PROJECT REPORT

on

Detection of plant diseases using Deep Learning

Major project for the partial fulfilment of :

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

Submitted by:

Harshak Bir Singh 171253

Arnav Sharma 171265

Under the supervision of

Mr. Prateek Thakral Sir

(Assistant Professor (Grade-2) of CSE dept)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY WAKNAGHAT, SOLAN-173234,

HIMACHAL PRADESH

CANDIDATE'S DECLARATION

I declare that the report on the topic " Detection of plant diseases using Deep Learning" in partial fulfillment of the requirements for award of degree of Bachelor of Technology in Computer Science and Engineering/Information Technology submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is authentic record of my own work carried out till 17th May under the supervision of Mr. Prateek Thakral (Assistant Professor (Grade-2) of CSE dept)
The matter in report is for the respective degree only and not used for other purposes.

Harshak Bir Singh (171253) [illegible]

Arnav Sharma (171265) [illegible]

This is to certify that the statement made by the candidates is true to my knowledge.

[illegible]

Mr. Prateek Thakral

(Assistant Professor (Grade-2) of CSE dept)

Department of Computer Science & Engineering and Information Technology

08/12/2020

Detection of plant diseases using Deep Learning

Introduction

The question of effective protection against plant diseases is closely related to the question of sustainable agriculture. The use of inexperienced pesticides can lead to long-term resistance to bacteria, greatly reducing disease resistance. Timely and accurate diagnosis of plant diseases is one of the pillars of agricultural accuracy. To avoid unnecessary waste of resources and resources and thus achieve healthy production in a dynamic environment, it is important that the identification of appropriate and timely diseases, including early prevention, has never been so important. There are several ways to diagnose plant diseases. Some diseases have no obvious symptoms or the result is too late, in this case a more complex analysis is needed. However, many diseases will have some form of action on the visible screen, so eye examination of the buttocks by a trained professional is the main method used to diagnose plant diseases.

To get an accurate diagnosis of plant diseases, veterinarians must have good visual skills in order to recognize the symptoms of the disease. Mutation changes in diseased plants can lead to misdiagnosis, as amateur growers and hobbyists can be much more difficult to determine than for geneticists. An automated system designed to help identify plant diseases through plant appearance and visual cues can be of great help to hobbyists through the garden process and specialists as a disease verification program.

Advances in computer recognition provide opportunities to increase and improve the practice of plant protection with precision and to expand the market for computer vision applications in the precise agricultural sector. To detect and differentiate plant diseases, common digital image processing methods such as color analysis and threshold application are used. In mechanical and cognitive science, ANN is a process of data processing inspired by the way biological systems, such as the brain, use information. A neural network or communication system is a calculation method used in computer science and other fields of study, based on a large number of neural units (artificial neurons). It mimics the way the blood brain solves the problem of a large number of biological neurons connected to axons. Each neuronal unit is connected to many other neuronal units, and the effect of the link in the activation of the connected neuronal object can be binding or inhibiting. Each neural unit can have a summary function that includes the values of all inputs. There may be a delay function or a limitation function on each connection to the unit itself, so the signal must cross the limit before it can transmit to other neurons. These programs are self-taught and training, rather than explicitly designed, and work best in areas where solutions or feature detection are difficult to master in traditional computer programs. Neural networks are usually made up of multilayer designs or cubes, and the signal path is torn from front to back. Distribution of back-to-back use is a stimulant to restore the weight of the "frontal" neural unit, sometimes combined with training with well-known positive effects. In terms of connectivity, modern networks have more fluidity in terms of promotion and prevention, and their communication is more complex and complex. Dynamic neural networks are the most advanced because they can vigorously establish new connections based on rules, or even create new neural units, while disabling other neural units. Although various neural networks are invisible, the purpose of neural networks is to solve problems in the same way as the human mind.

Current neural network projects typically use thousands or millions of neural units and millions of connections. Their complexity is still several orders of magnitude lower than that of the human brain, and they are closer to the computer power of the caterpillar. New brain research often innovates new models of neural networks. The new method is to use stretchable connections away from connecting processing layers instead of staying available to nearby neurons. In-depth studies of the different types of signals distributed by the axons over time, such as in-depth reading, are much more difficult to integrate than simply shutting down or closing a set of Boolean variables. Its input can take any value between 0 and 1. Also, the neuron has the weight of each input and the total deviation. Weight is a real number, which indicates the value of each input output. Bias is used to control the complexity of neuronal production 1. With neurons with very high bias, it is easy to produce 1, but when bias is severe, it is difficult to produce 1. India is an agricultural country, and about 70% of the population depends on agriculture.

Farmers can choose fruits and vegetables that are suitable for a wide variety of crops. The cultivation of high quality fruit and high quality products of these crops is very special. Managers should keep a close eye on plants so that the disease does not interfere with production. Image processing can be used in the agricultural sector, and its uses are as follows:
Look for diseased leaves, stems and fruits.

Examine the affected area.

Get the shape of the affected area.

Identify the color of the affected area.

Find the size and shape of the fruit.

The latest developments in Smartphones and computer views will make their high-definition cameras a very interesting diagnostic tool. It is estimated that by 2020, the number of smartphones worldwide will reach 500 to 6 billion. At the end of 2015, 69% of the world's population had access to mobile broadband streaming. From 2011 to 2012, this has had a huge impact on image recognition. Although CNN has been trained to distribute back to back decades, and the use of NN's GPU (including CNN) has been around for many years, CNN's rapid implementation is making great use of Ciresan and his GPU colleagues. The ability to share needs to improve computer perception. In 2011, this approach achieved more human performance for the first time in a visual pattern recognition competition. Also in 2011, it won the ICDAR China Handwriting Contest, and in May 2012, it won the ISBI Image Segmentation Contest. Until 2011, CNN did not play a major role in computer vision conferences, but in June 2012, Ciresan et al. At a major conference, CVPR demonstrated how CNN's largest team in the GPU could significantly improve its visual recordings. Compared to the in-depth machine learning method, it won the huge ImageNet competition with huge profits. He also won the ICPR competition, the theme of which was the analysis of major cancer diagnostic images, and the MICCAI challenge on the same topic the following year. In 2013 and 2014, with similar trends in large-scale visual recognition, the error rate of ImageNet operations using in-depth reading dropped significantly. The

Wolfram Image Recognition project has highlighted this development. Some researchers estimate that ImageNet's October 2012 victory brought "a radical change in learning" that transformed the intelligence industry. So, here, we use the most in-depth reading technology to show the feasibility of our approach using public data of 9000 images of healthy and infected tomato leaves. The model can be used in a smartphone app to identify 5 types, the type of tomato disease, without testing, its accuracy is 99.84%.

Introduction to in-depth reading

In-depth learning is a specific set of machine learning and machine learning a specific subset of artificial intelligence. Artificial intelligence is a vast field for building machines that can think intelligently. Machine learning is a component or subset of artificial intelligence (AI), which enables the system to learn and develop automatically from experience without the need for a clear program. Machine learning focuses on building computer programs that can access data and use it to read for themselves. In-depth learning is a collection of algorithms used in machine learning to perform high-quality data extraction using model formats made with multiple off-line modifications.

Introduction to Convolutional Neural Networks

In a convolutional neural network, the image begins to pass through the convolutional neural layer, which will convert the image into a matrix according to our size, where the size of the matrix we use is 256 x 256 x3. 3 here is a color picture. The image matrix is repeated with a filter or main matrix to find the feature map matrix. CNN uses channels to define the number of pixels to be converted to the input matrix. If the stride value is 1, the filter matrix will change the cell to 1, and if the value is 2, it will decrease by 2 pixels. If filters are not used properly, they are suitable for image installation, a fill will be used, and two cases will be provided: 1) Fill in the image with zero to make it fit; 2) Insert part of the image in the filter does not match, Called active filler. It has ReLU, which represents an offline repair unit. Output is f (x) = max (0, x). Converts negative value to zero. If the image is too large, part of the collection layer will reduce the number of parameters. Collections can be of various types, such as 1) the largest collection, 2) the standard collection, and 3) the total collection. The largest collection picks up the largest item on the modified feature map. Selecting the largest item can also use standard collection and total of all items in the feature map call such as total collection. A fully connected layer is called the FC layer, where we insert a matrix into a vector and feed it to a fully connected layer like a neural network. Finally, it uses functions, such as softmax or Sigmoid, to differentiate the effect such as cats, dogs, cars, trucks, and to diagnose plant leaf diseases. 1.2 Problem statement In analyzing the above problems, we will achieve the following: In this project, we try to explore the possibility of using two well-known technologies (eg neural network and imaging) in the classification of plant diseases. Therefore, the combination of these two methods can make the separation more relevant and reliable. The most important thing is to differentiate plant diseases by neural networks, because they can read a non-linear map between input and output. The ability to learn unfamiliar chaotic systems with the help of neural networks can work better than traditional analysis and other computer-based methods.

Purpose

The objectives of the project are as follows: Create a strong differentiation of plant disease by in-depth study.

This goal is achieved through the use of CNN and image processing. So, for us, we will use two popular technologies, neural networks and image processing, to diagnose diseases. Useful data is taken from a large number of data sets and segmented, so the combination of these two technologies can make the segmentation more relevant and reliable. As for the solution to the previous problem, the answer depends on the method used to classify it. Therefore, the process we will use has proven to be very suitable for the division of labor into entry systems. Many of the methods that can be helpful in class problems have shown that there are not enough disease planning activities in the admission systems.

Materials and methods

The complete model development process using CNN's in-depth plant diagnostics is presented in detail. The whole process is divided into several required sections in the following sections. The first is to use deep neural networks to collect images of the separation process.

Data set

We have published a database on the popular Plantvillage database, which contains approximately 5,000 photos of 14 plants and 26 diseases. We have chosen to process 9,000 images of tomato leaves; In addition to healthy leaves, our data contains samples of five tomato diseases, and six stages, as shown below: (0) Range: bacterial spots.

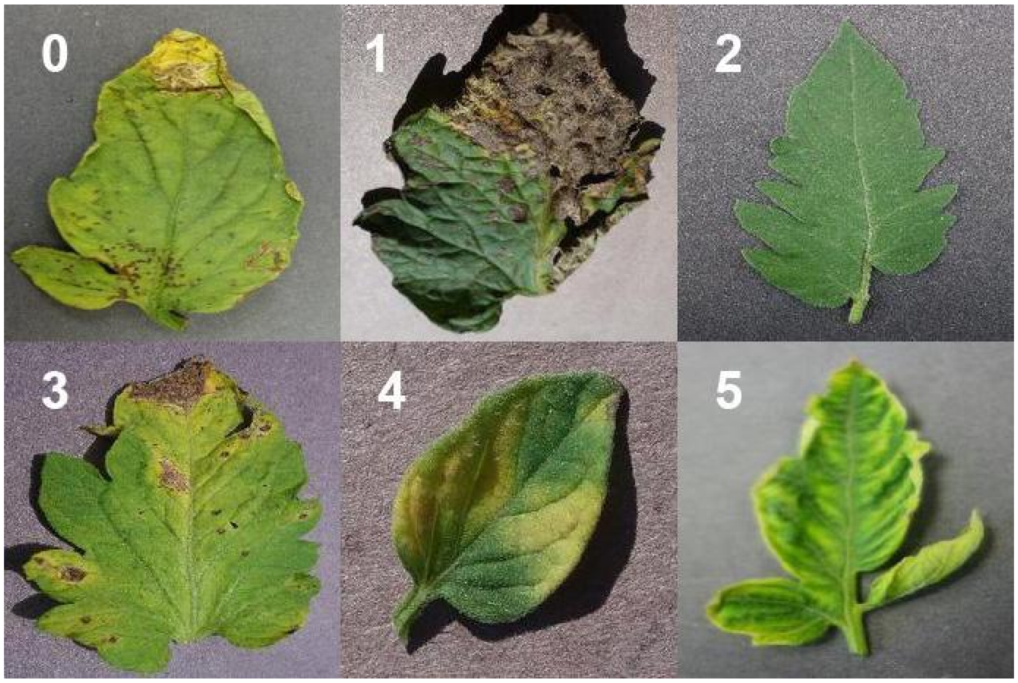
Category (1): Initial injury.

Section (2): Health.

Stage (3): Distinguished leaf disease of the leaf.

Category (4): Leaf mold.

Stage (5): Yellow curl infection.

- 

Adjust the image to 150 150 to speed up the calculation without affecting the quality of the data. From the training phase to the evaluation of the performance of the visual algorithm, appropriate data sets are required for all phases of object recognition research. We are fortunate to receive data on diseased leaves from the well-known Plant Village Dataset. This data set was found in the SP Mohanty git repository.

Image editing and labeling Images Downloaded on the Internet come in a variety of shapes and sizes. In order to obtain a better feature, the final image, intended to be used as a set of deep neural data separator data, is prepared to ensure consistency. Image editing usually involves removing background noise with low frequency, adjusting the density of a single particle image, removing reflections, and hiding certain parts of the image. Image previewing is a way to improve data. In addition, the process of making an image involves hand cutting all the images to form a square around the leaves to highlight the location of the leaves of the plants. In the data set photo category, low resolution images and less than 500 pixel sizes are not considered valid image set data.

Neural Network Training

A proposal to train a deep convolutional neural network to create an image separation model from a set of data proposed. Tensor Flow is an open source software library that can use data flow graphs for numerical calculations. Graph nodes represent mathematical operations, and the edges of the graph represent the various data matrices (tensors) interacting with each other. Flexible builds allow you to use a single API to move statistics across multiple CPUs or CPUs or desktops, servers or mobile devices. In machine learning, the convolutional neural network is a type of implantal response of the neural network, and the pattern of connections between neurons is inspired by the organization of the visual cortex of animals. A single cortical neuron responds to a stimulus in a limited space called the reception field. The receiving fields of different neurons meet partially, thus covering the field of view. The response of a single neuron to stimulation in its receptor field can be statistically measured by convolution. Transformation networks are inspired by biological processes and are different from multilayer perceptrons designed to use small processing. One of the great advantages of convolutional networks is that shared resources are used in the convolutional layer, which means that each pixel in the layer uses the same filter (weight library). This not only reduces memory, but also improves performance. The convolutional neural network is also called the SIANN, which derives its name from its weight-sharing structure and signals of translation flexibility. The convolutional layer is a fundamental part of the convolutional neural network. Layer parameters contain a set of readable nuclei, with a small reception field but extending to the full depth of the input volume. The adjusted line unit (Re LU) replaces non-linearity saturation. The Trigger function can flexibly adjust the configuration parameters and improve the accuracy of additional unaccounted calculation costs. In a neural implant network, a modifier is an activation function, defined as: F (x) = Max (0, x) When x is a neuron implant. This is also called ramp function, which is similar to half-wave adjustment in electrical engineering. This activation function was first introduced to dynamic networks by Hahn Losr et al. In the 2000 issue of "Nature", this text had a strong biological basis and a mathematical basis. Its use in convolutional networks is more effective than the more widely used logmoid sigmoid colon (inspired by the theory of chance; see the order of things) and its active hyperbolic tangent. The pace of CNN and ReLU training is often faster. This method works for the removal of each convolutional layer and the fully connected layer. On CNN, hidden neurons are isolated from "feature maps." The neurons on the feature map share the same weights and discrimination. Neurons in the same search map map. These neurons are different because they are connected to different neurons in the lower layer. Thus, in the first hidden layer, the neurons on the feature map will connect to different regions of the input image. The hidden layer is separated by feature maps, where each neuron in the feature map looks the same feature, but in a different location in the input image. Basically, a feature map is the result of applying convolution to an image. The convolutional layer is the mainstay of CNN. Layer parameters formed by a set of filters (or cores) can be read. The reception fields of these filters (or cores) are small, but expand to the full depth of the input volume. In a step forward, each filter will stabilize the width and height of the input volume, calculate the dot output between the input and input, and produce a two-dimensional map of the filter. As a result, the network reads filters that are activated when a specific type of object is detected in a specific location in the installation. Setting start maps for all filters by depth depth will create a complete convolutional layer. Therefore, each product item can also be interpreted as the release of a neuron that looks at a small fraction of the input and sharing parameters of the neurons in the same performance map. When using high-input inputs, such as images, it does not work to connect neurons to all neurons in the previous volume because such network constructions do not take into account the local formation of data. Transformation networks take advantage of spatial adjustment by using local connectivity patterns between neurons in adjacent layers: each neuron is connected to only a small fraction of the input volume. The level of this connection is a hyperparameter called the receptor field of the neuron. The connection is local to the space (width and height), but it always extends to the full depth of the input volume. Such a design ensures a readable filter produces the strongest response to local input patterns. Three hyperparameters control the magnitude of the output volume of the convolutional layer: depth, step and zero padding. 1. Output volume depth controls the number of neurons in a layer connected to the same input volume area. All of these neurons will learn to explode with various input signals. For example, if the first convolutional layer takes the first image as insertion, neurons of different depths will be induced in the presence of multiple edges or colored spots. 2. The step length controls how the depth column is mapped around the size of the area (width and height). When the span is 1, a new column of neuron depth is given a local position, and only one local interval. This results in high volatility in the reception curves between columns, and results in a large amount of output. On the other hand, if a higher step is used, the receiving fields will pass slightly and the final output volume will have a smaller local size. 3. The step length controls how the depth column is mapped by the size of the space (width and height). When the span is 1, a new column of neuron depth is given a local position, and only one local interval. This results in high volatility in the reception curves between columns, and results in a large amount of output. On the other hand, if a higher step is used, the receiving fields will pass slightly and the final output volume will have a smaller local size.

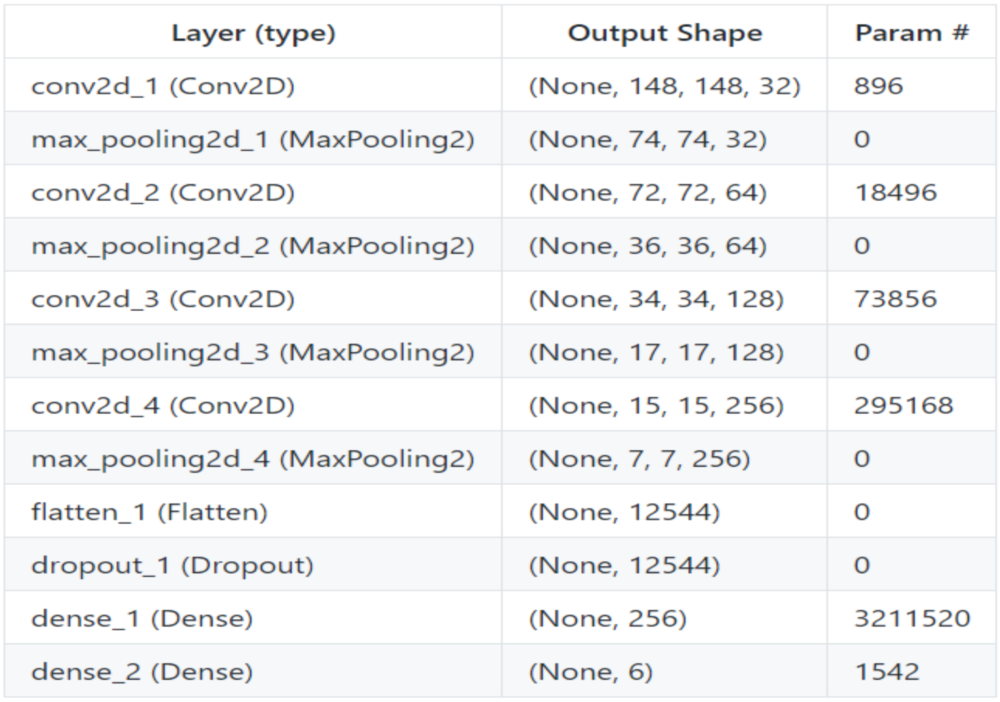
Perform the test.

A common way to measure the performance of artificial neural networks is to divide data into training sets and test sets, then train the neural network in the training set, and then use the predictive test set. Therefore, as the initial results of the test set and the predictable results of our model are known, the accuracy of the forecast can be calculated. 1.4.5 Utilities. One PC is used to train and evaluate the process of its entire plant disease diagnostic model described in this document. CNN training is done in graphics processing mode (GPU) On this particular machine, each training iteration took about three hours, and its basic features are shown in Table 2.

SYSTEM DEVELOPMENT

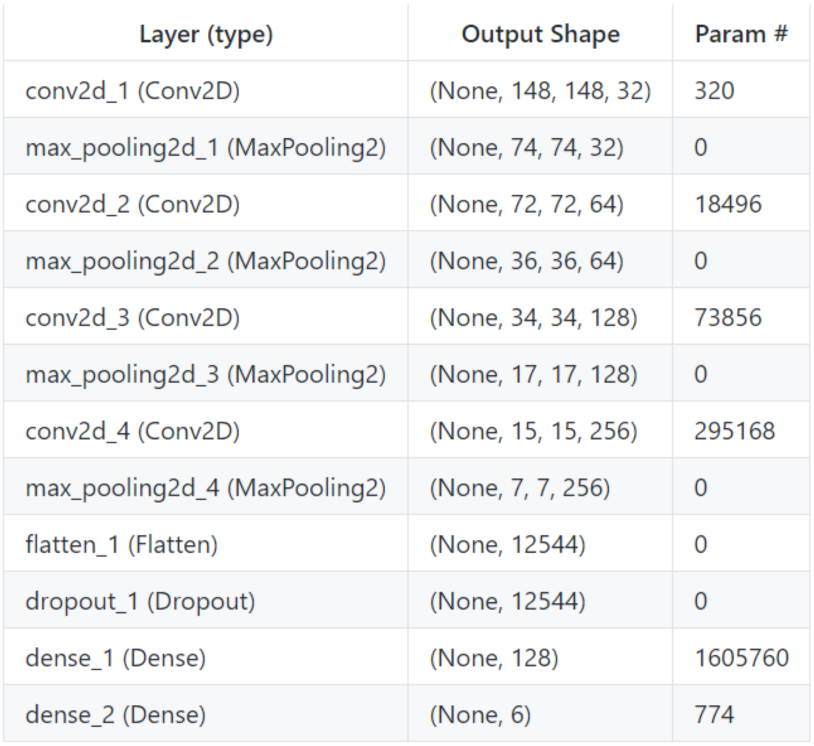
Our model takes immature images as input, so we used CNN (Convolutional Neural Networks) to extract the features, which resulted in the model having two parts: The first part of the model (including extraction), which was the same as full- color method and gray scale method, consists of layers -4 variables with the function of Reu activation, each followed by a Max Pooling layer. The second part after the sharp layer consists of two dense layers in both directions, but in full color the first one has 256 hidden units making a total of 3,601,478 trained network parameters, on the other hand the gray scale method has a hidden component of 128,4 and 4,4 as complete training parameters, we have reduced the size of the gray scale network to avoid overheating, because the last layer of both has Softmax as activation with 6 outputs representing six classes. After training the model we store it and use it to differentiate between different diseases.

Full color model summary:



Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_1 (MaxPooling2)	(None, 74, 74, 32)	0
conv2d_2 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_2 (MaxPooling2)	(None, 36, 36, 64)	0
conv2d_3 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_3 (MaxPooling2)	(None, 17, 17, 128)	0
conv2d_4 (Conv2D)	(None, 15, 15, 256)	295168
max_pooling2d_4 (MaxPooling2)	(None, 7, 7, 256)	0
flatten_1 (Flatten)	(None, 12544)	0
dropout_1 (Dropout)	(None, 12544)	0
dense_1 (Dense)	(None, 256)	3211520
dense_2 (Dense)	(None, 6)	1542

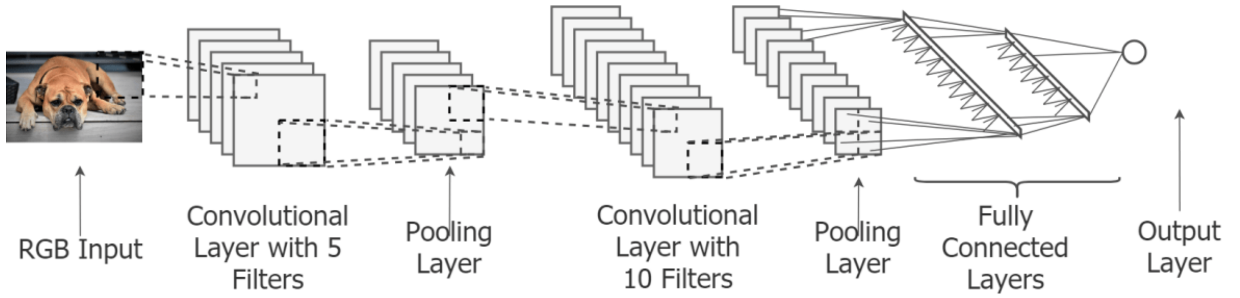
Gray-Scale Model Summary:



Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 148, 148, 32)	320
max_pooling2d_1 (MaxPooling2)	(None, 74, 74, 32)	0
conv2d_2 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_2 (MaxPooling2)	(None, 36, 36, 64)	0
conv2d_3 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_3 (MaxPooling2)	(None, 17, 17, 128)	0
conv2d_4 (Conv2D)	(None, 15, 15, 256)	295168
max_pooling2d_4 (MaxPooling2)	(None, 7, 7, 256)	0
flatten_1 (Flatten)	(None, 12544)	0
dropout_1 (Dropout)	(None, 12544)	0
dense_1 (Dense)	(None, 128)	1605760
dense_2 (Dense)	(None, 6)	774

Mathematical Model

Convolution neural networks (CNNs) are a family of deep networks that can use a local data structure (e.g. images) to learn data, so that the algorithm can generate something useful. For example, if I give CNN a personal photo, this deep neural network first needs to learn some local features (e.g. eyes, nose, mouth, etc.). These local factors are studied through the layers of convolution. CNN will then look at what local features are present in the given image and generate activation patterns (or opening vectors) that represent the global presence of these local feature maps. These activation patterns are produced by layers that are fully connected to CNN. For example, if an image is not a person, the activation pattern will be different from what we give the image of a person. There are three things that are different from typical CNN. It is the layers of convolution, the layers of integration and the layers that are fully connected. The convolution layer has many letters. These characters (sometimes called convolution filters) that exist in the convolution layer, read the local features in the image (e.g., what a human eye looks like). Such a local feature read by the convolution layer is called a feature map. After that these features are verified above the image. This convolution function will lead to a matrix (sometimes called an activation map). The activation map produces a maximum value in a given area, if the feature displayed in the convolution filter is present in that input field. The integration layer enables these features to be read by CNN's consistent translation (e.g. whether the human eye is present [x = 10, y = 10] or [x = 12, y = 11] positions, the output layer output will be the same). Fully integrated layers are responsible for generating various activation patterns based on the set of feature maps used and the areas in the image, feature maps are designed for it. This looks like CNN's

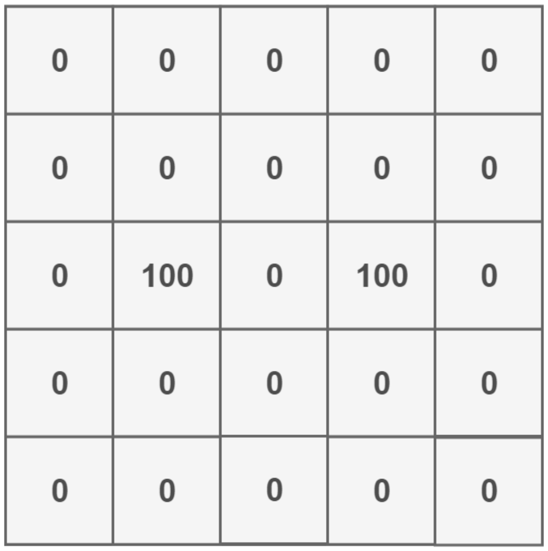

RGB Input
Convolutional Layer with 5 Filters
Pooling Layer
Convolutional Layer with 10 Filters
Pooling Layer
Fully Connected Layers
Output Layer

Convolution Layer:

The Convolution function removes the maximum value of a given position when a convolution feature is present in that area, the other subtracts the lower value. Specifically, in the given area of the convolution kernel, we take the smart multiplication of each kernel cell value and the corresponding pixel value of the kernel cell, and then take the sum of that. The exact value is determined by the kernel width of the following formula and height, h convolution output, input, convolution kernel.

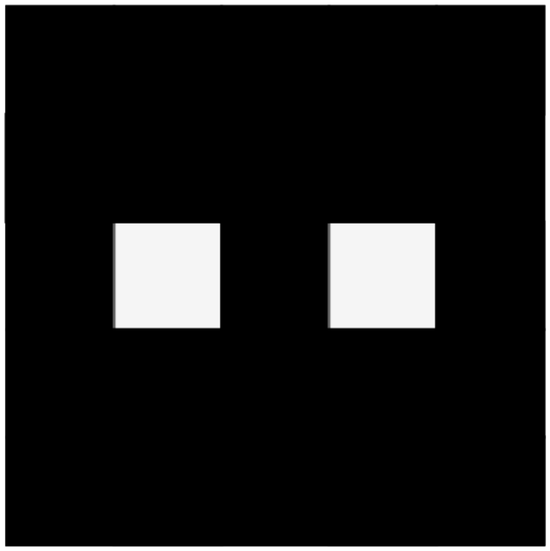
$$\mathrm{h}_{i,j} = \sum_{k=1}^{m} \sum_{l=1}^{m} w_{k,l} x_{i+k-1,j+l-1}$$

It is not enough to know what the convolution operation does, we also need to understand what the convolution output represents. Just imagine colours for the values in the convolution output (0 \black, 100 \ white). If visualized ,this image will represent a binary image that lights up at the location the eyes are at.



0	0	0	0	0
0	0	0	0	0
0	100	0	100	0
0	0	0	0	0
0	0	0	0	0

Convolution Output



Convolution Output
(as an Image)

Pooling Layer:

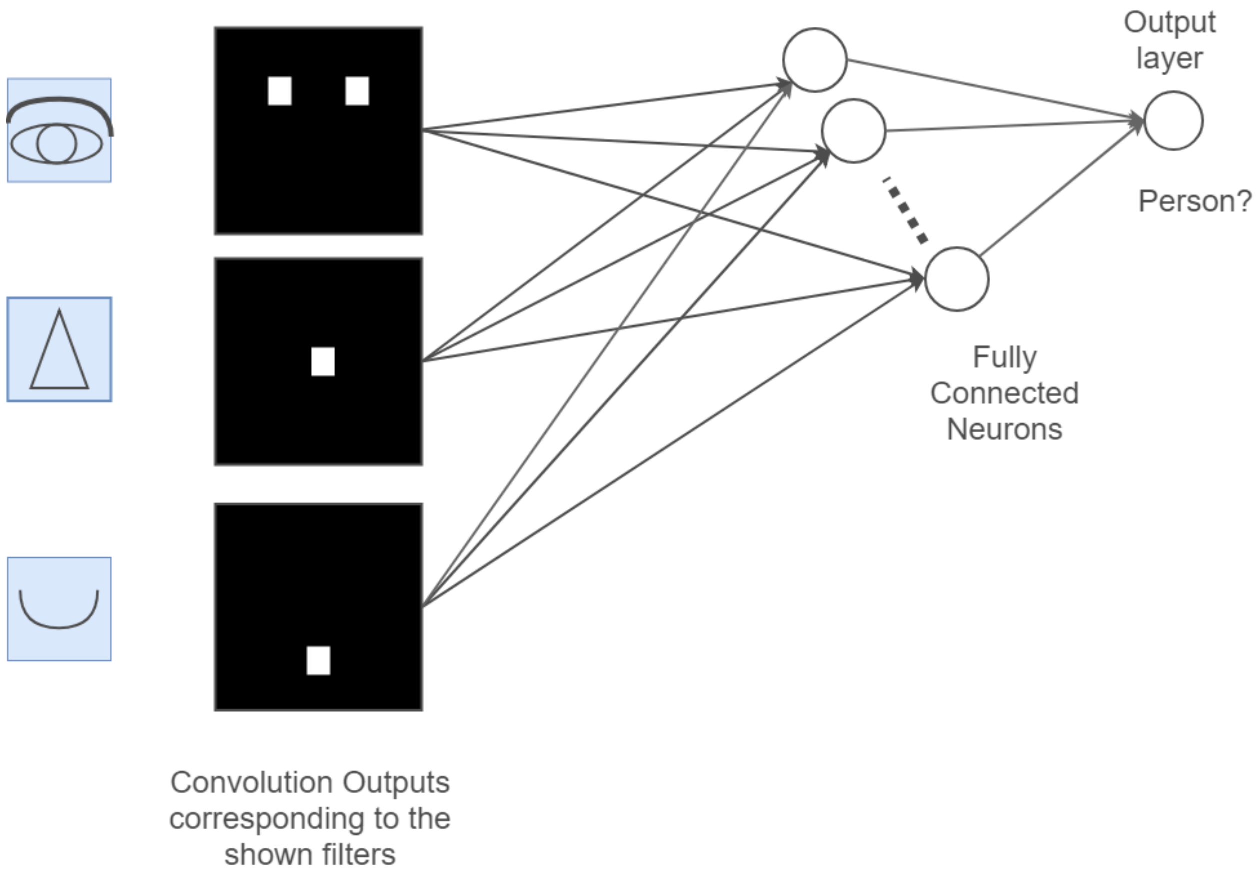
The Pooling layer (or sometimes called subsampling) makes CNN less flexible in terms of convolution. There are two different integration methods used (max-pooling and average-pooling). Specifically, merging function, in a given area, yields a higher number of inputs that fall within the kernel mathematically

$$\mathrm{h}_{i,j} = max\{x_{i+k-1,j+l-1} \forall 1 \leq k \leq m \text{ and } 1 \leq l \leq m\}$$

Fully Connected Layers:

Fully connected layers will incorporate features studied by the various letters of convolution so that the network can create a global representation of the whole picture. The neurons in the fully connected layer will be activated based on whether the various enterprises represented by the elements of convolution actually exist in the input. As fully connected neurons are activated for this, they will produce different start-up patterns based on what features are present in the input images. This provides an integrated representation of the presence in the image, in the output layer, that the output layer can easily use to easily distinguish the image.

Weaving Together Now all we have to do is put all of this together, building a model for the end, from immature images, to decisions. And when connected CNN will look like this. In summary, the layers of convolution will learn various local features in the data (e.g., what the eye looks like), and the integration layer will make CNN more consistent in the translation of these features (e.g. if the eye appears slightly translated into two images fully connected, say, we got two eyes, nose and mouth, so this has to be human, and to work for the right result.



Experimental Development

TENSORFLOW: TensorFlow is an open source software library for high number counting. Its flexible design allows easy deployment of various platforms (CPUs, GPUs, TPUs), and from desktops to server clusters on mobile and side devices.

Originally developed by researchers and engineers of the Google Brain team within the Google AI organization, it comes with strong support for machine learning and in-depth learning and the flexible numerical context is used in many other scientific fields.

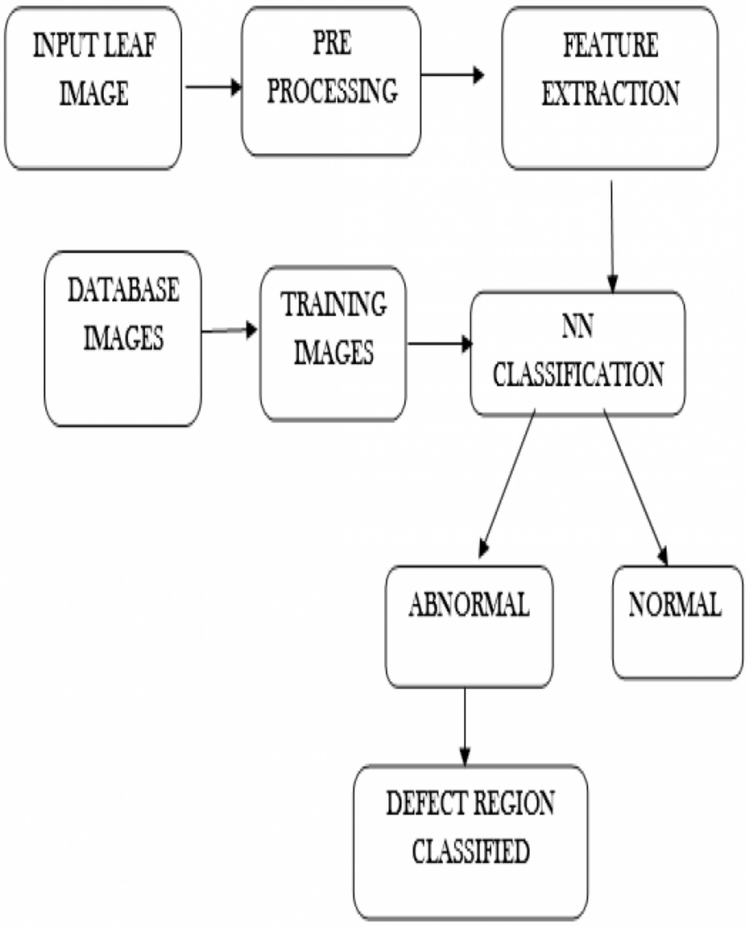
KERAS:

Kera is a high-quality network API, written in Python language and capable of running over TensorFlow, Theano or CNTK. Built with a focus on enabling rapid testing. Being able to move from vision to success with minimal delays that may be essential to doing good research. Kera was developed and maintained by Francois Chollet and is part of the Tensorflow backbone, enabling Tensorflow to select a high-quality API. Use Kera if you need an in-depth reading library: Allows quick and easy scanning (using user friendliness, layout, extension). Duplicate networks and Convolution are supported by both. It can work on both CPU and GPU

Python:

Python is a translated, high-performance, common programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes the readability of the code with its remarkable use of white color. Its language structure and object-oriented approach aim to assist editors in writing clear, logical code for small and large projects. Python typed harder and collected garbage. It supports multiple editing paradigms, including process, object-focused, and efficient operation. Python is often described as a "battery-powered" language because of a common library.

Flowchart of proposed system


INPUT LEAF IMAGE
PRE PROCESSING
FEATURE EXTRACTION
DATABASE IMAGES
TRAINING IMAGES
NN CLASSIFICATION
ABNORMAL
NORMAL
DEFECT REGION CLASSIFIED

PROPOSED APPROACH:

Step 1: Take the input as a picture (picture of a disease pest).
Step 2: Processing images of plants.
Step 3: Train the model with a new plant disease.
Step 4: CNN verification phase where we can increase efficiency before performing any tests, similar to the development environment.
Step 5: Check the model
Step 6: Shipping the model

PERFORMANCE ANALYSIS

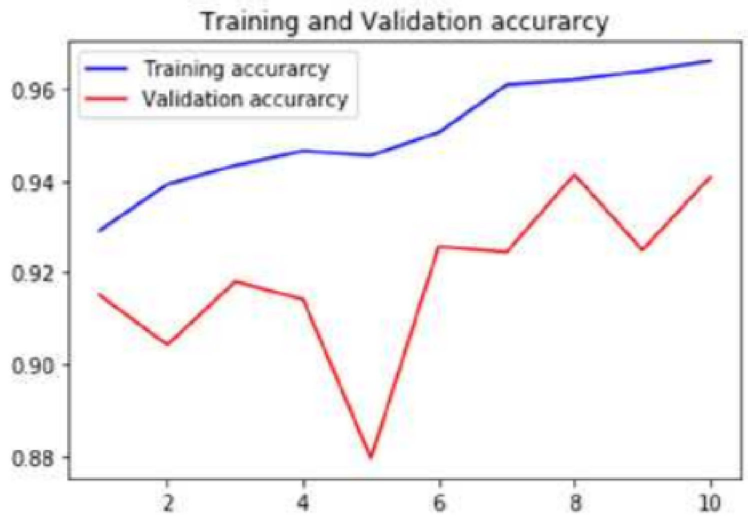
Existing system method:
They focused on the construction of two buildings: the construction of AlexNet and the construction of GoogLeNet.
The AlexNet architecture has 5 layers of convolution, followed by 3 layers of fully connected (FC), and finally ends with a softMax layer. The first two layers of convolution (conv {1,2}) each are followed by the norm and the cohesive layer, and the last layer of convolution (conv5) is followed by the single cohesive layer. A fully integrated final layer (fc8) has 38 effects in our modified version of AlexNet (equivalent to the total number of classes in our database), which feeds the softMax layer. All the first 7 layers of AlexNet have a ReLu non-linearity activation unit associated with it, and the first two fully connected layers (fc {6, 7}) have an associated drop-out layer, with a drop-off rate of 0.5. AlexNet buildup contains 60 million parameters.
The build of GoogLeNet is much deeper and wider than AlexNet. It has 22 layers. while still having a much lower number of parameters (5 million parameters) on the network than AlexNet (60 million parameters). Key Feature used for the construction of GoogLeNet by the original modules. The first module uses 1 1, 3 3, and 5 5 combinations and a multi-component layer at the same time, which is why it enables it to capture different features equally. A total of 9 implementation modules are used in the version of the GoogLeNet architecture we use in our experiment. They have a total of 60 test settings, which differ from the following parameters:
1) Selection of in-depth learning structures:
AlexNet GoogLeNet.

2) Choice of training method:
Transfer Learning, Training from Scratch.

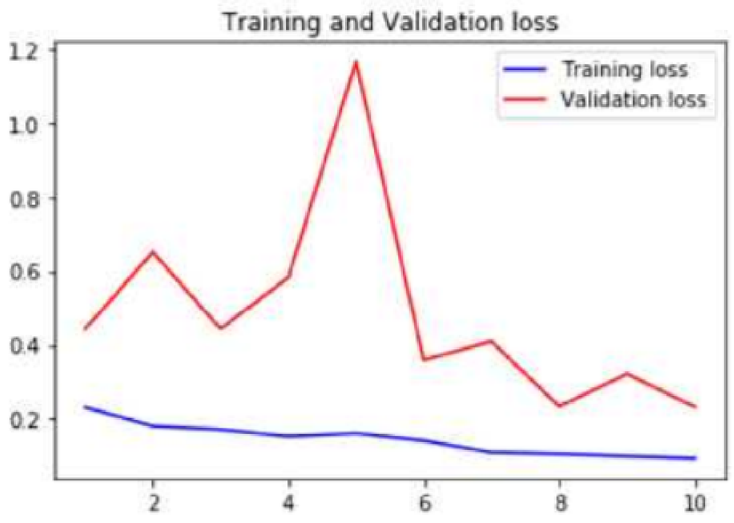
3) Data type selection:
Grayscale Scale Divided.

4) Distribution options for test setup:
Train: 80%, Test: 20%, Train: 60%, Test: 40%, Train: 50%, Test: 50%, Train: 40%, Test: 60%, Train: 20%, Test: 80%. They ran a total of 30 epochs, each of these 60 tests. Also use the following parameters for all tests: Solver Type: Stochastic Gradient Descent, basic reading rate: 0.005, reading rate policy parameter: Step (decreased by 10 factor every 30/3 times), Momentum: 0.9, Weight loss: 0.0005, Gamma: 0.1, Batch parameter size: 24 (in the case of GoogLeNet), 100 (in the case of AlexNet).

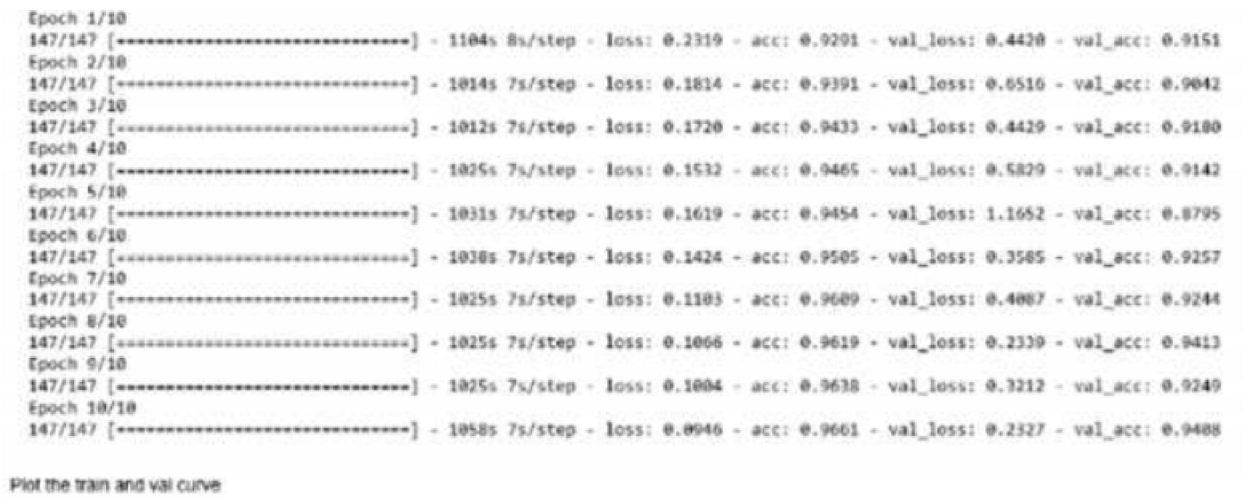
Obtained results



Accuracy of existing system



Loss of existing system



```
Epoch 1/10
147/147 [==============================] - 1104s 8s/step - loss: 0.2319 - acc: 0.9291 - val_loss: 0.4420 - val_acc: 0.9151
Epoch 2/10
147/147 [==============================] - 1014s 7s/step - loss: 0.1814 - acc: 0.9391 - val_loss: 0.6516 - val_acc: 0.9042
Epoch 3/10
147/147 [==============================] - 1012s 7s/step - loss: 0.1720 - acc: 0.9433 - val_loss: 0.4429 - val_acc: 0.9180
Epoch 4/10
147/147 [==============================] - 1025s 7s/step - loss: 0.1532 - acc: 0.9465 - val_loss: 0.5829 - val_acc: 0.9142
Epoch 5/10
147/147 [==============================] - 1031s 7s/step - loss: 0.1619 - acc: 0.9454 - val_loss: 1.1652 - val_acc: 0.8795
Epoch 6/10
147/147 [==============================] - 1038s 7s/step - loss: 0.1424 - acc: 0.9505 - val_loss: 0.3585 - val_acc: 0.9257
Epoch 7/10
147/147 [==============================] - 1025s 7s/step - loss: 0.1103 - acc: 0.9609 - val_loss: 0.4087 - val_acc: 0.9244
Epoch 8/10
147/147 [==============================] - 1025s 7s/step - loss: 0.1066 - acc: 0.9619 - val_loss: 0.2339 - val_acc: 0.9413
Epoch 9/10
147/147 [==============================] - 1025s 7s/step - loss: 0.1004 - acc: 0.9638 - val_loss: 0.3212 - val_acc: 0.9249
Epoch 10/10
147/147 [==============================] - 1058s 7s/step - loss: 0.0946 - acc: 0.9661 - val_loss: 0.2327 - val_acc: 0.9408
```

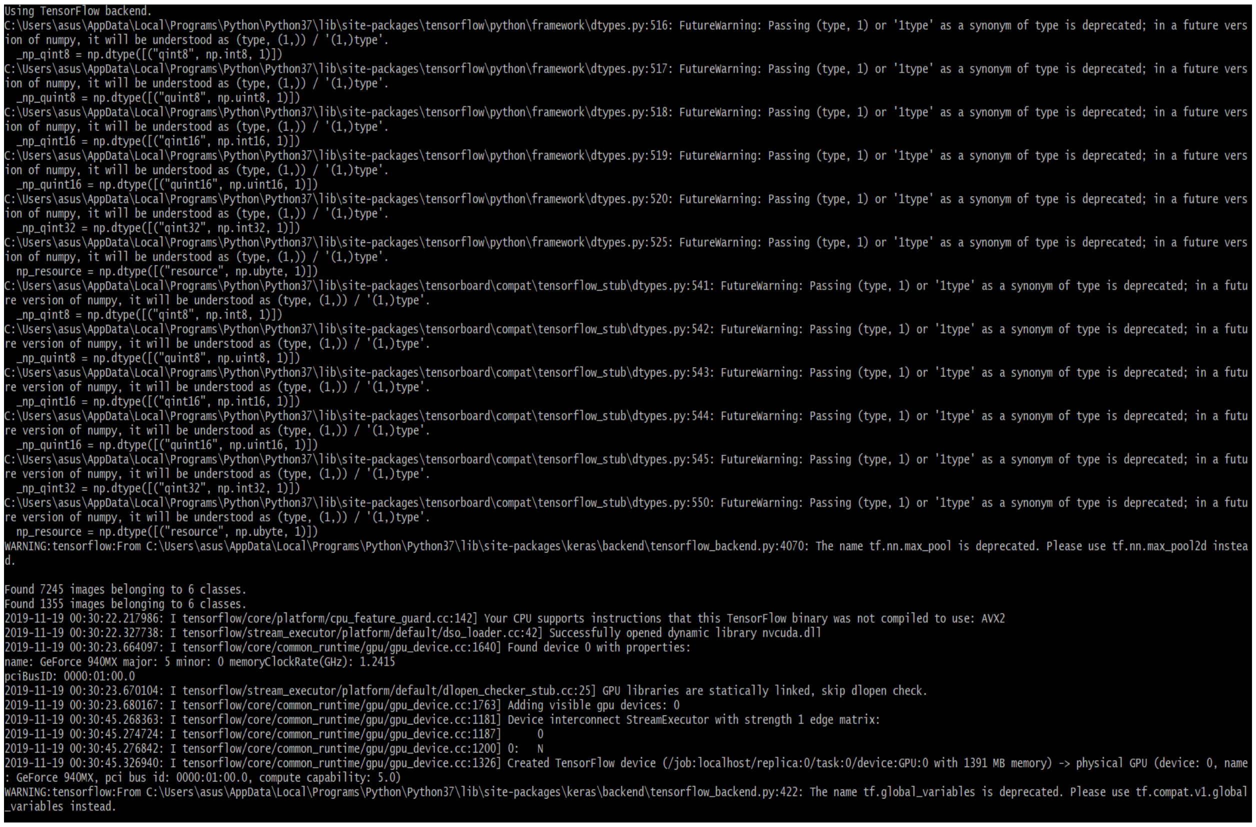
Plot the train and val curve

Training in existing system

Self Made System

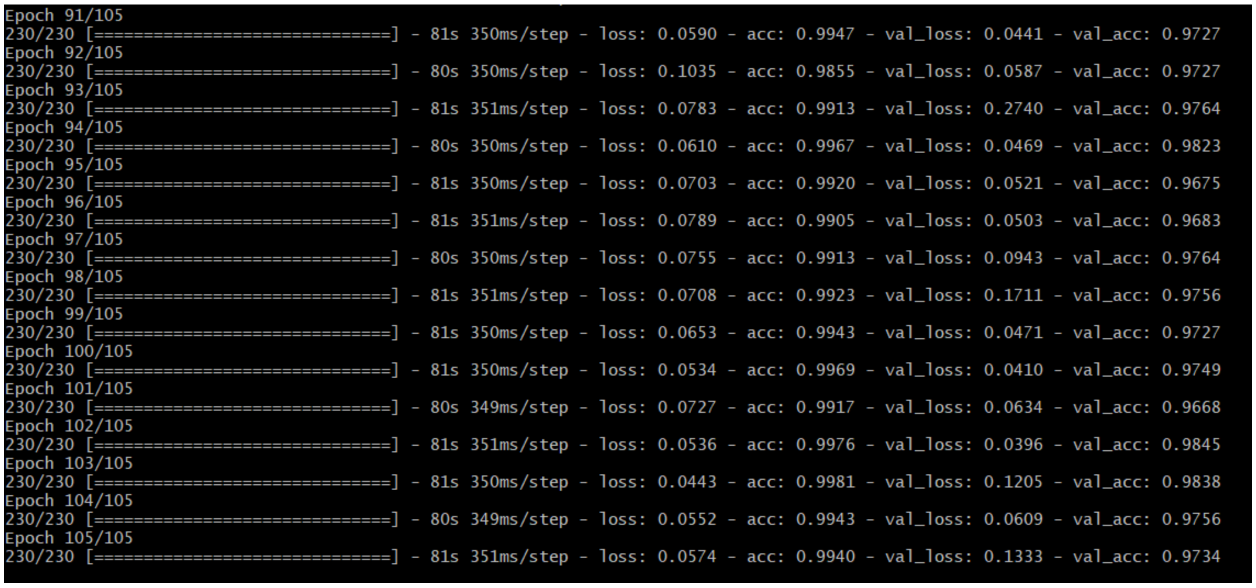
Training Phase

Total epochs =105 Batch size=230 Avg time taken for each epoch= 81s



```
Using TensorFlow backend.
C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future vers
ion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future vers
ion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future vers
ion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future vers
ion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future vers
ion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future vers
ion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:541: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a futu
re version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:542: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a futu
re version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:543: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a futu
re version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:544: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a futu
re version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:545: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a futu
re version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:550: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a futu
re version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
WARNING:tensorflow:From C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:4070: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instea
d.

Found 7245 images belonging to 6 classes.
Found 1355 images belonging to 6 classes.
2019-11-19 00:30:22.217986: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
2019-11-19 00:30:22.327738: I tensorflow/stream_executor/platform/default/dso_loader.cc:42] Successfully opened dynamic library nvcuda.dll
2019-11-19 00:30:23.664097: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1640] Found device 0 with properties:
name: GeForce 940MX major: 5 minor: 0 memoryClockRate(GHz): 1.2415
pciBusID: 0000:01:00.0
2019-11-19 00:30:23.670104: I tensorflow/stream_executor/platform/default/dlopen_checker_stub.cc:25] GPU libraries are statically linked, skip dlopen check.
2019-11-19 00:30:23.680167: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1763] Adding visible gpu devices: 0
2019-11-19 00:30:45.268363: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1181] Device interconnect StreamExecutor with strength 1 edge matrix:
2019-11-19 00:30:45.274724: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1187]      0
2019-11-19 00:30:45.276842: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1200] 0:   N
2019-11-19 00:30:45.326940: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1326] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 1391 MB memory) -> physical GPU (device: 0, name
: GeForce 940MX, pci bus id: 0000:01:00.0, compute capability: 5.0)
WARNING:tensorflow:From C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global
_variables instead.
```

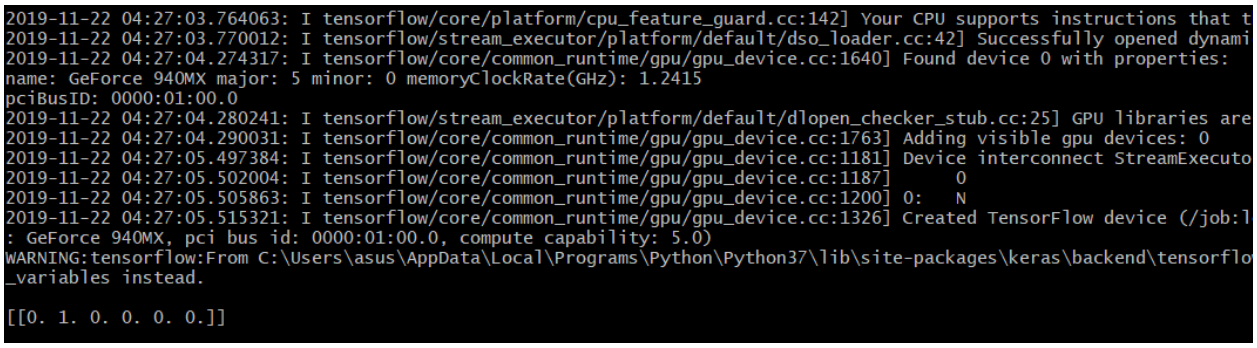


```
Epoch 91/105
230/230 [==============================] - 81s 350ms/step - loss: 0.0590 - acc: 0.9947 - val_loss: 0.0441 - val_acc: 0.9727
Epoch 92/105
230/230 [==============================] - 80s 350ms/step - loss: 0.1035 - acc: 0.9855 - val_loss: 0.0587 - val_acc: 0.9727
Epoch 93/105
230/230 [==============================] - 81s 351ms/step - loss: 0.0783 - acc: 0.9913 - val_loss: 0.2740 - val_acc: 0.9764
Epoch 94/105
230/230 [==============================] - 80s 350ms/step - loss: 0.0610 - acc: 0.9967 - val_loss: 0.0469 - val_acc: 0.9823
Epoch 95/105
230/230 [==============================] - 81s 350ms/step - loss: 0.0703 - acc: 0.9920 - val_loss: 0.0521 - val_acc: 0.9675
Epoch 96/105
230/230 [==============================] - 81s 351ms/step - loss: 0.0789 - acc: 0.9905 - val_loss: 0.0503 - val_acc: 0.9683
Epoch 97/105
230/230 [==============================] - 80s 350ms/step - loss: 0.0755 - acc: 0.9913 - val_loss: 0.0943 - val_acc: 0.9764
Epoch 98/105
230/230 [==============================] - 81s 351ms/step - loss: 0.0708 - acc: 0.9923 - val_loss: 0.1711 - val_acc: 0.9756
Epoch 99/105
230/230 [==============================] - 81s 350ms/step - loss: 0.0653 - acc: 0.9943 - val_loss: 0.0471 - val_acc: 0.9727
Epoch 100/105
230/230 [==============================] - 81s 350ms/step - loss: 0.0534 - acc: 0.9969 - val_loss: 0.0410 - val_acc: 0.9749
Epoch 101/105
230/230 [==============================] - 80s 349ms/step - loss: 0.0727 - acc: 0.9917 - val_loss: 0.0634 - val_acc: 0.9668
Epoch 102/105
230/230 [==============================] - 81s 351ms/step - loss: 0.0536 - acc: 0.9976 - val_loss: 0.0396 - val_acc: 0.9845
Epoch 103/105
230/230 [==============================] - 81s 350ms/step - loss: 0.0443 - acc: 0.9981 - val_loss: 0.1205 - val_acc: 0.9838
Epoch 104/105
230/230 [==============================] - 80s 349ms/step - loss: 0.0552 - acc: 0.9943 - val_loss: 0.0609 - val_acc: 0.9756
Epoch 105/105
230/230 [==============================] - 81s 351ms/step - loss: 0.0574 - acc: 0.9940 - val_loss: 0.1333 - val_acc: 0.9734
```

Screenshot of training

Testing Phase

Image entered

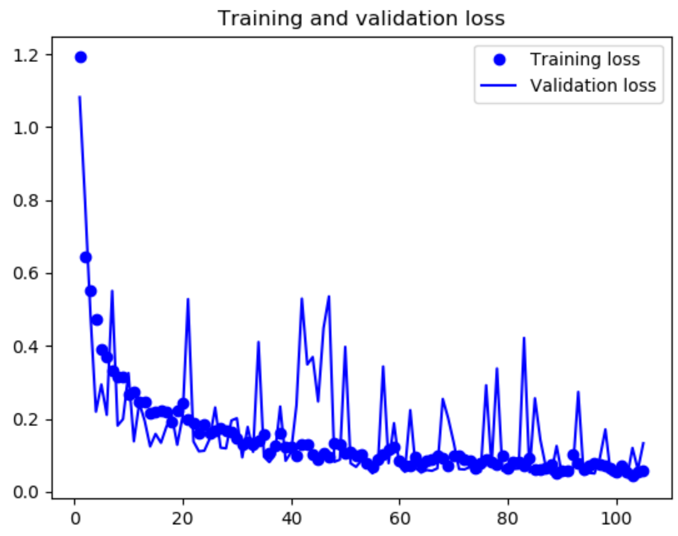


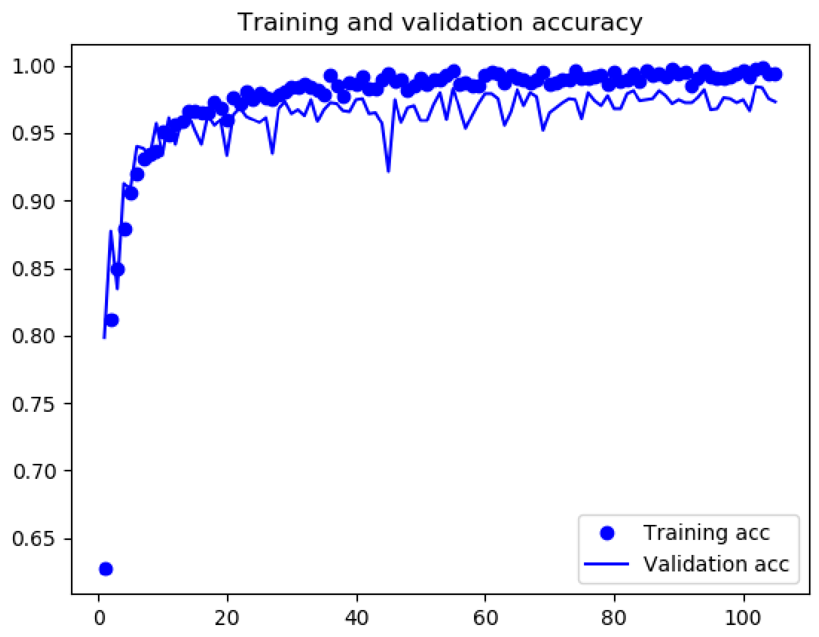
```
2019-11-22 04:27:03.764063: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that t
2019-11-22 04:27:03.770012: I tensorflow/stream_executor/platform/default/dso_loader.cc:42] Successfully opened dynami
2019-11-22 04:27:04.274317: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1640] Found device 0 with properties:
name: GeForce 940MX major: 5 minor: 0 memoryClockRate(GHz): 1.2415
pciBusID: 0000:01:00.0
2019-11-22 04:27:04.280241: I tensorflow/stream_executor/platform/default/dlopen_checker_stub.cc:25] GPU libraries are
2019-11-22 04:27:04.290031: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1763] Adding visible gpu devices: 0
2019-11-22 04:27:05.497384: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1181] Device interconnect StreamExecuto
2019-11-22 04:27:05.502004: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1187]      0
2019-11-22 04:27:05.505863: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1200] 0:   N
2019-11-22 04:27:05.515321: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1326] Created TensorFlow device (/job:l
: GeForce 940MX, pci bus id: 0000:01:00.0, compute capability: 5.0)
WARNING:tensorflow:From C:\Users\asus\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflo
_variables instead.

[[0. 1. 0. 0. 0. 0.]]
```

According to model the leaf belongs from Class 1 disease.

Obtained Results

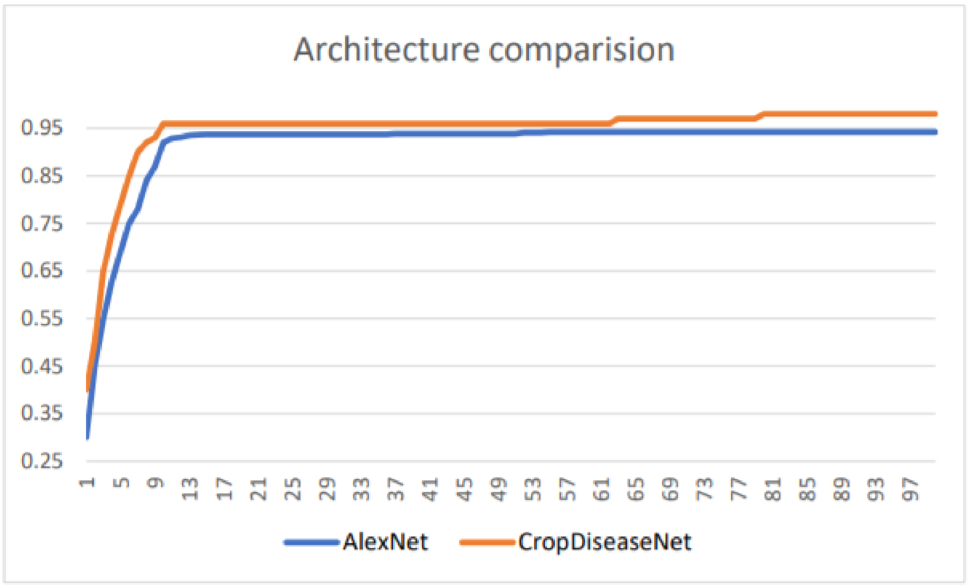




It is clearly visible how the accuracy of model (for both training and validation set) increases as subsequent epoch passes .Vice versa loss of model decreases.

Implementation and Performance details

We have performed the comparison between AlexNet and CropDiseaseNet Architecture. From the comparison we have obtained 99% accuracy for CropDiseaseNet architecture and for Alexnet we obtain 97% accuracy. So performance of CropDiseaseNet architecture is better than Alexnet in case of identifying disease from leaf image.



CONCLUSIONS

Object detection by neural convolution networks is widely used in modern times. With various medical and agricultural applications. With the increase or decrease in the number of layers of convolution, the accuracy can increase. The CNN certification phase is similar to the development area, where we can test accuracy and improve accuracy. Thereafter the accuracy measures in the test model and the final phase will be the phase distribution model. Our CropDiseaseNet Model achieves 98% accuracy.

FUTURE WORK

1) We will discover new regional diseases that are not available on public databases.

2) We will also add common plant diseases. Ex. grasshopper, black rot

3) We will also improve accuracy.

4) We will use a mobile app that detects the disease in real time.

Summary of Papers (REFERENCES)

Tittle	**Detection of Plant Disease using Image processing Approach**
Authors	Sushil R. Kamalpurkar , Ph.D. Karamveer Kakashaeb Wagh Institute of Engineering , Education & Research, Nashik , India.
Year of Publication	February 2016
Summary	The use of automated monitoring and management systems benefits the growing need for technological advancement. In the agricultural sector the loss of crops is largely due to the spread of disease. Especially the diagnosis and diagnosis of the disease is seen when the disease progresses to a severe stage. Therefore, it creates losses in terms of yield, time and money. The proposed system is able to detect the disease as soon as it emerges from the leaf. Therefore saving losses and reducing expert dependence to some extent is possible. It can provide help for someone with little knowledge of the disease. According to these objectives, we must eliminate the elements associated with this disease.
Web Link	http://www.ijsrp.org/research-paper-0216/ijsrp-p5011.pdf

Tittle	**Plant Disease Detection and its Solution using Image Classification**
Authors	Saradhambal.G Dhivya.R Latha.S R.Rajesh
Year of Publication	April 2018
Summary	Data mining technology has been introduced into the agricultural industry. This project uses a new concept to identify affected plants and provides solutions to the agricultural industry. Using the k-mean clustering algorithm, the infected area of the leaf was isolated and analyzed. Images are fed into our diagnostic application. It offers good choice in the agricultural community especially in remote villages. It works as an effective program to reduce contact time with an infected region. The feature removal process helps to remove the infected leaf and to isolate plant diseases. The embedded voice transmission system helps guide us through the entire process. As the future development of the project to expand open multimedia (Audio / Video) about diseases and their solution automatically once the disease is detected

Web Link	https://ijpam.eu/

Tittle	**Leaf Disease Detection using Image Processing**
Authors	Sujatha Radha Y Sravan Kumar Garine Uma Akhil VIT university Vellore Tamil Naidu
Year of Publication	March 2016
Summary	This study summarizes the extensive image processing used to diagnose leaf diseases by k-means clustering, SVM. This method can greatly support the accurate diagnosis of the leaf disease. There are five steps for the identification of a leaf disease called image detection, image processing first, classification, feature removal, separation. By using the amount of disease present in the leaf, we can use a sufficient amount of pesticides to effectively control the insects so that the yield can be increased. We can extend this approach by using different algorithms for segmentation, segmentation.

	By using this concept the identification of the disease is done on all types of leaves and also the user does not know the affected area of the percentage by identifying the disease well the user can easily fix the problem and at a lower cost
Web Link	https://www.researchgate.net/publication/318109025

Tittle	**Plant Disease Detection by Image Processing**
Authors	Monishanker Halder Ananya Sarkar Habibullah Bahar
Year of Publication	Feburary 2019
Summary	Data mining technology has been introduced into the agricultural industry. This project uses a new concept to identify affected plants and provides solutions to the agricultural industry. Using the k-mean clustering algorithm, the infected area of the leaf was isolated and analyzed. Images are fed into our diagnostic application. It offers good choice in the agricultural community especially in remote villages. It works as an effective

	program to reduce contact time with an infected region. The feature removal process helps to remove the infected leaf and to isolate plant diseases. The embedded voice transmission system helps guide us through the entire process.
Web Link	https://www.researchgate.net/publication/330956595

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ……………………….

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: Harshak Bir Singh & Arnav Sharma **Department:** ________________ **Enrolment No** 171253,171265

Contact No. ______________________________ **E-mail.** 171253@juitsolan.in and 171265@juitsolan.in

Name of the Supervisor: Mr. Prateek Thakral

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): ______________________

Detection of plant diseases using Deep Learning

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 32
- Total No. of Preliminary pages = 6
- Total No. of pages accommodate bibliography/references = 32

[illegible] [illegible]

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ………26……..(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

[illegible]

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

……

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ORIGINALITY REPORT

26%	13%	9%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Asian Institute of Technology Student Paper	7%
2	towardsdatascience.com Internet Source	6%
3	Submitted to Jaypee University of Information Technology Student Paper	5%
4	github.com Internet Source	3%
5	Satish Kumar Satti, K. Suganya Devi, Prasenjit Dhar, P. Srinivasan. "A machine learning approach for detecting and tracking road boundary lanes", ICT Express, 2020 Publication	1%
6	www.thushv.com Internet Source	1%
7	www.coursehero.com Internet Source	1%