AGRI-SENTINEL

USING UAV AND ROVER

A MINI PROJECT REPORT

Submitted by

(1) Chiranjeeve R (312422106301)
(2) Deva Narayanan N.A (312422106035)
(3) Hemath Sakthi SS (312422106057)
(4) ASHWIN SHARON R (312422106017)

In partial fulfilment of co-curricular activities Organized by

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERRING

St. JOSEPH'S INSTITUTE OF TECHNOLOGY

OMR , CHENNAI – 600119

Abstract

Agriculture field has a high impact on our life. Agriculture is the most important sector of our Economy. Proper management leads to a profit in agricultural products. Farmers do not expertise in leaf disease so they produce less production. Plant leaf diseases detection is the important because profit and loss are depends on production. CNN is the solution for leaf disease detection and classification. Main aim of this research is to detect the apple, grape, corn, potato and tomato plants leaf diseases. Plant leaf diseases are monitoring of large fields of crops disease detection, and thus automatically detected the some feature of diseases as per that provide medical treatment. Proposed Deep CNN model has been compared with popular transfer learning approach such as VGG16. Plant leaf disease detection has wide range of applications available in various fields such as Biological Research and in Agriculture Institute. Plant leaf disease detection is the one of the required research topic as it may prove benefits in monitoring large fields of crops, and thus automatically detect the symptoms of diseases as soon as they appear on plant leaves.

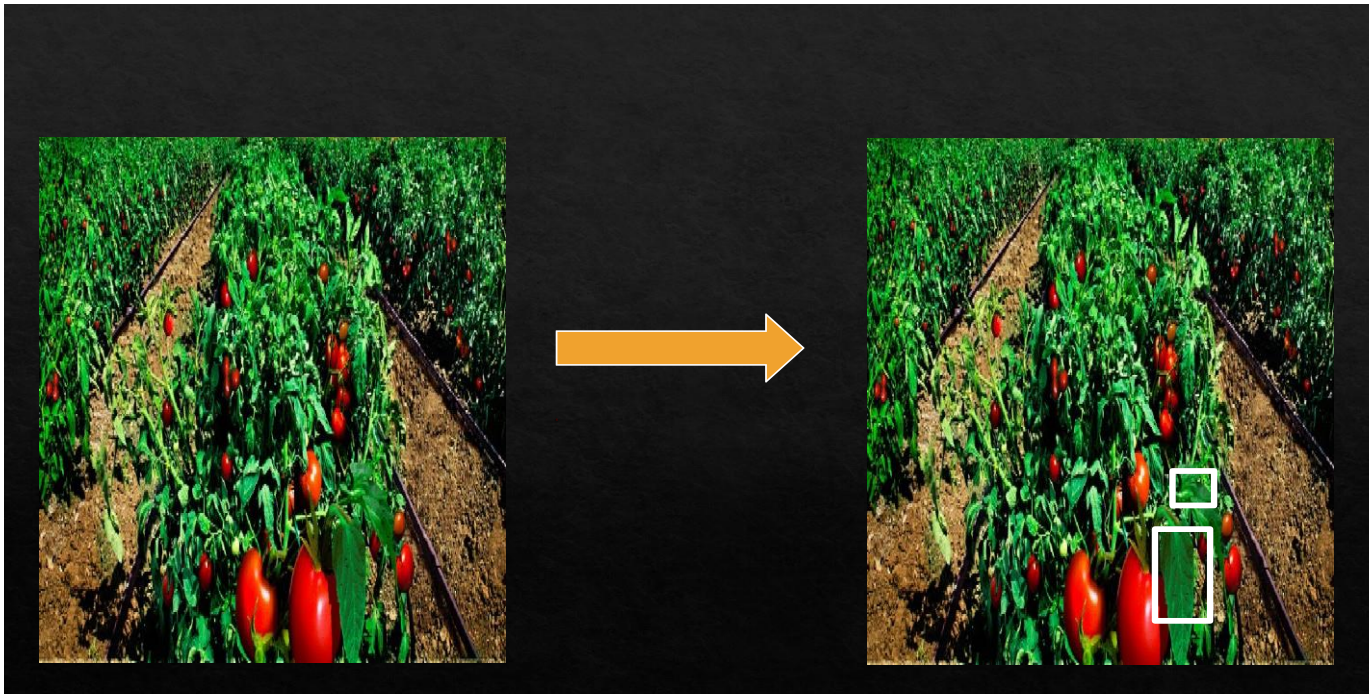
Acknowledgments

This thesis is based on research work conducted for"Leaf Disease Detection using Convolutional Neural Network". This work would not be possible without two people whose contributions can't be ignored.

I consider it an honor to work under my guides Dr. U.K Jaliya and Prof. Pranay Patel. This thesis is fruit of their valuable guidelines and directions. They were always available to monitor me for this work. I specially acknowledge Dr. U.K Jaliya for guide me in leaf disease detection research work and always support me. and thanks to Prof. Pranay Patel for always solve the coding problem in my dissertation work.

I am greatful to Birla Vishvakarma Mahavidhyalaya College of Engineering for providing everything from faculty guides to resources for this work. I would like to thank all other people who havedirectly or indirectly helped me to realize this work.

ARIELVIEW:



ROVERVIEW:

TABLE OF CONTENTS

Chapter 1: Introduction

1.1 Introduction of Leaf Disease Detection

The most important sector of our Economy is Agriculture. Various types of disease damages the plant leaves and effects the production of crop there for Leaf disease detection is important. Regular maintenance of plant leaves is the profit in agricultural products. Farmers do not expertise in leaf disease so they producing lack of production. Leaf disease detection is important because profit and loss depend on production. So that here use deep learning techniques to detectapple, grape, corn [11], potato, and tomato plant leaves diseases. That contains twenty-four types of leaf diseases and twenty-four thousand leaves images are used [13].

Apple, grape, corn, potato, and tomato plant leaves which are categorized total 24types of labels apple label namely: Apple scab, Black rot, apple rust, and healthy. Grape label namely: Black rot, Esca, healthy, and Leaf blight. Corn label namely: Corn Cercospora spot Gray spot, Corn rust, Corn healthy, Corn Northern Blight[11][13]. Potato label namely: Early blight, healthy, andLate blight. Tomato label namely: bacterial spot, early blight, healthy, late blight, leaf mold, septoria leaf spot, spider mite, target sport, mosaic virus[11][13].

The dataset consist of 31,119 images of apple, grape, potato and tomato, all Images are resized into 256 x 256,that images divided into two parts training and testing dataset[11][13].

1. Apple scab 2. Grape Esca 3.Corn leaf spot 4.potato Early Blight 5.Tomato Bacterial Spot

Fig.1 Leaves with Disease part [11]

In fig.1 we can see vegetable and fruit leaves like potato, tomato, corn, apple, grape with diseased part this disease can be easily detected using deep learning techniques [13].

This disease detected using convolutional neural network (CNN), and also this model is compared with VGG16. Images are resized into 224 x 224[13].

1.2 Applications

- Biological research
- Plant leaf disease detection also useful in agriculture institute.
- Some plant leaf disease detection automatic techniques are beneficial for large work of monitoring in farm of crops disease detection.

1.3 Objectives

- The objective of this research is to concentrate based on potato, tomato, corn, grapes, and apple leaf disease detection using CNN.
- CNN is used for examine the healthy and diseased plants leaves.

1.4 Motivation

- Identifying and recognition of leaves disease is the solution for saving the reduction of large farm in crop disease detection and profit in productivity, it is beneficial in agricultural institute, Biological research .

Chapter 2: HARDWARE (COMPONENT)

COMPONENTS USED

Drone components:

Drone Frame -Arms

-Motors

-Drone Propellers

Battery

-Flight Controller Board

-Sensors

-Camera

-Gimbal

-Speed controllers

Drone control station

Rover components:-

Arduino Uno

-DHT22 temperature and humidity Soil moisture sensor

-CNC mechanism

-Double shift gear motors

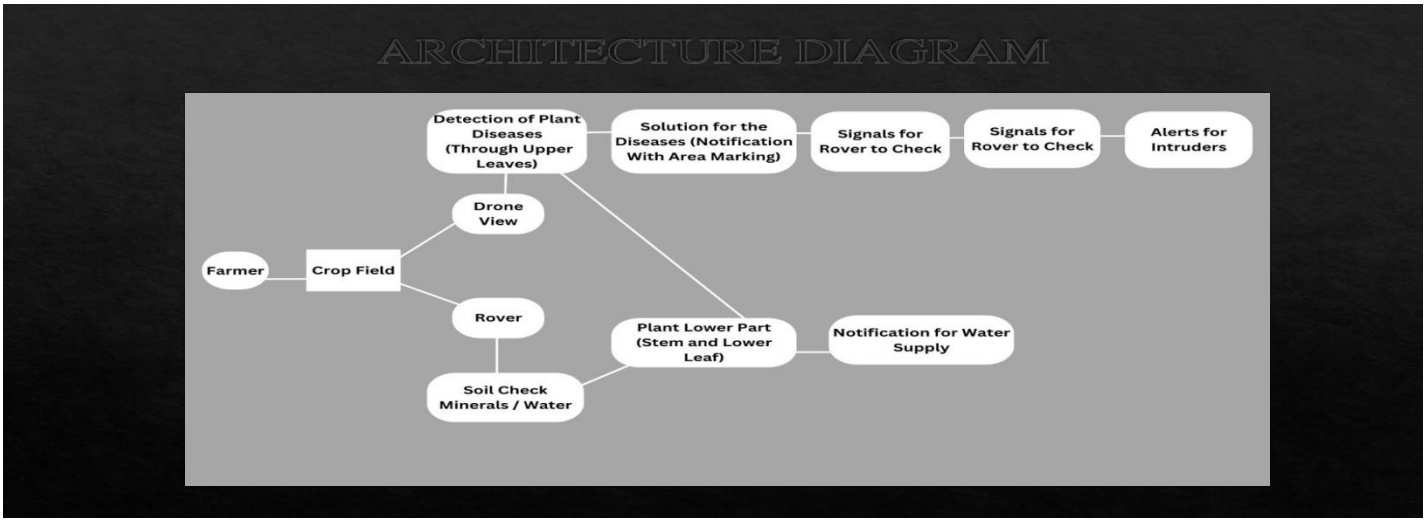
-Motorshield

-Servomotor

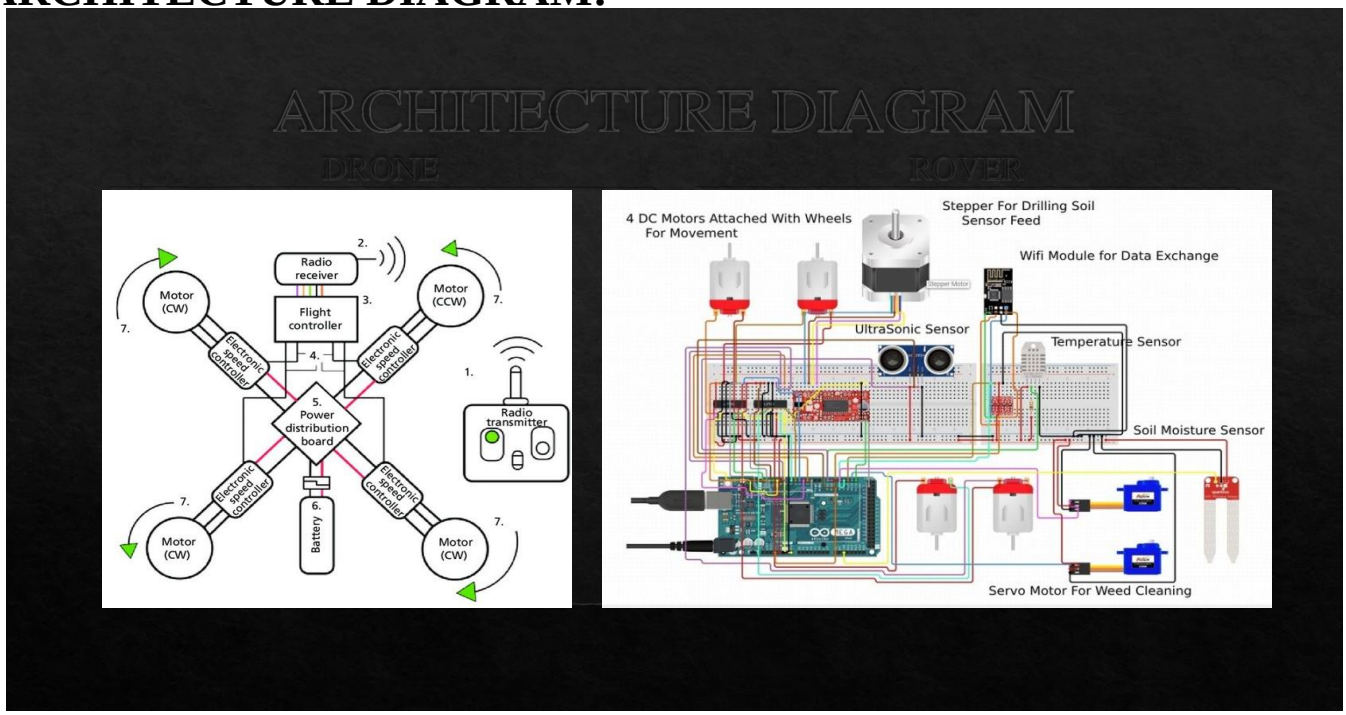
-Detectioncamera

-ESP8266module

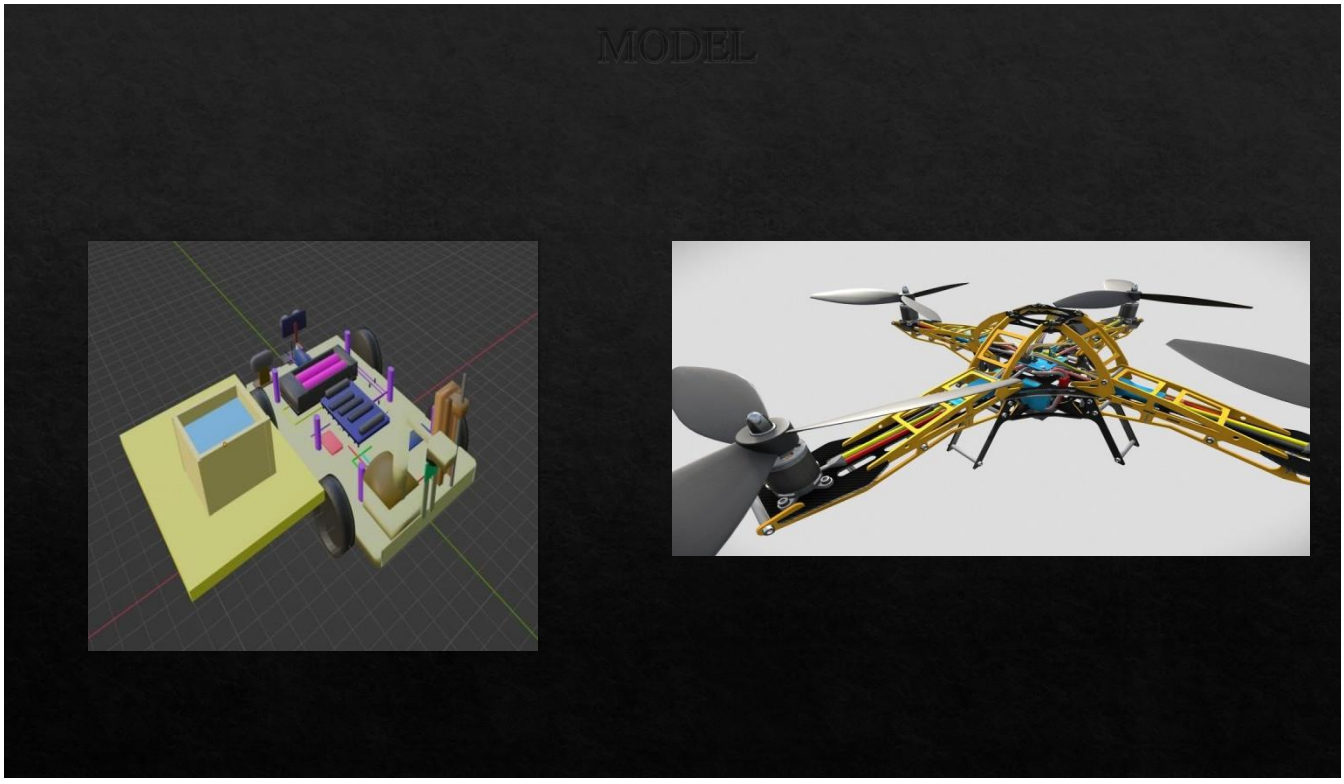
BLOCKDIAGRAM:



ARCHITECTURE DIAGRAM:



ROVER AND DRONE MODEL:



Chapter 3: Existing Work & Implementation work

3.1 Overview of Existing Work

Existing work related to leaf disease detection using CNN show to detect and classify leaf disease using image processing techniques that follow steps like

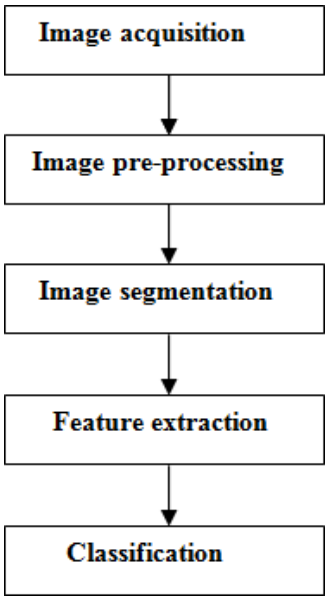


Fig.2 General Block Diagram of Feature Based Approach [13]

Image Acquisition: image acquisition in the first load the image in digital picture process and that consist capturing the image through digital camera and stores it in digital media for additional MATLAB operations [13][7].

B. Image Preprocessing: The main aim of image pre-processing is to enhance the image information contained unwanted distortions or to reinforce some image features for any processing [7][13]. Preprocessing technique uses various techniques like dynamic image size and form, filtering of noise, image conversion, enhancing image and morphological operations[7][13].

C. Image Segmentation: In image segmentation is used K-means cluster technique for partitioning of pictures into clusters during which a minimum of one part of cluster contain image with major space of unhealthy part [7][13]. The k means cluster algorithmic rule is applied to classify the objects into K variety of categories per set of features [13][7].

D. Feature extraction: After clusters are formed texture features are extracted using GLCM [13].

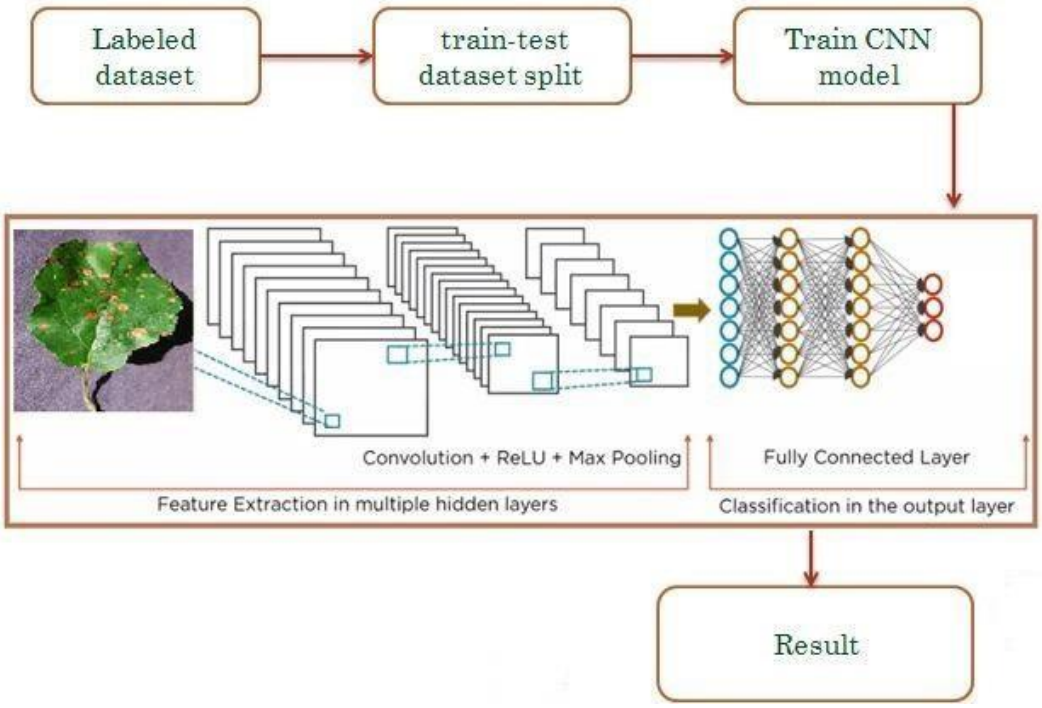
(Gray-Level Co-occurrence Matrix).

E. Classification: In classification is used for testing the leaf disease. The Random forest classifier is used for classification.[7][13]

3.2 Implementation work

Apple, grape, potato, and tomato plant leaves which are categorized total 24 types of labels apple label namely: Apple scab, Black rot, Apple rust, and healthy. Corn label namely: Corn Cercospora Gray spot, Corn rust, Corn healthy, Corn Blight [11][13]. Grape label namely: Black rot, Esca, healthy, and Leaf blight. Potato label namely: Early blight, healthy, and Late blight. Tomato label namely: bacterial spot, early blight, healthy, late blight, leaf mold, septoria leaf spot, spider mite, target sport, mosaic virus, and yellow leaf curl virus[11][13].

The dataset consist of 31,119 images of apple, corn, grape, potato and tomato, out of 31,119 images 24000 images are used. all Images are resized into 256 x 256,that images divided into two parts training and testing dataset, the whole range of the train test split using 80-20 (80% of the whole dataset used for the training and 20% for the testing)[11][13]. Then train CNN model.



Proposed workflow [13]

Convolutional neural networks (CNN) can be used for the computational model creation thatworks on the unstructured image inputs and converts to output labels of corresponding

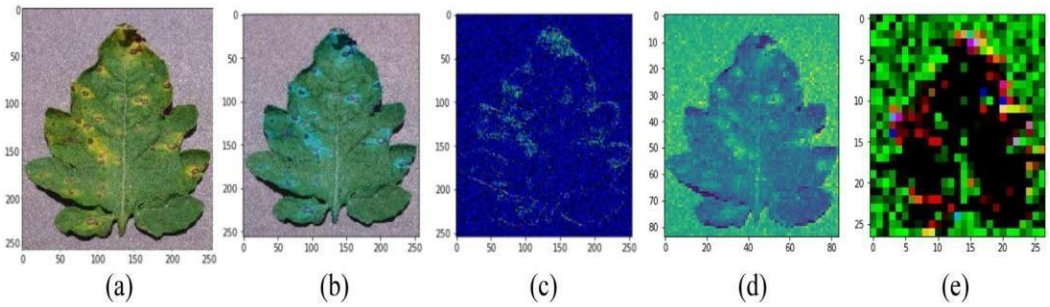
classification[13]. They belong to the category of multi-layer neural networks which can be trained to learn the required features for classification purposes [13].

Less pre-processing is required in comparison to traditional approaches and automatic feature extraction is performed for better performance. For the purpose of leaf disease detection, the best results could be seen with the use of a variation of the LeNet architecture [13].

LeNet consists of convolutional, activation, max-pooling and fully connected layer also LeNet is simple CNN model. This architecture used for the classification of the leaf diseases in LeNet model [13]. It consists of an additional block of convolution, activation and pooling layers in comparison to the original LeNet architecture. The model used in this paper been shown in Fig. 2.Each block consists of a convolution, activation and a max pooling layer. Three such blocks followed by fully connected layers and soft-max activation are used in this architecture. Convolution and pooling layers are used for feature extraction whereas the fully connected layers are used for classification. Activation layers are used for introducing non-linearity into the network [13].

Convolution layer applies convolution operation for extraction of features. With the increase in depth, the complexity of the extracted features increases. The size of the filter is fixed to 5

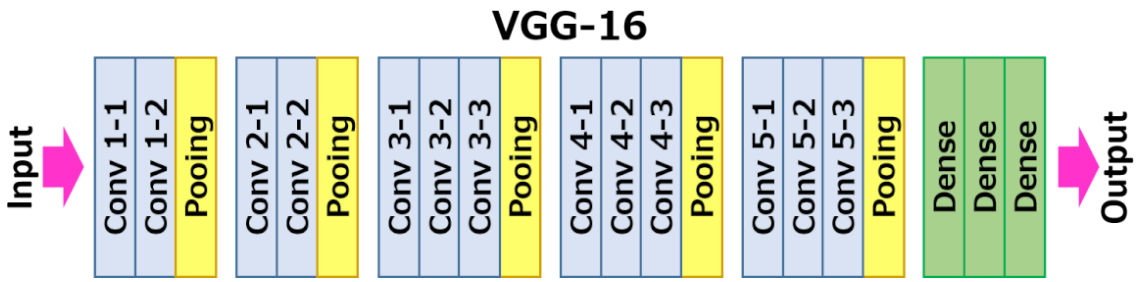
× 5 whereas number of filters is increased progressively as we move from one block to another. The number of filters is 20 in the first convolution block while it is increased to 50 in the second and 80 in the third. This increase in the number of filters is necessary to compensate for the reduction in the size of the feature maps caused by the use of pooling layers in each of the blocks. After the application of the convolution operation feature maps are zero padded, in order to preserve the size of the image. The max pooling layer is used for reduction in size of the feature maps, speeding up the training process, and making the model less variant to minor changes in input. The kernel size for max pooling is 2×2. Re-LUactivation layer is used in each of the blocks for the introduction of non-linearity. Also, Dropout regularization technique has been used with a keep probability of 0.5 to avoid over- fitting the train set. Dropout regularization randomly drops neurons in the network during iteration of training in order to reduce the variance of the model and simplify the network which aids in prevention of over fitting. Finally, the classification block consists of two sets fully connected neural network layers each with 500 and 10 neurons respectively. The second dense layer is followed by a soft max activation function to compute the probability scores for the ten classes [13].



Experimental result(a)input image(b)convolution layer-1(c) convolution layer-2 (d) convolution layer-3 (e)flatting layer [3].

Further, in every experiment, the overall accuracy over the whole period of training and testing regular intervals (for every epoch) will be computed. The overall accuracy score will be used for performance evaluation [3].

Transfer learning is a knowledge- sharing method that reduces the size of the training data, contains 224*224 image fix size. To transfer the learning of a pre-trained model to a new model Transfer learning is useful. Transfer learning has been used in various applications, such as plant classification, software defect prediction, activity recognition and sentiment classification [3]. In this, the performance of the proposed Deep CNN model has been compared with popular transfer learning approach VGG16.

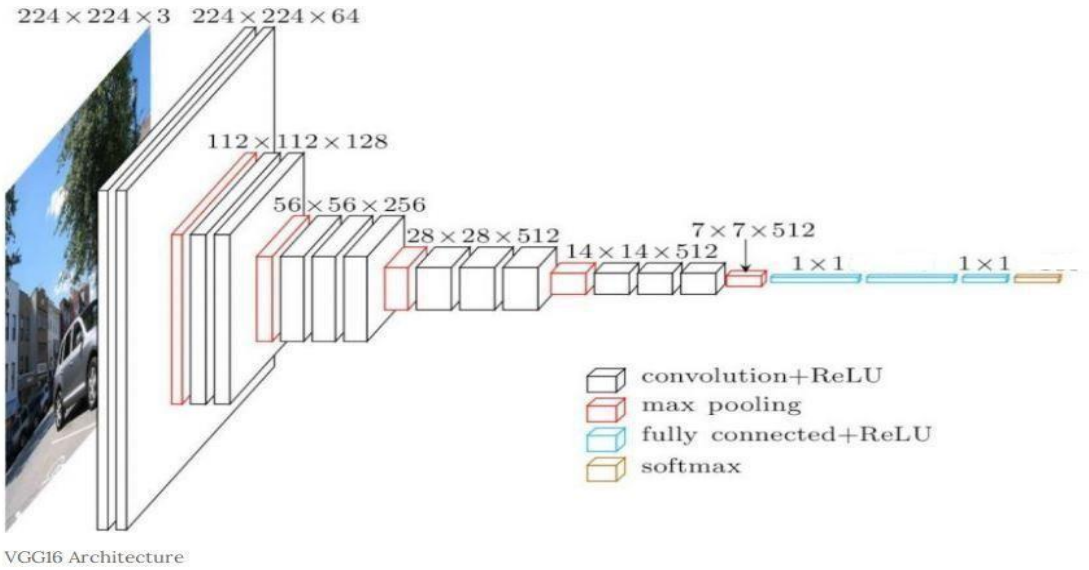


VGG16 layered architecture [18]

VGG16 is a convolutional neural network. The input to convolution layer size is 224 x 224 RGB fixed image size. The image is passed to convolutional layers, where the filters used with a very small receptive field which is the smallest size to capture the notion of left, right, up, and down, center: 3×3 [18]. Some of the configurations, it utilizes 1×1 convolution filters, which can be linear transformation followed by non-linearity of the input channels [18]. The convolution stride is fixed that is one pixel the spatial padding of convolution layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1 pixel for 3×3 convolution layers. Five max-pooling layers carried out spatial pooling, which follow some of the convolution layers (not all the conv. layers are followed by max-pooling).

Over a 2×2 pixel Max-pooling is performed [18].

Three layers follow a stack of convolutional layers. The first two have 4096 channels and third performs 1000-way ILSVRC classification and thus contains 1000 channels. The last layer is the soft-max layer. The configuration of the fully connected layers is the identify the leaf disease [18].



.VGG16 architecture [18]

All hidden layers are equipped with the rectification Re-Lu Is rectified linear unit is contain non- linearity on network, and Also noted that none of the networks contain Local Response Normalization (LRN), such LRN does not improve the performance on the dataset [18].

Chapter 4: Dataset, Implementation and Result

4.1 Dataset Detail

The Plant leaf diseases dataset with augmentation data-set, 39 different classes of plant leaf and background images are available. The data-set containing 61,486 images. We used six different augmentation techniques for increasing the data-set size. These techniques are 1)image flipping,

2) Gamma correction, 3) noise injection, 4) PCA color augmentation, 5) rotation, and 6) Scaling [11][13].

We use The Plant leaf diseases dataset with augmentation dataset only 30,052 images with 24 labels. The apple label namely: Apple scab, Black rot, apple rust, and healthy. Corn label namely: Corn Cercospora spot Gray spot, Corn rust, Corn healthy, Corn Northern Blight[13][11]. Grape label namely: Black rot, Esca, healthy, and Leaf blight. Potato label namely: Early blight, healthy, and Late blight. Tomato label namely: bacterial spot, early blight, healthy, late blight, leaf mold, septoria leaf spot, spider mite, target sport, mosaic virus, yellow leaf curl virus[13][11].

Classes	no. of images	used images
Apple_scab	1000	1000
Apple_black_rot	1000	1000
Apple_cedar_apple_rust	1000	1000
Apple_healthy	1645	1000
Corn_gray_leaf_spot	1000	1000
Corn_common_rust	1192	1000
Corn_northern_leaf_blight	1000	1000
Corn_healthy	1162	1000
Grape_black_rot	1180	1000
Grape_black_measles	1383	1000
Grape_leaf_blight	1076	1000
Grape_healthy	1000	1000
Potato_early_blight	1000	1000
Potato_healthy	1000	1000
Potato_late_blight	1000	1000
Tomato_bacterial_spot	2127	1000
Tomato_early_blight	1591	1000
Tomato_healthy	1909	1000
Tomato_late_blight	1000	1000
Tomato_leaf_mold	1000	1000
Tomato_septoria_leaf_spot	1707	1000
Tomato_spider_mites_two-spotted_spider_mite	1676	1000
Tomato_target_spot	1404	1000
Tomato_mosaic_virus	1000	1000
Total images	30052	24000

.LEAF DISEASE DATASET


Apple scab[11]


Apple black rot[11]


Cader apple rust[11]


Apple healthy


Corn spot[11]


Corn rust[11]


Corn healthy[11]


Corn Blight[11]


Grape rot[11]


Grape Esca [11]


Grape healthy[11]


Grape blight[11]

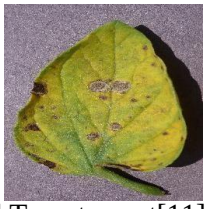

Potato Early Blight [11]


Potato healthy [11]


Potato Late blight[11]


Tomato blight[11]


Tomato Mold[11]


Tomato spot[11]


Tomato Bacterial Spot[11]


Tomato Early blight[11]


Tomato healthy


Tomato Two-spotted spider mite[11]


Tomato Target spot[11]


Tomato mosaic virus[11]

vegetable and fruits leaves with diseases [11] [13].

4.2 Tools & Technologies

4.2.1 PYTHON

Python as a language has a vast community behind it. Any problem which may be faced is simply resolved with a visit to Stack Overflow. Python is among the foremost standard language on the positioning that makes it very likely there will be straight answer to any question

Python has an abundance of powerful tools prepared for scientific computing Packages like NumPy, Pandas and SciPy area unit freely available and well documented. Packages like these will dramatically scale back, and change the code required to write a given program. This makes iteration fast.

Python as a language is forgiving and permits for program that appear as if pseudo code. This can be helpful once pseudo code given in tutorial papers must be enforced and tested. Using python this step is sometimes fairly trivial.

However, Python is not without its errors. The language is dynamically written and packages are area unit infamous for Duck writing. This may be frustrating once a package technique returns one thing that, for instance, looks like an array instead of being an actual array. Plus the actual fact that standard Python documentation does not clearly state the return type of a method, this can lead to a lot of trials and error testing that will not otherwise happen in a powerfully written language. This is a problem that produces learning to use a replacement Python package or library more difficult than it otherwise may be.

4.2.2 NUMPY

Numpy is python package which provide scientific and higher level mathematical abstractionswrapped in python. It is [19] the core library for scientific computing, that contains a strong n-dimensional array object, provide tools for integrating C, C++ etc. It is additionally useful in linear algebra, random number capability etc.

Numpy's array type augments the Python language with an efficient data structure used for numerical work. Numpy additionally provides basic numerical routines, like tools for locating Eigenvector.

4.2.3 SCIKIT LEARN

Scikit-learn [21] could be a free machine learning library for Python. It features numerous classification, regression and clustering algorithms like support vector machine, random forests, and k-neighbors', and it additionally supports Python numerical and scientific libraries like NumPy and SciPy. Scikit-learn is especially written in Python, with some core algorithms written in Python to get performance. Support vector machines are enforced by a python wrapper around LIBSVM .i.e., logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

4.2.4 TENSORFLOW

TensorFlow [22] is an open source software library for numerical computation using data flow graphs. Nodes inside the graph represent mathematical formula, whereas the graph edges represent the multidimensional knowledge arrays (tensors) communicated between them. The versatile architecture permits you to deploy computation to at least one or more CPUs orGPUs in a desktop, server, or mobile device with a single API. Tensor Flow was originally developed by researchers and engineers acting on the Google Brain Team at intervals Google's Machine Intelligence analysis organization for the needs of conducting machine learning and deep neural networks research, however, the system are general enough to be applicable in a wide range of alternative domains as well.

4.2.5 KERAS

Keras is[20] a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with attention on enabling quick experimentation. Having the ability to travel from plan to result with the smallest amount doable delay is key to doing great research. Keras permits for straightforward and quick prototyping (through user-friendliness, modularity, and extensibility).Supports each convolutional networks and recurrent networks, furthermore as combinations of the two Runs seamlessly on CPU and GPU. The library contains numerous implementations of usually used neural network building blocks like layers, objectives, activation functions, optimizers, and a number of tools to create operating with image and text data easier. The code is hosted on GitHub, and community support forums embody the GitHub issues page, a Glitter channeland a Slack channel.

4.2.6. COMPILER OPTION

Anaconda is also a premium open-source distribution of the Python and R programming languages for large-scale process, predictive analytics, and scientific computing, that aims to modify package managing and deployment.

4.2.7. JUPITER NOTEBOOK

The Jupyter Notebook is an open-source web application that enables you to make and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more [23].

4.3 Results

```
/content/drive/My Drive/leafdisease dataset/Apple___Apple_scab
/content/drive/My Drive/leafdisease dataset/Apple___Black_rot
/content/drive/My Drive/leafdisease dataset/Apple___Cedar_apple_rust
/content/drive/My Drive/leafdisease dataset/Apple___healthy
/content/drive/My Drive/leafdisease dataset/Grape___Black_rot
/content/drive/My Drive/leafdisease dataset/Grape___Esca_(Black_Measles)
/content/drive/My Drive/leafdisease dataset/Grape___healthy
/content/drive/My Drive/leafdisease dataset/Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
/content/drive/My Drive/leafdisease dataset/Corn___Cercospora_leaf_spot Gray_leaf_spot
/content/drive/My Drive/leafdisease dataset/Corn___Common_rust
/content/drive/My Drive/leafdisease dataset/Corn___healthy
/content/drive/My Drive/leafdisease dataset/Corn___Northern_Leaf_Blight
/content/drive/My Drive/leafdisease dataset/Potato___Early_blight
/content/drive/My Drive/leafdisease dataset/Potato___healthy
/content/drive/My Drive/leafdisease dataset/Potato___Late_blight
/content/drive/My Drive/leafdisease dataset/Tomato___Bacterial_spot
/content/drive/My Drive/leafdisease dataset/Tomato___Early_blight
/content/drive/My Drive/leafdisease dataset/Tomato___healthy
/content/drive/My Drive/leafdisease dataset/Tomato___Late_blight
/content/drive/My Drive/leafdisease dataset/Tomato___Leaf_Mold
/content/drive/My Drive/leafdisease dataset/Tomato___Septoria_leaf_spot
/content/drive/My Drive/leafdisease dataset/Tomato___Spider_mites Two-spotted_spider_mite
/content/drive/My Drive/leafdisease dataset/Tomato___Target_Spot
/content/drive/My Drive/leafdisease dataset/Tomato___Tomato_mosaic_virus
X_data shape: (24000, 256, 256, 3)
y_data shape: (24000,)
```

Fig 8. Labled images

4.4 This output in images are resized and gives label name to all images.

```
X_train shape: (19200, 256, 256, 3)
X_test shape: (4800, 256, 256, 3)
Y_train shape: (19200,)
Y_test shape: (4800,)
```

Fig 9. Split dataset images

4.5 Fig 9. Contain Total 1500 dataset images are divided into two parts 1200 are in training part and 300 is the testing part.

```
Epoch 138/150
17280/17280 [==============================] - 28s 2ms/step - loss: 0.0016 - accuracy: 1.0000 - val_loss: 0.4220 - val_accuracy: 0.8958
Epoch 139/150
17280/17280 [==============================] - 28s 2ms/step - loss: 0.0015 - accuracy: 1.0000 - val_loss: 0.4293 - val_accuracy: 0.8964
Epoch 140/150
17280/17280 [==============================] - 28s 2ms/step - loss: 0.0015 - accuracy: 1.0000 - val_loss: 0.4233 - val_accuracy: 0.8995
Epoch 141/150
17280/17280 [==============================] - 28s 2ms/step - loss: 0.0015 - accuracy: 1.0000 - val_loss: 0.4267 - val_accuracy: 0.8974
Epoch 142/150
17280/17280 [==============================] - 28s 2ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.4367 - val_accuracy: 0.8995
Epoch 143/150
17280/17280 [==============================] - 28s 2ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.4282 - val_accuracy: 0.8995
Epoch 144/150
17280/17280 [==============================] - 28s 2ms/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.4326 - val_accuracy: 0.8984
Epoch 145/150
17280/17280 [==============================] - 28s 2ms/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.4363 - val_accuracy: 0.8969
Epoch 146/150
17280/17280 [==============================] - 28s 2ms/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.4301 - val_accuracy: 0.9005
Epoch 147/150
17280/17280 [==============================] - 28s 2ms/step - loss: 0.7371 - accuracy: 0.8725 - val_loss: 0.7075 - val_accuracy: 0.8307
Epoch 148/150
17280/17280 [==============================] - 28s 2ms/step - loss: 0.1158 - accuracy: 0.9592 - val_loss: 0.4037 - val_accuracy: 0.8896
Epoch 149/150
17280/17280 [==============================] - 28s 2ms/step - loss: 0.0253 - accuracy: 0.9957 - val_loss: 0.3917 - val_accuracy: 0.8932
Epoch 150/150
17280/17280 [==============================] - 28s 2ms/step - loss: 0.0155 - accuracy: 0.9987 - val_loss: 0.3777 - val_accuracy: 0.9000
```

Fig.10 Train CNN model

4.6 Fig . 10. Consist output of train the convolutional neural network. Train 1080 samples and validate on 120 samples

```
4800/4800 [==============================] - 5s 943us/step
accuracy: 90.229166%
```

Fig .11. Test CNN model

4.7 Fig 11. Consist output of convolutional neural network testing accurancy score 90.23%

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 254, 254, 32)	896
activation_1 (Activation)	(None, 254, 254, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 127, 127, 32)	0
conv2d_2 (Conv2D)	(None, 125, 125, 64)	18496
activation_2 (Activation)	(None, 125, 125, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 62, 62, 64)	0
conv2d_3 (Conv2D)	(None, 60, 60, 128)	73856
activation_3 (Activation)	(None, 60, 60, 128)	0
max_pooling2d_3 (MaxPooling2	(None, 30, 30, 128)	0
conv2d_4 (Conv2D)	(None, 28, 28, 256)	295168
activation_4 (Activation)	(None, 28, 28, 256)	0
max_pooling2d_4 (MaxPooling2	(None, 14, 14, 256)	0
conv2d_5 (Conv2D)	(None, 12, 12, 512)	1180160
activation_5 (Activation)	(None, 12, 12, 512)	0
max_pooling2d_5 (MaxPooling2	(None, 6, 6, 512)	0
flatten_1 (Flatten)	(None, 18432)	0
dense_1 (Dense)	(None, 128)	2359424
dense_2 (Dense)	(None, 256)	33024
dense_3 (Dense)	(None, 20)	5140
activation_6 (Activation)	(None, 20)	0

Total params: 3,966,164
Trainable params: 3,966,164
Non-trainable params: 0

Table.3. CNN model summary table

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 224, 224, 64)	1792
conv2d_2 (Conv2D)	(None, 224, 224, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 112, 112, 64)	0
conv2d_3 (Conv2D)	(None, 112, 112, 128)	73856
conv2d_4 (Conv2D)	(None, 112, 112, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 56, 56, 128)	0
conv2d_5 (Conv2D)	(None, 56, 56, 256)	295168
conv2d_6 (Conv2D)	(None, 56, 56, 256)	590080
conv2d_7 (Conv2D)	(None, 56, 56, 256)	590080
max_pooling2d_3 (MaxPooling2	(None, 28, 28, 256)	0
conv2d_8 (Conv2D)	(None, 28, 28, 512)	1180160
conv2d_9 (Conv2D)	(None, 28, 28, 512)	2359808
conv2d_10 (Conv2D)	(None, 28, 28, 512)	2359808
max_pooling2d_4 (MaxPooling2	(None, 14, 14, 512)	0
conv2d_11 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_12 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_13 (Conv2D)	(None, 14, 14, 512)	2359808
max_pooling2d_5 (MaxPooling2	(None, 7, 7, 512)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_1 (Dense)	(None, 4096)	102764544
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 4096)	16781312
dropout_2 (Dropout)	(None, 4096)	0
dense_3 (Dense)	(None, 2)	8194

Total params: 134,268,738
Trainable params: 134,268,738
Non-trainable params: 0

Table. 4. VGG16 model summary table

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None,    )               752670
=================================================================
Total params: 15,467,358
Trainable params: 752,670
Non-trainable params: 14,714,688
```

Table. 4. Transfer learning VGG16 model summary

Epochs	CNN Accuracy	VGG16 Accuracy
150	**90.229166 %**	**51.166334 %**
120	**86. 666667%**	**50.140005 %**
90	**86.133333 %**	**47.157776 %**
60	**85.085556 %**	**46.872223 %**
30	**84.166664 %**	**45.708333 %**

comparison table of CNN vs. VGG16

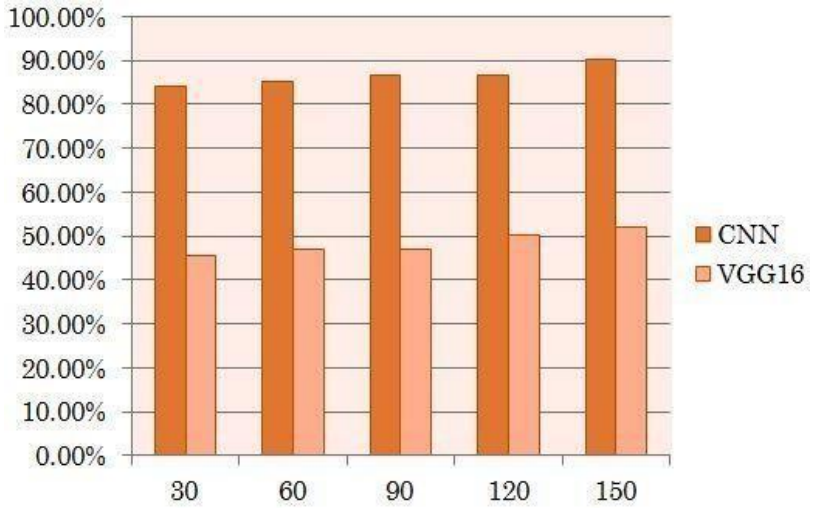


Chart of CNN vs. VGG16

Chapter 5: Conclusion and Future work

Conclusion

We have studied about existing system feature based approach. It's done by image processing technique in this we have studied steps like image Acquisition, image pre- processing, Image Segmentation, features extraction, classification.

Proposed system to achieve this purpose, we have use CNN and get accuracy is 90.23%. Wehave also use VGG16 model to detect leaf disease but in our case CNN has better result thanVGG16.

In future we can add more classes of leaves and disease type.