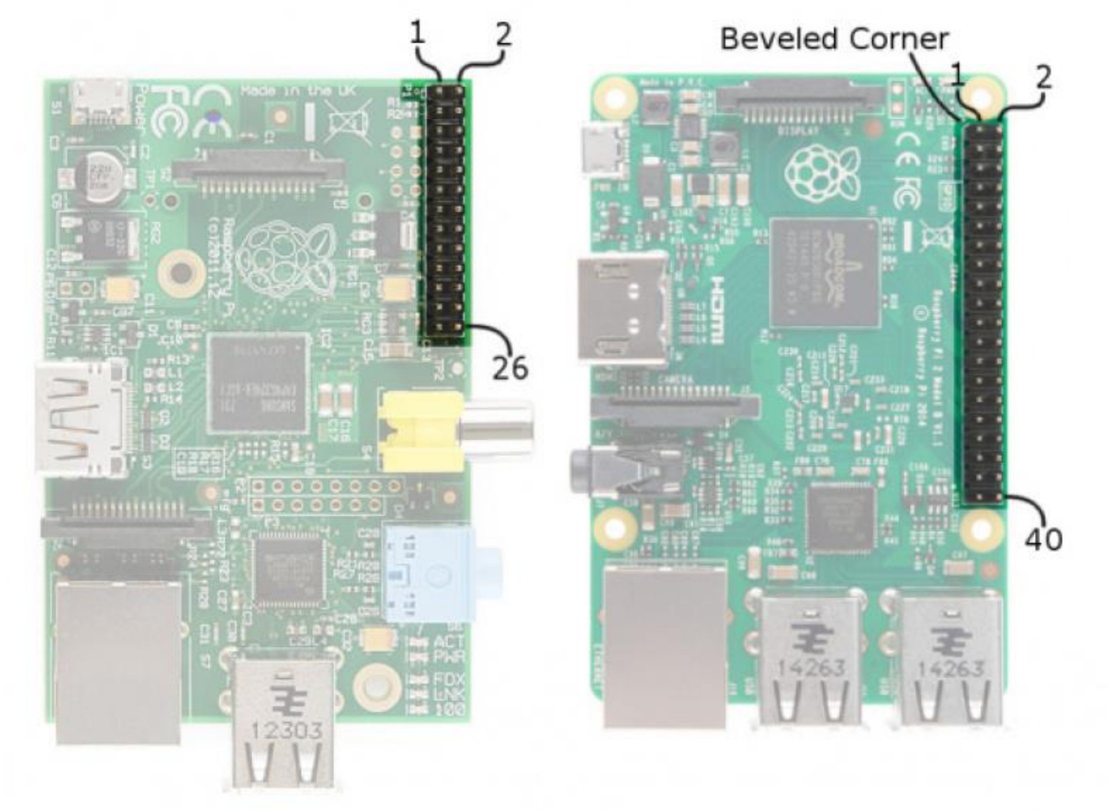
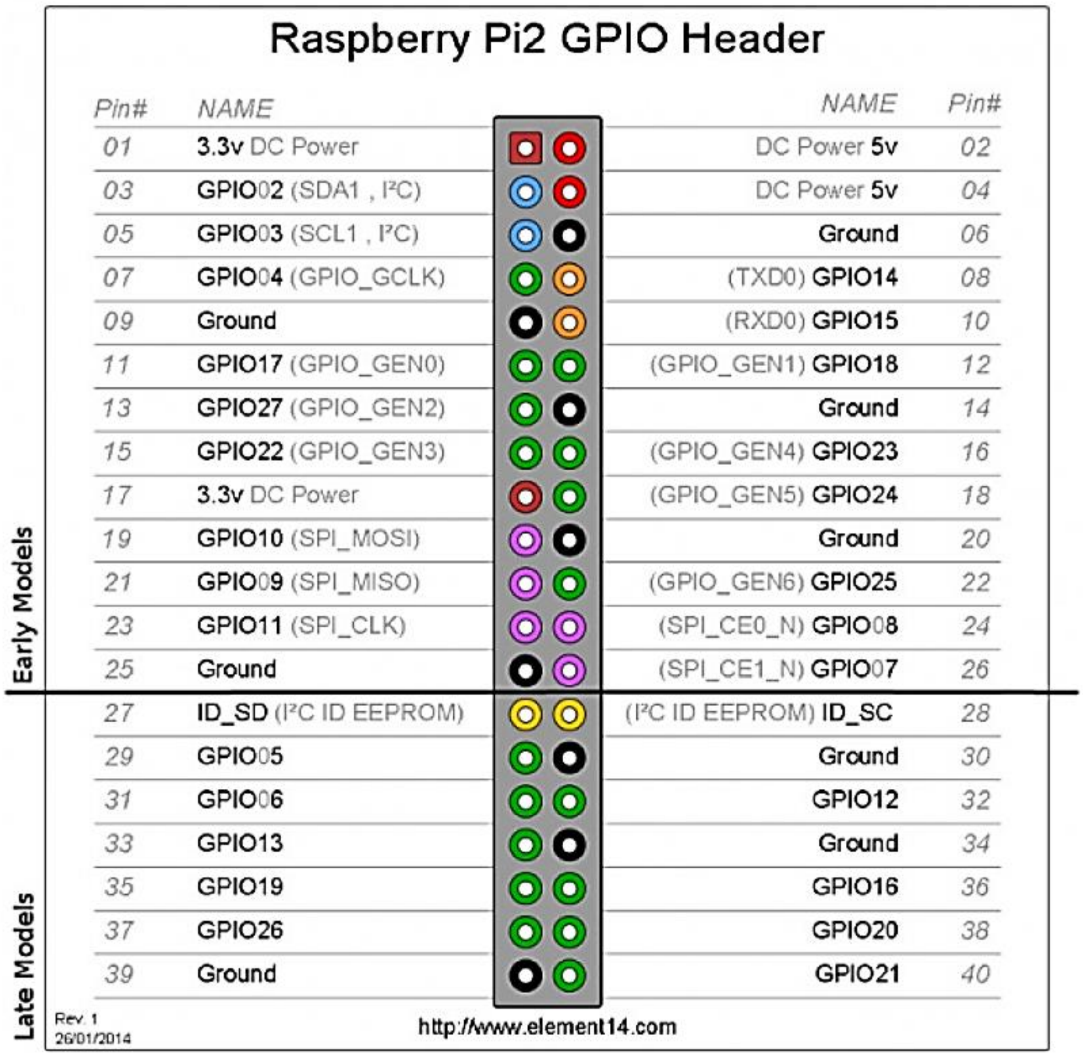


Fig. 3.6: Header configuration for Pi computers

There are (at least) two, different numbering schemes you may encounter when referencing Pi pin numbers: (1) Broadcom chip-specific pin numbers and (2) P1 physical pin numbers. You’re usually free to use either number-system, but many programs require that you declare which scheme you’re using at the very beginning of your program.



Raspberry Pi2 GPIO Header

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

Early Models

Late Models

Rev. 1
26/01/2014

http://www.element14.com

Fig. 3.7: Element14 pin description, annotated

3.3.1.1 Processor

The Broadcom BCM2835 SoC used in the first generation Raspberry Pi is somewhat equivalent to the chip used in first modern generation smartphones[clarification needed] (its CPU is an older ARMv6 architecture),[25] which includes a 700 MHz ARM1176JZF-S processor, VideoCore IV graphics processing unit (GPU),[26] and RAM. It has a level 1 (L1) cache of 16 KB and a level 2 (L2) cache of 128 KB. The level 2 cache is used primarily by the GPU. The SoC is stacked underneath the RAM chip, so only its edge is visible.

The Raspberry Pi 3+ uses a Broadcom BCM2837B0 SoC with a 1.4 GHz 64-bit quad-core ARM Cortex-A53 processor, with 512 KB shared L2 cache.[29]

3.3.1.2 Performance

The Raspberry Pi 3, with a quad-core ARM Cortex-A53 processor, is described as 10 times the performance of a Raspberry Pi 1. This was suggested to be highly dependent upon task threading and instruction set use. Benchmarks showed the Raspberry Pi 3 to be approximately 80% faster than the Raspberry Pi 2 in parallelised tasks.

Raspberry Pi 2 V1.1 included a quad-core Cortex-A7 CPU running at 900 MHz and 1 GB RAM. It was described as 4–6 times more powerful than its predecessor. The GPU was identical to the original. In parallelised benchmarks, the Raspberry Pi 2 V1.1 could be up to 14 times faster than a Raspberry Pi 1 Model B+.

While operating at 700 MHz by default, the first generation Raspberry Pi provided a real-world performance roughly equivalent to 0.041 GFLOPS. On the CPU level the performance is similar to a 300 MHz Pentium II of 1997–99. The GPU provides 1 Gpixel/s or 1.5 Gtexel/s of graphics processing or 24 GFLOPS of general purpose computing performance. The graphical capabilities of the Raspberry Pi are roughly equivalent to the performance of the Xbox of 2001.

The LINPACK single node compute benchmark results in a mean single precision performance of 0.065 GFLOPS and a mean double precision performance of 0.041 GFLOPS for one Raspberry Pi Model-B board. A cluster of 64 Raspberry Pi Model B computers, labelled "Iridis-pi", achieved a LINPACK HPL suite result of 1.14 GFLOPS (n=10240) at 216 watts for c. US$4000.

3.3.1.3 Overclocking

Most Raspberry Pi chips could be overclocked to 800 MHz, and some to 1000 MHz. There are reports the Raspberry Pi 2 can be similarly overclocked, in extreme cases, even to 1500 MHz (discarding all safety features and over-voltage limitations). In the Raspbian Linux distro the overclocking options on boot can be done by a software command running "sudo raspi-config" without voiding the warranty.

In those cases the Pi automatically shuts the overclocking down if the chip reaches 85 °C (185 °F), but it is possible to override automatic over-voltage and overclocking settings (voiding the warranty); an appropriately sized heat sink is needed to protect the chip from serious overheating.

Newer versions of the firmware contain the option to choose between five overclock ("turbo") presets that when used, attempt to maximise the performance of the SoC without impairing the lifetime of the board. This is done by monitoring the core temperature of the chip, the CPU load, and dynamically adjusting clock speeds and the core voltage. When the demand is low on the CPU or it is running too hot the performance is throttled, but if the CPU has much to do and the chip's temperature is acceptable, performance is temporarily increased with clock speeds of up to 1 GHz depending on the individual board and on which of the turbo settings is used.

3.3.1.4 RAM

On the older beta Model B boards, 128 MB was allocated by default to the GPU, leaving 128 MB for the CPU. On the first 256 MB release Model B (and Model A), three different splits were possible. The default split was 192 MB (RAM for CPU), which should be sufficient for standalone 1080p video decoding, or for simple 3D, but probably not for both together. 224 MB was for Linux only, with only a 1080p framebuffer, and was likely to fail for any video or 3D. 128 MB was for heavy 3D, possibly also with video decoding (e.g. XBMC). Comparatively the Nokia 701 uses 128 MB for the Broadcom VideoCore IV.

For the later Model B with 512 MB RAM initially there were new standard memory split files released(arm256_start.elf, arm384_start.elf, arm496_start.elf) for 256 MB, 384 MB and 496 MB CPU RAM (and 256 MB, 128 MB and 16 MB video RAM). But a week or so later the RPF released a new version of start.elf that could read a new entry in config.txt (gpu_mem=xx) and could dynamically assign an amount of RAM (from 16 to 256 MB in 8 MB steps) to the GPU, so the older method of memory splits became obsolete, and a single start.elf worked the same for 256 and 512 MB Raspberry Pis.

The Raspberry Pi 2 and the Raspberry Pi 3 have 1 GB of RAM. The Raspberry Pi Zero and Zero W have 512 MB of RAM.

3.3.1.5 Networking

The Model A, A+ and Pi Zero have no Ethernet circuitry and are commonly connected to a network using an external user-supplied USB Ethernet or Wi-Fi adapter. On the Model B and B+ the Ethernet port is provided by a built-in USB

Ethernet adapter using the SMSC LAN9514 chip. The Raspberry Pi 3 and Pi Zero W (wireless) are equipped with 2.4 GHz WiFi 802.11n (150 Mbit/s) and Bluetooth 4.1 (24 Mbit/s) based on Broadcom BCM43438 FullMAC chip with no official support for Monitor mode but implemented through unofficial firmware patching and the Pi 3 also has a 10/100 Ethernet port. The Raspberry Pi 3B+ features dual-band IEEE 802.11b/g/n/ac WiFi, Bluetooth 4.2, and Gigabit Ethernet (limited to approximately 300 Mbit/s by the USB 2.0 bus between it and the SoC).

3.3.1.6 Peripherals

The Raspberry Pi may be operated with any generic USB computer keyboard and mouse. It may also be used with USB storage, USB to MIDI converters, and virtually any other device/component with USB capabilities.

Other peripherals can be attached through the various pins and connectors on the surface of the Raspberry Pi.

3.3.1.7 Video

The video controller can generate standard modern TV resolutions, such as HD and Full HD, and higher or lower monitor resolutions as well as older NTSC or PAL standard CRT TV resolutions. As shipped (i.e., without custom overclocking) it can support these resolutions: 640×350 EGA; 640×480 VGA; 800×600 SVGA; 1024×768 XGA; 1280×720 720p HDTV; 1280×768 WXGA variant; 1280×800 WXGA variant; 1280×1024 SXGA; 1366×768 WXGA variant; 1400×1050 SXGA+; 1600×1200 UXGA; 1680×1050 WXGA+; 1920×1080 1080p HDTV; 1920×1200 WUXGA.

Higher resolutions, such as, up to 2048×1152, may work or even 3840×2160 at 15 Hz (too low a frame rate for convincing video). Note also that allowing the highest resolutions does not imply that the GPU can decode video formats at those; in fact, the Pis are known to not work reliably for H.265 (at those high resolutions), commonly used for very high resolutions (most formats, commonly used, up to Full HD, do work).

Although the Raspberry Pi 3 does not have H.265 decoding hardware, the CPU is more powerful than its predecessors, potentially fast enough to allow the decoding of H.265-encoded videos in software. The GPU in the Raspberry Pi 3 runs at higher clock frequencies of 300 MHz or 400 MHz, compared to previous versions which ran at 250 MHz.

The Raspberry Pis can also generate 576i and 480i composite video signals, as used on old-style (CRT) TV screens and less-expensive monitors through standard connectors – either RCA or 3.5 mm phono connector depending on models. The television signal standards supported are PAL-BGHID, PAL-M, PAL-N, NTSC and NTSC-J.

3.3.1.8 Real-time Clock

None of the current Raspberry Pi models have a built-in real-time clock, so they are unable to keep track of the time of day independently. As a workaround, a program running on the Pi can retrieve the time from a network time server or from user input at boot time, thus knowing the time while powered on. To provide consistency of time for the file system, the Pi does automatically save the time it has on shutdown, and re-installs that time at boot.

A real-time hardware clock with battery backup, such as the DS1307, may be added (often via the I²C interface).

3.2.2 CAMERA MODULE V2

The Raspberry Pi Camera Module v2 replaced the original Camera Module in April 2016.

The Raspberry Pi Camera Module v2 replaced the original Camera Module in April 2016. The v2 Camera Module has a Sony IMX219 8-megapixel sensor (compared to the 5-megapixel OmniVision OV5647 sensor of the original camera).

The Camera Module as shown in Figure 3.8 and 3.9 can be used to take high-definition video, as well as stills photographs. It’s easy to use for beginners, but has plenty to offer advanced users if you’re looking to expand your knowledge. There are lots of examples online of people using it for time-lapse, slow-motion, and other video cleverness. You can also use the libraries we bundle with the camera to create effects.

You can read all the gory details about IMX219 and the Exmor R back-illuminated sensor architecture on Sony's website, but suffice to say this is more than just a resolution upgrade: it's a leap forward in image quality, colour fidelity, and low-light performance. It supports 1080p30, 720p60 and VGA90 video modes, as well as still capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi.

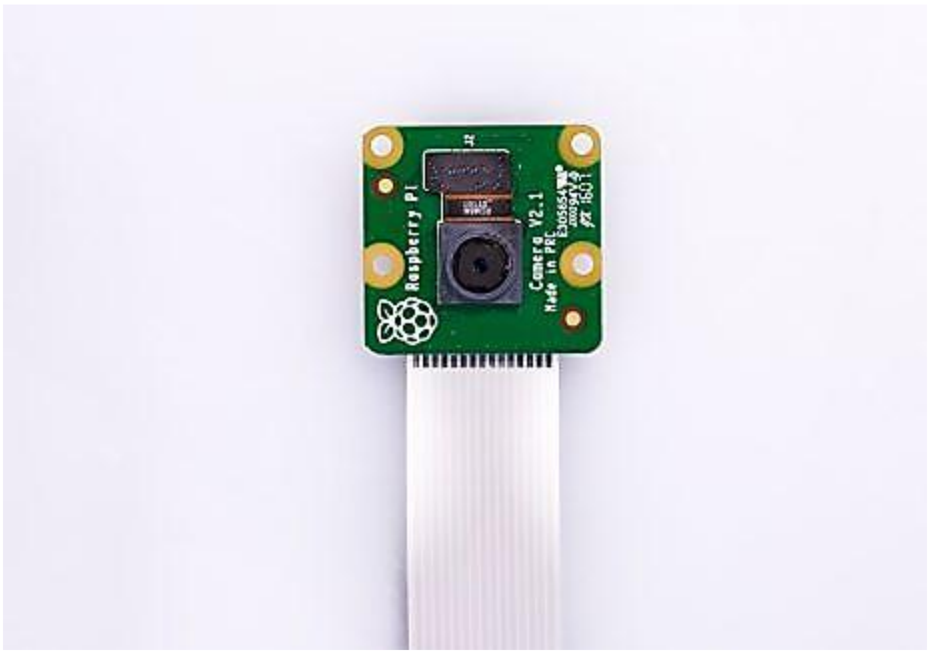


Fig. 3.8: Raspberry pi Camera Module V2 front

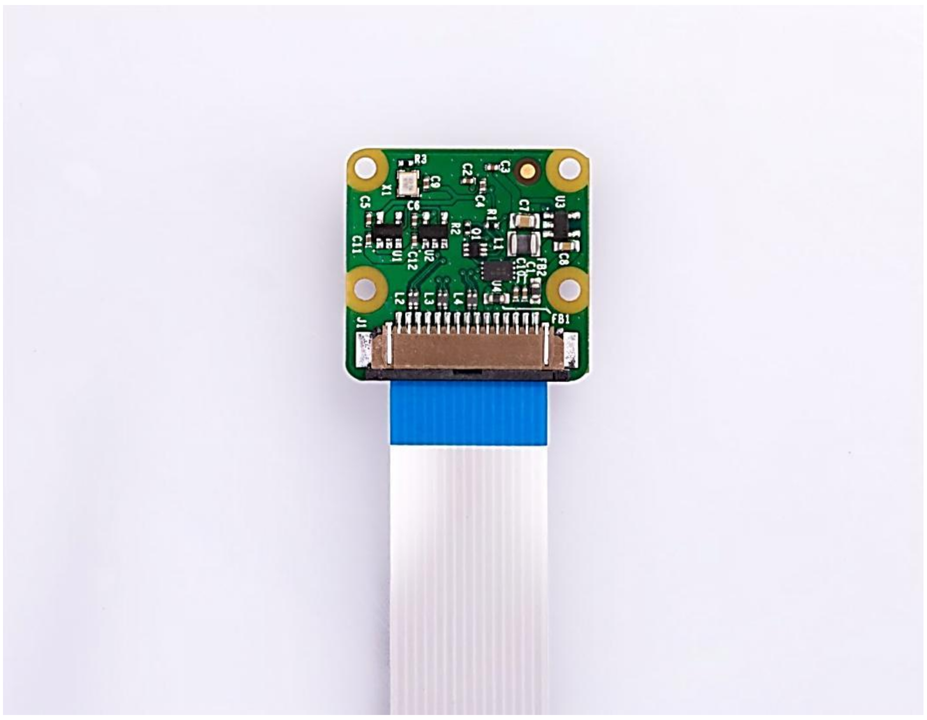


Fig. 3.9: Raspberry pi Camera Module V2 Rear

The camera works with all models of Raspberry Pi 1, 2, and 3. It can be accessed through the MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Picamera Python library. See the Getting Started with Picamera resource to learn how to use it.

The camera module is very popular in home security applications, and in wildlife camera traps.

3.3.2.1 PiEncoder

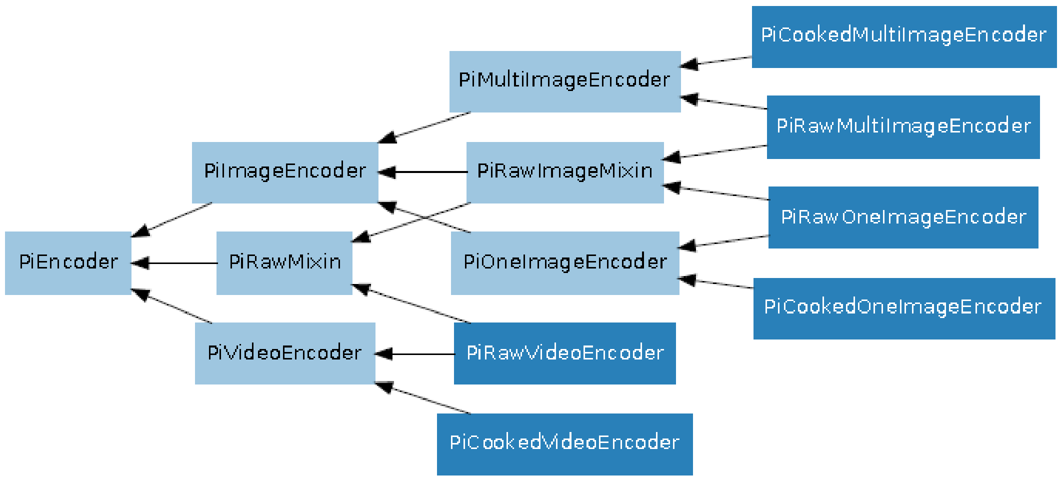


Fig. 3.10: The inheritance diagram for the following classes

The parent parameter specifies in Figure 3.10, the PiCamera instance that has constructed the encoder. The camera_port parameter provides the MMAL camera port that the encoder should enable for capture (this will be the still or video port of the camera component). The input_port parameter specifies the MMAL port that the encoder should connect to its input. Sometimes this will be the same as the camera port, but if other components are present in the pipeline (e.g. a splitter), it may be different.

3.3 Software

To implement the project we need to operate the Pi3 using Linux operating system, while communicating the board using the SSH. The programming will be implemented using Python C programing with the assistance with image processing tools OpenCV language.

3.3.1 Linux OS

Linux was originally developed for personal computers based on the Intel x86 architecture, but has since been ported to more platforms than any other operating system. Because of the dominance of the Linux kernel-based Android OS on smartphones, Linux has the largest installed base of all general-purpose operating systems. Linux is also the leading operating system on servers and other big iron systems such as mainframe computers, and the only OS used on TOP500 supercomputers (since November 2017, having before gradually eliminated all competitors). It is used by around 2.3% of desktop computers. The Chromebook, which runs the Linux kernel-based Chrome OS, dominates the US K–12 education market and represents nearly 20% of the sub-$300 notebook sales in the US. Linux also runs on embedded systems—devices whose operating system is typically built into the firmware and is highly tailored to the system. This includes TiVo and similar DVR devices, network routers, facility automation controls, televisions, video game consoles and smartwatches. Many smartphones and tablet computers run Android and other Linux derivatives.

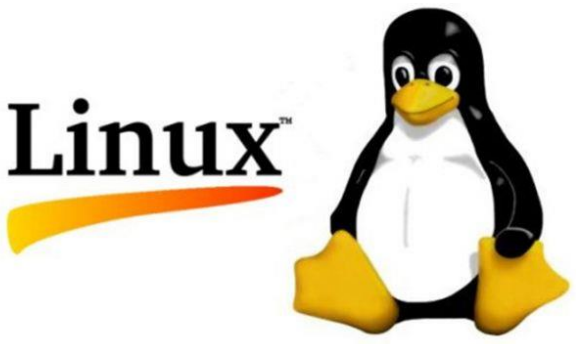


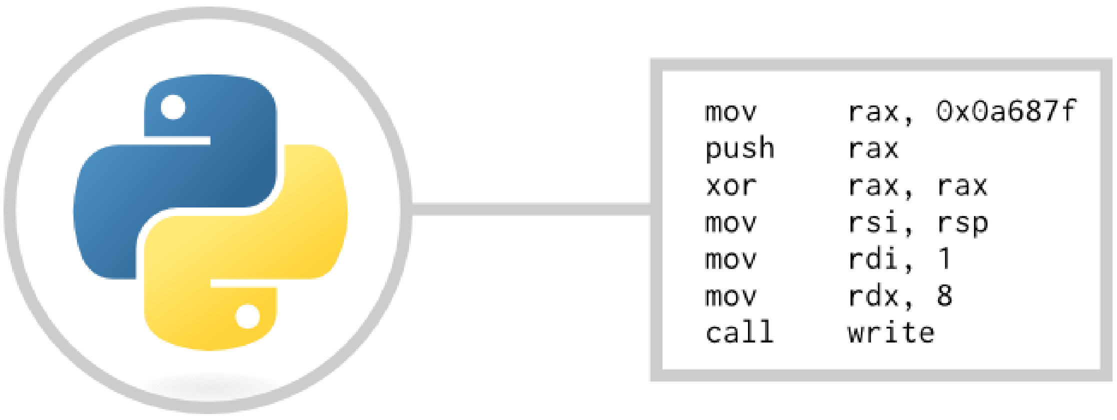
Fig. 3.11: Linux Operating system

The development of Linux is one of the most prominent examples of free and open-source software collaboration. The underlying source code may be used, modified and distributed—commercially or non-commercially—by anyone under the terms of its respective licenses, such as the GNU General Public License.

Some of the most popular and mainstream Linux distributions are Arch Linux, CentOS, Debian, Fedora, Gentoo Linux, Linux Mint, Mageia, openSUSE and Ubuntu, together with commercial distributions such as Red Hat Enterprise Linux and SUSE Linux Enterprise Server. Distributions include the Linux kernel, supporting utilities and libraries, many of which are provided by the GNU Project, and usually a large amount of application software to fulfil the distribution's intended use. Desktop Linux distributions include a windowing system, such as X11, Mir or a Wayland implementation, and an accompanying desktop environment such as GNOME or KDE Plasma; some distributions may also include a less resource-intensive desktop, such as LXDE or Xfce. Distributions intended to run on servers may omit all graphical environments from the standard install, and instead include other software to set up and operate a solution stack such as LAMP. Because Linux is freely redistributable, anyone may create a distribution for any intended use.

3.3.2 Python C

A significant limitation of CPython is the use of a global interpreter lock (GIL) on each CPython interpreter process, which effectively disables concurrent Python threads within one process. Concurrency can only be achieved with separate CPython interpreter processes managed by a multitasking operating system. This complicates communication between concurrent Python processes, though the multiprocessing module mitigates this somewhat. Much discussion took place on whether to remove the GIL from CPython. A set of "free threading" patches to CPython was submitted by Greg Stein, which effectively replaced GIL with fine-grained locking. However the patches were rejected due to the execution overhead they introduced into single-process code.



```
mov     rax, 0x0a687f
push    rax
xor     rax, rax
mov     rsi, rsp
mov     rdi, 1
mov     rdx, 8
call    write
```

Fig. 3.12: Python C Programming

3.3.3 Computer Vision Programs

3.3.3.1 OpenCV

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

OpenCV (Open Source Computer Vision Library) is an open-source computer vision library that contains many different functions for computer vision and machine learning. It was created by Intel and originally released in 2000. OpenCV has many different algorithms related to computer vision that can perform a variety of tasks including facial detection and recognition, object identification, monitoring moving objects, tracking camera movements, tracking eye movements, extracting 3D models of objects, creating an augmented reality overlay with a scenery, recognizing similar images in an image database, etc. OpenCV has interfaces for C++, Java Python, MATLAB, etc. and it supports various operating systems such as Windows, Android, Mac OS, Linux, etc.

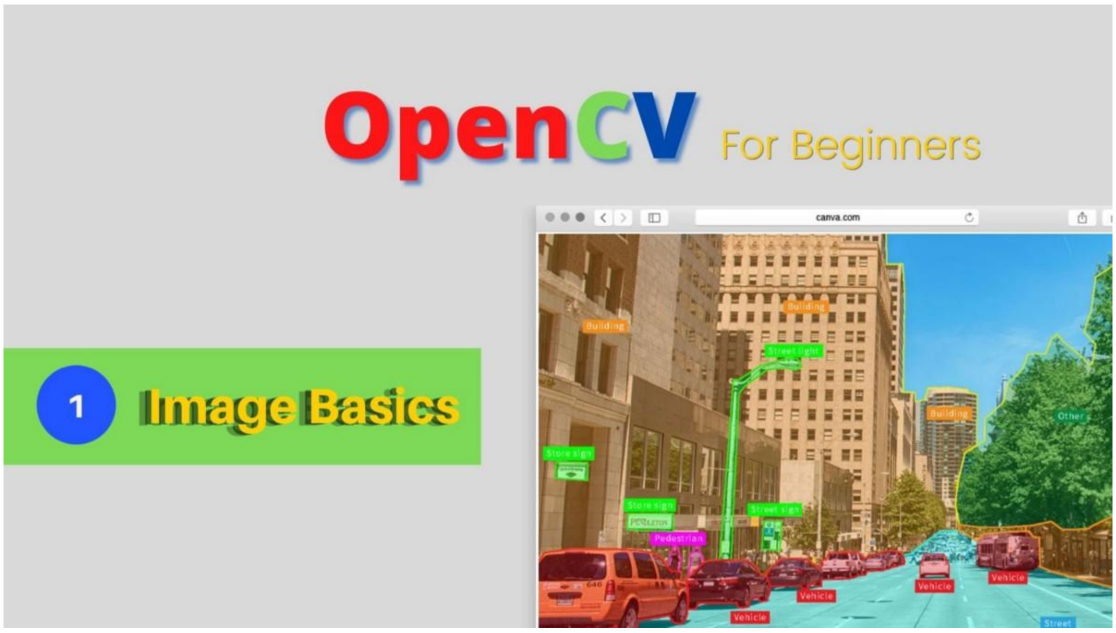


Fig. 3.13: OpenCV

Applications of Computer Vision

<u>Robotics Application</u>

- Localization − Determine robot location automatically
- Navigation
- Obstacles avoidance

- Assembly (peg-in-hole, welding, painting)
- Manipulation (e.g. PUMA robot manipulator)
- Human Robot Interaction (HRI) − Intelligent robotics to interact with and serve people

Medicine Application

- Classification and detection (e.g. lesion or cells classification and tumor detection)
- 2D/3D segmentation
- 3D human organ reconstruction (MRI or ultrasound)
- Vision-guided robotics surgery

Industrial Automation Application

- Industrial inspection (defect detection)
- Assembly
- Barcode and package label reading
- Object sorting
- Document understanding (e.g. OCR)

Security Application

- Biometrics (iris, finger print, face recognition)
- Surveillance − Detecting certain suspicious activities or behaviors

Transportation Application

- Autonomous vehicle
- Safety, e.g., driver vigilance monitoring

Features of OpenCV Library

Using OpenCV library, you can −

- Read and write images
- Capture and save videos
- Process images (filter, transform)
- Perform feature detection
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

3.3.3.2 SimpleCV

SimpleCV is an open-source computer vision framework that can be used for building various computer vision applications. SimpleCV is simple (as the name suggests!) and you can use various advanced computer vision libraries with it such as OpenCV without learning all the CV concepts in-depth such as file formats, buffer management, color spaces, eigenvalues, bit depths, matrix storage, bitmap storage, etc. SimpleCV allies you to experiment in computer vision using the images or video streams from webcams, FireWire, mobile phones, Kinects, etc. It is the best framework if you need to perform some quick prototyping. You can use SimpleCV with Mac, Windows, and Ubuntu Linux operating systems.

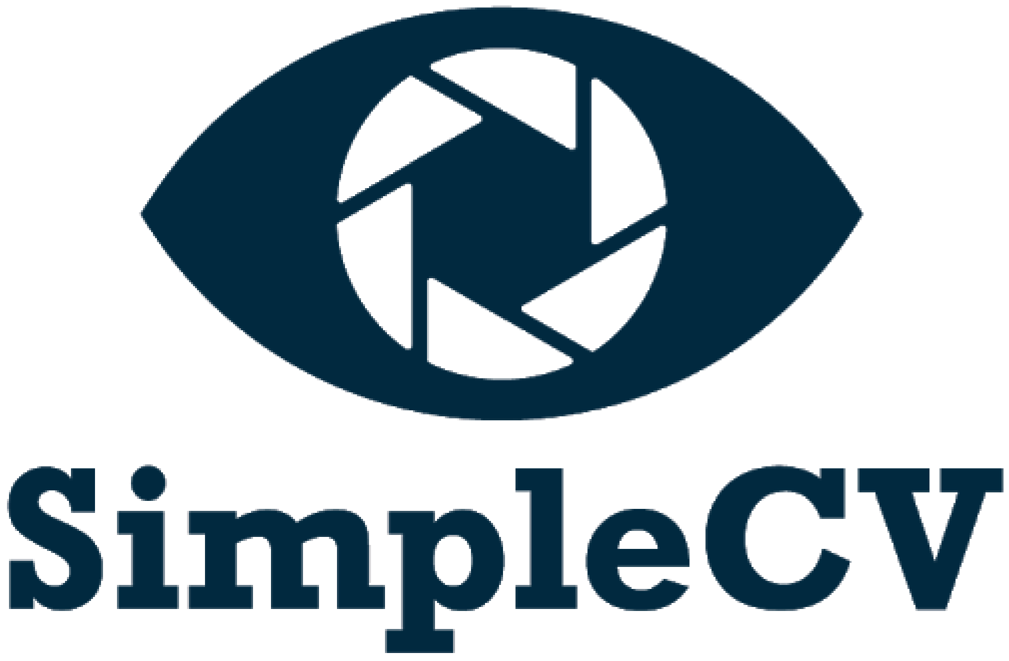
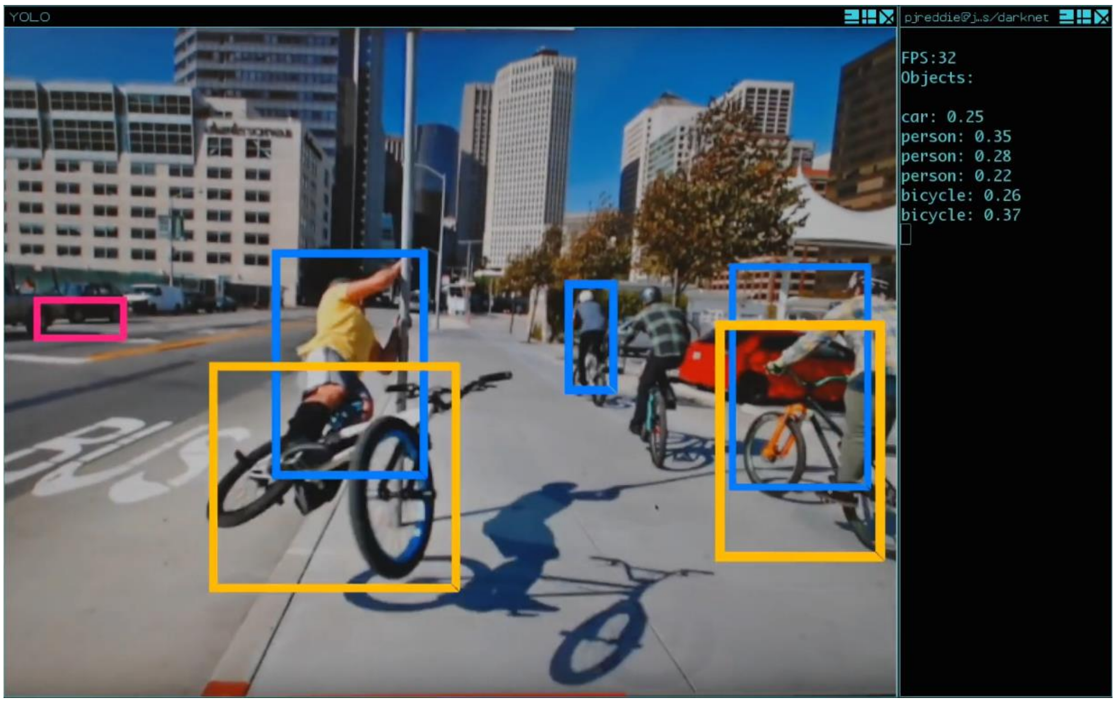


Fig. 3.14: Simple CV

3.3.3.3 YOLO

YOLO or You only look once! is a latest and cutting edge real-time object detection system. It was created by Joseph Redmon and Ali Farhadi from the University of Washington and it is extremely fast and accurate as compared to the other object detectors. The YOLO algorithm is so fast as compared to other object detection algorithms because it applies a neural network to the full image in order to classify the objects. The neural network then partitions the image into regions and predicts probabilities for each region. On the other hand, the rest of the commonly used object detection algorithms apply the neural network to an image at many different locations and scales. So YOLO is fast as It looks at the whole image so its predictions are informed by a holistic context of the image.

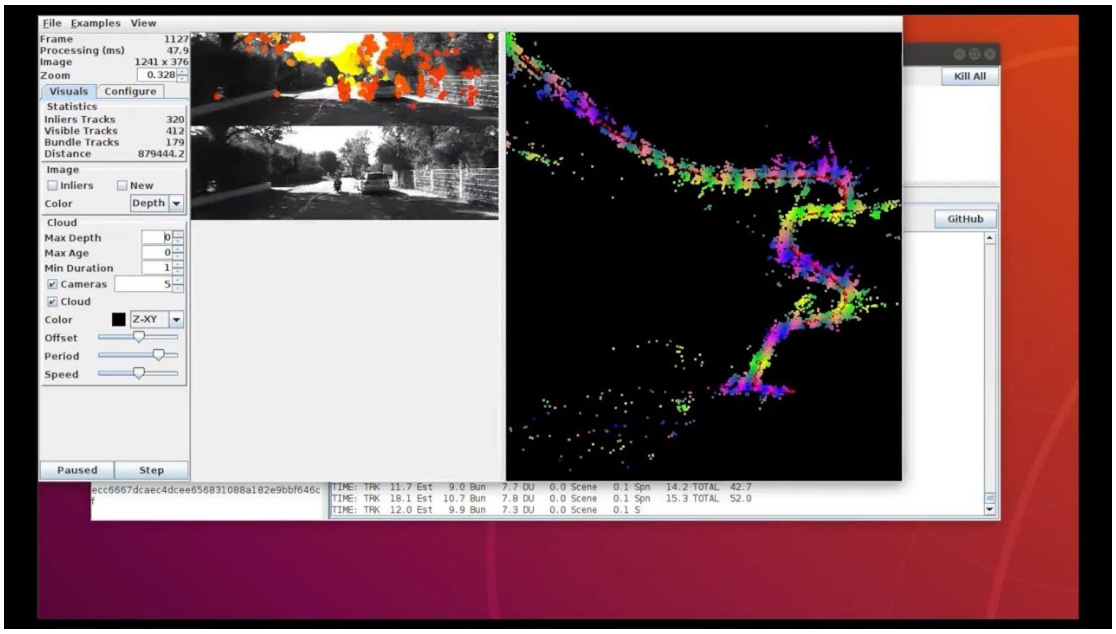


```
FPS:32
Objects:

car: 0.25
person: 0.35
person: 0.28
person: 0.22
bicycle: 0.26
bicycle: 0.37
```

Fig. 3.15: Yolo Real time application example

3.3.3.4 BoofCV

BoofCV is an open-source library that is written specifically for real-time computer vision. It was released under an Apache 2.0 license for both academic and commercial use. Thee are options for various branches of CV in BoofCV including low-level image processing, feature detection and tracking, camera calibration, etc. Some of the packages in BoofCV include Image processing functions with image processing functions that operate on pixels, Geometric vision for extracting image features using 2D and 3D geometry, Calibration that has functions to determine the camera’s intrinsic and extrinsic parameters, Recognition for recognizing complicated visual objects, etc.



```
Frame                 1127
Processing (ms)       47.9
Image           1241 x 376
Zoom                 0.328
Statistics
Inliers Tracks         320
Visible Tracks         412
Bundle Tracks          179
Distance          879444.2
Image
[ ] Inliers    [ ] New
Color            Depth
Cloud
Max Depth                0
Max Age                  0
Min Duration             1
[✓] Cameras              5
[✓] Cloud
Color             Z-XY
Offset
Period
Speed
```

```
ecc6667dcaec4dcee656831088a182e9bbf646c
f
```

```
TIME: TRK  11.7 Est   9.0 Bun   7.7 DU   0.0 Scene   0.1 Spn   14.2 TOTAL  42.7
TIME: TRK  18.1 Est  10.7 Bun   7.8 DU   0.0 Scene   0.1 Spn   15.3 TOTAL  52.0
TIME: TRK  12.0 Est   9.9 Bun   7.3 DU   0.0 Scene   0.1 S
```

Fig. 3.16: BoofCV example

System Design

4.1 Circuit diagram

4.1.1 Camera Module:

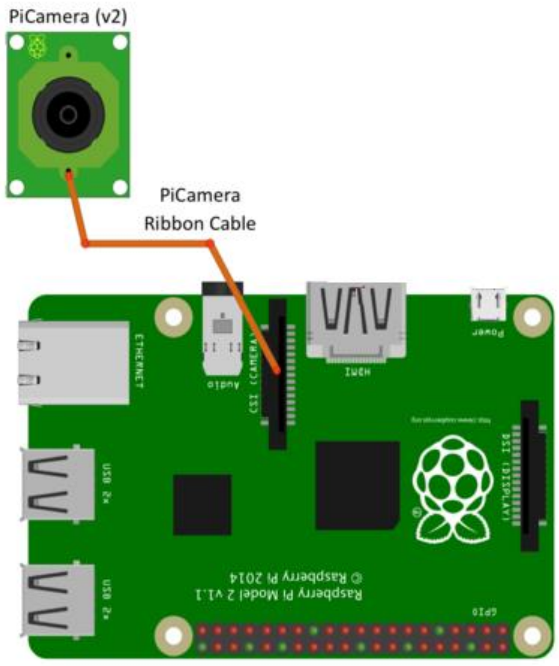


Fig. 4.1: Project Top Level Circuit Diagram

In this chapter, we will introduce the circuit diagram of the proposed project for Plant Wheat Detection Using Artificial Intelligence. In Figure 4.1, shows the circuit connection of the raspberry pi 4 to the raspberry pi 4 camera module V2 through using the camera pins implemented in the raspberry pi 3 CSI (Camera Socket Interface).

The Camera Module can be used to take high-definition live stream, as well as stills photographs. It’s easy to use for beginners, but has plenty to offer advanced users if you’re looking to expand the user knowledge. There are lots of examples online of people using it for time-lapse, slow-motion, and other video cleverness.

The gory details about IMX219 and the Exmor R back-illuminated sensor architecture on Sony’s website, but suffice to say this is more than just a resolution upgrade: it’s a leap forward in image quality, colour fidelity, and low-light performance. It supports 1080p30, 720p60 and VGA90 video modes, as well as still capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi.

PIN connections

- Camera module is connected to the Raspberry Pi CSI

4.2 Algorithms

This section will present the training algorithms for the camera module to get wheat spike training, as well as the detection algorithms for the camera module to detect wheat spike.

4.2.1 Training Algorithm

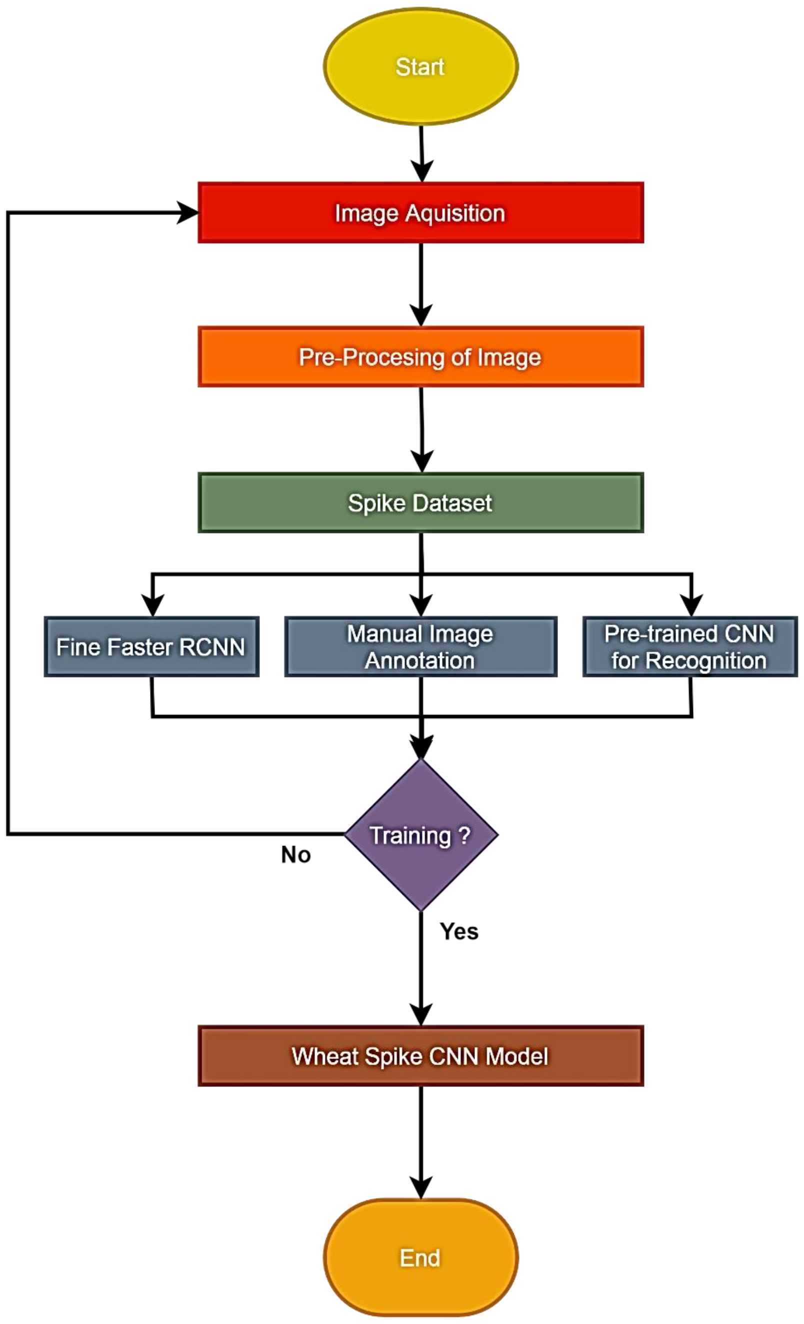


Fig. 4.2: Training Flow Chart

In this algorithm, the microprocessor starts by

1. Loading the camera module and getting the live stream video.
2. Then ensure that the camera is open and there is a live stream.
3. If the camera and live stream existed, the microprocessor will operate preprocesing to the frame captured
4. Build tensorflow dataset by following three operations
    a. Fine Faster RCNN
    b. Manual Image Annotation
    c. Or Pre trained CNN for recognition.
5. If training is completed and build the dataset.
6. Finally end the wheat spike CNN model.

4.2.2 Detection Algorithm

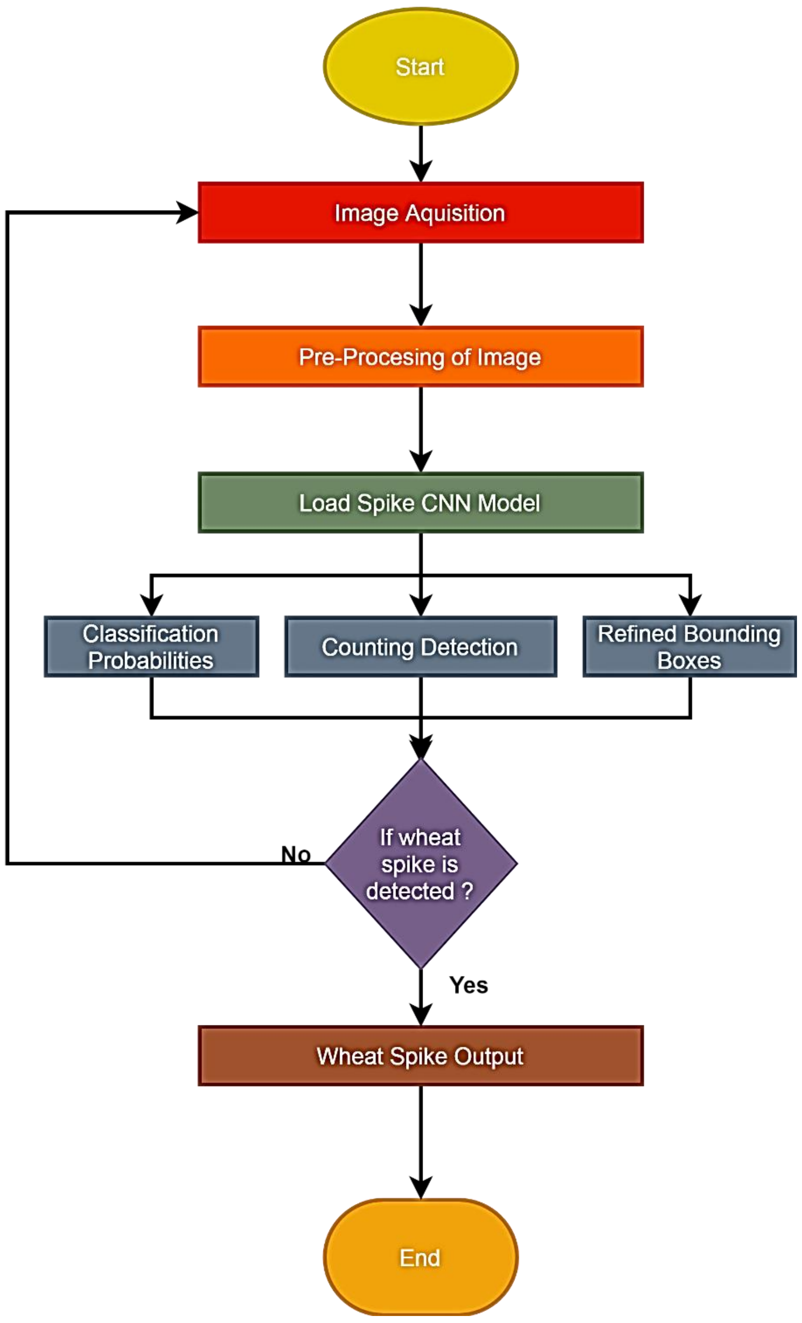


Fig. 4.4: Detection Flow Chart

In this algorithm, the microprocessor starts by

1. Loading the camera module and getting the live stream video.
2. Then ensure that the camera is open and there is a live stream.
3. If the camera and live stream existed, the microprocessor will operate pre-processing to the frame captured
4. Load pre trained wheat spike CNN model.
5. Then operate the following parameters using the CNN model as:

    a. Classification probabilities
    b. Counting of detected wheat spike
    c. Draw refined bounding boxes
6. If wheat spike is detected
    a. The system outputs the number of detected counting as well as a green bounding box for each detected wheat spike.

4.3 Summary

This chapter presents the connection of camera module to the Raspberry Pi 4. All these connections are illustrated. Furthermore, the algorithm of training wheat spike model and detection of wheat spike. Finally, the summary of this chapter is presented.

Chapter V
Implementation and Testing

5.1 Hardware implementation

Figure 5.1, shows the hardware implementation for the proposed Plant Wheat Detection Using Artificial Intelligence prototype. The figure illustrates the connection of the raspberry pi with the Camera Module. The camera module will be responsible for training and recognizing the wheat spike.

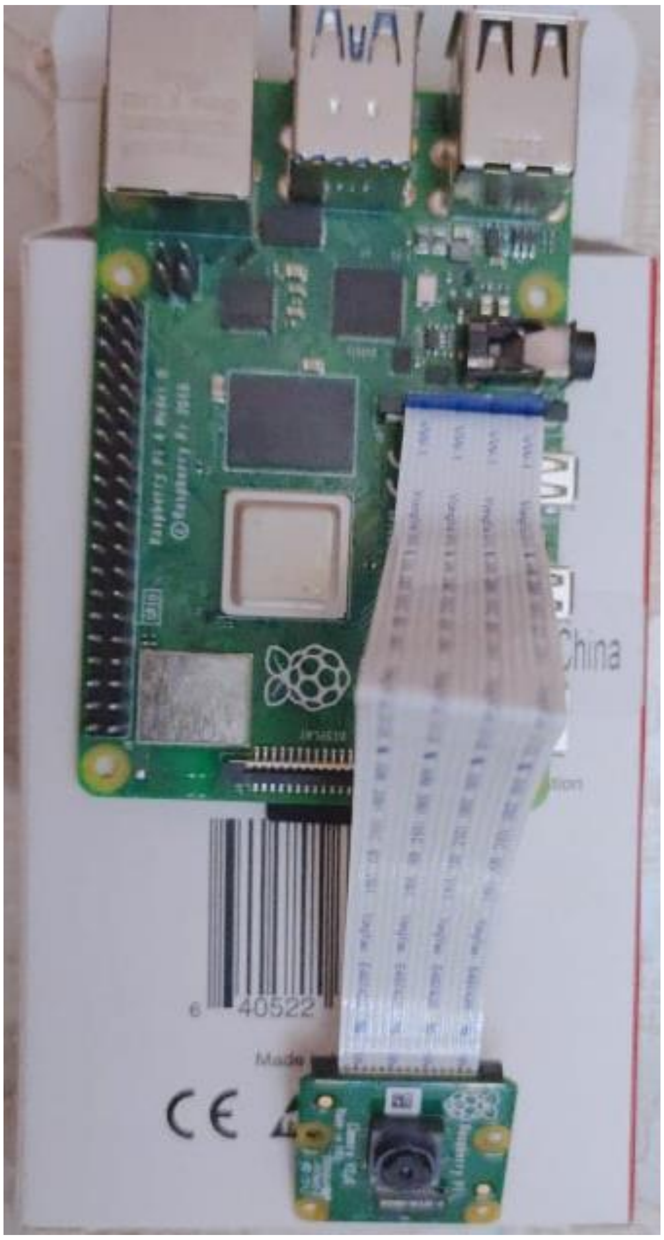


Fig. 5.1: Plant Wheat Detection Using Artificial Intelligence Hardware

5.1.2 Problem encountered

1- Training for wheat grains of different types of wheat plants.

2- Classify the different diseases of the wheat plant.

3-Wheat Leaf detection classifying features.

4-Differentiate between the wheat spike color and the leaf disease.

5.2 Software implemented

5.2.1 Raspberry Pi Configuration

1-Download the distribution from the raspberrypi.org

2-Extract the image file

3-Insert the SD card into your SD card reader

4-Download the Win32DiskImager utility

5-Extract the executable from the zip file and run the Win32DiskImager utility

6-Select the image file you extracted above.

7-Click Write and wait for the write to complete.

8-Updating and enabling Raspberry pi

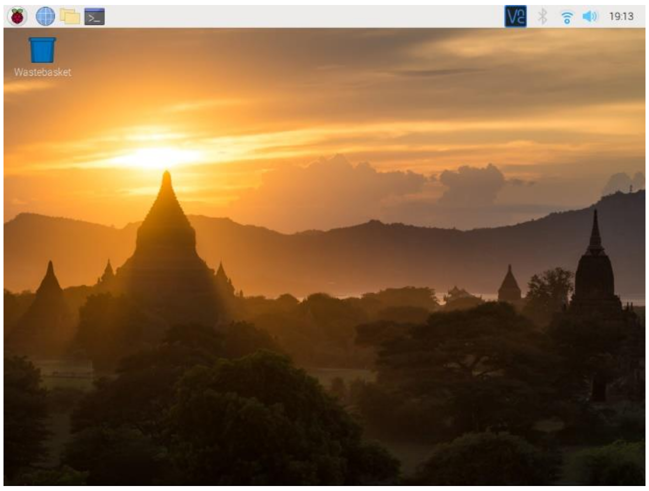
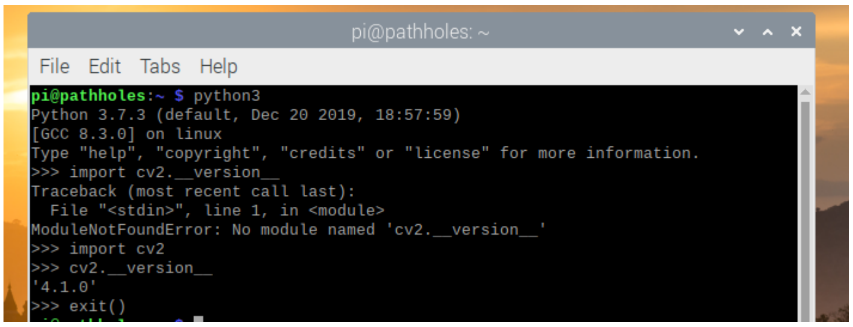


Fig. 5.2: Raspberry pi Buster Linux Desktop

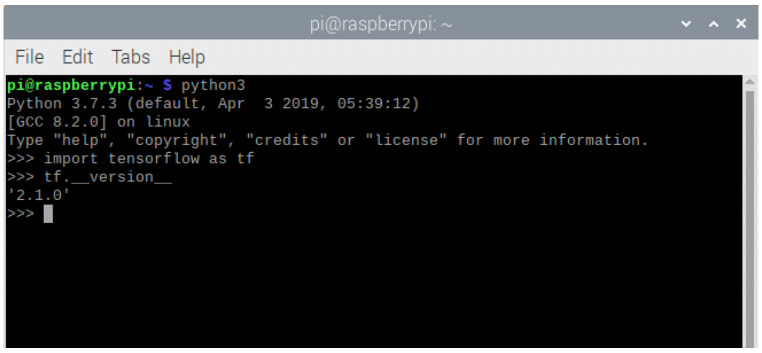
9- installing OpenCV to python3



```
pi@pathholes:~ $ python3
Python 3.7.3 (default, Dec 20 2019, 18:57:59)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2.__version__
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'cv2.__version__'
>>> import cv2
>>> cv2.__version__
'4.1.0'
>>> exit()
```

Fig. 5.3: OpenCV installation to Python3

10-installing TensorFlow



```
pi@raspberrypi:~ $ python3
Python 3.7.3 (default, Apr  3 2019, 05:39:12)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> tf.__version__
'2.1.0'
>>> 
```

Fig. 5.4: TensorFlow installation to Python3

5.2.2 Wheat Spike Detection

1- downloading pre trained wheat spike model

2- train the wheat spike model with more images as shown in Figure 5.5

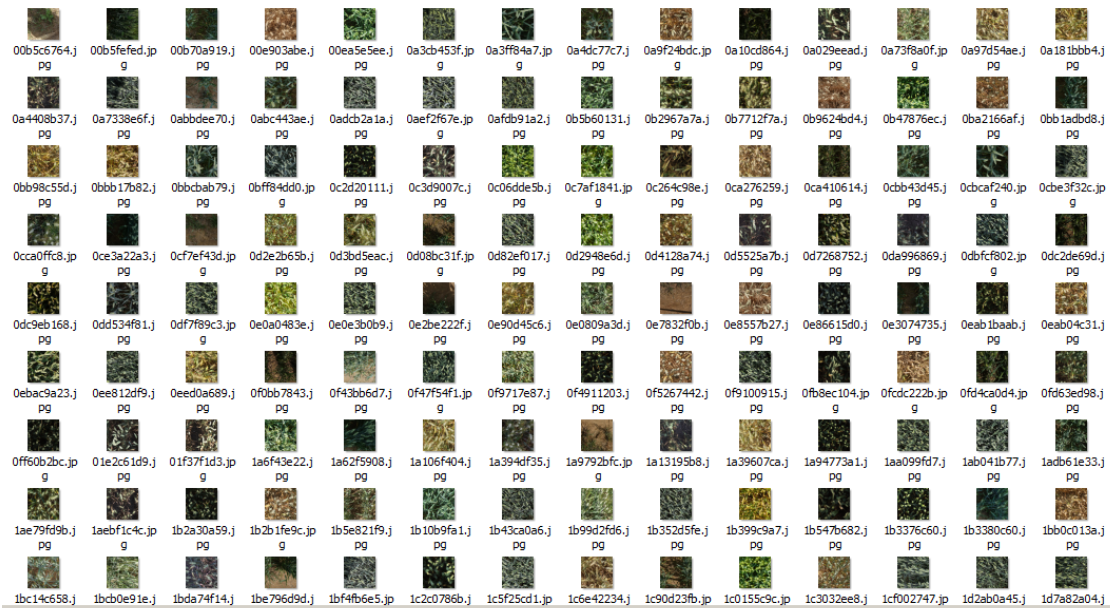


Fig. 5.5: Images used for training

In Figure 5.5, shows sample of images that is used to train the wheat spike model. This set contain 3442 jpg images with a resolution of 1024x 1024 pixels.

3- create python code to load the wheat spike CNN model and get fixed images and evaluate the wheat spike.

```
import numpy as np
import tensorflow as tf
import cv2
import matplotlib.pyplot as plt
import os
import pandas as pd
from tqdm import tqdm
```

```
print("Loading Libraries")
model_path = '/home/pi/wheat/inceptinV2_frozen_inference_graph.pb'
test_images_path = '/home/pi/wheat/test/'
sample_csv = '/home/pi/wheat/sample.csv'
print("Loading Paths")
sub = pd.read_csv(sample_csv)
sub.head()
print("Sample CSV File")
with tf.compat.v1.gfile.FastGFile(model_path, 'rb') as f:
    graph_def = tf.compat.v1.GraphDef()
    graph_def.ParseFromString(f.read())
submission = pd.DataFrame(columns=list(sub.columns))
plt.figure(figsize=(20,10))
with tf.compat.v1.Session() as sess:
        sess.graph.as_default()
        tf.import_graph_def(graph_def, name='')
        img = cv2.imread('test1.jpg')
        rows = img.shape[0]
        cols = img.shape[1]
        inp = cv2.resize(img, (640, 480))
        inp = inp[:, :, [2, 1, 0]]  # BGR2RGB
        out = 
sess.run([sess.graph.get_tensor_by_name('num_detections:0'),

sess.graph.get_tensor_by_name('detection_scores:0'),

sess.graph.get_tensor_by_name('detection_boxes:0'),

sess.graph.get_tensor_by_name('detection_classes:0')],
                        feed_dict={'image_tensor:0': inp.reshape(1, 
inp.shape[0], inp.shape[1], 3)})

        # Saving detected bounding boxes.
        num_detections = int(out[0][0])
        pred_str = ''
        for i in range(num_detections):
            classId = int(out[3][0][i])
            score = float(out[1][0][i])
            bbox = [float(v) for v in out[2][0][i]]
            if score > 0.3:
                x = int(bbox[1] * cols)
                y = int(bbox[0] * rows)
                xmax = int(bbox[3] * cols)
                ymax = int(bbox[2] * rows)
                cv2.rectangle(img, (x, y), (xmax, ymax), (0,0,255), 
2)
                cv2.imwrite('result.jpg')
                pred = '{} {} {} {} {} 
'.format(np.round(score,2),x,y,xmax-x,ymax-y)
                pred_str = pred_str + pred
```

4- open terminal over the Raspberry Pi.

5-python3 wheatp.py

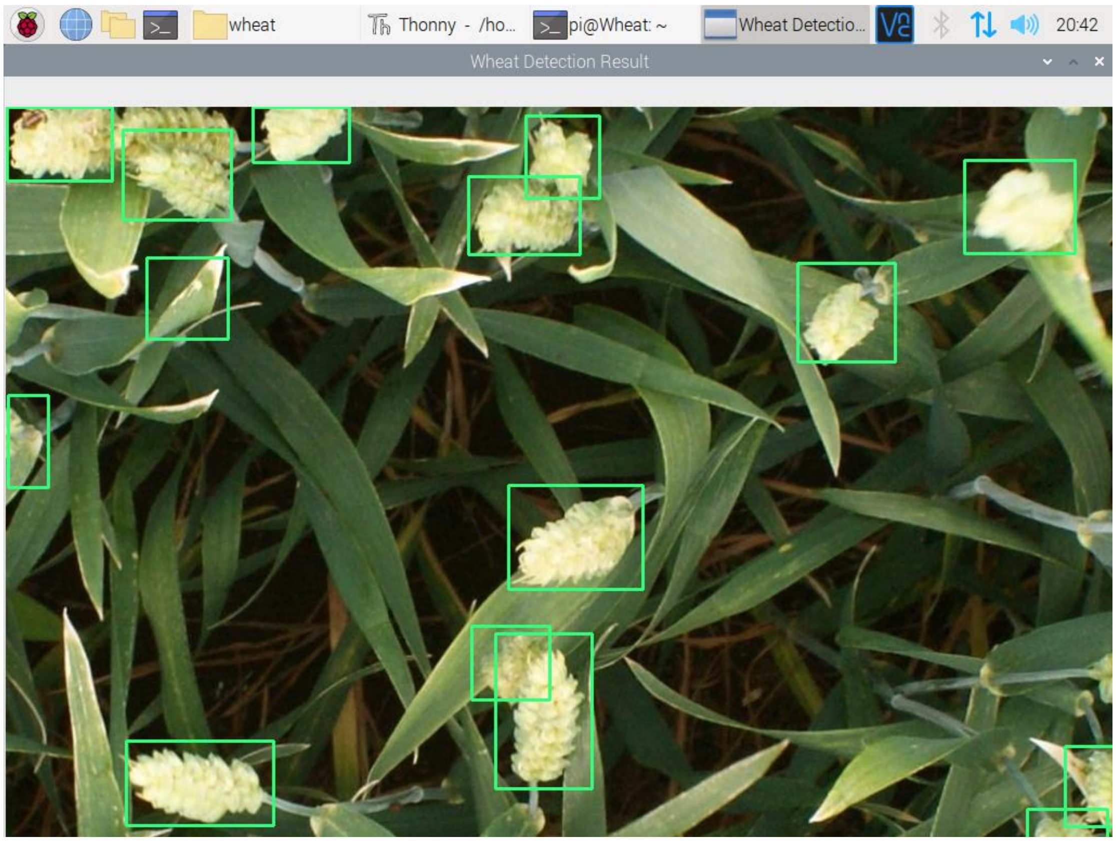


Fig. 5.6: Wheat spike detection result 1

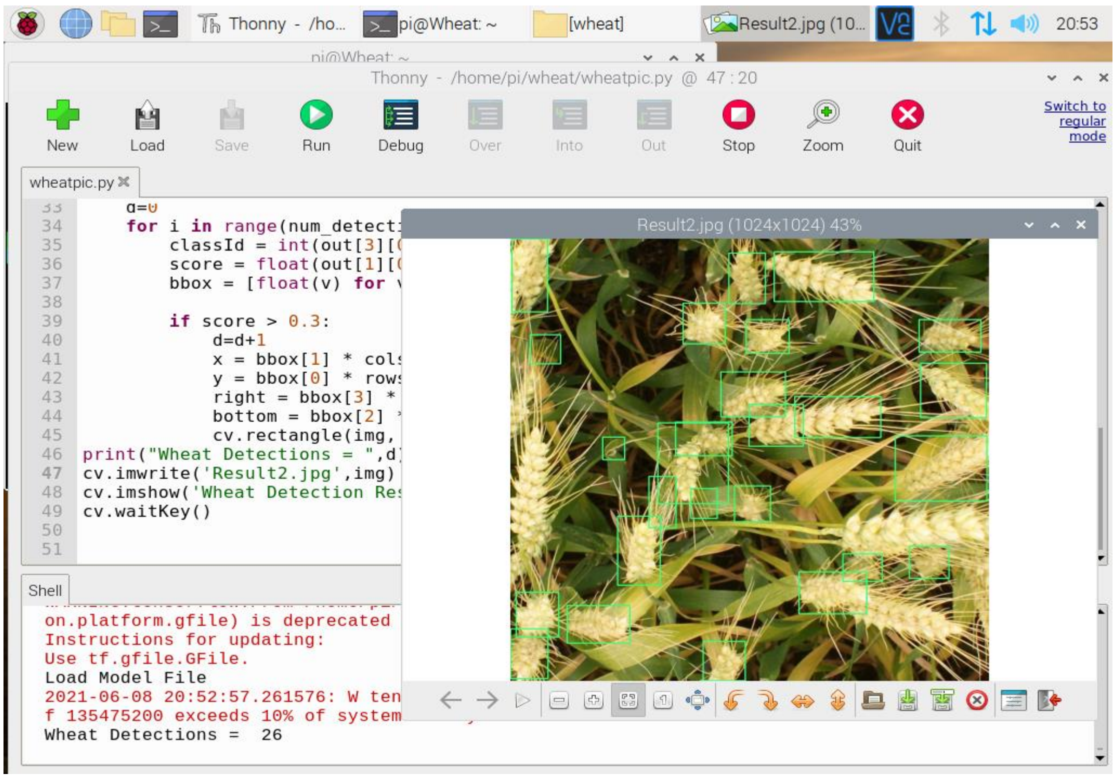
In Figure 5.6 present the wheat detection spike result for a test fixed image for 720 milli second showing a successful detection of 15 wheat spike from the total 13 wheart spike with 83 percent but not showing the number of detection, so the program is modified to get the detection number and showing the result over the live stream.

6- create python3 wheatpic.py

```
import numpy as np
import tensorflow as tf
import cv2 as cv
print("Load Main Libraries")
# Read the graph.
with
tf.compat.v1.gfile.FastGFile('inceptinV2_frozen_inference_graph.pb',
'rb') as f:
    graph_def = tf.compat.v1.GraphDef()
    graph_def.ParseFromString(f.read())
    print("Load Model File")
with tf.compat.v1.Session() as sess:
    # Restore session
    sess.graph.as_default()
    tf.import_graph_def(graph_def, name='')
    # Read and preprocess an image.
    img = cv.imread('test5.jpg')
    rows = img.shape[0]
    cols = img.shape[1]
    inp = cv.resize(img, (300, 300))
    inp = inp[:, :, [2, 1, 0]]  # BGR2RGB
    # Run the model
    out =
sess.run([sess.graph.get_tensor_by_name('num_detections:0'),

sess.graph.get_tensor_by_name('detection_scores:0'),

sess.graph.get_tensor_by_name('detection_boxes:0'),

sess.graph.get_tensor_by_name('detection_classes:0')],
                   feed_dict={'image_tensor:0': inp.reshape(1,
inp.shape[0], inp.shape[1], 3)})
    # Visualize detected bounding boxes.
    num_detections = int(out[0][0])
    d=0
    for i in range(num_detections):
        classId = int(out[3][0][i])
        score = float(out[1][0][i])
        bbox = [float(v) for v in out[2][0][i]]

        if score > 0.3:
            d=d+1
            x = bbox[1] * cols
            y = bbox[0] * rows
            right = bbox[3] * cols
            bottom = bbox[2] * rows
            cv.rectangle(img, (int(x), int(y)), (int(right),
int(bottom)), (125, 255, 51), thickness=2)
print("Wheat Detections = ",d)
cv.imwrite('Result5.jpg',img)
cv.imshow('Wheat Detection Result', img)
cv.waitKey()
```

7- open terminal over the Raspberry Pi.

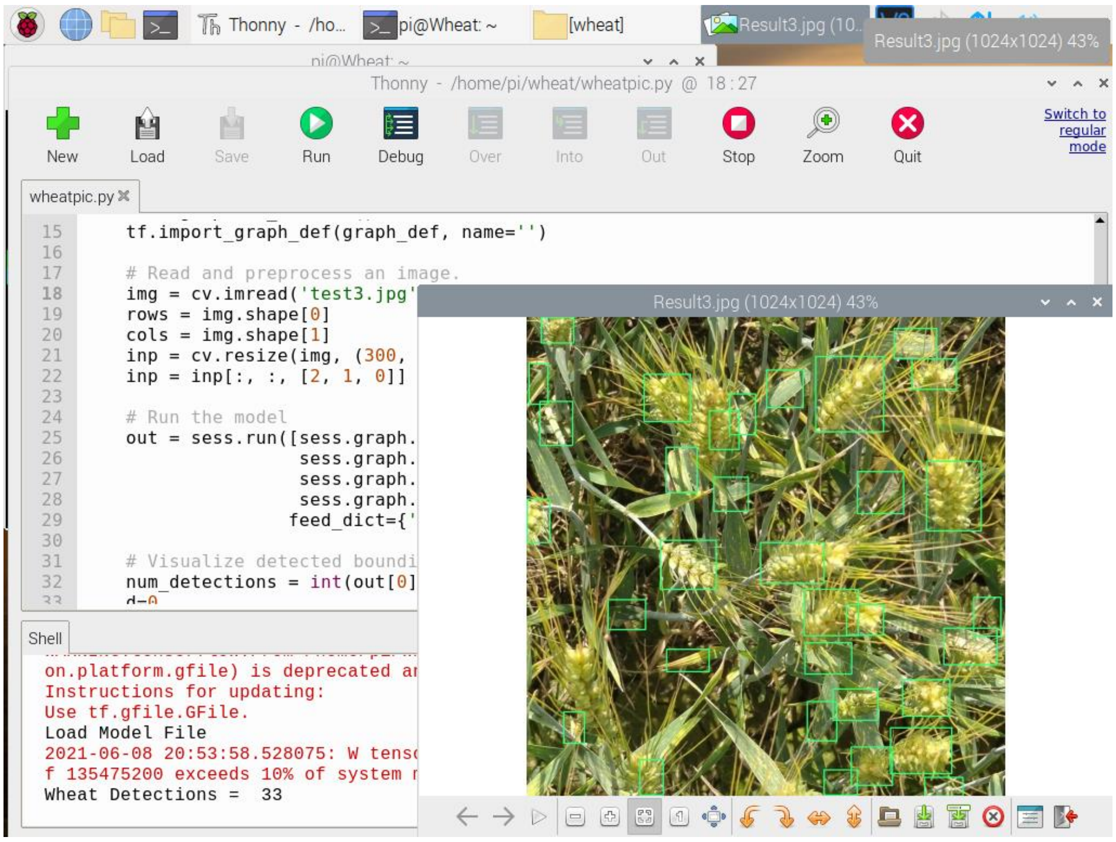
8-python3 wheatpic.py



```
    d=0
    for i in range(num_detect
        classId = int(out[3][(
        score = float(out[1][(
        bbox = [float(v) for \

        if score > 0.3:
            d=d+1
            x = bbox[1] * cols
            y = bbox[0] * rows
            right = bbox[3] *
            bottom = bbox[2] *
            cv.rectangle(img,
print("Wheat Detections = ",d)
cv.imwrite('Result2.jpg',img)
cv.imshow('Wheat Detection Res
cv.waitKey()
```

```
on.platform.gfile) is deprecated
Instructions for updating:
Use tf.gfile.GFile.
Load Model File
2021-06-08 20:52:57.261576: W ten
f 135475200 exceeds 10% of system
Wheat Detections =  26
```

Fig. 5.7: Wheat spike detection result 2



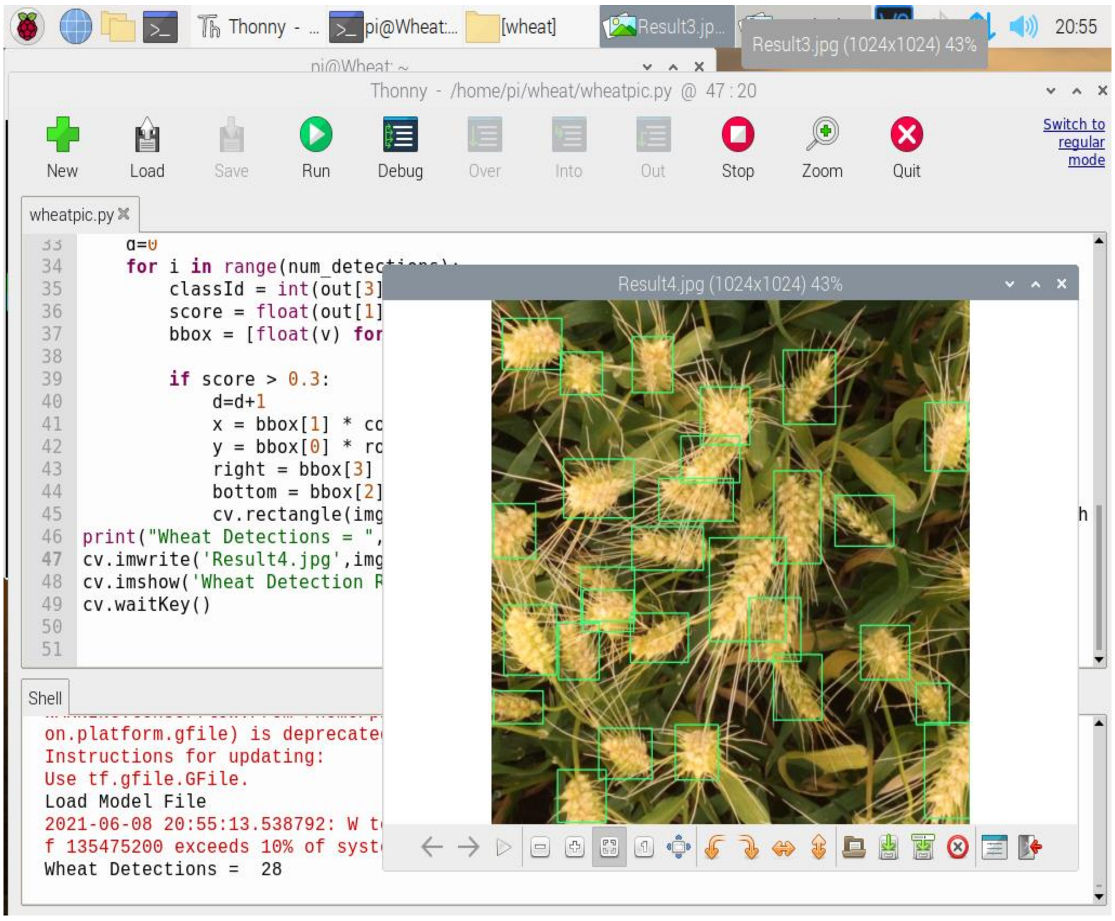
```
    tf.import_graph_def(graph_def, name='')

    # Read and preprocess an image.
    img = cv.imread('test3.jpg'
    rows = img.shape[0]
    cols = img.shape[1]
    inp = cv.resize(img, (300,
    inp = inp[:, :, [2, 1, 0]]

    # Run the model
    out = sess.run([sess.graph.
                    sess.graph.
                    sess.graph.
                    sess.graph.
                   feed_dict={'

    # Visualize detected boundi
    num_detections = int(out[0]
```

```
on.platform.gfile) is deprecated ar
Instructions for updating:
Use tf.gfile.GFile.
Load Model File
2021-06-08 20:53:58.528075: W tenso
f 135475200 exceeds 10% of system r
Wheat Detections =  33
```

Fig. 5.8: Wheat spike detection result 3

The result of wheat spike detection is shown in Figure 5.7 shows the 26 wheat spike detected from a total of 24 wheat spike in 735 milli seconds with 83 percent and showing the number of detection. Also, The result of wheat spike detection is shown in Figure 5.8 shows the 33 wheat spike detected from a total of 35 wheat spike in 1534 milli seconds with 83 percent and showing the number of detection.

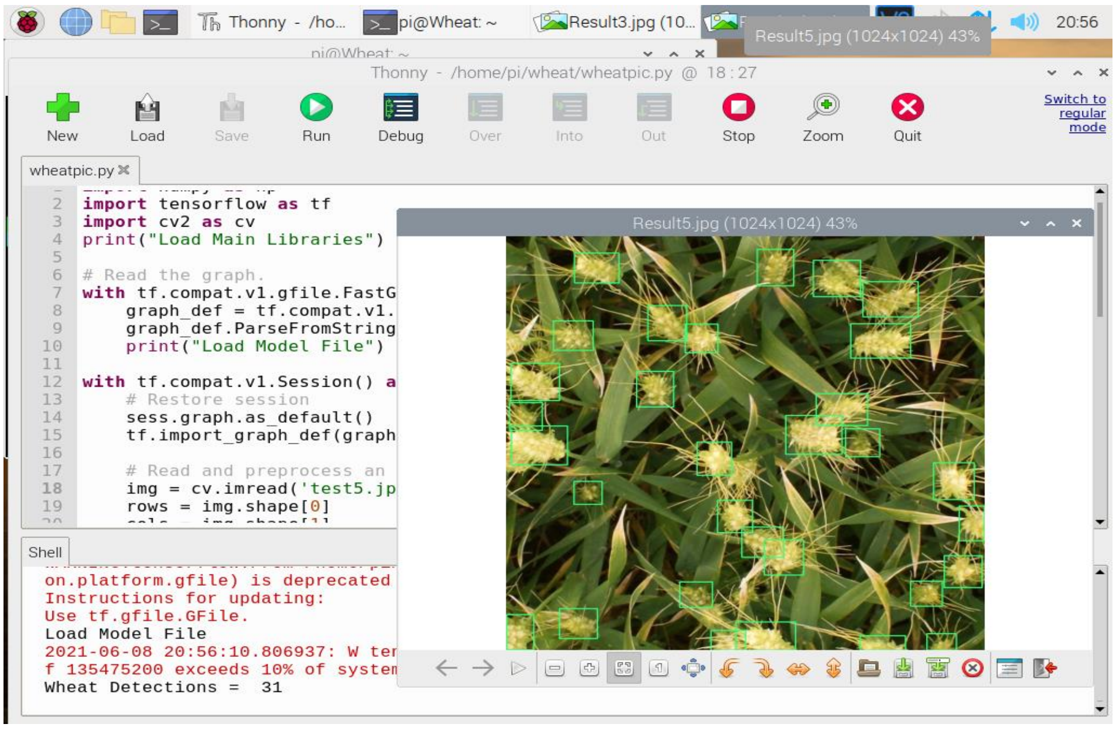
The result of wheat spike detection is shown in Figure 5.9 shows the 28 wheat spike detected from a total of 28 wheat spike in 1216 milli seconds with 100 percent and showing the number of detection. Also, The result of wheat spike detection is shown in Figure 5.10 shows the 31 wheat spike detected from a total of 31 wheat spike in 1176 milli seconds with 100 percent and showing the number of detection.


Thonny - /home/pi/wheat/wheatpic.py @ 47 : 20
New
Load
Save
Run
Debug
Over
Into
Out
Stop
Zoom
Quit
Switch to regular mode
wheatpic.py
Result4.jpg (1024x1024) 43%
Result3.jpg (1024x1024) 43%
Shell
20:55

```
    d=0
    for i in range(num_detec
        classId = int(out[3]
        score = float(out[1]
        bbox = [float(v) for

        if score > 0.3:
            d=d+1
            x = bbox[1] * co
            y = bbox[0] * ro
            right = bbox[3]
            bottom = bbox[2]
            cv.rectangle(img
print("Wheat Detections = ",
cv.imwrite('Result4.jpg',img
cv.imshow('Wheat Detection R
cv.waitKey()
```

```
on.platform.gfile) is deprecate
Instructions for updating:
Use tf.gfile.GFile.
Load Model File
2021-06-08 20:55:13.538792: W t
f 135475200 exceeds 10% of syst
Wheat Detections =  28
```

Fig. 5.9: Wheat spike detection result 4



```
import tensorflow as tf
import cv2 as cv
print("Load Main Libraries")

# Read the graph.
with tf.compat.v1.gfile.FastG
    graph_def = tf.compat.v1.
    graph_def.ParseFromString
    print("Load Model File")

with tf.compat.v1.Session() a
    # Restore session
    sess.graph.as_default()
    tf.import_graph_def(graph

    # Read and preprocess an
    img = cv.imread('test5.jp
    rows = img.shape[0]
```

```
on.platform.gfile) is deprecated
Instructions for updating:
Use tf.gfile.GFile.
Load Model File
2021-06-08 20:56:10.806937: W ter
f 135475200 exceeds 10% of system
Wheat Detections =  31
```

Fig. 5.10: Wheat spike detection result 5

9- create python3 color.py

```
from collections import deque
import numpy as np
import argparse
import imutils
import cv2
import urllib
ap = argparse.ArgumentParser()
# ap.add_mutually_exclusive_group("-b", type=int, default=64)
ap.add_argument("-v", "--video", help="path to the (optional) video
file")
ap.add_argument("-b", "--buffer", type=int, default=64, help="max
buffer size")
args = vars(ap.parse_args())
lower = {'red':(166, 84, 141), 'green':(66, 122, 129), 'blue':(97,
100, 117), 'yellow':(23, 59, 119), 'orange':(0, 50, 80)}
upper = {'red':(186,255,255), 'green':(86,255,255),
'blue':(117,255,255), 'yellow':(54,255,255), 'orange':(20,255,255)}
 colors = {'red':(0,0,255), 'green':(0,255,0), 'blue':(255,0,0),
'yellow':(0, 255, 217), 'orange':(0,140,255)}
 if not args.get("video", False):
    camera = cv2.VideoCapture(0)
else:
    camera = cv2.VideoCapture(args["video"])
while True:
    (grabbed, frame) = camera.read()
    if args.get("video") and not grabbed:
        break
     frame = imutils.resize(frame, width=900)
     blurred = cv2.GaussianBlur(frame, (11, 11), 0)
    hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
    for key, value in upper.items():
        kernel = np.ones((9,9),np.uint8)
        mask = cv2.inRange(hsv, lower[key], upper[key])
        mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
        mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
        cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
            cv2.CHAIN_APPROX_SIMPLE)[-2]
        center = None
        if len(cnts) > 0:

            c = max(cnts, key=cv2.contourArea)
            ((x, y), radius) = cv2.minEnclosingCircle(c)
            M = cv2.moments(c)
            center = (int(M["m10"] / M["m00"]), int(M["m01"] /
M["m00"]))
            if radius > 5.5:
                cv2.circle(frame, (int(x), int(y)), int(radius),
colors[key], 5)
                cv2.putText(frame,key + "color", (int(x-
radius),int(y-radius)), cv2.FONT_HERSHEY_SIMPLEX, 0.6,colors[key],2)
                print(key)
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
    # press 'q' to stop the loop
    if key == ord("q"):
        break
camera.release()
cv2.destroyAllWindows()
```

10- open terminal over the Raspberry Pi.

11-python3 color.py

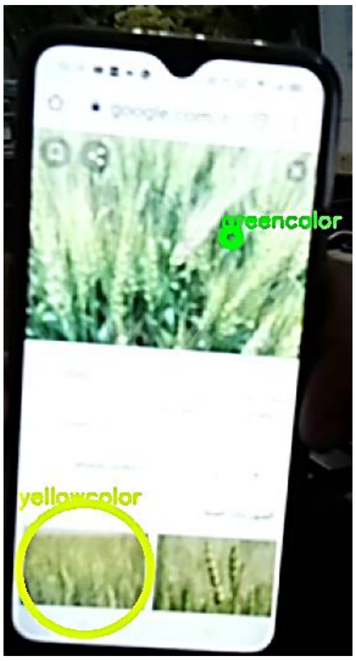


Fig. 5.11: Colour Detection Result

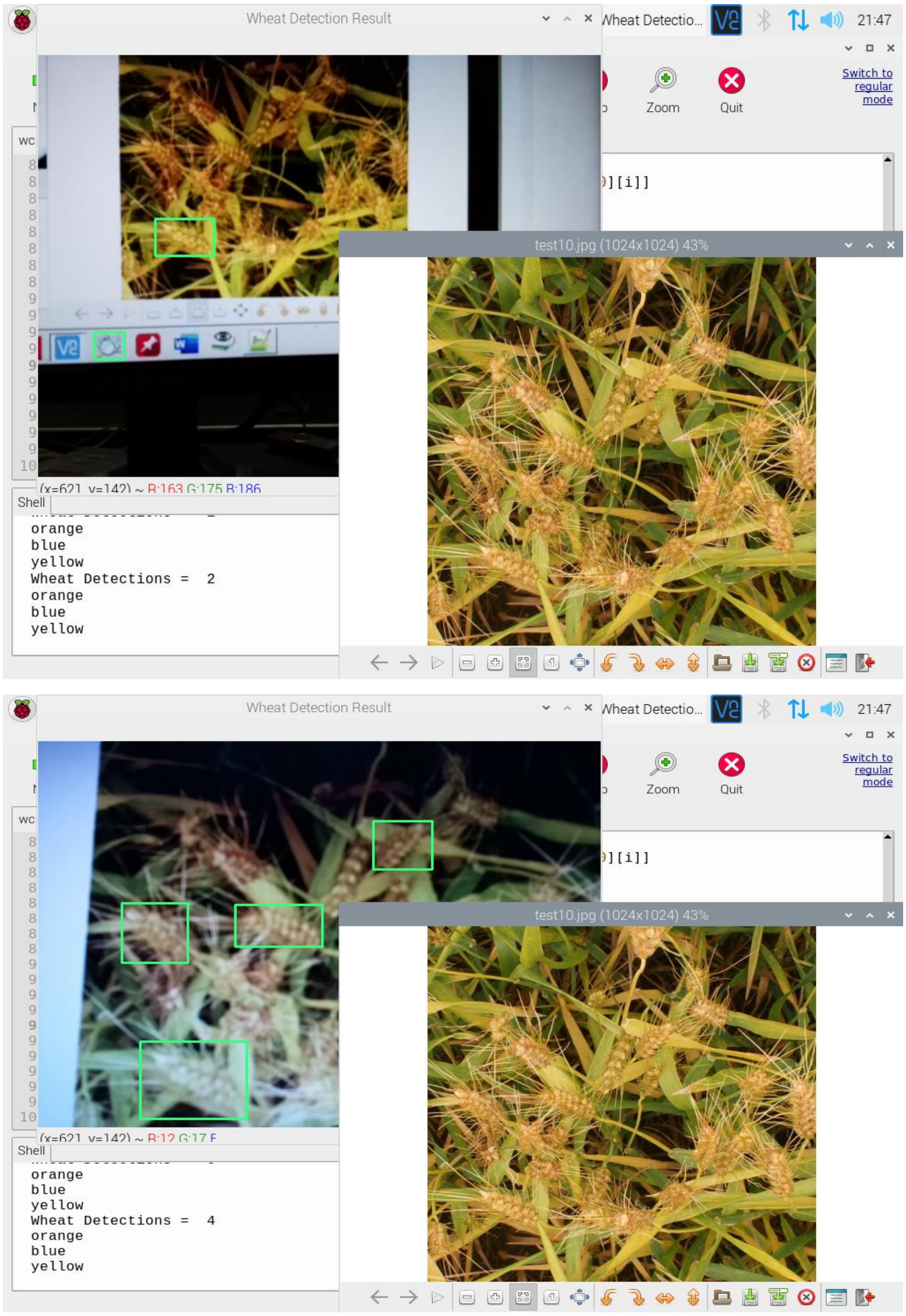
Fingure 11, shows the color detection result using live stream camera module and detecting the color in the image and detecting the green and yellow and the zones of the detected colors. All these values have performed in 23 milli seconds with 100 percent detection. So we add the color detection to enhance the detection of white spike and modify the program to operate over live stream.

12- create python3 wheatstream.py

```
import numpy as np
import tensorflow as tf
import cv2 as cv
print("Load Main Libraries")
cap = cv.VideoCapture(0)
# Read the graph.
with
tf.compat.v1.gfile.FastGFile('inceptinV2_frozen_inference_graph.pb',
'rb') as f:
    graph_def = tf.compat.v1.GraphDef()
    graph_def.ParseFromString(f.read())
    print("Load Model File")
with tf.compat.v1.Session() as sess:
    # Restore session
    sess.graph.as_default()
    tf.import_graph_def(graph_def, name='')
    # Read and preprocess an image.
#       img = cv.imread('test5.jpg')
    while True:
        ret, img = cap.read()
        rows = img.shape[0]
        cols = img.shape[1]
        inp = cv.resize(img, (300, 300))
        inp = inp[:, :, [2, 1, 0]]  # BGR2RGB
        # Run the model
        out =
sess.run([sess.graph.get_tensor_by_name('num_detections:0'),

sess.graph.get_tensor_by_name('detection_scores:0'),

sess.graph.get_tensor_by_name('detection_boxes:0'),

sess.graph.get_tensor_by_name('detection_classes:0')],
                       feed_dict={'image_tensor:0': inp.reshape(1,
inp.shape[0], inp.shape[1], 3)})
        # Visualize detected bounding boxes.
        num_detections = int(out[0][0])
        d=0
        for i in range(num_detections):
            classId = int(out[3][0][i])
            score = float(out[1][0][i])
            bbox = [float(v) for v in out[2][0][i]]

            if score > 0.3:
                d=d+1
                x = bbox[1] * cols
                y = bbox[0] * rows
                right = bbox[3] * cols
                bottom = bbox[2] * rows
                cv.rectangle(img, (int(x), int(y)), (int(right),
int(bottom)), (125, 255, 51), thickness=2)
        print("Wheat Detections = ",d)
        cv.imshow('Wheat Detection Result', img)
#      cv.waitKey(1)
        if cv.waitKey(100) == ord('q'):
            break
# When everything done, release the capture
img.release()
cv.destroyAllWindows()
```

13- open terminal over the Raspberry Pi.

14-python3 wheatstream.py



```
(x=621 y=142) ~ R:163 G:175 B:186
```

```
orange
blue
yellow
Wheat Detections =  2
orange
blue
yellow
```

```
(x=621 y=142) ~ R:12 G:17 F
```

```
orange
blue
yellow
Wheat Detections =  4
orange
blue
yellow
```

Fig. 5.12: wheat spike stream result

The result of wheat spike detection is shown in Figure 5.12 shows the 10 wheat spike detected from a total of 12 wheat spike in 1100 milli seconds with 80 percent and showing the number of detection.

Chapter VI
COST ANALYSIS

Table 6.1 Cost analysis

Item	**Cost**
Raspberry Pi 4	2000 L.E.
Camera Module	800 L.E.
Total	**2800 LE**

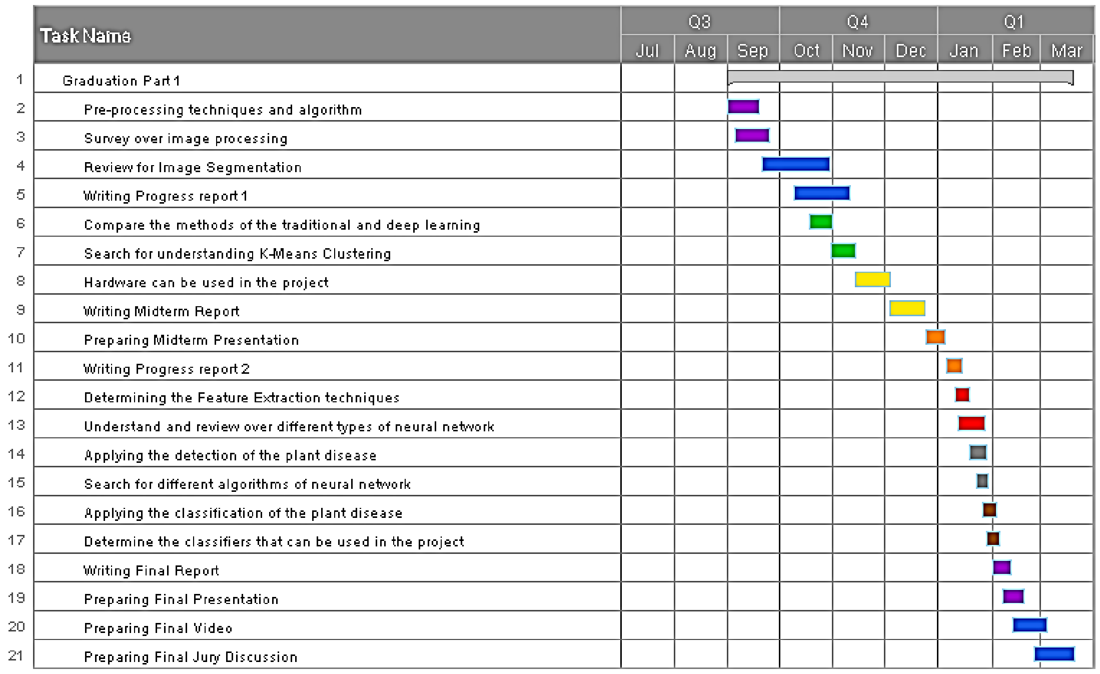
Chapter VII
TIME PLAN

7.1 Time Table

Table 7.1 Time plan

Task No.	Task Holder	Task Description	Exact Execution Time
1.	Mahmoud	Buying Components	2 weeks
2.	Marco	Testing Components	2 weeks
3.	Mahmoud	Operating System installation over the Raspberry pi board	2 weeks
4.	Marco	Operating the camera module over the Raspberry Pi	2 weeks
5.	Mahmoud Marco	Progress Report one	2 weeks
6.	Mahmoud	Training the wheat objects	1 week
7.	Marco	Detecting the wheat objects	1 week
8.	Mahmoud	Counting the wheat detected	1 week
9.	Marco	Color detection method	1 week
10.	Mahmoud Marco	Progress Report Two	2 weeks
11.	Mahmoud	Increase the performance speed and time	2 weeks
12.	Marco	Increase the accuracy of video stream detection	2 weeks
13.	Mahmoud	Project assembly	2 weeks
14.	Marco	Project Testing	2 weeks
15.	Mahmoud Marco	Final Preparation	2 weeks

7.2 Gantt Chart



	Task Name
1	Graduation Part 1
2	Pre-processing techniques and algorithm
3	Survey over image processing
4	Review for Image Segmentation
5	Writing Progress report 1
6	Compare the methods of the traditional and deep learning
7	Search for understanding K-Means Clustering
8	Hardware can be used in the project
9	Writing Midterm Report
10	Preparing Midterm Presentation
11	Writing Progress report 2
12	Determining the Feature Extraction techniques
13	Understand and review over different types of neural network
14	Applying the detection of the plant disease
15	Search for different algorithms of neural network
16	Applying the classification of the plant disease
17	Determine the classifiers that can be used in the project
18	Writing Final Report
19	Preparing Final Presentation
20	Preparing Final Video
21	Preparing Final Jury Discussion

Chapter VIII
CONCLUSION AND FUTURE WORK

8.1 Conclusion

Manual examination is not as accurate to examine crop growing stages because of the possibility of the human mistake and errors. While machine examination or automatic examination can easily examine crop growing stages and increase productivity because it provides fast and accurate examine result. The magnificent yielding of the crops depends upon vital growing phases so that during the growing season, the plant can be capitalized on suitable weather conditions. However, an understanding of the crops can contribute to the assessment of crop conditions and production potential during the growing season.

The project have segmented an image by using k-clustering algorithm using cluster to generate the initial centroid and the final segmented result is compare with k-means clustering algorithm and we can conclude that the proposed clustering algorithm has better segmentation and after formation of clusters apply the deep learning algorithm to find out the matching image, the project aims to find the value of z in each stage and compare those with the dataset images and the value having less z value will be the final value. Deep learning algorithm gives the appropriate result of the diseases and takes less amount of time for detection than other methods.

8.2 Future Work

This project can have a future work as follows:

- Increasing the wheat spike database and increasing its accuracy.
- Adding a wheat leaf disease to evaluate the wheat disease types
- Increase the accuracy of detection over live stream.
- Decrease the detection time using faster algorithms for wheat spike detection.
- Classify the type of wheat spike.

References

[1] M. P. Reynolds and N. E. Borlaug, “Applying innovations and new technologies for international collaborative wheat improvement,” Journal of Agricultural Science, vol. 144, no. 2. Cambridge University Press, pp. 95–110, Apr-2006.

[2] N. Brisson, P. Gate, D. Gouache, G. Charmet, F. X. Oury, and F. Huard, “Why are wheat yields stagnating in Europe? A comprehensive data analysis for France,” F. Crop. Res., vol. 119, no. 1, pp. 201–212, Oct. 2010.

[3] B. Schauberger, T. Ben-Ari, D. Makowski, T. Kato, H. Kato, and P. Ciais, “Yield trends, variability and stagnation analysis of major crops in France over more than a century,” Sci. Rep., vol. 8, no. 1, pp. 1–12, Dec. 2018.

[4] M. Reynolds et al., “Breeder friendly phenotyping,” Plant Science. Elsevier Ireland Ltd, p. 110396, Jan-2020.

[5] J. Crain, S. Mondal, J. Rutkoski, R. P. Singh, and J. Poland, “Combining High-Throughput Phenotyping and Genomic Information to Increase Prediction and Selection Accuracy in Wheat Breeding,” Plant Genome, vol. 11, no. 1, pp. 1–14, Mar. 2018.

[6] A. Hund, L. Kronenberg, J. Anderegg, K. Yu, A. W.-A. in crop Breeding, and U. 2019, Non-invasive phenotyping of cereal growth and development characteristics in the field. Cambridge, 2019.

[7] A. Walter, F. Liebisch, and A. Hund, “Plant phenotyping: from bean weighing to image analysis,” Plant Methods, vol. 11, no. 1, p. 14, Mar. 2015.

[8] S. Madec, X. Jin, H. Lu, B. De Solan, S. Liu, F. Duyme, E. Heritier, and F. Baret, “Ear density estimation from high resolution RGB imagery using deep learning technique,” Agric. For. Meteorol., vol. 264, pp. 225–234, Jan. 2019.

[9] M.M Hasan., J.P. Chopin, H. Laga, and S.J Miklavcic, “Detection and analysis of wheat spikes using convolutional neural networks”, Plant Methods, 14(1), 100, 2018.

[10] Zhang, N., Chaisattapagon, C. Effective criteria for weed identification in wheat fields using machine vision. Trans. ASAE 38 (3), 965–974 (1995).

[11] Moshou, D.; Bravo, C.; West, J.; Wahlen, S.; McCartney, A.; Ramon, H. Automatic detection of “yellow rust”in wheat using reflectance measurements and neural networks. Comput. Electron. Agric., 44, 173–188 (2004).

[12] Moshou, D.; Bravo, C.; Oberti, R.; West, J.; Bodria, L.; McCartney, A.; Ramon, H. Plant disease detection based on data fusion of hyper-spectral and multi-spectral fluorescence imaging using Kohonenmaps. Real-Time Imaging, 11, 75–83 (2005).

[13] Guevara-Hernandez, F., Gomez-Gil, J. A machine vision system for classification of wheat and barley grain kernels. Spanish J. Agric. Res. 9 (3), 672–680 (2011).

[14] Tian, Y., Zhao, C., Lu, S., Guo, X. SVM-based multiple classifier system for recognition of wheat leaf diseases. In: World Automation Congress (WAC), 2012. IEEE, pp. 189–193 (2012).

[15] Moshou, D.; Pantazi, X.-E.; Kateris, D.; Gravalos, I. Water stress detection based on optical multi sensor fusion with a least squares support vector machine classifier. Biosyst. Eng, 117, 15–22 (2014).

[16] Majumdar, D., Ghosh, A., Kole, D.K., Chakraborty, A., Majumder, D.D. Application of fuzzy C-means clustering method to classify wheat leaf images based on the presence of rust disease. In: Paper presented at the Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) (2014), pp. 277–284 (2015).

[17] Olgun, M., Onarcan, A.O., Ozkan, K., Işik, Ş., Sezer, O., Ozgişi, K., Koyuncu, O., Wheat grain classification by using dense SIFT features with SVM classifier. Comput. Electron. Agric. 122, 185–190 (2016).

[18] Jiang, G., Wang, X., Wang, Z., Liu, H. Wheat rows detection at the early growth stage based on Hough transform and vanishing point. Comput. Electron. Agric. 123,211–223 (2016).

[19] Mondal, D., Kole, D.K. A time efficient leaf rust disease detection technique of wheat leaf images using Pearson correlation coefficient and rough fuzzy C-means. In: Information Systems Design and Intelligent Applications. Springer, pp. 609–618 (2016).

[20] Pantazi, X.-E.; Moshou, D.; Alexandridis, T.K.; Whetton, R.L.; Mouazen, A.M. Wheat yield prediction using machine learning and advanced sensing techniques. Comput. Electron. Agric., 121, 57–65 (2016).

[21] Senthilnath, J.; Dokania, A.; Kandukuri, M.; Ramesh, K.N.; Anand, G.; Omkar, S.N., (2016) Detection of tomatoes using spectral-spatial methods in remotely sensed RGB images captured by UAV. Biosyst. Eng., 146, 16–32.

[22] Zhang, J., Wang, N., Yuan, L., Chen, F., Wu, K. Discrimination of winter wheat disease and insect stresses using continuous wavelet features extracted from foliar spectral measurements. Biosyst. Eng. 162, 20–29 (2017a).

[23] Shi, Y., Huang, W., Luo, J., Huang, L., Zhou, X. Detection and discrimination of pests and diseases in winter wheat based on spectral indices and kernel discriminant Analysis. Comput. Electron. Agric. 141, 171–180 (2017).

[24] Su, Y.; Xu, H.; Yan, L., (2017) Support vector machine-based open crop model (SBOCM): Case of rice production in China. Saudi J. Biol. Sci., 24, 537–547.

[25] Su, Y.; Xu, H.; Yan, L., (2017) Support vector machine-based open crop model (SBOCM): Case of rice production in China. Saudi J. Biol. Sci., 24, 537–547.

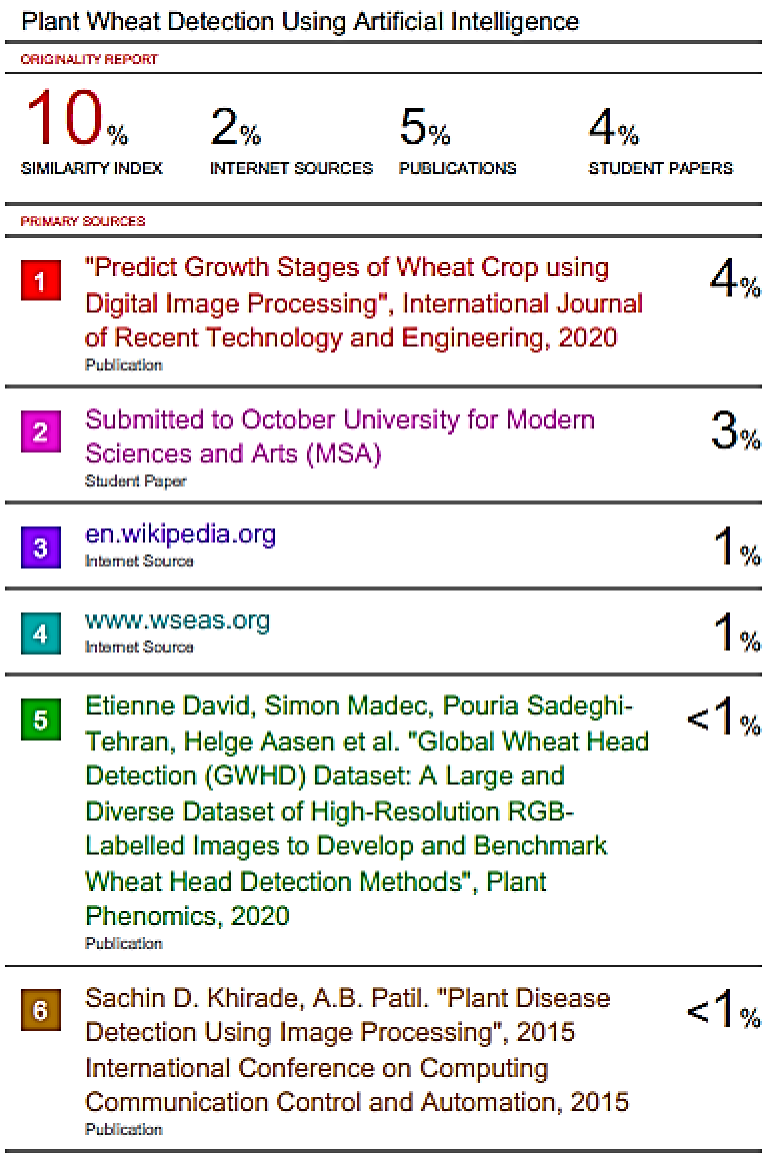
[26] Pantazi, X.E.; Tamouridou, A.A.; Alexandridis, T.K.; Lagopodi, A.L.; Kontouris, G.; Moshou, D.,(2017) Detection of Silybum marianum infection with Microbotyum Silybum using VNIR field spectroscopy. Comput. Electron. Agric., 137, 130–137.

[27] Pantazi, X.E.; Tamouridou, A.A.; Alexandridis, T.K.; Lagopodi, A.L.; Kashefi, J.; Moshou, D.,(2017) Evaluation of hierarchical self-organizing maps for weed mapping using UAS multispectral imagery. Comput. Electron. Agric., 139, 224–230.

[28] Ebrahimi, M.A.; Khoshtaghaza, M.H.; Minaei, S.; Jamshidi, B., (2017) Vision-based pest detection based on SVM classification method. Comput. Electron. Agric., 137, 52–58.

[29] Zhang, S., Wu, X., Yu, Z., Zhang, L., (2017b). Leaf image-based cucumber disease recognition using sparse representation classification. Comput. Electron. Agric. 134,135–141.

Appendix



Plant Wheat Detection Using Artificial Intelligence

ORIGINALITY REPORT

10%	2%	5%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"Predict Growth Stages of Wheat Crop using Digital Image Processing", International Journal of Recent Technology and Engineering, 2020 Publication	4%
2	Submitted to October University for Modern Sciences and Arts (MSA) Student Paper	3%
3	en.wikipedia.org Internet Source	1%
4	www.wseas.org Internet Source	1%
5	Etienne David, Simon Madec, Pouria Sadeghi-Tehran, Helge Aasen et al. "Global Wheat Head Detection (GWHD) Dataset: A Large and Diverse Dataset of High-Resolution RGB-Labelled Images to Develop and Benchmark Wheat Head Detection Methods", Plant Phenomics, 2020 Publication	<1%
6	Sachin D. Khirade, A.B. Patil. "Plant Disease Detection Using Image Processing", 2015 International Conference on Computing Communication Control and Automation, 2015 Publication	<1%

الملخص

في القرن العشرين ، هناك زيادة بمقدار ثلاثة أضعاف في عدد سكان العالم. من الصعب جدًا علينا تلبية الطلب على الغذاء لأعداد السكان المتزايدة. التطور في الزراعة مدعوم بالتكنولوجيا المتقدمة. تقيس السلطات العليا أو الحكومة في جميع أنحاء العالم وفقًا لارتفاع تجميع بعض الحبوب المهمة بما في ذلك القمح والأرز والذرة ، عددًا كبيرًا من الحالات لتأكيد الطلب على الغذاء للأمة. البلدان النامية بسرعة مثل مصر نشطة للغاية ولديها تخطيط مسبق للتنمية الزراعية في البلدان.

تعتبر الزراعة من أهم وأقدم احتلال في مصر. نظرًا لأن الاقتصاد المصري يقوم على الإنتاج الزراعي ، فإن العناية القصوى بإنتاج الغذاء ضرورية. تسبب الآفات مثل الفيروسات والفطريات والبكتيريا عدوى للنباتات مع فقدان الجودة والكمية للإنتاج. هناك قدر كبير من الخسارة للمزارع في الإنتاج. ومن ثم فإن الرعاية المناسبة للنباتات ضرورية لنفسه.

يحلل هذا المشروع اكتشاف ومراحل نمو محصول القمح من خلال التقاط صورة رقمية للمحصول من وقت لآخر. في وقت لاحق ، يتم نقل بيانات الصورة المجمعة إلى جهاز كمبيوتر. تم إجراء مزيد من التحليل للصور باستخدام OpenCV و Raspberry pi. تقلص الصبغة الخضراء لمحصول القمح مع تقدم العمر. في مراحل النمو المبكرة ، يكون لمحصول القمح الحد الأقصى من الأصباغ الخضراء التي تصبح الحد الأدنى في مرحلة النضج. لذلك تم تحليل نضج محصول القمح بقياس نسبة الصبغة الخضراء الموجودة في محصول القمح. يتم قياس نسبة اللون الأخضر الموجودة في محصول القمح باستخدام تقنية معالجة الصور الرقمية.

جامعة أكتوبر للعلوم الحديثة والآداب
كلية الهندسة

قسم هندسة نظم الاتصالات و الالكترونيات

أنظمة أكتشاف القمح النباتي باستخدام الذكاء الاصطناعي

مشروع التخرج
قدم ضمن متطلبات درجة البكالوريوس فى الهندسة الكهربية والحاسبات
الجزء الثانى

إعداد
ماركو ايمن جاد
محمود يوسف العزب

إشراف
د. محمد جمال

٢٠٢٠-٢٠٢١