PLANT DISEASE DETECTION USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

ABIRAMI P	**211420105003**
JEEVITHA R	**211420105041**
PRIYANKA S	**211418105082**

in partial fulfilment for the award of the degreeof

BACHELOR OF ENGINEERING

in

ELECTRICAL AND ELECTRONICS ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MAY 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **" PLANT DISEASE DETECTION USING MACHINE LEARNING "** is the bonafide work of **" ABIRAMI P (211420105003), JEEVITHA R (211420105041), PRIYANKA S (211420105082)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. S.SELVI, M.E, Ph.D.	**Dr. S.DEEPA, M.E, Ph.D.**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	**PROFESSOR**
Department of Electrical and	Department of Electrical and
Electronics Engineering,	Electronics Engineering,
Panimalar Engineering College,	Panimalar Engineering College,
Chennai-600 123	Chennai-600 123

Submitted for End Semester Project Viva Voce held on................................at Panimalar Engineering College, Chennai**.**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We would like to express our sincere thanks to our respected Chairman and Chancellor, Sathyabama University **Dr. JEPPIAR M.A., B.L., Ph.D.,** for being in the vanguard of scientific progress in our college.

We would like to express our deep gratitude to our Beloved Secretary and Correspondent **Dr. P.CHINNADURAI M.A., Ph.D.,** for extending his neverending support to us.

We extend our sincere thanks to our Professor & Head of the Department, Electrical and Electronics Engineering, **Dr. S. SELVI M.E, Ph.D.** and a heartfeltthanks to our Professor **Dr. D. SILAS STEPHEN M.E, Ph.D.** for his timely guidance in technical front and for instilling immense confidence in us for completing our project successfully.

We are thankful and forever indebted to our Project Supervisor , **Dr. N. MANOJKUMAR M.E, Ph.D.,** Professor for his insightful feedback and prompt assistance in completion our project.

We also extend our thanks to **All Staff Members** of Electrical and Electronics Engineering for their support and technical assistance. On a personal note, we would like to express our heartfelt thanks to our beloved **Parents** and our Friends, for their help and wishes in successfully completing this project. Thanks to **Almighty** for giving us the strength to do this project successfully.

ABSTRACT

It is often observed that when the crops and trees are affected by various diseases it affects the grainand agricultural production of the country as a whole. Organically farmers and cultivators observethe plants with naked eye without any accurate judgment for the detection and identification of a disease. This method proves be time consuming, expensive and even inaccurate at times. Artificial detection using image processing techniques offers fast and precise results. Advances in artificial intelligence and machine learning provide an opportunity to expand and improve the practice of precise crop protection and extend the market of deep learning applications in the field of professional agriculture. Text book way of training facilitates a system implementation in practice. Various steps required for implementing this plant disease recognition model are discussed throughout the report, starting from compiling images in order to create an integrated database, assured by agricultural experts present online, a deep learning framework to perform the deep CNNtraining. This is a new approach in detecting plant diseases using the deep convolutional neural network which are trained and finely tuned so as to fit accurately to the dataset of affected plant's leaves that was gathered for various plant diseases. The uniqueness of the developed model lies in its working; healthy leaves are in line with other classes, which enables the model to distinguish between diseased leaves and healthy ones or from the background noise using deep CNN, output will be displayed in the LCD display via Arduino UNO.

Keywords: Deep Convolutional neural networks, Deep learning, Image Classification, Plant disease,Rectified Linear Units.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
ReLu	Rectified Linear Unit
R-CNN	Region Based Convolutional Neural Network
SSD	Single Shot Detector
GDP	Gross Domestic Product
UN	United Nations
GUI	Graphical User Interface
HYV	High Yielding Variety
MLP	Multilayered Perceptron
KNN	K Nearest Neighbor
BPNN	Back Propagation Neural Network
SVM	Support Vector Machine
AI	Artificial Intelligence
FCM	Fuzzy C Means Clustering
PNN	Probabilistic Neural Network

CHAPTER 1
INTRODUCTION

1.1 Introduction to Agronomics

The art of Agronomics or crop cultivation is the methodical practice done primarily by the homo sapiens in which they use land to grow crops, fruits, vegetables rearing and management of livestock. The word agriculture is essentially a *latin* word agriculture which rounds up to just two words soil and culture.

According to recent studies, more or less 50% of the entire world's population and manifestly the 75% of India's population is directly or indirectly engaged in an agricultural activity of some sort. It is known that India’s agriculture stratum reports for approximately eighteen percent of India's GDP *vis-à-vis* gross domestic product and in fact provides employment opportunities to half of the country’s workforce. Substantially, India is the world’s largest exporter and producer of grains, lentils, rice, wheat and numerous spice products. Miscellaneous industries like the forestry and pisciculture commercially also come under all of these prevalent agricultural practices. It also fulfils the basic needs of the vast human population like food, shelter and clothing.

Statistically, one fifth of the aggregate exports of the country are agricultural goods. Agricultural statistics and predictions play a major role and hold formidable significance to a larger population at hand. In an attempt to enhance the quality as well as surge the quantity of the produce, various new state of the art techniques have been proposed in the field of agricultural sphere. Better and fairly advanced high yielding seeds, fertilizers, customized irrigation techniques are some of the practices which are quite prevalent and have increased the harvest yield and thus increased the land available under cultivation. To run a successful agrobusiness one needs to focus on a few fundamentals like HYV Seeds, soil conditioners, manure, fertilizer, machinery and handful labor. The human populace is outgrowing itself persistently over the years which can be sustained only by looking for better cultivation practices and expanding the use of ultra-advance power-assisted technologies and machineries. India possesses a vivid agricultural palette which consists of diverse crops, the most prominent of which are rice, corn maize and wheat. Despite having a colossal and enormous potential in the agricultural sector, the overall end yield of crops per hectare area in India is marginally lower than that of the widely accepted international standards.

The farming sector plays a climacteric role in the believed diversification of the economic sector to a larger dimension. On the ground level there might be some minute trackable progress in this dormant field, but there are certain problems which are still quite common and have been persistently affecting the Indian farmers for the past decades. It is also conjectured that one-fifth of comprehensive agricultural production is crumbled to dust due to lack of proficiency and incompetence in garnering, harvesting, transporting and stockpiling of the of the subsidized crops and end products.

1.1.1 Motivation behind the project

It is well known fact that both the Central and State federal Governments churn significant amount of revenues from the agriculture sector. Roadways and Railway industry also derive a considerable part of their income from the dissemination of agriculturally produced goods. Thus, it can be safely said that agriculture is an indispensable understructure in the country's economy and can't be ignored that easily. As per the Economic Survey of the financial year 2019-20, agriculture sector puts more than 50 per cent of the total workforce in India to a gainful employment and contributes just around 17-18 per cent to the country's GDP at large. The survey also suggests that the farmers of the new building nation are now keener to adapt relatively new farm mechanization and smart techniques at a faster pace as compared to recent years. The crops and plants growing in abysmal conditions are very much susceptible to various kinds of complex diseases which are proportionately very difficult and baffling to espy through naked eyes without any prior knowledge in the field.
The 21st century contemporary farms and agricultural activities/operations show a major leap from activities few decades ago, predominantly because of technological evolution, with the introduction of sensors, tools, embedded devices, machinery, and largely the whole realm of information technology. With the current market demand in the new era agriculture frequently involves using sophisticated technologies such as humidity, temperature and moisture sensors, aerial overview from drones, robotic equipment and GPS technology industrial science has become a customary standard to some markets. These advanced devices generally associated with the precision agricultural

techniques/methods empower the agrobusinesses to be more rewarding, profitable, efficient, non-vulnerable, and more ecofriendly to the nature.

1.1.2 Technology and agriculture

With the disruption in the technological innovations and advancements in farming marshalled by robotic engineering evolution and bleeding edge sensor technologies have collectively reshaped the future of farming practices over the course of time. For the prolonged centuries, farmers have taken over more and more tricks up their sleeves in the forever pursuit of foremost greater yields, the popular credence that more quantity the better notwithstanding plays a preeminent part in this day and age of farming activities, which eventually leaves all the small-scale operations to be practically non-viable. Intelligent and intuitive robotic systems have a fascinating capacity to change the landscape of the current financially rewarding model of farming so that it becomes reasonably feasible and accessible to be a modest producer thus also deranging today's agribusiness panorama.

The modern twenty first century programmed and AI technologies have the potential to solve the archaic problems known in the traditional farming practices. An artificially automated agronomic system, can make crop production remarkably more efficient, long lasting, minimize costs and thereby boost substantial quality. Advanced devices to monitor vegetable growth, improve monitoring and nurturing soil quality are currently in development. According to a UN body, The Food and Agriculture Organization reportedly 20–40% of universal annual crop yields are squandered each year to pests and diseases, making tons of unaware and immature pesticide decisions. Intelligent self-regulating devices, such as robots and drones enabling the culturists to shed the excessive chemical use by perceiving a potential threat to the whole crop beforehand and thus taking a well thought decision and using a precise chemical application or technique to eliminate the pests. Crop monitoring drones well equipped with RGB high definition cameras can identify a potential threat or pest feeding up a crop.

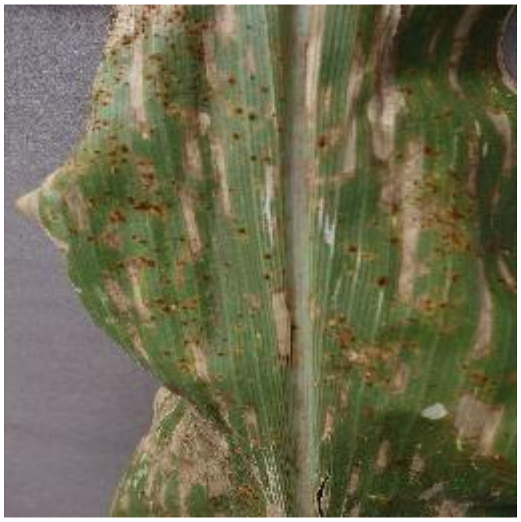
These types of practices have given rise to a new branch called as the precision agriculture. Precision agriculture advancements are majorly revamping the old school cultivation as it introduces some *modus operandi* for supervising and increasing the yield of crops. In recent times cultivators are switching on to the advanced methods of precision agriculture methodologies for several reasons:

- Relatively more yield of crop productivity and better safety standards for workers.
- Uses less input materials by adding more value in a longer run.
- Substantially low disposal of chemicals into water sources, thus more environment friendly.
- Well-engineered and dependable supervision and monitoring of air and water quality. It puts the producers themselves in saddle giving a much broader control over plant processing, storage and dispersal.

1.1.3 Some of the common corn leaf diseases

- **“Cercospora” / Gray Leaf Spot**

This type of disorder in a plant leaf incurs in the form of slate gray to brownish tan which is rectangle shaped abrasions on leaves, sheaths of leaf or may be present in the leaf husks. The fungus forms spores on the underside of leaves. This type of condition arises in extended periods of overly warm, cloudy days and consistently high ranges of humidity. With increased usage of reduced tillage and continuous corn more cases are being observed.



Common Rust

As the name suggests this can be identified when rust like tiny, cinnamon colored powdery spots surface on the upper and lower side of the corn leaf. When the infection is matured enough this powdery rust spores can be rubbed off. Regions with moderate to cool temperatures and high humidity give rise to this type of infection. The fungus arrives each season from crops grown in more southern regions.

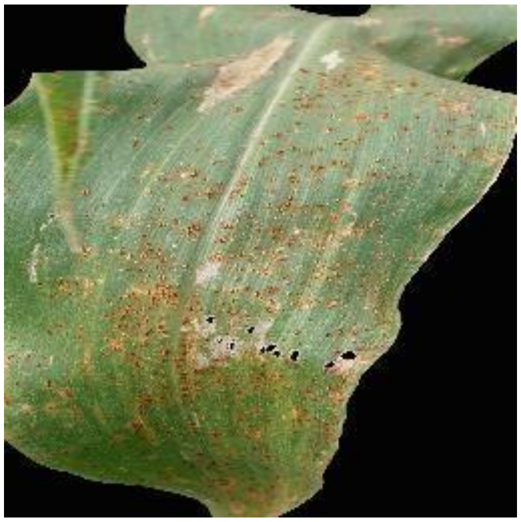


Figure 1.2 Common Rust infected leaf

- **Northern and Southern Leaf Blight**

This type of disease is observed when elliptical to cigar-shaped, gray-green lesions turn

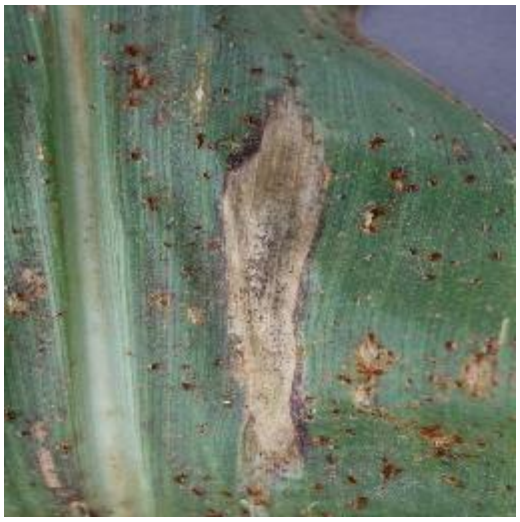


Figure 1.3 Northern Leaf Blight infected leaf

into tan-brown color. This infection starts from the lower leaves and with the progression of season the disease spreads to the upper canopy of the plant. High humidity and moderate temperatures are preferredconditions for this disease. Under humid conditions, lesions may appear dark, and fuzzy because the fungus forms spores on the dead tissues.

1.2 Problem Statement

To develop a neural network system which recognizes a corn plant infected with bacterial and infectious diseases by identifying the early symptoms from its visual appearance. Using this method provides an edge in precise prediction over identifying manually through naked eyes which usually leads to human error. Using this intelligent automated technique potential hazards can be known beforehand and necessary actions can be made, thus avoiding crop failure in a larger domain.

1.3 Objectives

- Creating a compilation of the dataset containing leaf images over diverse pictures.
- Training and developing a CNN model on collected dataset using TensorFlow and implementing the Keras.
- Building a classifier that can predict the plant disease on different labels, also improving on its prediction accuracy as compared to the previous models and approaches.

1.4 Methodology

Deep learning is a modern technique prevalent in current times primarily used for processing of image and its analysis on the data extracted from it. Instead of using image segmentation and feature extraction we can take use of this approach in current problem at hand. A supervised learning model is developed for the detection of these corn leaf disorders. Fundamentally speaking this approach is based on Convolution Neural Network. To classify these images, we need to apply various filters thus identifying each pixel in a specific image which is made neural network ready. This labelled and well-arranged data is then used to extract some unique patterns and features.

The model first created essentially has three layers thus reducing the complexity. The first layer of the structure consists of strides of 3 x 3 filters followed by max pool layer and activation function as a relu function layer. The next layer will flatten out all the extracted features from the previous layer and then feed it forward to the SoftMax layer.

In the approach of building the proposed model, we opted for Rectified linear unit relu instead of sigmoid function because it is more efficient and returns faster results for training deep neural networks and improves the overall accuracy. The proposed CNN will be refined with continuous forward and backward propagation among the nodes.

Forward propagation: On first training the model at an instance model attempts to find certain patterns and features feeding them forward thus giving predicted result at the end flattened labels layer. This mechanism followed by nodes or neurons is defined as forward propagation. Every node in the network hold some value of features is then fed forward to the next nodes.

Backward propagation: Following the forward propagation mechanism in one iteration, the neural network propagates to the previous nodes in a backward fashion. This process helps in calculating the mean value error which defines new bias and weights selection to increase accuracy in the predicted outputs thus and reducing the error to its minimal value. The continuous flow of these values from one node to another makes the model find its inefficiencies and improve on them to give accurate predictions and minimize the error.

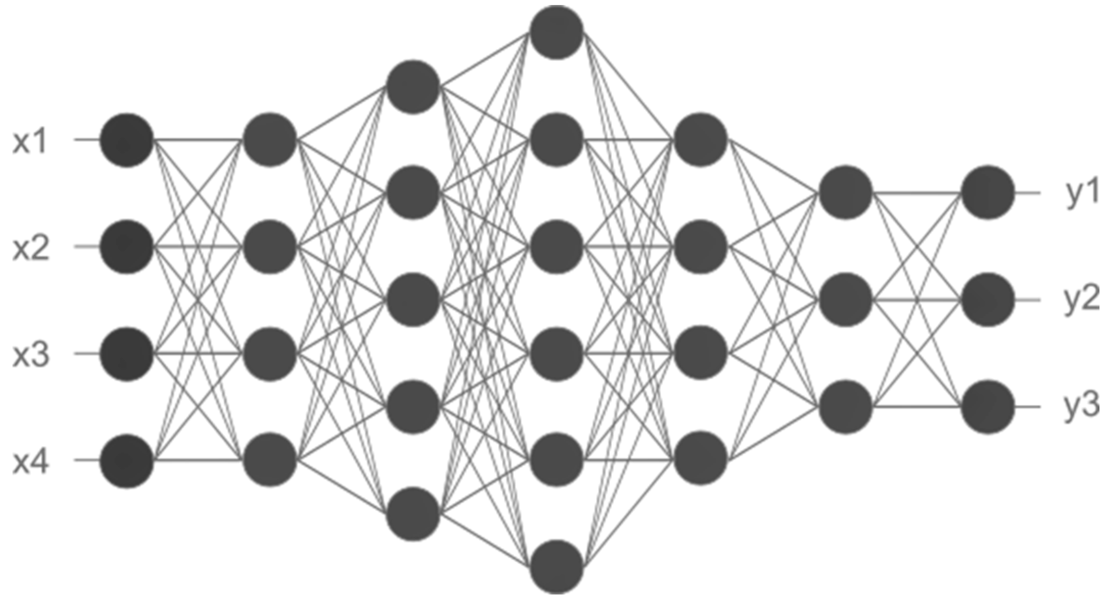


Figure 1.4 Forward and Back Propagation from nodes in fully connected CNN

The architecture of the Convolutional neural networks (CNNs) is constructed up of collective and diverse layers of fields having reception. Each node or even minute neuron collections are responsible for the processing fragments and chunks of the input image fed to one end of the network. Collaborating the outputs of all these collections are further tiled in order to the overlapping of input regions, thus resulting in the representation of the original input image in much higher resolution; this process is replicated intuitively for every layer in the network. The Tiling technique allows CNNs to evaluate translation of the input image. Convolutional neural networks can be composed of various local and global pooling layers. These layers contribute in combining the output received from these neuron clusters and combinations of the overall convolutional and fully connected layers.

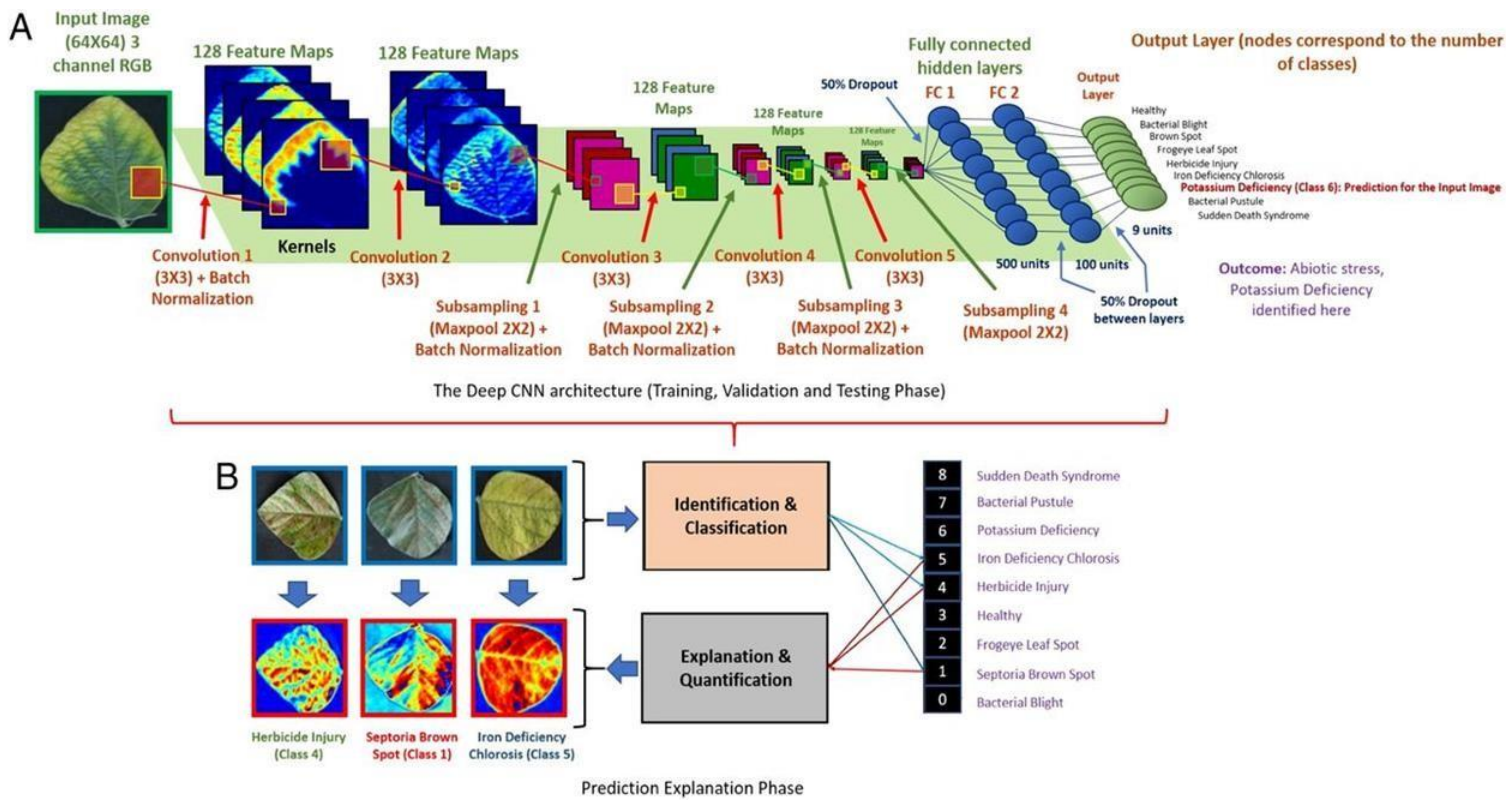

A
Input Image (64X64) 3 channel RGB
128 Feature Maps
128 Feature Maps
128 Feature Maps
128 Feature Maps
128 Feature Maps
50% Dropout
Fully connected hidden layers
FC 1
FC 2
Output Layer
Output Layer (nodes correspond to the number of classes)
Healthy
Bacterial Blight
Brown Spot
Frogeye Leaf Spot
Herbicide Injury
Iron Deficiency Chlorosis
Potassium Deficiency (Class 6): Prediction for the Input Image
Bacterial Pustule
Sudden Death Syndrome
9 units
Kernels
Convolution 1 (3X3) + Batch Normalization
Convolution 2 (3X3)
Convolution 3 (3X3)
Convolution 4 (3X3)
Convolution 5 (3X3)
500 units
100 units
Outcome: Abiotic stress, Potassium Deficiency identified here
Subsampling 1 (Maxpool 2X2) + Batch Normalization
Subsampling 2 (Maxpool 2X2) + Batch Normalization
Subsampling 3 (Maxpool 2X2) + Batch Normalization
Subsampling 4 (Maxpool 2X2)
50% Dropout between layers
The Deep CNN architecture (Training, Validation and Testing Phase)
B
Identification & Classification
Explanation & Quantification
8 Sudden Death Syndrome
7 Bacterial Pustule
6 Potassium Deficiency
5 Iron Deficiency Chlorosis
4 Herbicide Injury
3 Healthy
2 Frogeye Leaf Spot
1 Septoria Brown Spot
0 Bacterial Blight
Herbicide Injury (Class 4)
Septoria Brown Spot (Class 1)
Iron Deficiency Chlorosis (Class 5)
Prediction Explanation Phase

Figure 1.5 Architecture of the deep CNN used

Convolutional Neural networks constitutes local and global pooling layers, which work on a collaborative level and finally the output of neuron clusters is forwarded ahead. The confusional matrices are integrated to form a fully connected layer having nonlinearity applied to each point after the end of each layer.

- Building up the neural network, it also we used rectified linear unit layer and a fully connected layer. This ReLU layer will perform the function of an activation function, and will ensure non-linearity all over as the data moves to each layer in the network. If the network is deprived of this layer the data which is fed into each layer will end up losing the required dimensionality that needs tobe conserved across the whole computation.

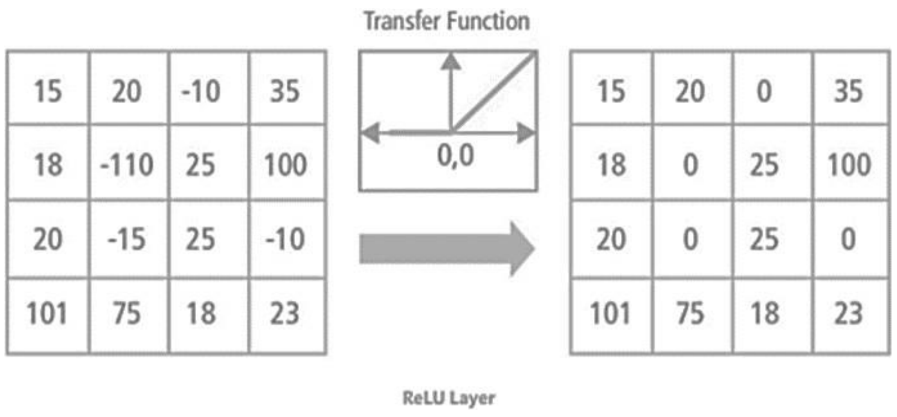


Figure 1.6 ReLu Layer Function

- The convolutional layer runs by placing a filter over the matrix array of image pixels. This then creates a convolved feature map, in layman's language it's like looking at an image through a window, which allows you to see only specific features you might not otherwise be able to see.

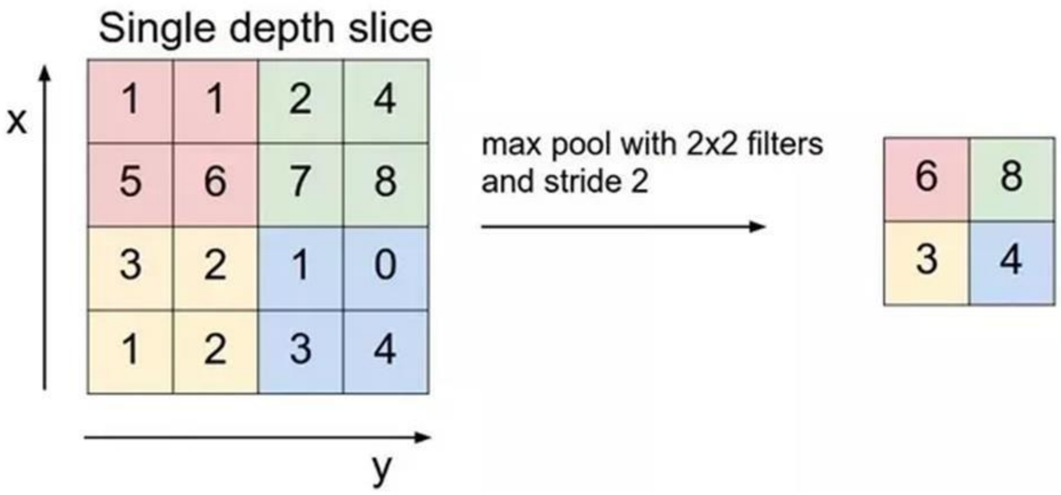


Figure 1.7 Max pooling function

- Now in the next step the Pooling layer effectively downsamples or decreases the sample size of a particular feature map. This also makes processing much optimized as it reduces the number of parameters the network needs to process. The output of this process is a pooled feature map.

- The next prominent step is max pooling, which essentially extracts out the maximum input to a more simplified and optimized convolved feature, or sometimes concept of average pooling is taken to use which simply takes the average. These steps then collectively lead to feature extraction which includes the network building up a picture of the image data with respect to the mathematical rules.
- In the end of the neural network the features are then labelled out to a flattened layer to make certain accurate predictions into different categories.

1.5 Organization

training data and testing data we want to make sure that the labels are right so on that basis we can train our model so we are going to encode the labels fist. After that we'll resize the image to 50 cross 50 pixel and we are going to read it as a grayscale image then we are going to split the data of approximately 24,000 images for training and 50 for testing once this is done we are going to reshape the data appropriately for TensorFlow. For the implementation of deep learning models, we are going to build themodel and calculate the loss which is nothing but categorical cross entropy value. Then we are going to reduce the loss by using Adam optimizer with a learning rate set aside to point double zero 1 then we aregoing to train the deep neural network for about 10 epochs or even 15 epochs or iterations and finally weare going to make predictions from the built classifier.

1.5.1 CNN

The fundamental unit of this project is a Convolutional Neural Network (CNN) which is an optimized Deep Learning technique which can take image characteristics, appoint significance to certain features namely bias and weights at all the nodes to different viewpoints/protests in the picture and have the option to separate one from the other. These are less complex in comparison to other algorithms used for classification. These networks are more intuitive in making important decisions and do not require much of the manual work.

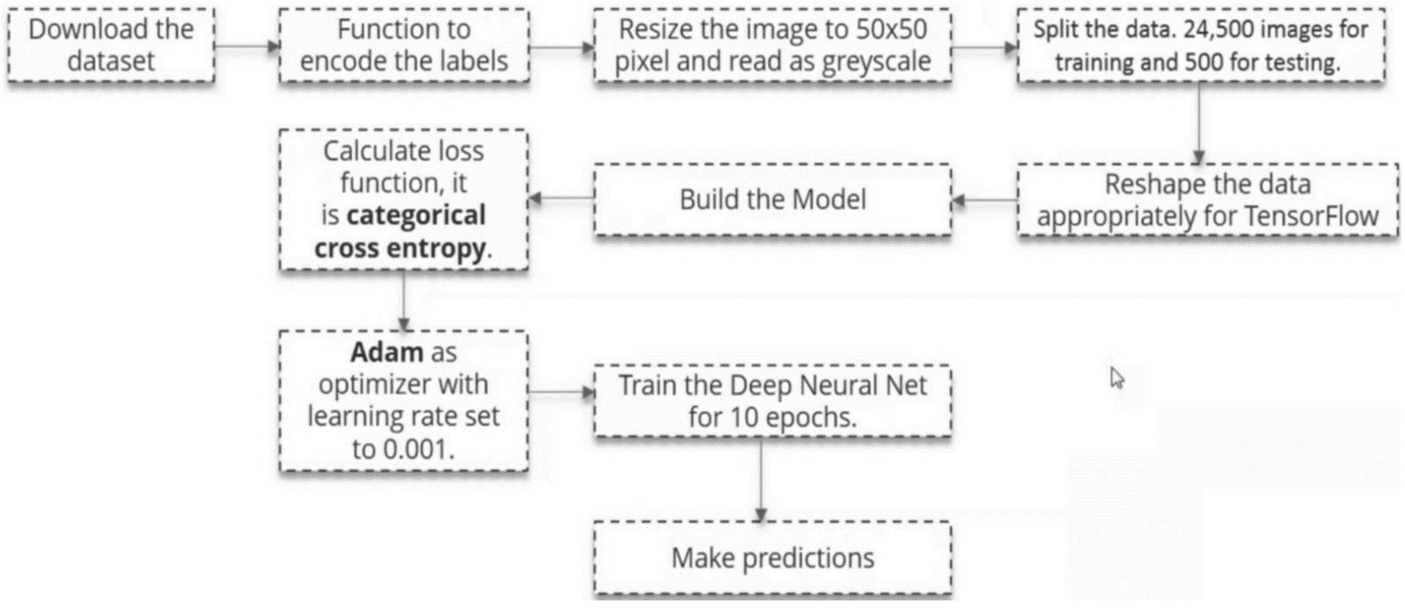

Download the dataset
Function to encode the labels
Resize the image to 50x50 pixel and read as greyscale
Split the data. 24,500 images for training and 500 for testing.
Reshape the data appropriately for TensorFlow
Build the Model
Calculate loss function, it is **categorical cross entropy**.
Adam as optimizer with learning rate set to 0.001.
Train the Deep Neural Net for 10 epochs.
Make predictions

Figure 1.8 Workflow to be followed.

1.5.2 TensorFlow

We use TensorFlow in our approach as it is open-source for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.

Google's TensorFlow is currently most trending python library used for image recognition and classification purposes.

These are nothing but multidimensional arrays, two dimensional tables or multi-dimensional. Its main feature includes its ease of use in implementing complex deep learning models with few lines of code.

1.5.3 Tkinter

We Tkinter in the later stages of development to create a usable GUI that that is more user friendly. The user is able to select an image for processing from the device itself from a dialog box prompt. This Python GUI toolkit proves to a great help for connecting with the operating system controls.

1.5.4 Jupyter Notebook

Using the anaconda navigator, the Jupyter Notebook we were able create our CNN model in Python language. It is basically an open-source web application that can be used for data cleaning and image transformation, TensorFlow, statistical and analytical modeling, data visualization and machine learning.

CHAPTER 2

LITERATURE SURVEY

[2.1] Detection and classification of plant leaf diseases through deep learning Algorithm

***M. Akila, Prabu Deepan* [2018]**

The proposed framework had a particular automated learning model in this paper, this framework depends on a particular convolutionary neural system for the discovery of leaf illness. Different cameras and assets caught the informational collection images. Pests and illness are not an issue in farming, as indicated by look into, as solid plants and plants developing in rich soil can withstand bother/ailment. Indicators, for example, Faster Region-based Convolutionary Neural Network (Faster R-CNN), Fully Convolutionary Region-based Network(R-FCN), and Single Shot Detector (SSD) were utilized. The dataset was isolated into three sets to play out the test, for example Set of approval, set of preparing and set of tests. The principal appraisal is performed on the approval set, after the neural system preparing is performed on the preparation set and the last assessment is performed on the test set-to assess results on crude information, they use preparing and approval sets to execute the preparation procedure and test set.

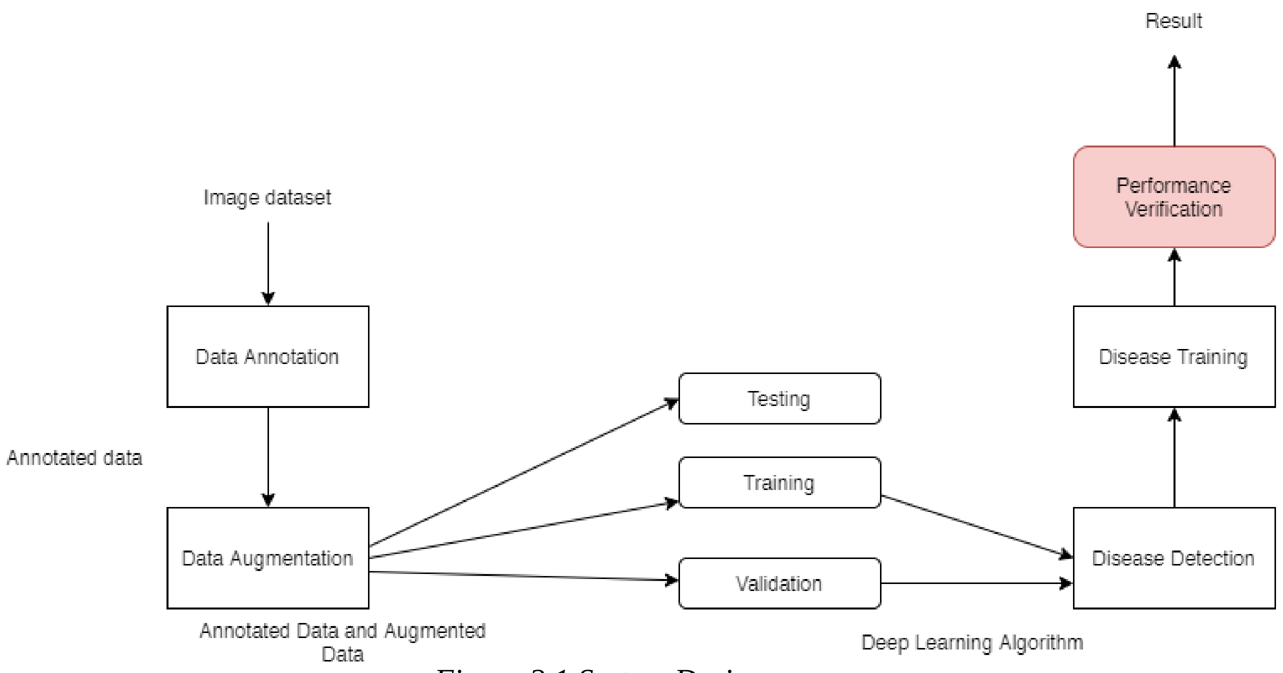
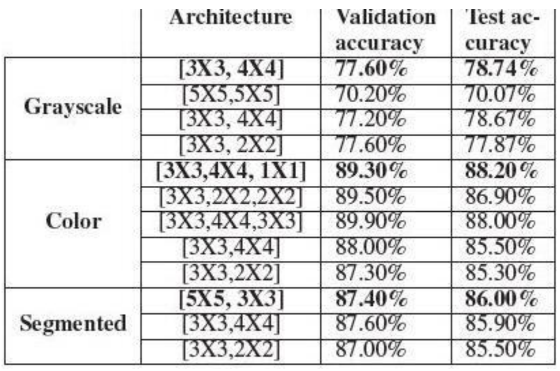


Figure 2.1 System Design

[2.2] Identification of soybean plant disease using the Convolutionary Neural Network

Jiangsheng Gui, Mor Mbaye **[2019]**

In this report, soybean plant contamination is recognized utilizing convolutional neural system. The arrangement of information contains pictures from normal sources. In the analysis of sickness, the procedure productivity of 99.32% is accomplished. The test likewise shows that the preparation set's information increment improves the system's exhibition. This present paper's proposed CNN model comprises of three convolutionary layers, trailed by a maxpooling layer for each layer Fully associated with MLP is the last layer. The enactment capacity of ReLu is applied to every convolution layer's output the yield layer is given to the softmax layer that gives the four yield classes of likelihood circulation. The arrangement of information was isolated into preparing (70%), approval (10%) and testing (20%)



	Architecture	Validation accuracy	Test accuracy
Grayscale	**[3X3, 4X4]**	**77.60%**	**78.74%**
	[5X5,5X5]	70.20%	70.07%
	[3X3, 4X4]	77.20%	78.67%
	[3X3, 2X2]	77.60%	77.87%
Color	**[3X3,4X4, 1X1]**	**89.30%**	**88.20%**
	[3X3,2X2,2X2]	89.50%	86.90%
	[3X3,4X4,3X3]	89.90%	88.00%
	[3X3,4X4]	88.00%	85.50%
	[3X3,2X2]	87.30%	85.30%
Segmented	**[5X5, 3X3]**	**87.40%**	**86.00%**
	[3X3,4X4]	87.60%	85.90%
	[3X3,2X2]	87.00%	85.50%

Figure 2.2 Classification results from different models.

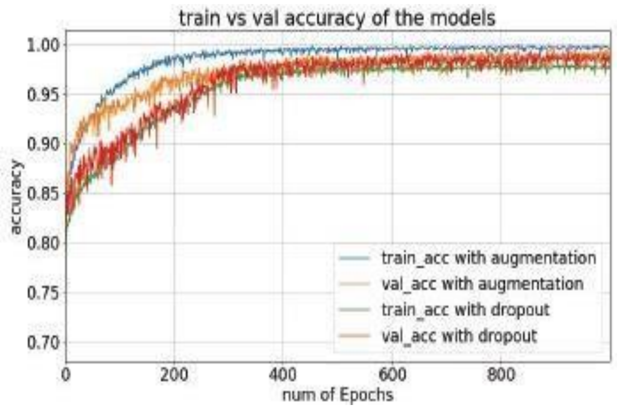


Figure 2.1.1 Training vs Validation accuracy

From the order investigation, it tends to be seen that shading pictures are more exact than dark images. Color pictures are in this way better for extraction of the component. The model result likewise demonstrates the model to be overfitting. Overfitting happens when the layout is fruitful in fitting the preparation cluster. It at that point turns into a speculation of new occurrences not in the preparation set.

[2.3] A Deep Learning Approach Classification of Banana Leaf Diseases

***Bassem Bouaziz, Jihen Amara, Alsayed Algerawy* [2017]**

This paper proposed to characterize and arrange banana ailment through the convolutionary neural system. The proposed model will help ranchers in banana plant illness detection. With the signs, the rancher will snap a photo of the leaf and the procedure can survey the sort of disease. The specialist utilized profound neural systems to discover two unmistakable banana ailments that join banana shimmering Sigatoka and banana in the genuine scene and under troublesome conditions, for example, lighting, muddled setting, totally extraordinary picture resolution, size, cause, and orientation. Once numerous analysts had the option to see the impacts of keen grouping. This has affirmed that with almost no framework exertion, the proposed approach would significantly help Associate in Nursing to accurately distinguish leaf infections. We utilized a profound learning approach in this paper to portray and depict the ailment of banana leaves. The structure of the technique comprises of two components, for example picture preparing and profound arrangement dependent on perusing.

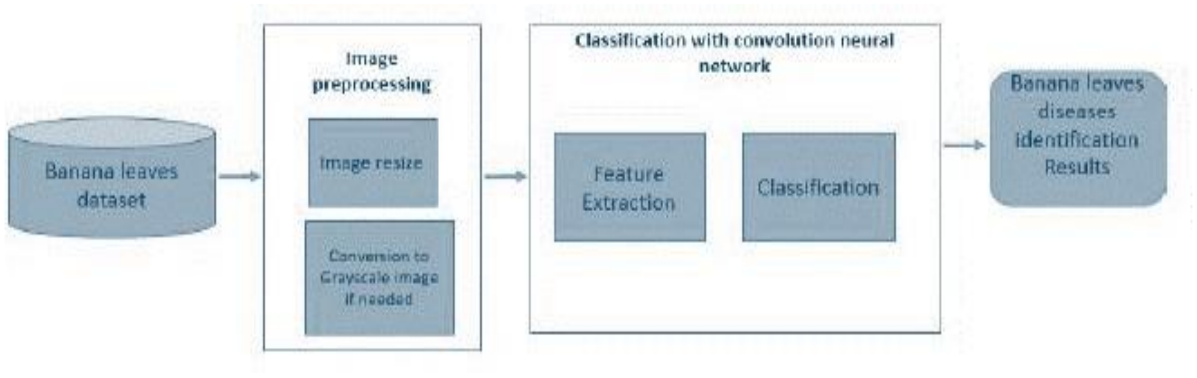

Banana leaves dataset
Image preprocessing
Image resize
Conversion to Grayscale image if needed
Classification with convolution neural network
Feature Extraction
Classification
Banana leaves diseases identification Results

Figure 2.2: Proposed framework architecture.

[2.4] Using image processing techniques, plant leaf disease detection

***Richards E.Woods and Rafel C.Gonzalez* [2016]**

This paper incorporates an overview of different methods for arrangement that can be utilized for research, farming, and so forth. This paper gives an outline of the different procedures used to characterize plant leaf malady. Plant infection distinguishing proof is the way to forestalling misfortunes in horticultural item yield. This paper investigated various procedures for fragmenting the plant's ailment partition. This paper utilizes picture preparing strategy to introduce an example of various techniques for plant leaf malady recognition. There are numerous techniques for the distinguishing proof and arrangement of ailments in programmed or PC vision, however this exploration zone is as yet absent. It is difficult to recognize all the sickness utilizing a solitary method We arrive at the accompanying resolution from the examination of the above characterization strategies. Maybe the easiest of all calculations to foresee the class of a test model is the k-closest neighbor strategy. The time multifaceted nature of making expectations is an undeniable downside of the k-NN technique. Neural systems are additionally lenient of loud information sources. In any case, it's difficult to comprehend calculation structure in the neural network's was seen as effective in characterizing high-dimensional informational collections with the best accessible AI calculations.

[2.5] Detection of damaged paddy leaf disease by image processing

Priyanka B Raj, Hegde Soumya G, Dr. Neha Mangla, Pooja R **[2018]**

A respectable approach is proposed in this paper, for example picture handling of the paddy leaf by histogram, to maintain a strategic distance from the impact of these maladies for an enormous scope. Through this strategy, one can recognize the infection at an exceptionally essential stage and accordingly find a way to diminish yield misfortune in time. The proposed strategy is planned for setting up a certified plant leaves sickness reviewing framework. Distinctive leaf tests are considered for exploratory purposes. The accompanying advances are followed for analysis:

(A) Image Enhancement,

(B) Pre-process picture

(C) Detection of paddy sickness. Histogram Equation:

M = A(i) = = = [I] Zi(x) dx ——————(1) mv = A(iv) = = Z v i(ik) k=1 = = = l v k / l = ——

—-(3) For v=1,2,........................ ,N where m k=1 v is a measure of strength.

For the grading of diseases, a histogram-

based system is developed by referring to the scale of the disease scoring in Table I.

1 1 0 0 (,)lo g (,) 4 n m Paddy iv i x y G A nm Z xy − − = = = ∑∑ −−−−−− Where Paddy G = grade value, (,) Z xy i = Role of matrix image, value of n = row and value of m = column

DISEASE GRADING VALUE BY PICKS VALUE

Table: 1

Disease Grade	Training Sample	Testing Sample	Classifier Accuracy
Blast	196	82	87%
Bacterial Leaf Blight	205	83	92%
Rice tungro	210	71	90%

Figure 2.4 Prediction Accuracy Comparison

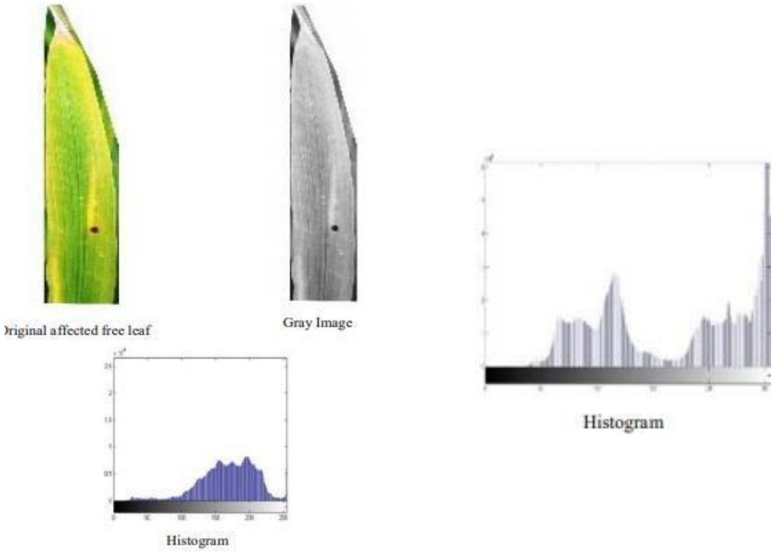


Figure 2.5 Filters applied on the set of images

[2.6] Identifying medicinal plants that are safe and tainted with Canny Edge Detection Algorithm

Vinita M. Tajane, N.J.Janwe professor **[2019]**

This paper presents a method for the application of canny edge detection algorithm on medicinal plants ' healthy and disease samplesFarmers suffer from the problem that arises from different ty pes of plant characteristics / diseases.
The very first step is to apply canny edge detection algorithm to safe and diseased samples in ord er to identify the plant disease.

CANNY EDGE ALGORITHM DETECTION:

The methodology here is that the images of healthy and infected plants were taken for the first ti me. Then apply the algorithm for canny edge detection to samples.
Canny edge detection algorithm which retains the structural properties for further processing of images.
The general purpose of edge detection is to reduce the amount of data in an image substantiallyT he objective of this algorithm for the following criteria:
 Detection:

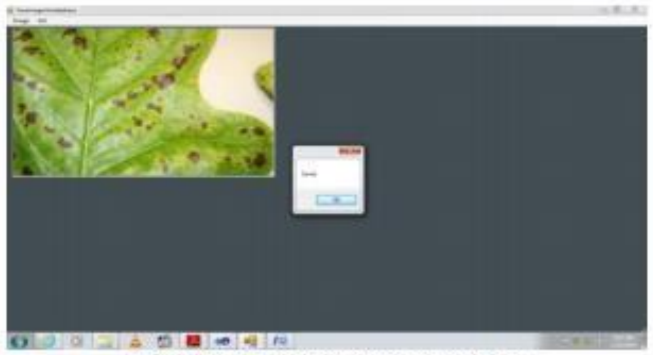
It should increase the probability of detecting actual edge points while decreasing the likelihood of wrongly detecting non-edge points.
This is equivalent to maximizing the ratio of signal to noise.

 ii) Localization: the edges found should be as close to the actual edges as possible.

iii) Number of responses: not more than one observed edge will result in one real edge. The algor ithm for detection of the Canny Edge runs in 5 separate steps:
1. Smoothing: photo blur to eliminate noise. Implemented with Basic Kernel Size (N) and Gaussi an Envelope Parameter Sigma by Gaussian Filtering.
2. Finding gradients: where the gradients of the image have large magnitudes, the edges should be marked.
3.Nonaximum suppression: edges should be labeled only as local maxima. Find gradient directio n and perform non-maxima suppression using these directions.
 4. Double thresholding: thresholding is used to evaluate possible edges.

5. Hysteresis edge tracking: Final edges are determined by suppressing all edges not connected t

Healthy Image Save into Database



Infected Image Save into Database

Figure 2.6 SS from the MATLAB application

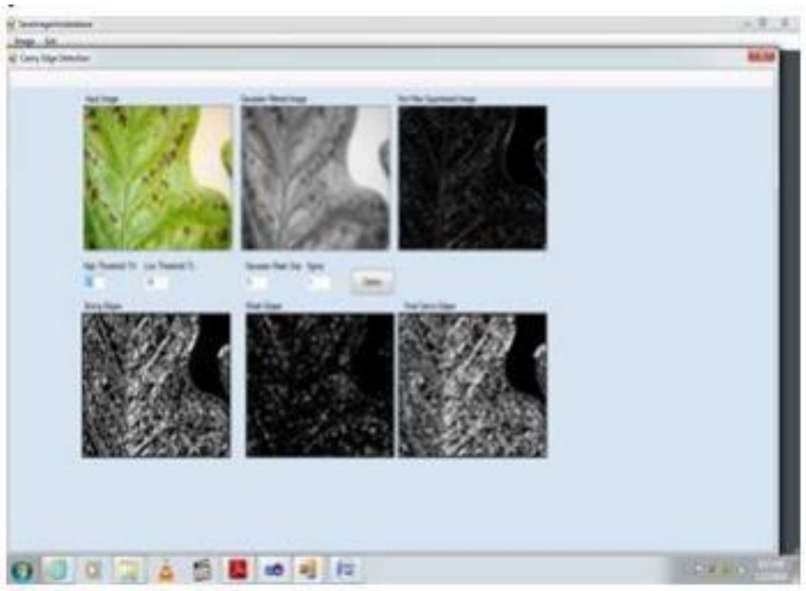


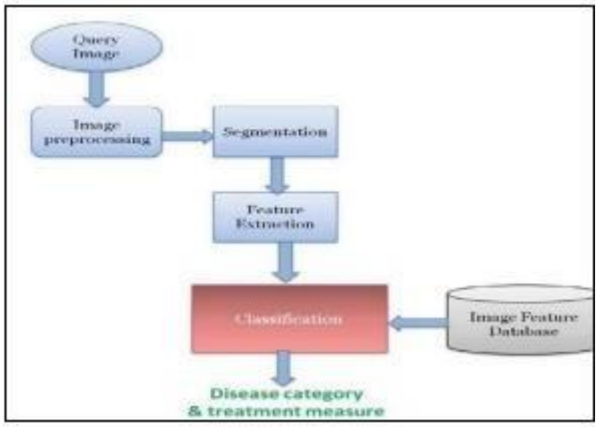
Figure 2.7 Feature Extraction

[2.7] Detection of cotton plants with leaf disease using image processing techniques

***P. Gulve, Tambe Sharayu S., Ms. S.S. Kanse Pranita* [2016]**

Important case for providing better care to protect the crop is proper identification of the disease Spatial FCM & PNN classifier image processing gives the best result in identifying the type of disease in cotton plants.
In which the acquisitionof images is carried out. A median filter is used to obtain pre-

processing later. The preprocessed images of the leaf are then segmented using the clustering app roach of Spatial FCM.Then the color features texture features such as power, entropy, similarity, contrast, edges are extracted from the diseased image of the leaf using repeated texture configura tion and then compared to regular image of cotton leaf.



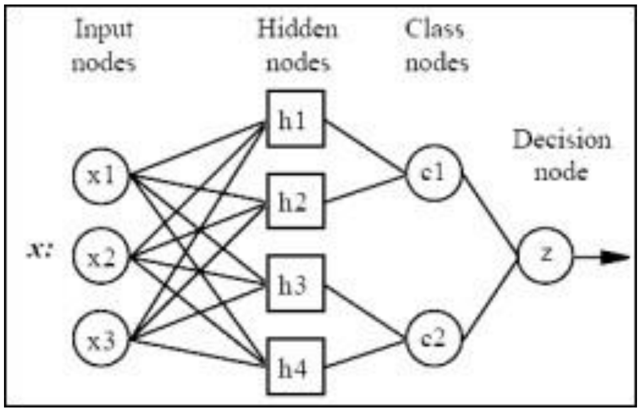


Figure 2.8 Processing Techniques Workflow and system overview

This research paper was used to analyze some technique for the identification of cotton plant leaf disease Normal human eye on cotton leaf can not discern differences in color and texture. The software can be used to capture color and texture characteristicsAnother software used will compare with the trained dat abase and can be identified according to this disease.
Technology for image processing used to deploy the classification system. Thus, image processing with spatial fcm and pnn used to automatically detect and classify the diseases in the system.

[2.8] Detection of plant diseases using Leaf patterns.

A. Ranjith Ram, Vishnu S. **[2015]**

We discuss the different methodologies for the detection of plant disease in this review paper. St udies show that relying on experts ' pure nakedeye observation to diagnose and identify diseases can be time-consuming and expensive, particularly in rural areas and developing countries.
So, we present a solution based on quick, automated, cheap and precise image processing.

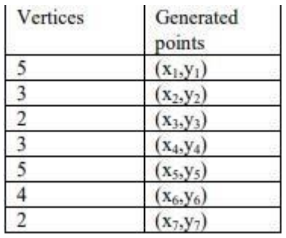
This paper discusses and describes the methods of image processing used to classify plant diseas es. The primary plant disease identification strategies are: BPNN, SVM

The analysis indicates that this technique of disease detection shows a good potential with the abi lity to detect diseases of the plant leaf and certain limitations. Therefore, in existing research, the re is room for change.

[2.9] Rice Leaf Disease Detection Using Chaos and Fractal Image Processing

V.Surendrababu, Dr.C.P.Sumathi, E.Umapathe **[2017]**

In this paper, the specialists saw some significant research issues in bedlam and fractal measurements in picture handling for rice leaf illness location. Due to self-likeness, the fractal for the leaf example of the sick leaf at the last stage will have a similar example for each type of rice leaf ailment during the underlying stage. The reproduction of the fractal would in this way permit the malady to be distinguished early. The procedures of reproducing the fractal from impact malady influenced leaf designs through fractal measurement and tumult hypothesis to be done similarly as different sorts of leaf infections.



Vertices	Generated points
5	(x_1,y_1)
3	(x_2,y_2)
2	(x_3,y_3)
3	(x_4,y_4)
5	(x_5,y_5)
4	(x_6,y_6)
2	(x_7,y_7)

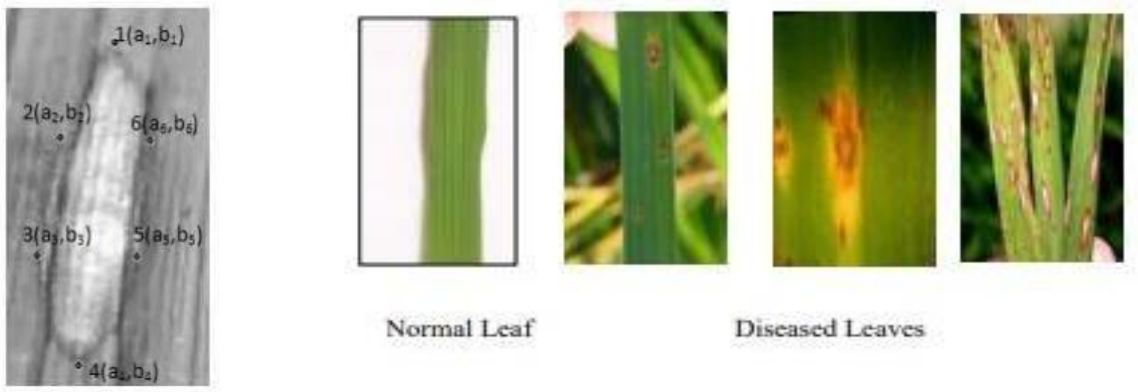


Figure 2.9 Leaf Disease classification based on feature points

Along these lines for each kind of infection that has been distinguished, there is characterization. Besides, the proposed strategies and results can be of incredible intrigue and give setting to future research in the field of mayhem and fractal measurement of picture preparing for different applications.

[2.10] Detection of the affected area of the plant leaf using image processing technology

Vijai Singh, A K Misra, Varsha **[2014]**

The image processing approach is used in this paper to achieve accuracy in plant leaf disease detection. In this article, the method of image processing is used to detect plant leaf disease accuracy. Citrus leaf di sease detection is conducted using the clustering algorithm k-mean.

The percentage calculated using region and image properties for the affected area of the leaf. That's the a dvantage to the farmer, reducing the exact amount of pesticides needed to cover the area being infected. There are three critical stages in the experimental work; in the initial stage, the image acquisition throug h which digital camera records the actual plant sample image.

The image is subjected to image preprocessing tools in the next stage, reducing the size and complexity of the image.

For the segmentation process, the precise digital information is used that separates the diseased portion o f the leaf samplesEventually, the region of the segmented portion of the leaf is determined using algorith ms for image processing.

Photo processing: The acquisition of photographs is the first stage involving a digital camera to capture a photo.The images are taken in the garden or fields from various lighting conditions. For current work, a total of 100 images of citrus-diseased leaf were collected.

The captured citrusdiseased leaf image is shown in Fig.1(a) Preprocessing: eliminate the undesirable dist ortion of these images in the preprocessing of the image.

It is reshaped to the size of 256 pixels and enhances the contrast of some images.

Fig.shows a resized citrus leaf image and an enhanced contrast image, resulting in the image being used for further operations such as creating clusters in the segmentation process.

The best way to overcome growing plant diseases through digital images combined with image processi ng, pattern recognition, classification techniques.

Image Processing

The most ideal approach to defeat developing plant illnesses through computerized pictures joined with and efficiency. The significant kinds of contagious, viral, and bacterial-based plant illnesses. Seen some broad conditions are yellow and earthy colored spots, early and late burning, and so on. Investigating and identifying the various ailments of plant leaf through accessible catching gadgets and computerized picture preparing and getting an upgraded picture or giving the data**.**

The database contains an enormous array of pictures of safe and polluted leaves in either indigenous or f oreign repositories. Photographs are in love with a traditional camera.

An image generally constitutes of three channels RGB which are red green and blue.

Up to the current finish, we tend to perform a preprocessing step wherever each image in our dataset is r esized to 60 or 60 pixels and regenerates to a gray scale.

Deep-Learning-Based Classification

The neural network is made up of several layered neurons. The neurons are interconnected in adjacent la yers. These neurons learn to convert inputs to corresponding outputs (labels) (pre-extracted and pre-processed features).

The Conventional Neural Network comprises of different layers working together on a set of data to find some patterns and thus create a learning map of some unique features in the process. This type of neural network is trained multiple times over thousands of piles of

data.CNN consists of three main parts that are layers to be defined, pooled and totally linked. The convo lution and pooling layers' act as feature extractors from the images of the input while the fully connecte d layer acts as the classifier.

Determination is the primary objective of removing features from each source image automatically. The pooling layer reduces the mobility of these features.

Fully attached to the SoftMax activation mechanism at the end of the template, the trained high-level features are used to group the input objects into predefined classes.

2.11 Tabular comparison of techniques used in the past recent years *[Table 2.1]*

Year	**Contributors / Authors**	**Technique Used**	**Accuracy**
2019	Raj Priyanka, Dr. Neha Mangla,	Canny edge detection algorithm and CBIR	**92.3%**
2019	Jiangsheng Gui , Mor Mbaye	CNN Network	**90.5%**
2018	Jihen Amara, Bassem Bouaziz, Alsayed Algerawy	Deep learning	**83%**
2018	Rafel C.Gonzalez and Richards E.Woods	Classification of image processing techniques	**89%**
2017	M. Akila, P. Deepan(Assistant Professor)	Histogram approach	**71.2%**
2016	Dr. Neha Mangla, Priyanka B Raj, Soumya G Hegde, Pooja R	Canny edge detection algorithm and CBIR	**92.3%**
2016	Sharayu S. Tambe , Gulve Pranita	PNN and extract color and texture feature	**86%**
2015	Vishnu S, A. Ranjith Ram	BPNN, SVM to detect plant leaf disease	**73%**
2015	V.Surendrababu, Dr.C.P.Sumathi, E.Umapathy	Chaos and fractal dimension	**81.2%**

2014	Vijai Singh, Varsha, A K Misra	Capture, resize and enhancing contrast	**71.6%**

CHAPTER 3
SYSTEM DEVELOPMENT

3.1 Dataset Collection

The first priority when training a model is to create an accurate dataset. Our compilation consists of various images of diverse flora. In the development of this, we prioritized some of the commercial/cash grain crops, cereal crops, vegetable crops and fruit plants such as corn maize and potato. Infected bacterial or viral leaves, healthy leaves all of them were collected from diverse range of sources like images download from the Internet, or simply taking pictures using any camera device.

This specific model is optimized for corn(maize) plant.

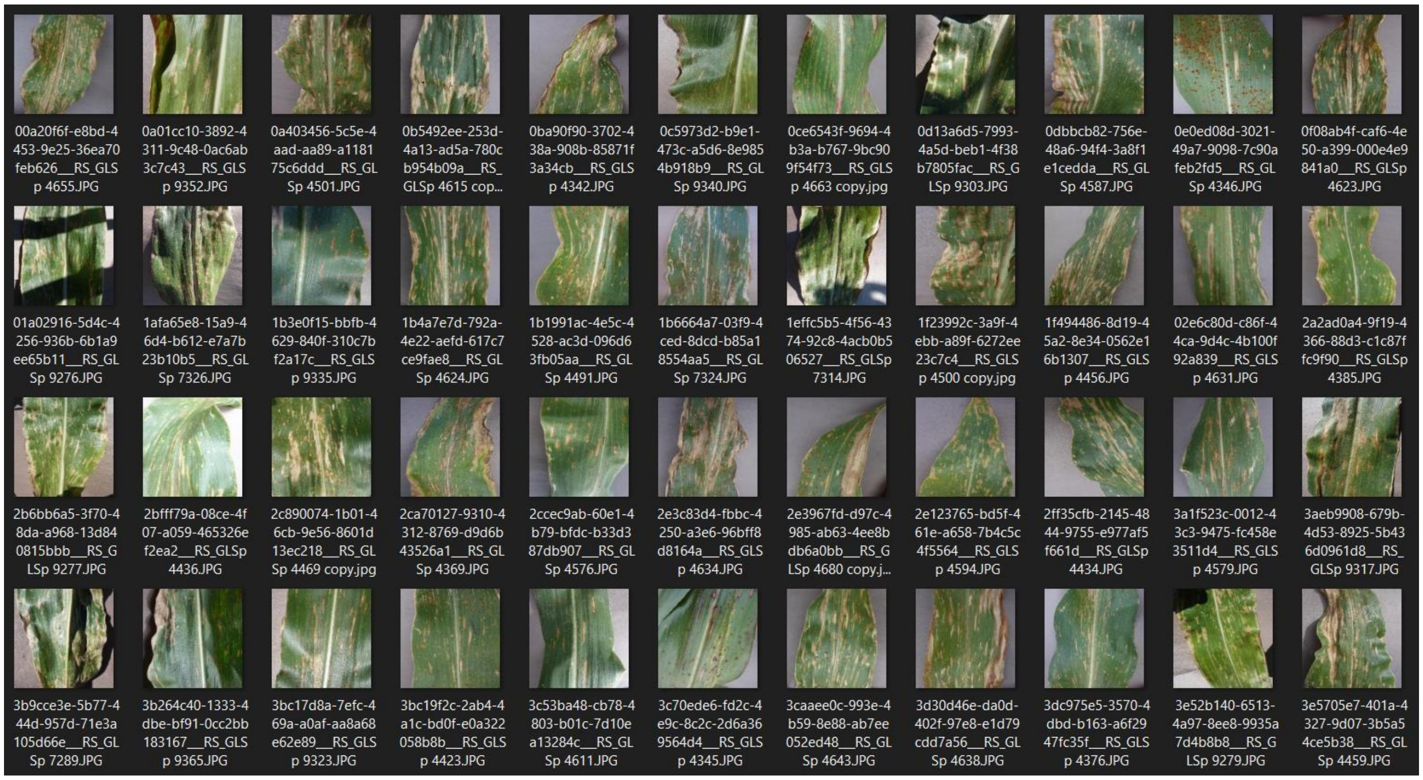
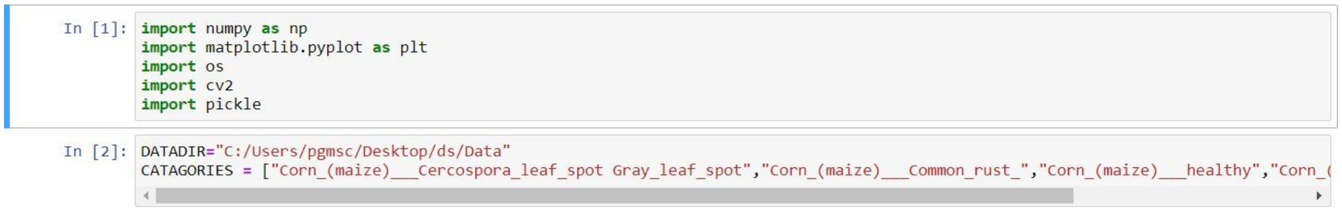

00a20f6f-e8bd-4453-9e25-36ea70feb626__RS_GLSp 4655.JPG
0a01cc10-3892-4311-9c48-0ac6ab3c7c43__RS_GLSp 9352.JPG
0a403456-5c5e-4aad-aa89-a118175c6ddd__RS_GLSp 4501.JPG
0b5492ee-253d-4a13-ad5a-780cb954b09a__RS_GLSp 4615 cop...
0ba90f90-3702-438a-908b-85871f3a34cb__RS_GLSp 4342.JPG
0c5973d2-b9e1-473c-a5d6-8e9854b918b9__RS_GLSp 9340.JPG
0ce6543f-9694-4b3a-b767-9bc909f54f73__RS_GLSp 4663 copy.jpg
0d13a6d5-7993-4a5d-beb1-4f38b7805fac__RS_GLSp 9303.JPG
0dbbcb82-756e-48a6-94f4-3a8f1e1cedda__RS_GLSp 4587.JPG
0e0ed08d-3021-49a7-9098-7c90afeb2fd5__RS_GLSp 4346.JPG
0f08ab4f-caf6-4e50-a399-000e4e9841a0__RS_GLSp 4623.JPG
01a02916-5d4c-4256-936b-6b1a9ee65b11__RS_GLSp 9276.JPG
1afa65e8-15a9-46d4-b612-e7a7b23b10b5__RS_GLSp 7326.JPG
1b3e0f15-bbfb-4629-840f-310c7bf2a17c__RS_GLSp 9335.JPG
1b4a7e7d-792a-4e22-aefd-617c7ce9fae8__RS_GLSp 4624.JPG
1b1991ac-4e5c-4528-ac3d-096d63fb05aa__RS_GLSp 4491.JPG
1b6664a7-03f9-4ced-8dcd-b85a18554aa5__RS_GLSp 7324.JPG
1effc5b5-4f56-4374-92c8-4acb0b506527__RS_GLSp 7314.JPG
1f23992c-3a9f-4ebb-a89f-6272ee23c7c4__RS_GLSp 4500 copy.jpg
1f494486-8d19-45a2-8e34-0562e16b1307__RS_GLSp 4456.JPG
02e6c80d-c86f-44ca-9d4c-4b100f92a839__RS_GLSp 4631.JPG
2a2ad0a4-9f19-4366-88d3-c1c87ffc9f90__RS_GLSp 4385.JPG
2b6bb6a5-3f70-48da-a968-13d840815bbb__RS_GLSp 9277.JPG
2bfff79a-08ce-4f07-a059-465326ef2ea2__RS_GLSp 4436.JPG
2c890074-1b01-46cb-9e56-8601d13ec218__RS_GLSp 4469 copy.jpg
2ca70127-9310-4312-8769-d9d6b43526a1__RS_GLSp 4369.JPG
2ccec9ab-60e1-4b79-bfdc-b33d387db907__RS_GLSp 4576.JPG
2e3c83d4-fbbc-4250-a3e6-96bff8d8164a__RS_GLSp 4634.JPG
2e3967fd-d97c-4985-ab63-4ee8bdb6a0bb__RS_GLSp 4680 copy.j...
2e123765-bd5f-461e-a658-7b4c5c4f5564__RS_GLSp 4594.JPG
2ff35cfb-2145-4844-9755-e977af5f661d__RS_GLSp 4434.JPG
3a1f523c-0012-43c3-9475-fc458e3511d4__RS_GLSp 4579.JPG
3aeb9908-679b-4d53-8925-5b436d0961d8__RS_GLSp 9317.JPG
3b9cce3e-5b77-444d-957d-71e3a105d66e__RS_GLSp 7289.JPG
3b264c40-1333-4dbe-bf91-0cc2bb183167__RS_GLSp 9365.JPG
3bc17d8a-7efc-469a-a0af-aa8a68e62e89__RS_GLSp 9323.JPG
3bc19f2c-2ab4-4a1c-bd0f-e0a322058b8b__RS_GLSp 4423.JPG
3c53ba48-cb78-4803-b01c-7d10ea13284c__RS_GLSp 4611.JPG
3c70ede6-fd2c-4e9c-8c2c-2d6a369564d4__RS_GLSp 4345.JPG
3caaee0c-993e-4b59-8e88-ab7ee052ed48__RS_GLSp 4643.JPG
3d30d46e-da0d-402f-97e8-e1d79cdd7a56__RS_GLSp 4638.JPG
3dc975e5-3570-4dbd-b163-a6f2947fc35f__RS_GLSp 4376.JPG
3e52b140-6513-4a97-8ee8-9935a7d4b8b8__RS_GLSp 9279.JPG
3e5705e7-401a-4327-9d07-3b5a54ce5b38__RS_GLSp 4459.JPG

Figure 3.1 Infected leaves samples

3.1.1 Creating the dataset:

Here we are going to use the collected Plant leaf Dataset, which contains 80,000 RGB images in 8 categories. We will use 65000 for training and the rest 15000 for testing purposes. At first, we will import and then load the data to feed it into the CNN network.

- Importing the libraries and loading the data:



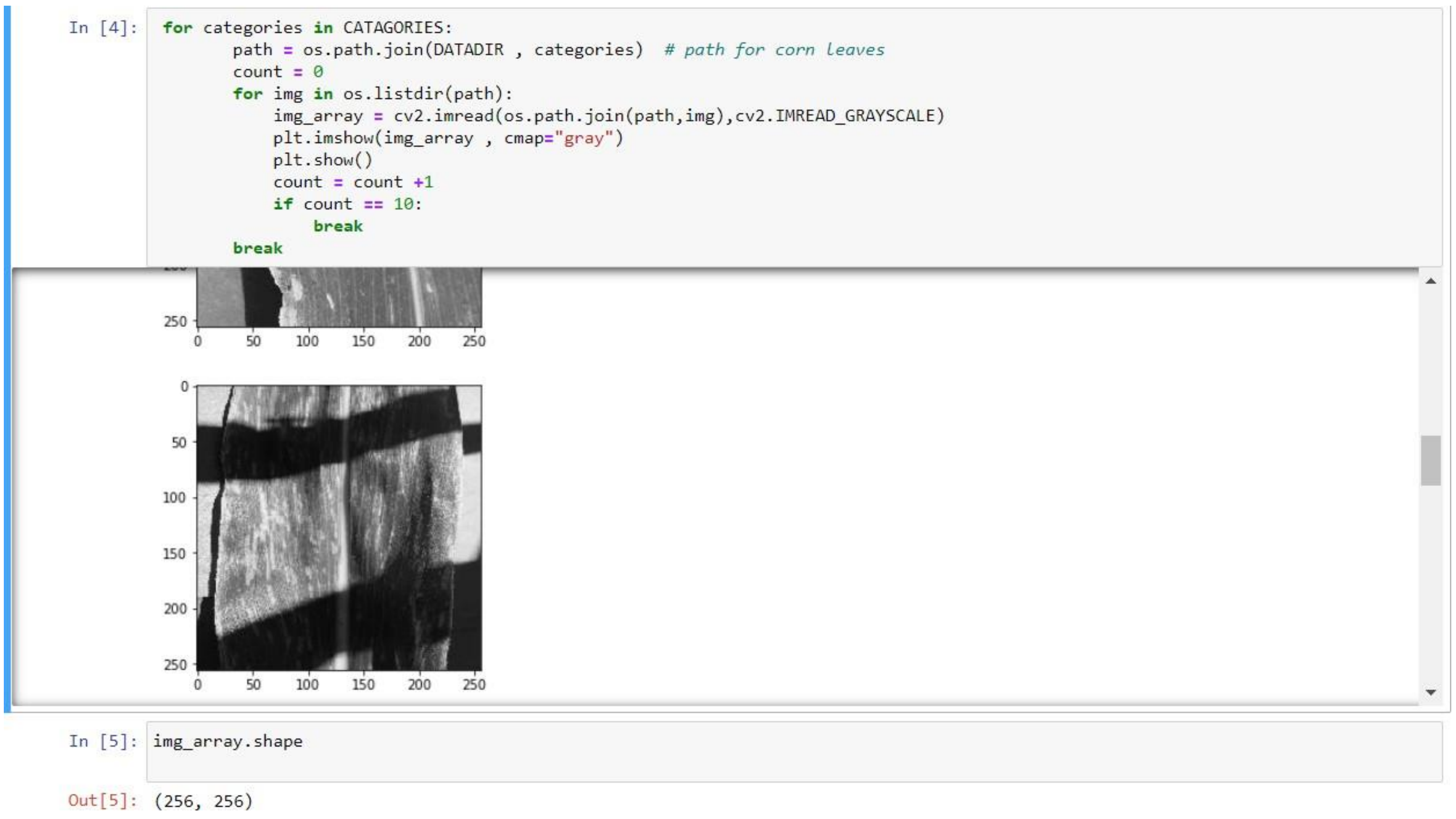
```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import os
        import cv2
        import pickle

In [2]: DATADIR="C:/Users/pgmsc/Desktop/ds/Data"
        CATAGORIES = ["Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot","Corn_(maize)___Common_rust_","Corn_(maize)___healthy","Corn_(
```

Figure 3.2 Installing & Importing required dataset

Here we have loaded the data and mapped the images into various classes for the classification.

- Pre-processing the data for computation:

  We scaled and resized the images thus creating a workable training dataset. The loaded images are then displayed with applied transformations on them.

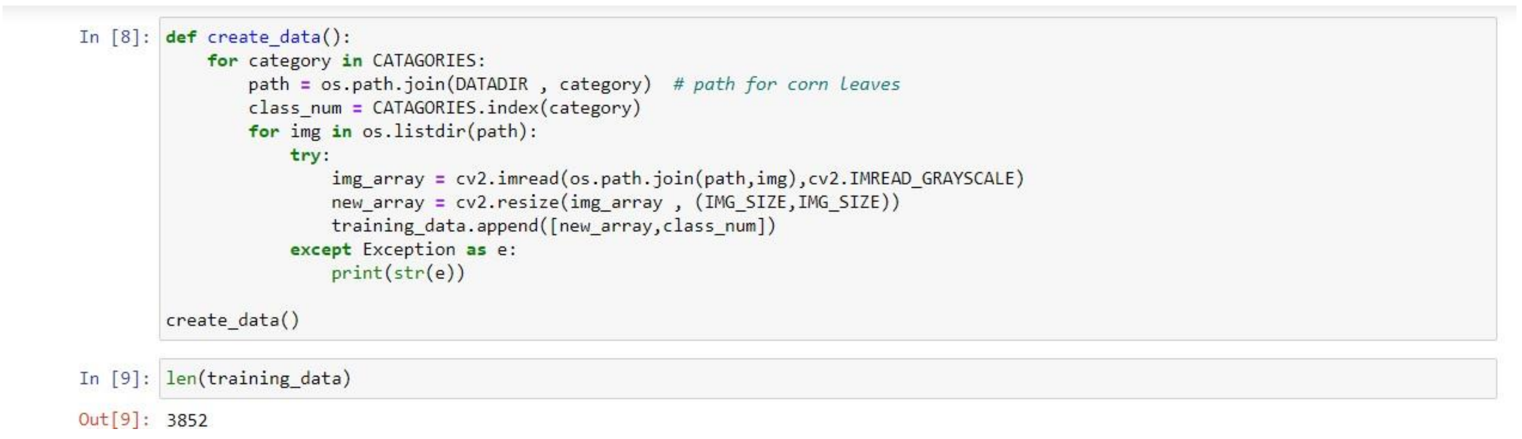


```
In [4]:  for categories in CATAGORIES:
                 path = os.path.join(DATADIR , categories)  # path for corn leaves
                 count = 0
                 for img in os.listdir(path):
                     img_array = cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
                     plt.imshow(img_array , cmap="gray")
                     plt.show()
                     count = count +1
                     if count == 10:
                         break
                 break
```

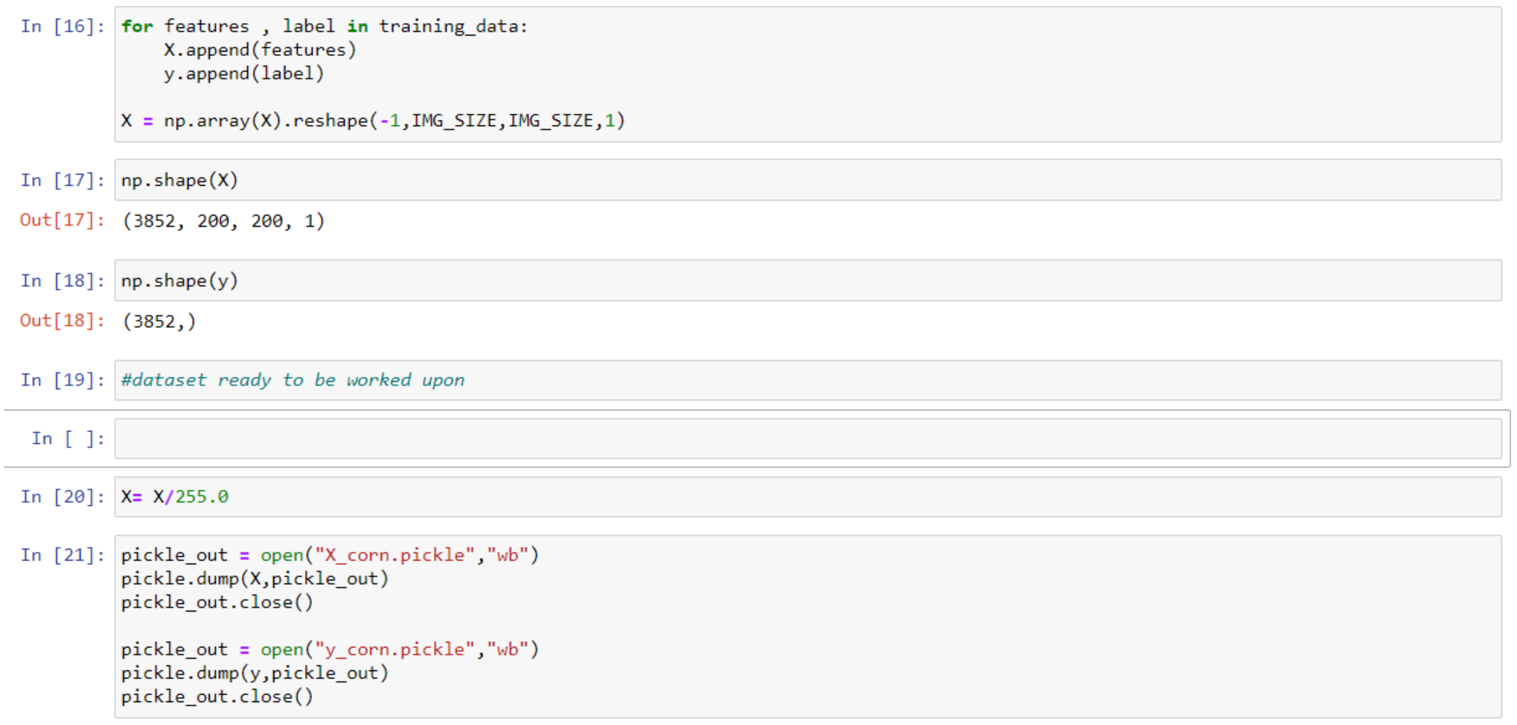
```
In [5]:  img_array.shape

Out[5]:  (256, 256)
```

Figure 3.3 Images loaded are displayed at random

To make the computation more discernible we convert the three channel RGB pictures into Grayscale. Here we observe that the images loaded from the training set have size 256*256. It shows that the pixel values will fall in the range of 0 to 255.



```
In [8]:  def create_data():
             for category in CATAGORIES:
                 path = os.path.join(DATADIR , category)  # path for corn leaves
                 class_num = CATAGORIES.index(category)
                 for img in os.listdir(path):
                     try:
                         img_array = cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
                         new_array = cv2.resize(img_array , (IMG_SIZE,IMG_SIZE))
                         training_data.append([new_array,class_num])
                     except Exception as e:
                         print(str(e))

         create_data()

In [9]:  len(training_data)

Out[9]:  3852
```



```
In [16]: for features , label in training_data:
             X.append(features)
             y.append(label)

         X = np.array(X).reshape(-1,IMG_SIZE,IMG_SIZE,1)

In [17]: np.shape(X)

Out[17]: (3852, 200, 200, 1)

In [18]: np.shape(y)

Out[18]: (3852,)

In [19]: #dataset ready to be worked upon

In [ ]:

In [20]: X= X/255.0

In [21]: pickle_out = open("X_corn.pickle","wb")
         pickle.dump(X,pickle_out)
         pickle_out.close()

         pickle_out = open("y_corn.pickle","wb")
         pickle.dump(y,pickle_out)
         pickle_out.close()
```

Figure 3.4 Applying transformations and filters

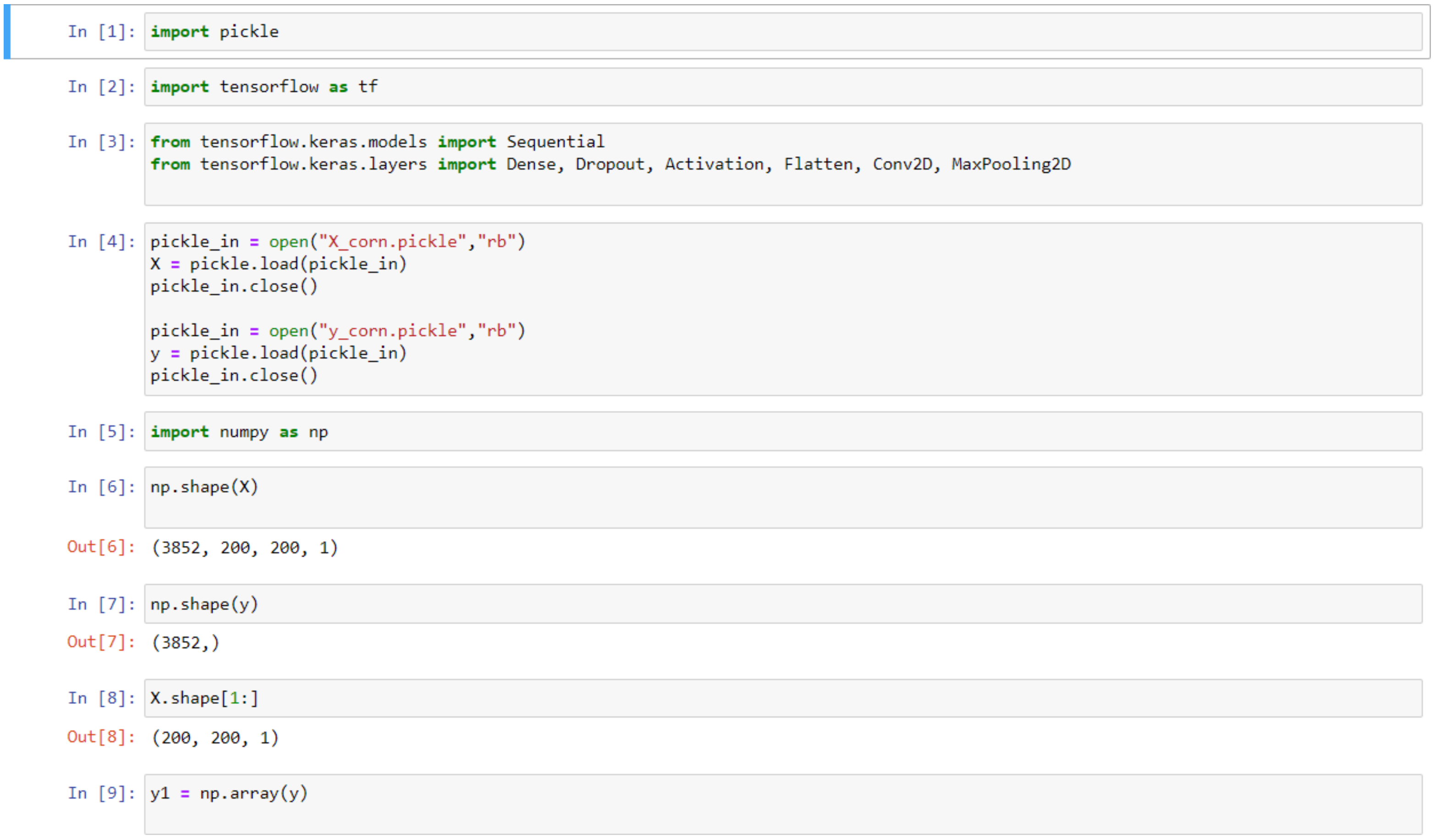
Here we displayed the images with the applied transformations for the computation. Now we scale the images into the range 0-1 for further feeding into the neural network.

3.2 Building the Neural Network Architecture:

In order to create the model, we will be feeding our system with the data set optimized for this computation. This dataset is split into two distinguishable parts; feature set and label set. The feature set consists of the array of images and the label tags consists of the class for the respective image. Building the neural network requires configuring the layers of the model, then compiling the model. In this deep learning model, we will chain together these layers multiple times for implanting the prediction.

The neural network is made up of several layered neurons. The neurons are interconnected in adjacent la yers. These neurons learn to convert inputs to corresponding outputs (labels) (pre-extracted and pre-processed features).
The Conventional Neural Network comprises of different layers working together on a set of data to find some patterns and thus create a learning map of some unique features in the process. This type of neural network is trained multiple times over thousands of piles of
data.CNN consists of three main parts that are layers to be defined, pooled and totally linked. The convo lution and pooling layers' act as feature extractors from the images of the input while the fully connecte d layer acts as the classifier.

Now we import the required libraries for building the model as the input image is reshaped into a matrix of 200*200*1

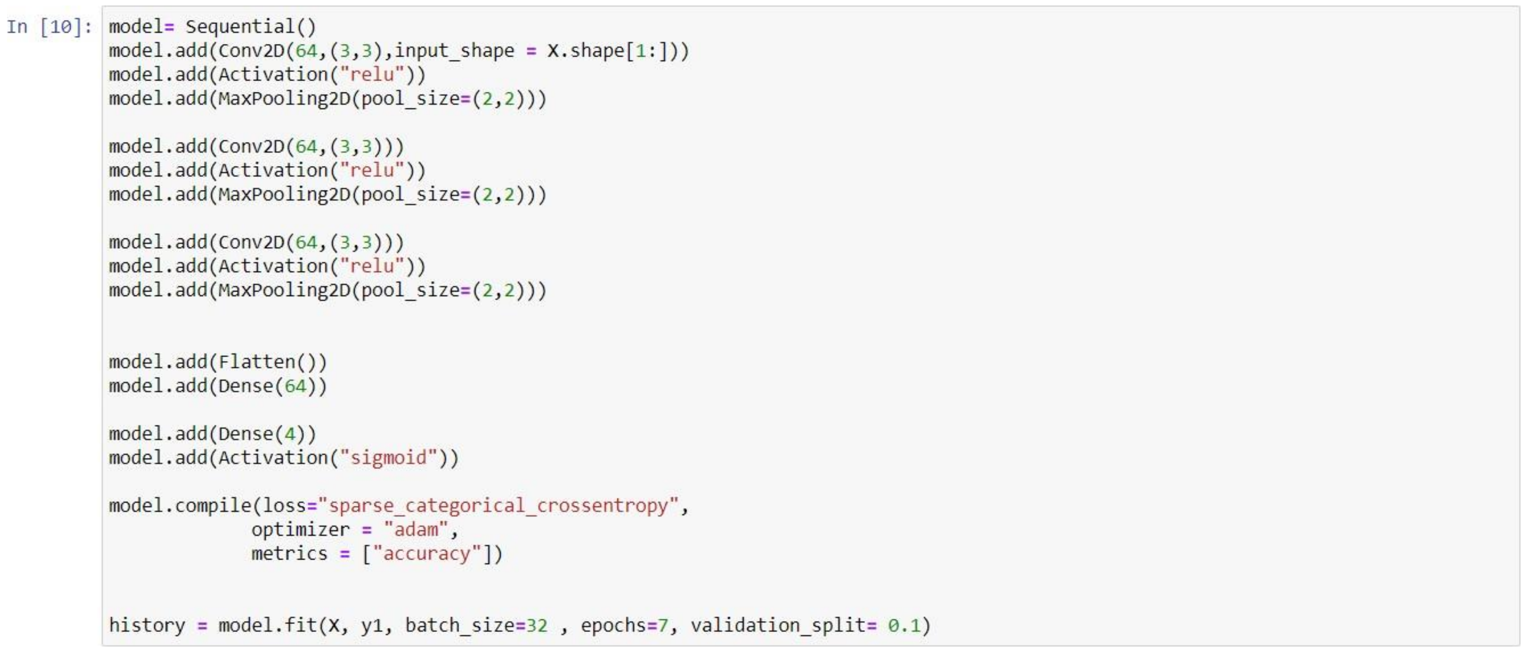


```
In [1]: import pickle

In [2]: import tensorflow as tf

In [3]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D

In [4]: pickle_in = open("X_corn.pickle","rb")
        X = pickle.load(pickle_in)
        pickle_in.close()

        pickle_in = open("y_corn.pickle","rb")
        y = pickle.load(pickle_in)
        pickle_in.close()

In [5]: import numpy as np

In [6]: np.shape(X)

Out[6]: (3852, 200, 200, 1)

In [7]: np.shape(y)

Out[7]: (3852,)

In [8]: X.shape[1:]

Out[8]: (200, 200, 1)

In [9]: y1 = np.array(y)
```

Figure 3.5 Rechecking for the transformed dimensions

Now we defined the convolution layer with 64 filters and a stride of 3 and activation function set as "relu" followed by a have pooling layer i.e. "max pool" layer. The same process is repeated with different parameters. After that we are using an activation layer as 'sigmoid'. We are using the "adam" optimizer to optimize the model and thus, reduced the overall loss.

The neural network is made up of several layered neurons. The neurons are interconnected in adjacent la yers. These neurons learn to convert inputs to corresponding outputs (labels) (pre-extracted and pre-processed features).
The Conventional Neural Network comprises of different layers working together on a set of data to find some patterns and thus create a learning map of some unique features in the process. This type of neural network is trained multiple times over thousands of piles of
data.CNN consists of three main parts that are layers to be defined, pooled and totally linked. The convo lution and pooling layers' act as feature extractors from the images of the input while the fully connecte d layer acts as the classifier.

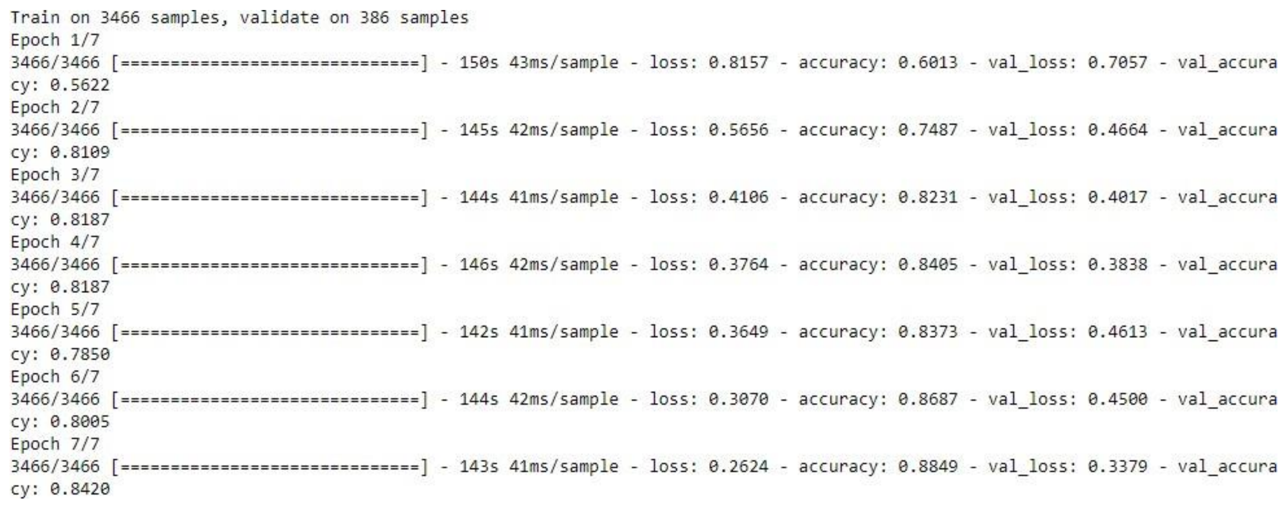


```
In [10]: model= Sequential()
         model.add(Conv2D(64,(3,3),input_shape = X.shape[1:]))
         model.add(Activation("relu"))
         model.add(MaxPooling2D(pool_size=(2,2)))

         model.add(Conv2D(64,(3,3)))
         model.add(Activation("relu"))
         model.add(MaxPooling2D(pool_size=(2,2)))

         model.add(Conv2D(64,(3,3)))
         model.add(Activation("relu"))
         model.add(MaxPooling2D(pool_size=(2,2)))


         model.add(Flatten())
         model.add(Dense(64))

         model.add(Dense(4))
         model.add(Activation("sigmoid"))

         model.compile(loss="sparse_categorical_crossentropy",
                      optimizer = "adam",
                      metrics = ["accuracy"])


         history = model.fit(X, y1, batch_size=32 , epochs=7, validation_split= 0.1)
```

Figure 3.6 Building the CNN model

Consequently, in order to start the training of our model, model.fit function method is used; as the name suggests this method literally "fits" the model onto the given training data. Here we developed three models thereby increasing prediction accuracy with each iteration.

The first model will run for 7 iterations:



```
Train on 3466 samples, validate on 386 samples
Epoch 1/7
3466/3466 [==============================] - 150s 43ms/sample - loss: 0.8157 - accuracy: 0.6013 - val_loss: 0.7057 - val_accura
cy: 0.5622
Epoch 2/7
3466/3466 [==============================] - 145s 42ms/sample - loss: 0.5656 - accuracy: 0.7487 - val_loss: 0.4664 - val_accura
cy: 0.8109
Epoch 3/7
3466/3466 [==============================] - 144s 41ms/sample - loss: 0.4106 - accuracy: 0.8231 - val_loss: 0.4017 - val_accura
cy: 0.8187
Epoch 4/7
3466/3466 [==============================] - 146s 42ms/sample - loss: 0.3764 - accuracy: 0.8405 - val_loss: 0.3838 - val_accura
cy: 0.8187
Epoch 5/7
3466/3466 [==============================] - 142s 41ms/sample - loss: 0.3649 - accuracy: 0.8373 - val_loss: 0.4613 - val_accura
cy: 0.7850
Epoch 6/7
3466/3466 [==============================] - 144s 42ms/sample - loss: 0.3070 - accuracy: 0.8687 - val_loss: 0.4500 - val_accura
cy: 0.8005
Epoch 7/7
3466/3466 [==============================] - 143s 41ms/sample - loss: 0.2624 - accuracy: 0.8849 - val_loss: 0.3379 - val_accura
cy: 0.8420
```

Figure 3.7 Model In training phase

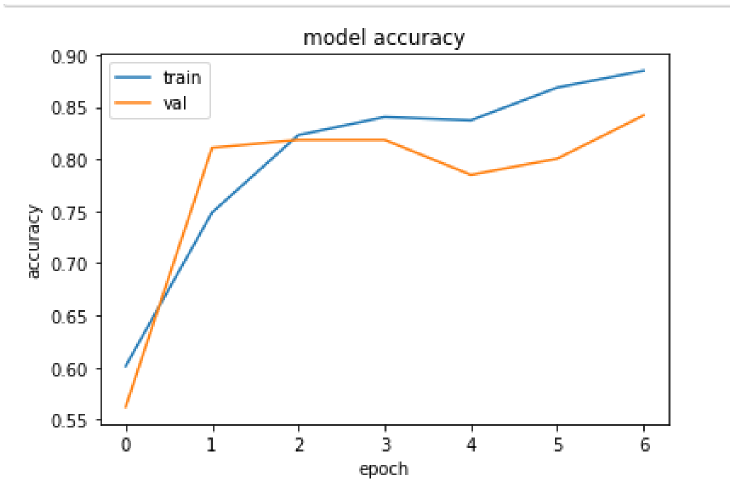
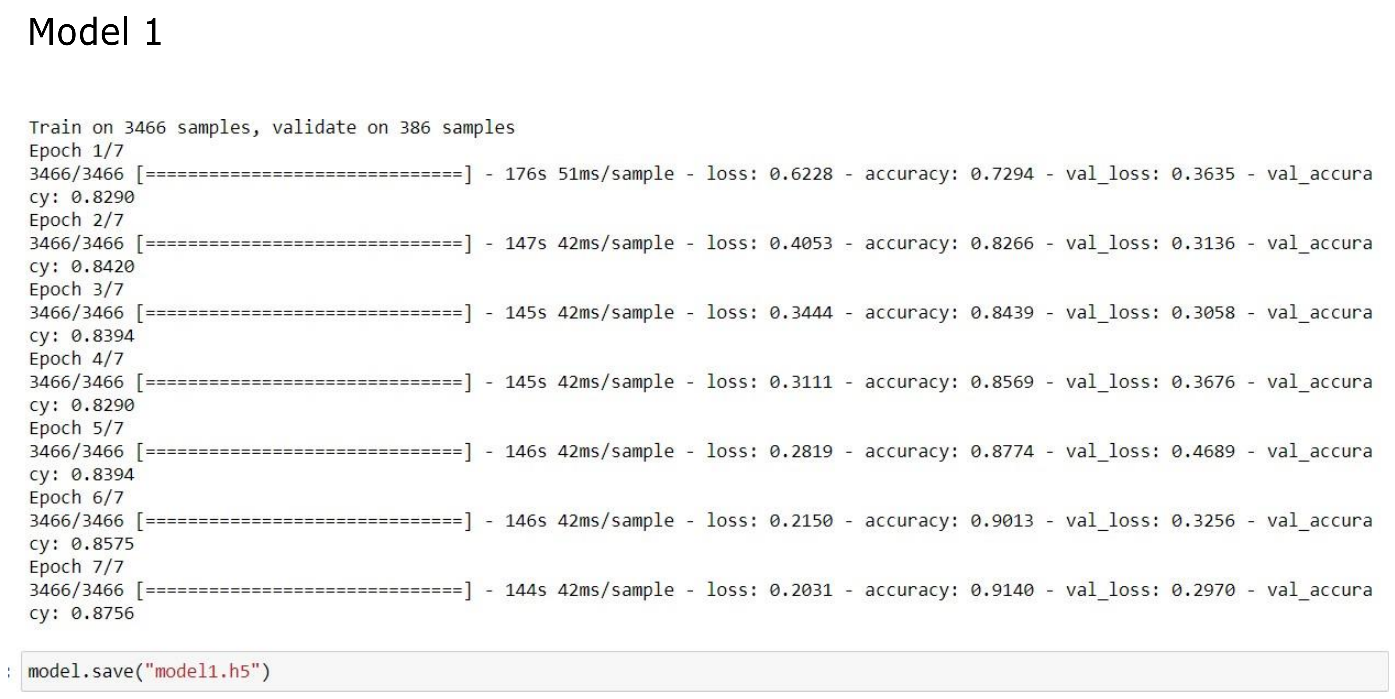


Figure 3.8 Model 0 Training vs Validation Accuracy

After the model is done training the loss and prediction accuracy metrics are also displayed. This model achieves an accuracy of 88% and 84% on validation data. All this data is then visualized into a graphical from thus making it easy to compare different models. Now, if you train your neural network for more epochs or change the activation function, you might get a different result that might have better accuracy.

The neural network is made up of several layered neurons. The neurons are interconnected in adjacent la yers. These neurons learn to convert inputs to corresponding outputs (labels) (pre-extracted and pre-processed features).
The Conventional Neural Network comprises of different layers working together on a set of data to find some patterns and thus create a learning map of some unique features in the process. This type of neural network is trained multiple times over thousands of piles of
data.CNN consists of three main parts that are layers to be defined, pooled and totally linked. The convo lution and pooling layers' act as feature extractors from the images of the input while the fully connecte d layer acts as the classifier.

Now to improve the accuracy of these models we will adjust their hyperparameters and tweak with the overall structure the neural network.



Model 1

```
Train on 3466 samples, validate on 386 samples
Epoch 1/7
3466/3466 [==============================] - 176s 51ms/sample - loss: 0.6228 - accuracy: 0.7294 - val_loss: 0.3635 - val_accura
cy: 0.8290
Epoch 2/7
3466/3466 [==============================] - 147s 42ms/sample - loss: 0.4053 - accuracy: 0.8266 - val_loss: 0.3136 - val_accura
cy: 0.8420
Epoch 3/7
3466/3466 [==============================] - 145s 42ms/sample - loss: 0.3444 - accuracy: 0.8439 - val_loss: 0.3058 - val_accura
cy: 0.8394
Epoch 4/7
3466/3466 [==============================] - 145s 42ms/sample - loss: 0.3111 - accuracy: 0.8569 - val_loss: 0.3676 - val_accura
cy: 0.8290
Epoch 5/7
3466/3466 [==============================] - 146s 42ms/sample - loss: 0.2819 - accuracy: 0.8774 - val_loss: 0.4689 - val_accura
cy: 0.8394
Epoch 6/7
3466/3466 [==============================] - 146s 42ms/sample - loss: 0.2150 - accuracy: 0.9013 - val_loss: 0.3256 - val_accura
cy: 0.8575
Epoch 7/7
3466/3466 [==============================] - 144s 42ms/sample - loss: 0.2031 - accuracy: 0.9140 - val_loss: 0.2970 - val_accura
cy: 0.8756
```

```
model.save("model1.h5")
```

Figure 3.9 Model 1 Training

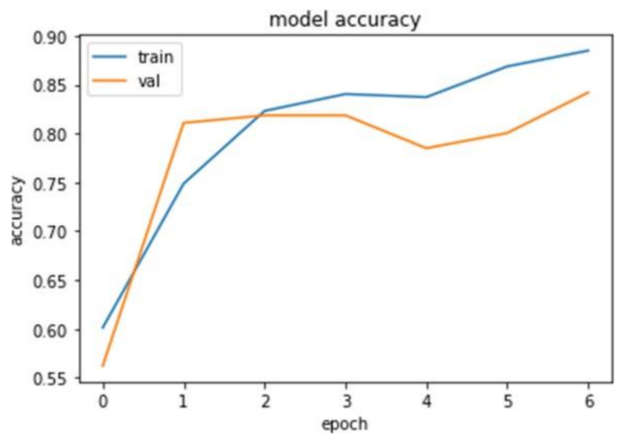
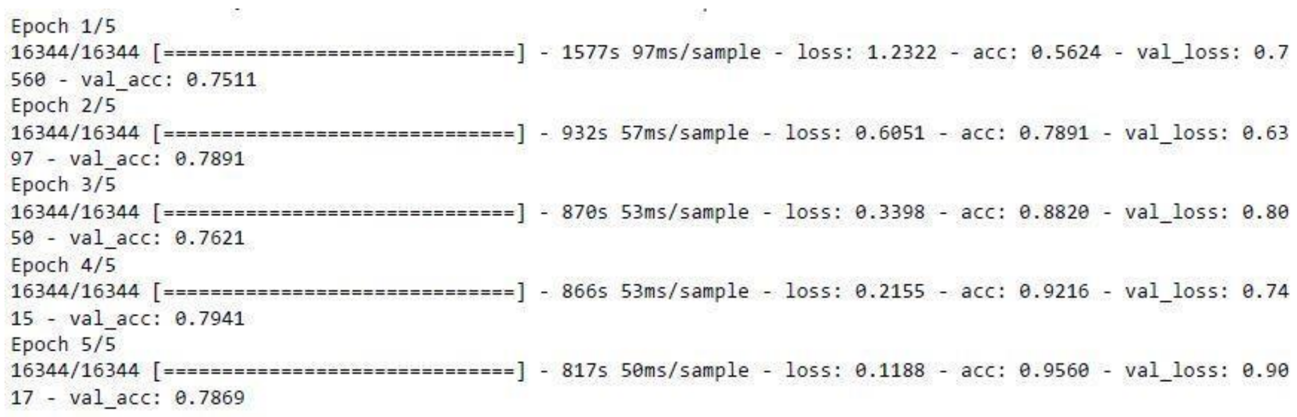


Figure 3.10 Model 1 Training vs Validation Accuracy

Model 2



```
Epoch 1/5
16344/16344 [==============================] - 1577s 97ms/sample - loss: 1.2322 - acc: 0.5624 - val_loss: 0.7
560 - val_acc: 0.7511
Epoch 2/5
16344/16344 [==============================] - 932s 57ms/sample - loss: 0.6051 - acc: 0.7891 - val_loss: 0.63
97 - val_acc: 0.7891
Epoch 3/5
16344/16344 [==============================] - 870s 53ms/sample - loss: 0.3398 - acc: 0.8820 - val_loss: 0.80
50 - val_acc: 0.7621
Epoch 4/5
16344/16344 [==============================] - 866s 53ms/sample - loss: 0.2155 - acc: 0.9216 - val_loss: 0.74
15 - val_acc: 0.7941
Epoch 5/5
16344/16344 [==============================] - 817s 50ms/sample - loss: 0.1188 - acc: 0.9560 - val_loss: 0.90
17 - val_acc: 0.7869
```

Figure 3.11 Model 2 Training

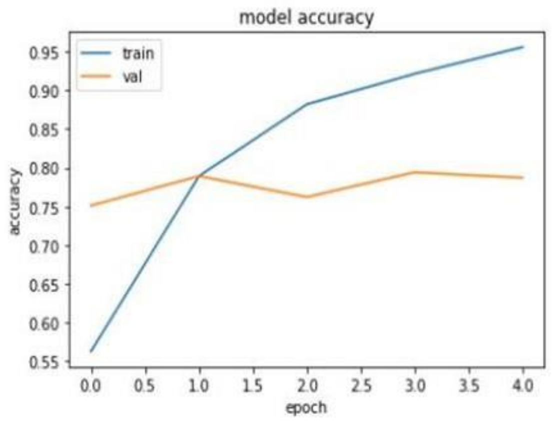
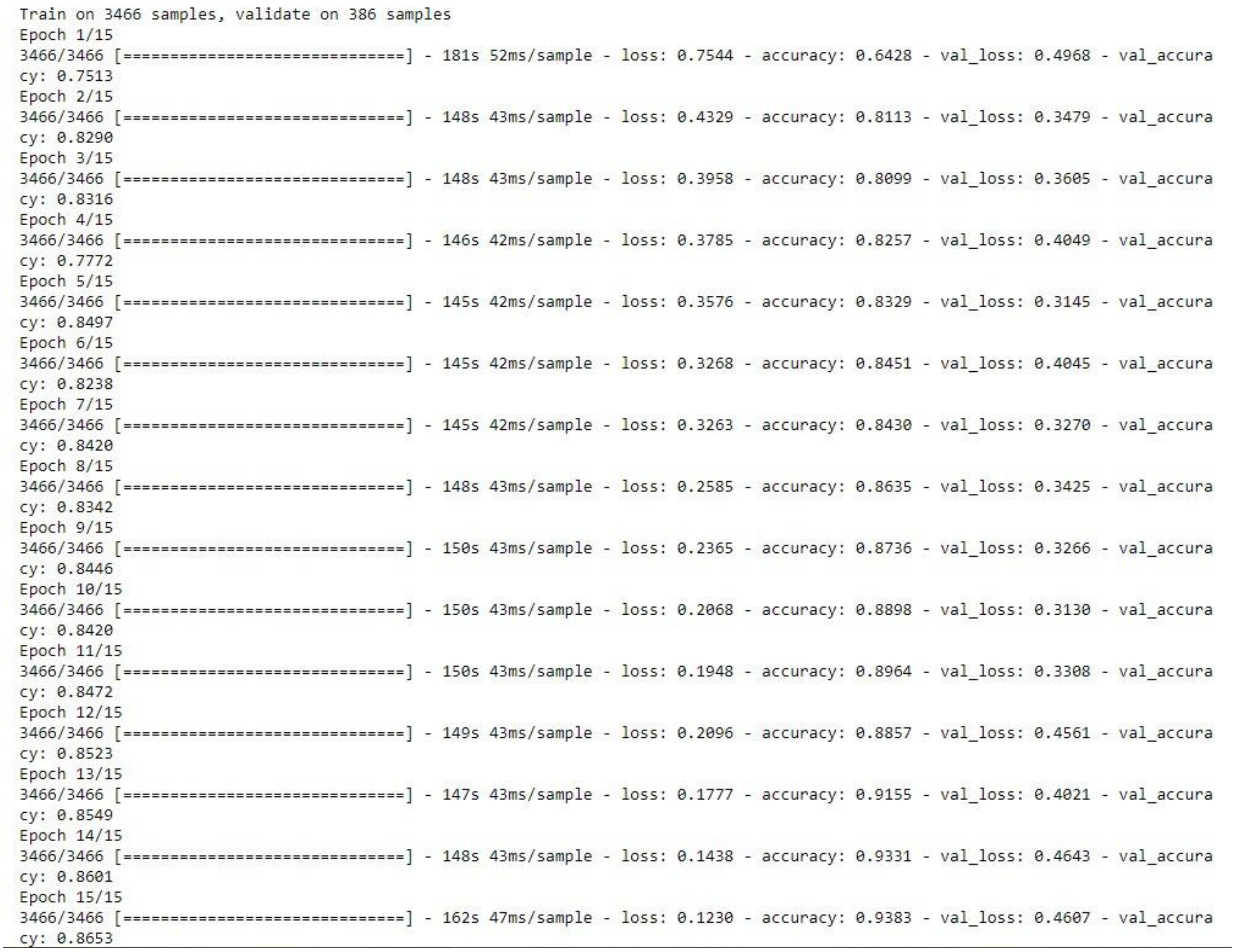


Figure 3.12 Model 2 Training vs Validation Accuracy

It turns out that the accuracy on the test dataset is a little less than the accuracy on the training dataset. This gap between training accuracy and test accuracy represents overfitting. Overfitting happens when a machine learning model performs worse on new, previously unseen inputs than it does on the training data. An overfitted model "memorizes" the noise and details in the training dataset to a point where it negatively impacts the performance of the model on the new data.

Model 3



```
Train on 3466 samples, validate on 386 samples
Epoch 1/15
3466/3466 [==============================] - 181s 52ms/sample - loss: 0.7544 - accuracy: 0.6428 - val_loss: 0.4968 - val_accura
cy: 0.7513
Epoch 2/15
3466/3466 [==============================] - 148s 43ms/sample - loss: 0.4329 - accuracy: 0.8113 - val_loss: 0.3479 - val_accura
cy: 0.8290
Epoch 3/15
3466/3466 [==============================] - 148s 43ms/sample - loss: 0.3958 - accuracy: 0.8099 - val_loss: 0.3605 - val_accura
cy: 0.8316
Epoch 4/15
3466/3466 [==============================] - 146s 42ms/sample - loss: 0.3785 - accuracy: 0.8257 - val_loss: 0.4049 - val_accura
cy: 0.7772
Epoch 5/15
3466/3466 [==============================] - 145s 42ms/sample - loss: 0.3576 - accuracy: 0.8329 - val_loss: 0.3145 - val_accura
cy: 0.8497
Epoch 6/15
3466/3466 [==============================] - 145s 42ms/sample - loss: 0.3268 - accuracy: 0.8451 - val_loss: 0.4045 - val_accura
cy: 0.8238
Epoch 7/15
3466/3466 [==============================] - 145s 42ms/sample - loss: 0.3263 - accuracy: 0.8430 - val_loss: 0.3270 - val_accura
cy: 0.8420
Epoch 8/15
3466/3466 [==============================] - 148s 43ms/sample - loss: 0.2585 - accuracy: 0.8635 - val_loss: 0.3425 - val_accura
cy: 0.8342
Epoch 9/15
3466/3466 [==============================] - 150s 43ms/sample - loss: 0.2365 - accuracy: 0.8736 - val_loss: 0.3266 - val_accura
cy: 0.8446
Epoch 10/15
3466/3466 [==============================] - 150s 43ms/sample - loss: 0.2068 - accuracy: 0.8898 - val_loss: 0.3130 - val_accura
cy: 0.8420
Epoch 11/15
3466/3466 [==============================] - 150s 43ms/sample - loss: 0.1948 - accuracy: 0.8964 - val_loss: 0.3308 - val_accura
cy: 0.8472
Epoch 12/15
3466/3466 [==============================] - 149s 43ms/sample - loss: 0.2096 - accuracy: 0.8857 - val_loss: 0.4561 - val_accura
cy: 0.8523
Epoch 13/15
3466/3466 [==============================] - 147s 43ms/sample - loss: 0.1777 - accuracy: 0.9155 - val_loss: 0.4021 - val_accura
cy: 0.8549
Epoch 14/15
3466/3466 [==============================] - 148s 43ms/sample - loss: 0.1438 - accuracy: 0.9331 - val_loss: 0.4643 - val_accura
cy: 0.8601
Epoch 15/15
3466/3466 [==============================] - 162s 47ms/sample - loss: 0.1230 - accuracy: 0.9383 - val_loss: 0.4607 - val_accura
cy: 0.8653
```

Figure 3.13 Model 3 Training

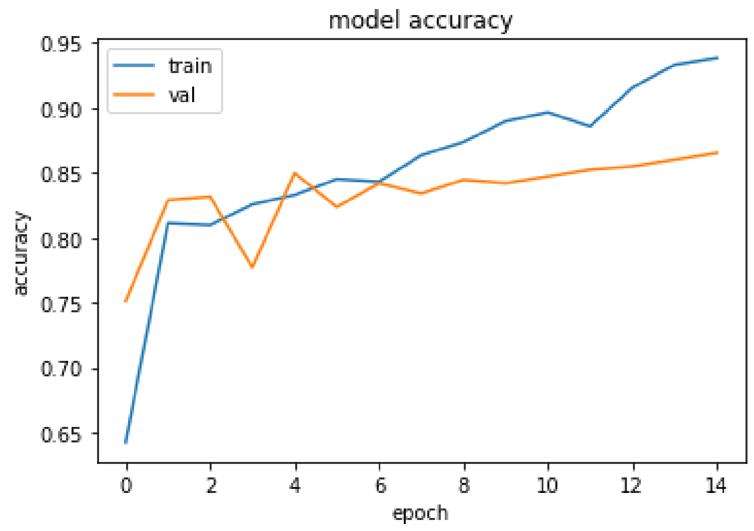


Figure Model 3.14 Training vs Validation Accuracy

Comparison Table: Models developed can be compared one to another with their metrics of final results in training in validation**.**

	Training Precision	Validation Accuracy	Overfitting/ Underfitting
Model 1	53%	50%	---
Model 2	95.2%	77%	Found
Model 3	93.8%	86.5%	Resolved

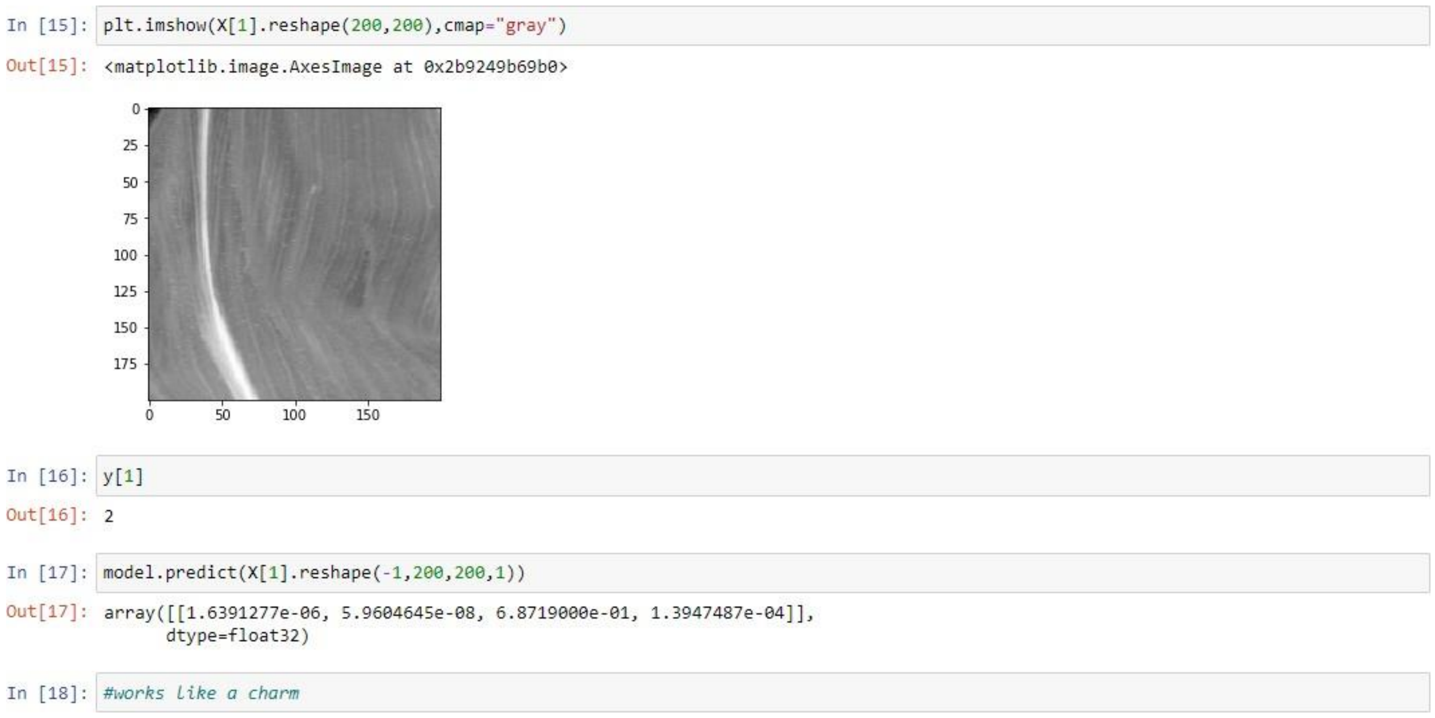
The neural network is made up of several layered neurons. The neurons are interconnected in adjacent la yers. These neurons learn to convert inputs to corresponding outputs (labels) (pre-extracted and pre-processed features).

The Conventional Neural Network comprises of different layers working together on a set of data to find some patterns and thus create a learning map of some unique features in the process. This type of neural network is trained multiple times over thousands of piles of

data.CNN consists of three main parts that are layers to be defined, pooled and totally linked. The convo lution and pooling layers' act as feature extractors from the images of the input while the fully connecte d layer acts as the classifier.

3.3 Making predictions with the trained model

Now that our model is trained, we can start making random predictions by feeding it some raw images. Generally, the CNN network gives out linear outputs known as logits which can't be interpreted. We have added a softmax layer at the end which converts the unfathomable logits to more specific probabilities values. Here, the model has predicted some probability score for each of the categorical label from various features.



```
In [15]:  plt.imshow(X[1].reshape(200,200),cmap="gray")
Out[15]:  <matplotlib.image.AxesImage at 0x2b9249b69b0>
```

```
In [16]:  y[1]
Out[16]:  2

In [17]:  model.predict(X[1].reshape(-1,200,200,1))
Out[17]:  array([[1.6391277e-06, 5.9604645e-08, 6.8719000e-01, 1.3947487e-04]],
                dtype=float32)

In [18]:  #works like a charm
```

Figure 3.15 Prediction values

The result is displayed in terms of arrays witch each element showing a probabilistic score. This score signifies the certainty in the prediction made into the labels. You can see which label has the highest confidence value. Here the trained model is 87% sure that it is a healthy leaf. It can be observed that the classification done by the model into the label is absolutely correct. This means that its working as expected.

CHAPTER 4

HARDWARE USED

ARDUINO

An Arduino is actually a microcontroller based kit which can be either used directly by purchasing from the vendor or can be made at home using the components, owing to its open source hardware feature. It is basically used in communications and in controlling or operating many devices. It was founded by Massimo Banzi and David Cuartielles in 2005.

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

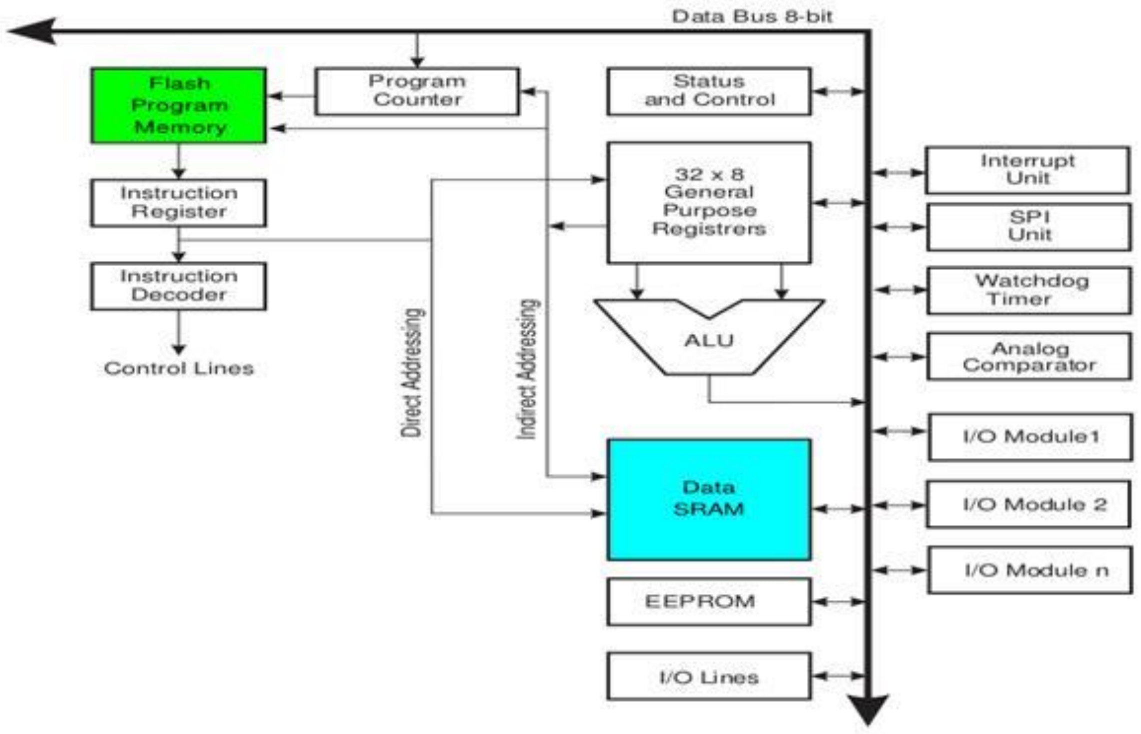
The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward.

The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions.

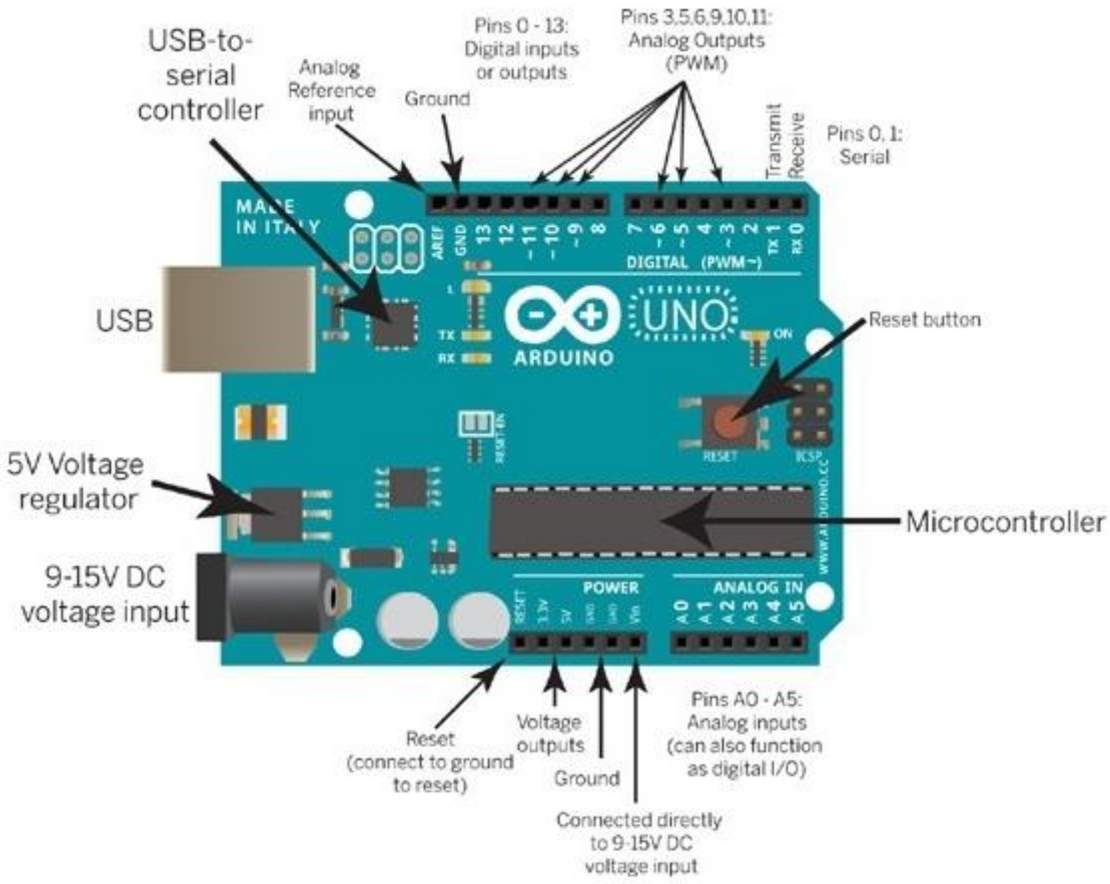
ARDUINO ARCHITECTURE:

Arduino’s processor basically uses the Harvard architecture where the program code and program data have separate memory. It consists of two memories- Program memory and the data memory.The code is stored in the flash program memory, whereas the data is stored in the data memory. The Atmega328 has 32 KB of flash memory for storing code (of which 0.5 KB is used for the bootloader), 2 KB of SRAM and 1 KB of EEPROM and operates with a clock speed of 16MHz.

The most important advantage with Arduino is the programs can be directly loaded to the device without requiring any hardware programmer to burn the program. This is done because of the presence of the 0.5KB of Bootloader which allows the program to be burned into the circuit. All we have to do is to download the Arduino software and writing the code.



ARDUINO PIN DIAGRAM


USB-to-serial controller
Analog Reference input
Ground
Pins 0 - 13: Digital inputs or outputs
Pins 3,5,6,9,10,11: Analog Outputs (PWM)
Transmit
Receive
Pins 0, 1: Serial
USB
Reset button
5V Voltage regulator
Microcontroller
9-15V DC voltage input
Reset (connect to ground to reset)
Voltage outputs
Ground
Connected directly to 9-15V DC voltage input
Pins A0 - A5: Analog inputs (can also function as digital I/O)

Arduino Uno consists of 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button

POWER JACK

Arduino can be power either from the pc through a USB or through external source like adaptor or a battery. It can operate on a external supply of 7 to 12V. Power can be applied externally through the pin Vin or by giving voltage reference through the IORef pin.

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

DIGITAL INPUTS

It consists of 14 digital inputs/output pins, each of which provide or take up 40mA current. Some of them have special functions like pins 0 and 1, which act as Rx and Tx respectively , for serial communication, pins 2 and 3-which are external interrupts, pins 3,5,6,9,11 which provides pwm output and pin 13 where LED is connected.

ANALOG INPUTS:

It has 6 analog input/output pins, each providing a resolution of 10 bits.

AREF

It provides reference to the analog inputs

RESET

It resets the microcontroller when low

5 STEPS TO PROGRAM AN ARDUINO

Programs written in Arduino are known as sketches. A basic sketch consists of 3 parts

1. Declaration of Variables
2. Initialization: It is written in the setup () function.
3. Control code: It is written in the loop () function.

- The sketch is saved with .ino extension. Any operations like verifying, opening a sketch, saving a sketch can be done using the buttons on the toolbar or using the tool menu.
- The sketch should be stored in the sketchbook directory.
- Chose the proper board from the tools menu and the serial port numbers.
- Click on the upload button or chose upload from the tools menu. Thus the code is uploaded by the bootloader onto the microcontroller.

BASIC ADRUINO FUNCTIONS

- **digitalRead**(pin): Reads the digital value at the given pin.
- **digitalWrite**(pin, value): Writes the digital value to the given pin.

- **pinMode**(pin, mode): Sets the pin to input or output mode.
- **analogRead**(pin): Reads and returns the value.
- **analogWrite**(pin, value): Writes the value to that pin.
- **serial.begin**(baud rate): Sets the beginning of serial communication by setting the bit rate.

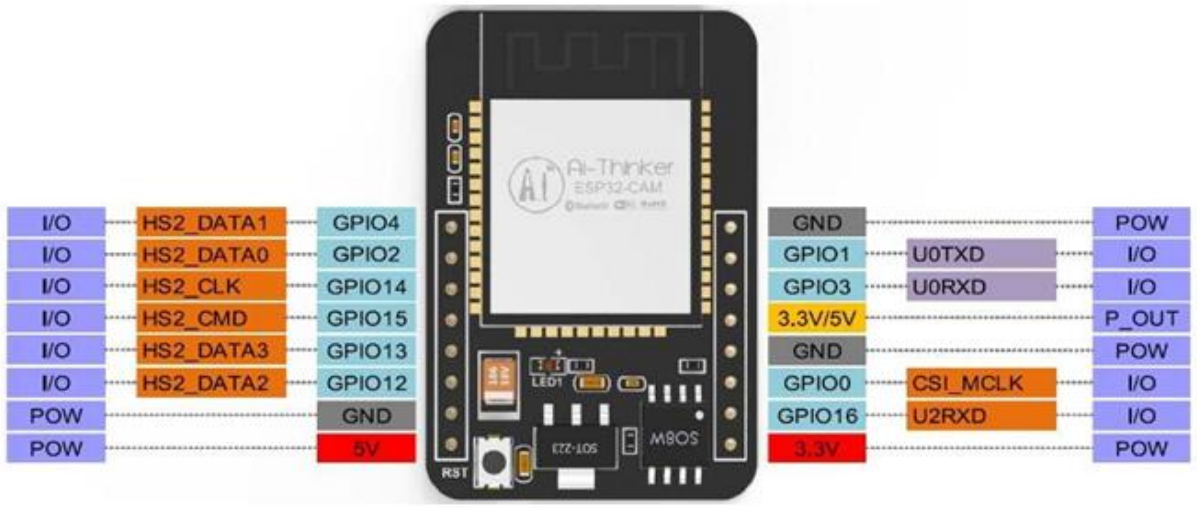
FEATURES

- It is inexpensive
- It comes with an open source hardware feature which enables users to develop their own kit using already available one as a reference source.
- The Arduino software is compatible with all types of operating systems like Windows, Linux, and Macintosh etc.
- It also comes with open source software feature which enables experienced software developers to use the Arduino code to merge with the existing programming language libraries and can be extended and modified.
- It is easy to use for beginners.
- We can develop an Arduino based project which can be completely stand alone or projects which involve direct communication with the software loaded in the computer.
- It comes with an easy provision of connecting with the CPU of the computer using serial communication over USB as it contains built in power and reset circuitry.

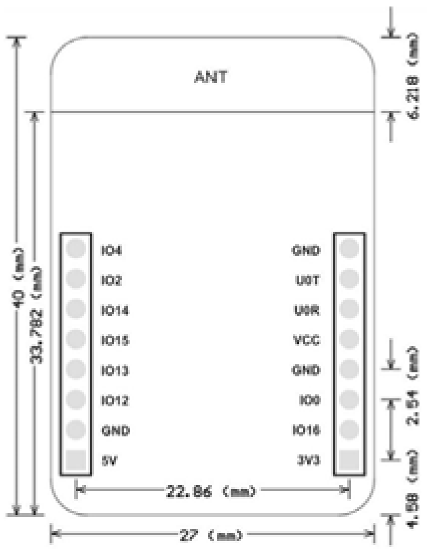
ESP32-CAM

INTRODUCTION

ESP32-CAM is a low-cost ESP32-based development board with an onboard camera, small in size. It is an ideal solution for IoT applications, prototypes, constructions, and DIY projects. The board integrates WiFi, traditional Bluetooth, and low-power BLE, with 2 high-performance 32-bit LX6 CPUs. It adopts a 7-stage pipeline architecture, on-chip sensor, Hall sensor, temperature sensor, and more, with its main frequency adjustment ranging from 80MHz to 240MHz. Fully compliant with WiFi 802.11b/g/n/e/i and Bluetooth 4.2 standards, it can be used as a master mode to build an independent network controller or as a slave to other host MCUs to add networking capabilities to existing devices. ESP32-CAM can be widely used in various IoT applications. It is suitable for home smart devices, industrial wireless control, wireless monitoring, QR wireless identification, wireless positioning system signals, and other IoT applications. It is an ideal solution for IoT applications.



Schematic Diagram



Dimension Diagram

Notes:

1. Please be sure that the power supply for the module should be at least 5V 2A, otherwise there may be water ripple appearing on the image.
2. ESP32 GPIO 32 pin is used to control the power of the camera, so when the camera is in working, pull GPIO 32 pin low.
3. Since IO pin is connected to camera XCLK, it should be left floating in use, and do not connect it to high/low level.
4. The product has been equipped with default firmware before leaving the factory, and we do not provide additional ones for you to download. So, please be cautious when you choose to burn other firmwares.

FEATURES:

• Up to 160MHz clock speed, Summary computing power up to 600 DMIPS

• Built-in 520KB SRAM, external 4MB PSRAM

• Supports UART/SPI/I2C/PWM/ADC/DAC

• Support OV2640 and OV7670 cameras, Built-in Flashlamp.

• Support image WiFI upload

• Support TF card

• Supports multiple sleep modes.

• Embedded Lwip and FreeRTOS

• Supports STA/AP/STA+AP operation mode

• Support SmartConfig/AirKiss technology

• Support for serial port local and remote firmware upgrades (FOTA)

SPECIFICATION:

• SPI Flash: default 32Mbit

• RAM: built-in 520KB + external 4MB PSRAM

• Dimension: 27*40.5*4.5 (±0.2) mm / 1.06*1.59*0.18"

• Bluetooth: Bluetooth 4.2 BR/EDR and BLE standards

• Wi-Fi: 802.11b/g/n/e/i

• Support Interface: UART, SPI, I2C, PWM

• Support TF card: maximum support 4G

• IO port: 9

• Serial Port Baud-rate: Default 115200 bps

• Image Output Format: JPEG (OV2640 support only), BMP, GRAYSCALE

• Spectrum Range: 2412~2484MHz

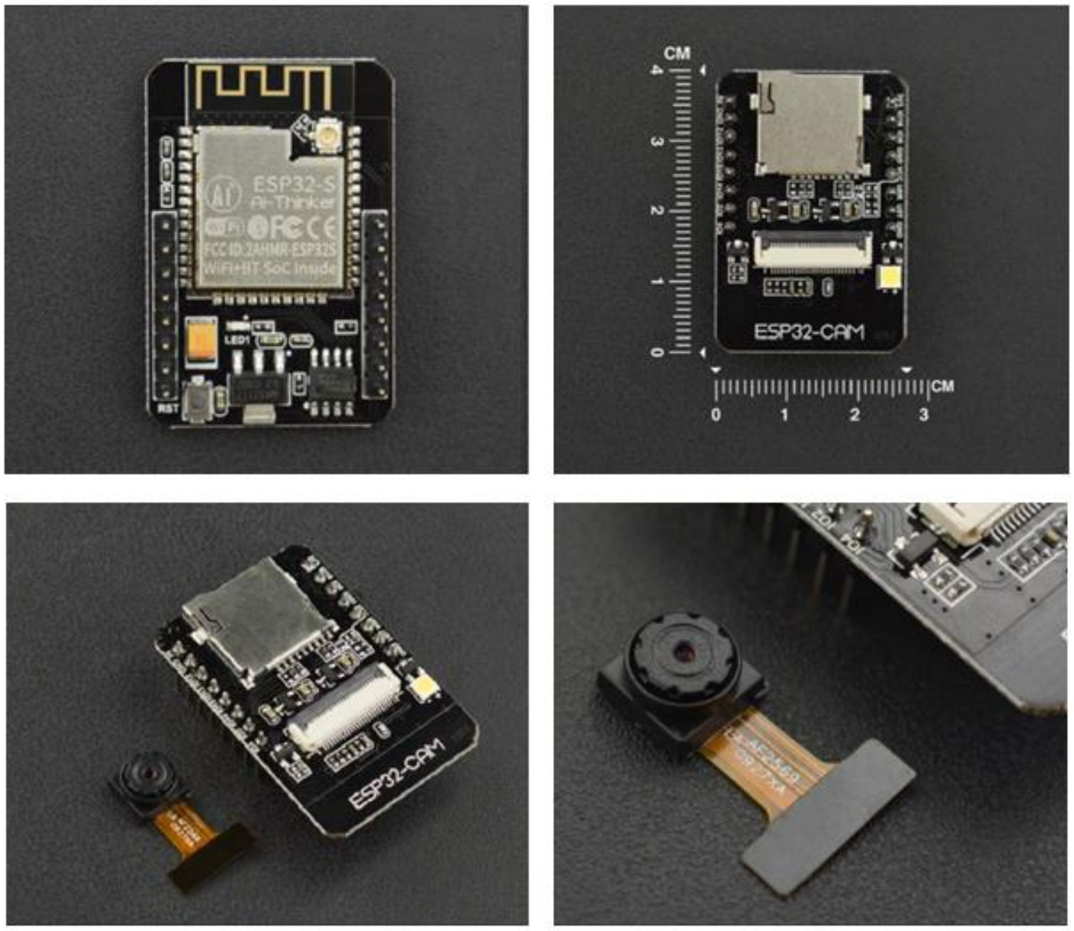
• Antenna: onboard PCB antenna, gain 2dBi

• Transmit Power:

  - 802.11b: 17±2dBm (@11Mbps)
  - 802.11g: 14±2dBm (@54Mbps)
  - 802.11n: 13±2dBm (@MCS7)

• Receiving Sensitivity:

  - CCK, 1Mbps: -90dBm
  - CCK, 11Mbps: -85dBm
  - 6Mbps (1/2 BPSK): -88dBm
  - 54 Mbps (3/4 64-QAM): -70dBm
  - MCS7 (65Mbps, 72.2Mbps): -67dBm
- Power consumption:
  - Turn off the flash: 180mA @ 5V
  - Turn on the flash and adjust the brightness to the maximum: 310mA @ 5V
  - Deep-sleep: the lowest power consumption can reach 6mA @ 5V
  - Modem-sleep: up to 20mA @ 5V
  - Light-sleep: up to 6.7mA @ 5V
- Security: WPA/WPA2/WPA2-Enterprise/WPS
- Power supply range: 5V
- Operating temperature: -20°C~85°C
- Storage environment: -40°C~90°C, <90%RH
- Weight: 10g



LCD DISPLAY

INTRODUCTION

Liquid crystal cell displays (LCDs) used to display of display of numeric and alphanumeric characters in dot matrix and segmental displays.They are all around us in laptop computers, digital clocks and watches, microwave, CD players and many other electronic devices. LCDs are common because they offer some real advantages over other display technologies. LCDs consume much less power than LED and gas-display displays because they work on the principle of blocking light rather than emitting it.

An LCD is made with either a passive matrix or an active matrix display grid. An active matrix has a transistor located at each pixel intersection, requiring less current to control the luminance of a pixel. For this reason, the current in an active matrix display can be switched on and off more frequently, improving the screen refresh time.Passive matrix LCD's have dual scanning, meaning that they scan the grid twice with current in the same

WORKING

When sufficient voltage is applied to the electrodes the liquid crystal molecules would be aligned in a specific direction. The light rays passing through the LCD would be rotated by the polarizer, which would result in activating/highlighting the desired characters. The power supply should be of +5v, with maximum allowable transients of 10mv. To achieve a better/suitable contrast for the display the voltage at pin 3 should be adjusted properly.

The ground terminal of the power supply must be isolated properly so that voltage is induced in it. The module should be isolated properly so that stray voltages are not induced, which could cause a flicking display.

LCD is lightweight with only a few, millimeters thickness since the LCD consumes less power, they are compatible with low power electronic circuits, and can be powered for long durations. LCD does not generate light and so light is needed to read the display. By using backlighting, reading is possible in the dark. LCDs have long life and a wide operating temperature range. Before LCD is used for displaying proper initialization should be done. LCDs with a small number of segments, such as those used in digital watches and pocket calculators, have individual electrical contacts for each segment. An external dedicated circuit supplies an electric charge to control each segment.

This display structure is unwieldy for more than a few display elements. Small monochrome displays such as those found in personal organizers, or older laptop screens. The pixels are addressed one at a time by row and column addresses. This type of display is called passive-matrix addressed because the pixel must retain its state between refreshes without the benefit of a steady electrical charge. As the number of pixels increases, this type of display becomes less feasible.

Very slow response times and poor contrast are typical of passive matrix addressed LCDs. High-resolution color displays such as modern LCD computer monitors and televisions use an active matrix structure. A matrix of thin-film transistors (TFTs) is added to the polarizing and color filters. Each pixel has its own dedicated transistor, allowing each column line to access one pixel. When a row line is activated, all of the column lines are connected to a row of pixels and the correct voltage is driven onto all of the column lines.

The row line is then deactivated and the next row line is activated. All of the row lines are activated in sequence during a refresh operation. Active-matrix addressed displays look "brighter" and "sharper" than passive-matrix addressed displays of the same size, and generally have quicker response times, producing much better images. A general purpose alphanumeric LCD, with two lines of 16 characters. So the type of LCD used in this project is16 characters * 2 lines with 5*7 dots with cursor, built in controller, +5v power supply, 1/16 duty cycle.

PIN DESCRIPTION FOR LCD

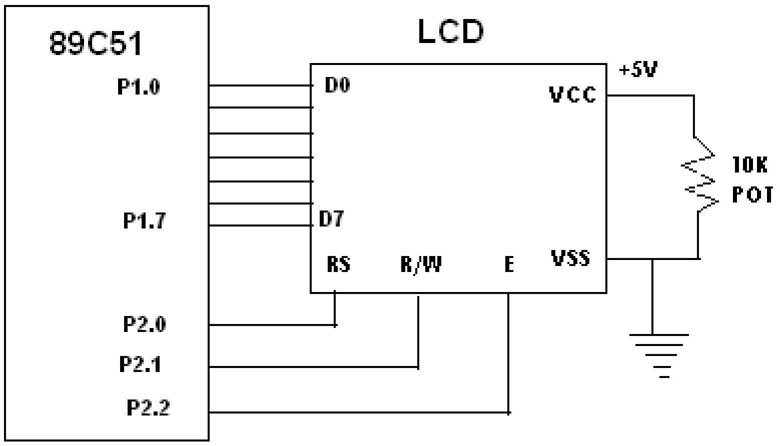
PIN NO	SYMBOL	FUNCTION
1	Vss	Ground terminal of Module
2	Vdd	Supply terminal of Module, + 5v
3	Vo	Power supply for liquid crystal drive

4	RS	Register select
		RS=0…Instruction register
		RS=1…Data register

5	R/W	Read/Write R/W=1…Read R/W=0…Write
6	EN	Enable
7-14	DB0-DB7	Bi-directional Data Bus. Data Transfer is performed once ,thru DB0-DB7,incase of interface data

		length is 8-bits;and twice, thru DB4-DB7 in the case of interface data length is 4-bits.Upper four bits first then lower four bits.
15	LAMP-(L-)	LED or EL lamp power supply terminals
16	LAMP+(L+) (E2)	Enable

LCD INTERFACING WITH MICROCONTROLLER



ADVANTAGES

- Consume much lesser energy when compared to LEDs.
- Utilizes the light available outside and no generation of light.
- Since very thin layer of liquid crystal is used, more suitable to act as display elements.
- Since reflectivity is highly sensitive to temperature, used as temperature measuring sensor.

DISADVANTAGES

- Angle of viewing is very limited.
- External light is a must for display.
- Since not generating its own light and makes use of external light for display, contrast is poor.
- Cannot be used under wide range of temperature.

APPLICATIONS

- Watches
- Fax & Copy machines & Calculators.

SOFTWARE USED

ARDUINO SOFTWARE (IDE)

The Arduino Integrated Development Environment or Arduino Software (IDE) contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

WRITING SKETCHES

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension **.**ino. The editor has features for cutting/pasting and for searching/replacing text.

The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information.

The bottom right hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension .pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.

✓ Verify
Checks your code for errors compiling it.

➜ Upload
Compiles your code and uploads it to the configured board. See uploading below for details.

Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

New

Creates a new sketch.

Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File | Sketchbookmenu instead.

Save

Saves your sketch.

Serial Monitor

Opens the serial monitor.

Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

FILE

- New
  Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.
- Open
  Allows loading a sketch file browsing through the computer drives and folders.
- Open Recent
  Provides a short list of the most recent sketches, ready to be opened.

- Sketchbook
  shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.

- Examples

  any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.
- Close

  closes the instance of the Arduino Software from which it is clicked.
- Save

  saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as.." window.
- Save as...

  Allows saving the current sketch with a different name.
- Page Setup

  It shows the Page Setup window for printing.
- Print

  sends the current sketch to the printer according to the settings defined in Page Setup.
- Preferences

  Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.
- Quit

  Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

EDIT

- Undo/Redo

  Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.
- Cut

  Removes the selected text from the editor and places it into the clipboard.
- Copy

  Duplicates the selected text in the editor and places it into the clipboard.

- Copy for Forum

  Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.
- Copy as HTML

  Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.
- Paste

  Puts the contents of the clipboard at the cursor position, in the editor.
- Select All

  Selects and highlights the whole content of the editor.
- Comment/Uncomment

  Puts or removes the // comment marker at the beginning of each selected line.
- Increase/Decrease Indent

  Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.
- Find

  Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.
- Find Next

  Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.
- Find Previous

  Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

SKETCH

- Verify/Compile

  Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.
- Upload

  Compiles and loads the binary file onto the configured board through the configured Port.

- Upload Using Programmer
  This will overwrite the boot loader on the board; you will need to use Tools > Burn Boot loader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so Tools -> Burn Bootloader command must be executed.
- Export Compiled Binary
  Saves a .hex file that may be kept as archive or sent to the board using other tools.
- Show Sketch Folder
  Opens the current sketch folder.
- Include Library
  Adds a library to your sketch by inserting include statements at the start of your code. For more details, seelibraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.
- Add File...
  Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side o the toolbar.

TOOLS

- Auto Format
  this formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.
- Archive Sketch
  Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.
- Fix Encoding & Reload
  Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.
- Serial Monitor
  Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.

- Board
  Select the board that you're using. See below for descriptions of the various boards.
- Port
  This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.
- Programmer
  For selecting a harware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.
- Burn Boot loader
  The items in this menu allow you to burn a bootloader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase new ATmega microcontrollers (which normally come without a bootloader). Ensure that you've selected the correct board from the Boards menu before burning the bootloader on the target board. This command also set the right fuses.

HELP

Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.

- Find in Reference
  this is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

SKETCHBOOK

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar.

The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

TABS, MULTIPLE FILES, AND COMPILATION

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

UPLOADING

Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241 (for an Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or/dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx, /dev/ttyUSBx or similar.

Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the File menu. Current Arduino boards will reset automatically and begin the upload.

With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board.

It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

LIBRARIES

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up.

If a sketch no longer needs a library, simply delete its include statements from the top of your code. There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.

To write your own library, see this tutorial.

THIRD-PARTY HARDWARE

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, boot loaders, and programmer definitions.

To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don't use "arduino" as the sub-directory name or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory.

For details on creating packages for third-party hardware, see the Arduino IDE 1.5 3rd party Hardware specification.

SERIAL MONITOR

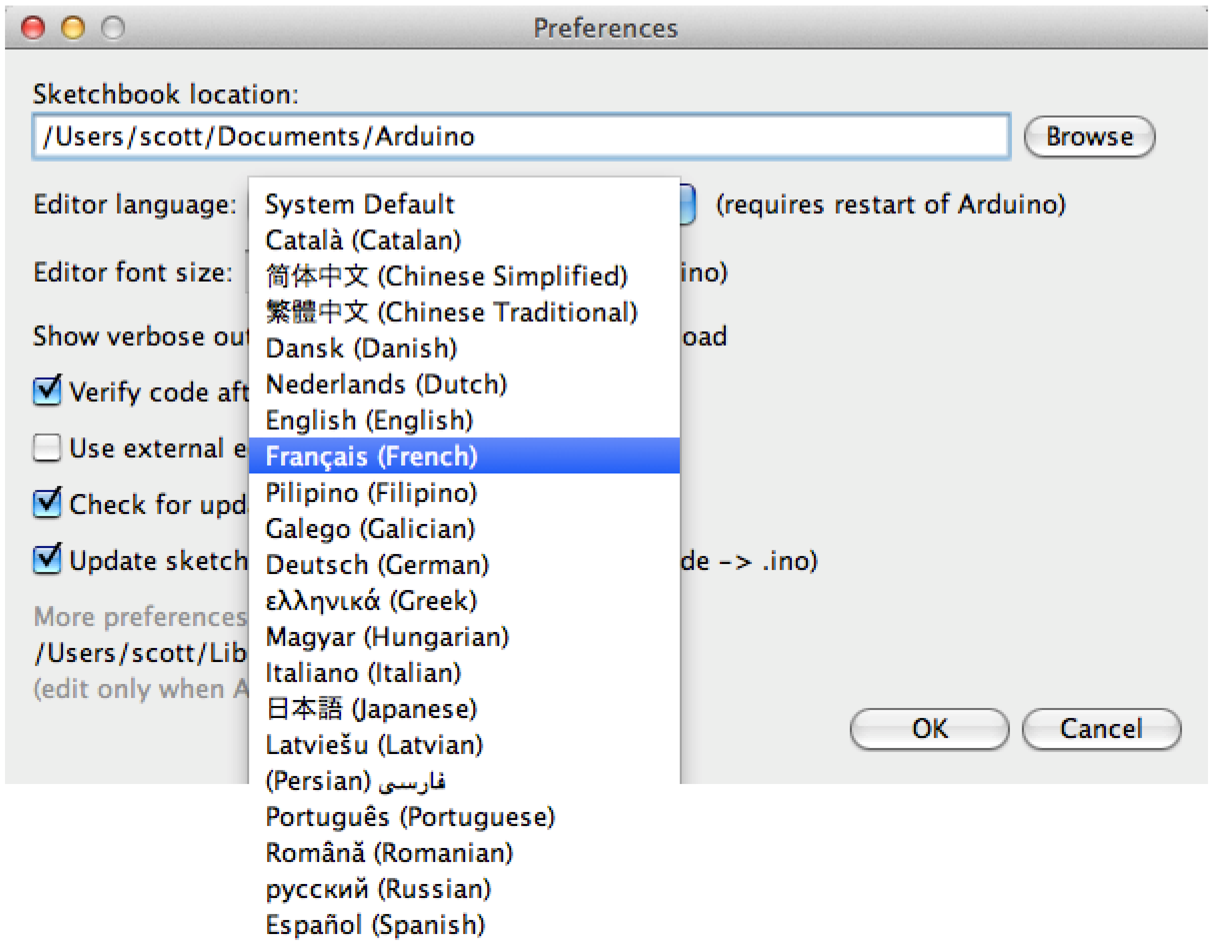
Displays serial data being sent from the Arduino or Genuino board (USB or serial board). To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to Serial.begin in your sketch. Note that on Windows, Mac or Linux, the Arduino or Genuino board will reset (rerun your sketch execution to the beginning) when you connect with the serial monitor.

You can also talk to the board from Processing, Flash, MaxMSP, etc (see the interfacing page for details).

REFERENCES

Some preferences can be set in the preferences dialog (found under the Arduino menu on the Mac, or File on Windows and Linux). The rest can be found in the preferences file, whose location is shown in the preference dialog.

LANGUAGE SUPPORT


Preferences

Sketchbook location:
/Users/scott/Documents/Arduino
Browse

Editor language: (requires restart of Arduino)
Editor font size:
Show verbose ou
✓ Verify code aft
Use external e
✓ Check for upd
✓ Update sketch
More preferences
/Users/scott/Lib
(edit only when A
OK
Cancel

System Default
Català (Catalan)
简体中文 (Chinese Simplified)
繁體中文 (Chinese Traditional)
Dansk (Danish)
Nederlands (Dutch)
English (English)
Français (French)
Pilipino (Filipino)
Galego (Galician)
Deutsch (German)
ελληνικά (Greek)
Magyar (Hungarian)
Italiano (Italian)
日本語 (Japanese)
Latviešu (Latvian)
(Persian) فارسی
Português (Portuguese)
Română (Romanian)
русский (Russian)
Español (Spanish)

Since version 1.0.1 , the Arduino Software (IDE) has been translated into 30+ different languages. By default, the IDE loads in the language selected by your operating system. (Note: on Windows and possibly Linux, this is determined by the locale setting which controls currency and date formats, not by the language the operating system is displayed in.)

If you would like to change the language manually, start the Arduino Software (IDE) and open the Preferences window. Next to the Editor Language there is a dropdown menu of currently supported languages. Select your preferred language from the menu, and restart the software to use the selected language. If your operating system language is not supported, the Arduino Software (IDE) will default to English.

You can return the software to its default setting of selecting its language based on your operating system by selecting System Default from the Editor Language drop-down. This setting will take effect when you restart the Arduino Software (IDE).

Similarly, after changing your operating system's settings, you must restart the Arduino Software (IDE) to update it to the new default language.

BOARDS

The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets and the file and fuse settings used by the burn boot loader command. Some of the board definitions differ only in the latter, so even if you've been uploading successfully with a particular selection you'll want to check it before burning the boot loader. You can find a comparison table between the various boards here.

Arduino Software (IDE) includes the built in support for the boards in the following list, all based on the AVR Core. The Boards included in the standard installation allows to add support for the growing number of new boards based on different cores like Arduino Due, Arduino Zero, Edison, Galileo and so on.

- ArduinoYùn
  An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- Arduino/Genuino Uno
  An ATmega328 running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- ArduinoDiecimila or Duemilanove w/ ATmega168
  An ATmega168 running at 16 MHz with auto-reset.
- Arduino Nano w/ ATmega328
  An ATmega328 running at 16 MHz with auto-reset. Has eight analog inputs.
- Arduino/Genuino Mega 2560
  An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- Arduino Mega
  An ATmega1280 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.

- Arduino Mega ADK

  An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- Arduino Leonardo

  An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- Arduino Micro

  An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- ArduinoEsplora

  An ATmega32u4 running at 16 MHz with auto-reset.
- Arduino Mini w/ ATmega328

  An ATmega328 running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM.
- Arduino Ethernet

  Equivalent to Arduino UNO with an Ethernet shield: An ATmega328 running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- ArduinoFio

  An ATmega328 running at 8 MHz with auto-reset. Equivalent to Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ATmega328, 6 Analog In, 14 Digital I/O and 6 PWM.
- Arduino BT w/ ATmega328

  ATmega328 running at 16 MHz. The boot loader burned (4 KB) includes codes to initialize the on-board Bluetooth module, 6 Analog In, 14 Digital I/O and 6 PWM..
- Lily Pad Arduino USB

  An ATmega32u4 running at 8 MHz with auto-reset, 4 Analog In, 9 Digital I/O and 4 PWM.
- Lily Pad Arduino

  An ATmega168 or ATmega132 running at 8 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328

  An ATmega328 running at 16 MHz with auto-reset. Equivalent to ArduinoDuemilanove or Nano w/ ATmega328; 6 Analog In, 14 Digital I/O and 6 PWM.
- Arduino NG or older w/ ATmega168

  An ATmega168 running at 16 MHz without auto-reset. Compilation and upload is equivalent to ArduinoDiecimila or Duemilanove w/ ATmega168, but the bootloader burned has a slower timeout (and blinks the pin 13 LED three times on reset); 6 Analog In, 14 Digital I/O and 6 PWM.

- Arduino Robot Control

  An ATmega328 running at 16 MHz with auto-reset.
- Arduino Robot Motor

  An ATmega328 running at 16 MHz with auto-reset.
- Arduino Gemma

  An ATtiny85 running at 8 MHz with auto-reset, 1 Analog In, 3 Digital I/O and 2 PWM.

EMBEDDED C

Embedded C is the most popular embedded software language in the world. Most embedded software is written in Embedded C.It is a set of language extensions for the C Programming language by the C Standards committee to address commonality issues that exist between C extensions for different embedded systems.

Historically, embedded C programming requires nonstandard extensions to the C language in order to support exotic features such as fixed-point arithmetic, multiple distinct memory banks, and basic I/O operations. The C programming language is perhaps the most popular programming language for programming embedded systems.

Most C programmers are spoiled because they program in environments where not only there is a standard library implementation, but there are frequently a number of other libraries available for use. The cold fact is, that in embedded systems, there rarely are many of the libraries that programmers have grown used to, but occasionally an embedded system might not have a complete standard library, if there is a standard library at all.

Few embedded systems have capability for dynamic linking, so if standard library functions are to be available at all, they often need to be directly linked into the executable. Oftentimes, because of space concerns, it is not possible to link in an entire library file, and programmers are often forced to "brew their own" standard c library implementations if they want to use them at all. While some libraries are bulky and not well suited for use on microcontrollers, many development systems still include the standard libraries which are the most common for C programmers.

C remains a very popular language for micro-controller developers due to the code efficiency and reduced overhead and development time. C offers low-level control and is considered more readable than assembly. Many free C compilers are available for a wide variety of development platforms.

The compilers are part of an IDEs with ICD support, breakpoints, single-stepping and an assembly window. The performance of C compilers has improved considerably in recent years, and they are claimed to be more or less as good as assembly, depending on who you ask. Most tools now offer options for customizing the compiler optimization. Additionally, using C increases portability, since C code can be compiled for different types of processors.

BASIC CONCEPTS OF EMBEDDED C AND EMBEDDED PROGRAMMING

Embedded C, even if it's similar to C, and embedded languages in general requires a different kind of thought process to use. Embedded systems, like cameras or TV boxes, are simple computers that are designed to perform a single specific task. They are also designed to be efficient and cheap when performing their task.

For example, they aren't supposed to use a lot of power to operate and they are supposed to be as cheap as possible. As an embedded system programmer, you will have simple hardware to work with. You will have very little RAM, ROM and very little processing power and stack space. The reason why most embedded systems use Embedded C as a programming language is because Embedded C lies somewhere between being a high level language and a low level language. Embedded C, unlike low level assembly languages, is portable.

It can run on a wide variety of processors, regardless of their architecture. Unlike high level languages, Embedded C requires less resources to run and isn't as complex. Some experts estimate that C is 20% more efficient than a modern language like C++. Another advantage of Embedded C is that it is comparatively easy to debug.

EMBEDDED C COMPILERS

There are a variety of different compilers on the market, manufactured by different companies that use Embedded C. One of the more popular ones is the Keil compiler. Because of this, Embedded C is also sometimes known as Keil C.

Embedded C has several keywords that are not present in C (learn more about the concept of keywords in this course). These keywords are associated with operations needed by microprocessors. You will need to be familiar with all of them to be able to write Embedded C programs.

Unsigned char data a;

Here, the unsigned char declaration is like a normal C declaration. We just added the data keyword, which tells the microcontroller to store the unsigned char a in the internal data memory.

bdata: The bdata keyword lets you store a declared variable in the bit addressable memory. Take a look at this example:

Unsigned char bdata a;

This is similar to the data declaration we showed you above. You have to access bdata variables in a different way, however.

Using: This keyword lets you execute a function by letting it access a register bank. There are three possible values: 1, 2 and 3.

EMBEDDED SYSTEMS PROGRAMMING

Embedded systems programming is different from developing applications on a desktop computers. Key characteristics of an embedded system, when compared to PCs, are as follows:

- Embedded devices have resource constraints(limited ROM, limited RAM, limited stack space, less processing power)
- Components used in embedded system and PCs are different; embedded systems typically uses smaller, less power consuming components. Embedded systems are more tied to the hardware.

EMBEDDED SYSTEMS USING DIFFERENT TYPE OF LANGUAGES:

- Machine Code
- Low level language, i.e., assembly
- High level language like C, C++, Java, Ada, etc.
- Application level language like Visual Basic, scripts, Access, etc.

DIFFERENCE BETWEEN C AND EMBEDDED C

- Though **C and embedded C** appear different and are used in different contexts, they have more similarities than the differences. Most of the constructs are same; the difference lies in their applications.
- C is used for desktop computers, while **embedded C** is for microcontroller based applications. Accordingly, C has the luxury to use resources of a desktop PC like memory, OS, etc. While programming on desktop systems, we need not bother about memory. However, embedded C has to use with the limited resources (RAM, ROM, I/Os) on an embedded processor. Thus, program code must fit into the available program memory. If code exceeds the limit, the system is likely to crash.
- Compilers for C (ANSI C) typically generate OS dependant executables. **Embedded C** requires compilers to create files to be downloaded to the microcontrollers/microprocessors where it needs to run. Embedded compilers give access to all resources which is not provided in compilers for desktop computer applications.
- Embedded systems often have the real-time constraints, which is usually not there with desktop computer applications.

Embedded systems often do not have a console, which is available in case of desktop applications.
So, what basically is different while programming with **embedded C** is the mindset; for embedded applications, we need to optimally use the resources, make the program code efficient, and satisfy real time constraints, if any. All this is done using the basic constructs, syntaxes, and function libraries of ‘C’.

ADVANTAGES

- It is small and reasonably simpler to learn, understand, program and debug.
- C Compilers are available for almost all embedded devices in use today, and there is a large pool of experienced C programmers.

- Unlike assembly, C has advantage of processor-independence and is not specific to any particular microprocessor/ microcontroller or any system. This makes it convenient for a user to develop programs that can run on most of the systems.
- As C combines functionality of assembly language and features of high level languages, C is treated as a ‘middle-level computer language’ or ‘high level assembly language’
- It is fairly efficient
- It supports access to I/O and provides ease of management of large embedded projects

Many of these advantages are offered by other languages also, but what sets C apart from others like Pascal, FORTRAN, etc. is the fact that it is a middle level language; it provides direct hardware control without sacrificing benefits of high level languages.

RESULT AND DISCUSSION

In recent years, there has been a growing interest in employing artificial intelligence (AI) techniques, particularly Convolutional Neural Networks (CNNs), for plant disease detection. This paper presents a novel approach utilizing CNNs for the detection of plant diseases. The system integrates an ESP32CAM for image capture, MATLAB for image analysis, and an Arduino with an LCD display for real-time output visualization. Through experimentation and analysis, the effectiveness of the proposed method in accurately identifying plant diseases is demonstrated. The results indicate the potential of CNN-based systems for early disease detection, thus aiding in crop management and ensuring food security.

Agricultural productivity is crucial for global food security, and plant diseases pose a significant threat to crop yields. Early detection and timely intervention are essential to mitigate the impact of diseases on crop production. Traditional methods of disease detection rely heavily on manual inspection by agricultural experts, which can be time-consuming and often subjective. With advancements in AI and computer vision, there is an opportunity to automate and improve the efficiency of disease detection processes.

Convolutional Neural Networks have shown remarkable performance in various image recognition tasks, including plant disease detection. By leveraging the power of deep learning, CNNs can learn discriminative features directly from images, enabling automated disease identification with high accuracy. In this paper, we propose a CNN-based approach for plant disease detection, integrating image capture, analysis, and output visualization components.

Methodology

Image Capture

The ESP32CAM module is utilized for capturing images of plant leaves. This module provides high-resolution images and can be easily interfaced with microcontrollers such as Arduino. Images of healthy and diseased plant leaves are collected under different lighting conditions and angles to create a diverse dataset for training the CNN model.

Image Analysis

MATLAB is employed for image preprocessing, feature extraction, and CNN model training. The collected images are preprocessed to enhance contrast, remove noise, and standardize image sizes. A pre-trained CNN model, such as VGG or ResNet, is fine-tuned using transfer learning to adapt it to the task of plant disease classification. The final layers of the CNN are modified to output disease labels corresponding to different classes.

Output Visualization

An Arduino microcontroller is used to interface with an LCD display, providing a real-time visualization of the disease detection results. Once an image is captured by the ESP32CAM, it is processed by the MATLAB-based CNN model, and the predicted disease label is transmitted to the Arduino. The Arduino then displays the result on the LCD screen, indicating whether the plant leaf is healthy or diseased and specifying the type of disease detected.

Results and Discussion

Experimental Setup

The proposed system is evaluated using a dataset comprising images of various plant diseases, including but not limited to powdery mildew, leaf spot, and blight. The dataset is divided into training, validation, and testing sets, ensuring proper evaluation of the CNN model's performance. Different configurations of the CNN architecture are experimented with to optimize performance metrics such as accuracy, precision, recall, and F1-score.

Performance Evaluation

The CNN model trained using the proposed methodology achieves promising results in terms of disease detection accuracy. The model demonstrates high accuracy rates on both the training and testing datasets, indicating its ability to generalize well to unseen data. Through confusion matrix analysis and ROC curve evaluation, it is observed that the CNN model effectively discriminates between healthy and diseased plant leaves, with minimal false positives and false negatives.

Real-time Implementation

The real-time implementation of the proposed system using Arduino and an LCD display showcases its practical applicability. Upon capturing an image of a plant leaf, the system rapidly processes the image and provides instant feedback regarding the presence of any disease. This real-time feedback mechanism is crucial for enabling timely intervention and preventing the spread of diseases in agricultural fields.

Limitations and Future Work

While the proposed system shows promising results, there are several limitations and areas for future improvement. One limitation is the reliance on a pre-existing dataset, which may not cover all possible plant diseases or variations in environmental conditions. Future work could involve the collection of a larger and more diverse dataset to enhance the robustness of the CNN model.

Additionally, the computational resources required for real-time image processing may pose challenges, particularly in resource-constrained environments. Optimization techniques such as model compression and hardware acceleration can be explored to improve the efficiency of the system.

Furthermore, the system's performance may vary under different lighting conditions and with variations in leaf textures and shapes. Incorporating additional preprocessing steps and data augmentation techniques can help address these challenges and improve the overall robustness of the system.

summary

In summary, this paper presents a novel approach for plant disease detection using Convolutional Neural Networks. By integrating image capture, analysis, and output visualization components, the proposed system offers a practical solution for automated disease identification in agricultural settings. Experimental results demonstrate the effectiveness of the CNN-based approach in accurately detecting plant diseases in real-time. Future research directions include enhancing the system's robustness, scalability, and efficiency to facilitate its widespread adoption in agriculture for improved crop management and food security.

CHAPTER 5
CONCLUSIONS

5.1 Conclusions of the work

In our attempt at this project we were able to develop various convolutional neural networks which give satisfactory results with upmost precision. There were various approaches in the recent years of image segmentation and classification to detect the infectious plant leaves. Here we took the less explored method of convolutional neural networks. The model was able to detect and classify plant leaves on the basis of diseases. Model was able to reach an accuracy of 94% giving precise results predominantly.
The complete course of development was discussed from creating dataset of the images to model predictions and interface.

.

- Biologically leaf diseases sometimes exhibits similar symptoms which becomes difficult to fathom for artificial intelligence. This essentially means that there is a large scope for further research and development in this field.
- According to recent research complexity of plant diseases modifies drastically with the time, therefore thus we need a model which is adaptive to these types of conditions.
- The model shows a great potential backed up with accurate results.
- This type of automated system can prove to a great help to the agribusiness in terms of economic and environmental aspects.

5.2 Future Scope

- There is always a scope for improvement in the prediction accuracy by running the system through various versions and models of the CNN architecture. Moreover, it can also expand to a wider variety of crops giving it more fuzzy problems to work with.
- A standalone web application or mobile app can be developed which implements the same technique and provides the accessibility to the end user.
- As the field of research is ever growing with modern advancements in biotechnology and robotic engineering this type of system can be combined with monitoring robots and drones which may analyze the whole field crop with continuous and consistent analysis and hence notifying for any potential hazards and trigger warnings beforehand.

REFERENCES

[1] Hiroya Kondou, Hatuyoshi Kitamura, Yutaka Nishikawa, Yoshitaka Motonaga and Atsushi Hashimoto, Shape Evaluationby Digital Camera for Grape Leaf.

[2] Prasad Babu M.S. and Srinivasa Rao B. (2007) Leaves Recognition Using Back Propagation Neural Network-Advice For Pest and Disease Control On Crops.

[3] Maliappis M.T., Ferentinos K.P., Passam H.C. and Sideridis A.B. (2008) World Journal of Agricultural Sciences 4(5), 640-647.

[4] Ryszard S. Choras (2007) International Journal Of Biology And Biomedical Engineering,1(1), 6-16.

[5] Suhasini P.S., Sri Rama Krishna K., Murali Krishna I.V. (2009) Journal Of Theoretical And Applied Information Technology, 6(1), 116-122.

[6] Zhi-Chun Huang, Patrick P. Chan, Wing W.Y. N.G., Daniel S. Yeung (2010) ninth international conference on machine learning and cybernetics, 11-14, 719-724

[7] Huang J., Kumar S.R., Mitra M., Zhu W.J. and Zabih R. (1997) IEEE Int. Conf. Computer Vision and Pattern Recognition, 762-768.

[8] Del Bimbo A., Mugnaini M., Pala P. and Turco F. Picasso, (1997) 2nd International Conference on Visual Information Systems, 125-131.

[9] Jau-Ling Shih & Ling-Hwei Chen, color image retrieval based on primitives of color moments.

[10]Priti Maheswary, Namita Srivastav (2008) international conference on computer and electrical engineering, 821-824.

[11] Mona Sharma, Sameer Singh (2001) Seventh Australian and New Zealand Intelligent Information Systems Conference, 117-121

[12] Xinhong Zhang, Fan Zhang (2008) Congress on Image and Signal Processing, IEEE computer society, 773-776.

[13] Hui Yu, Mingjing Li, Hong-Jiang Zhang, Jufu Feng (2003)

International Conference on Image Processing.