CLASSIFICATION OF PLANT DISEASES (LEAF) USING ALEXNET ARCHITECTURE

Major project submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

(2016-2020)

BY

Guntreddi Sai Kiran	**16241A1270**
Guntha Sai Roshik	**16241A1269**
Pallypati Harshavardhan Reddy	**16241A12A0**
Mamillapalli Mithun Kumar	**16241A1287**

Under the Esteemed guidance of

Dr. N. Rajasekhar

Professor, Dept. of IT

DEPARTMENT OF INFORMATION TECHNOLOGY

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

HYDERABAD

APRIL 2020

CERTIFICATE

This is to certify that it is a bonafide record of Mini Project work entitled **"Classification Of Plant Diseases (Leaf) using Alexnet Archcitecture"** done by **Guntreddi Sai Kiran (16241A1270), Guntha Sai Roshik (16241A1269), Pallypati Harshavardhan Reddy (16241A12A0)**, **Mamillapally Mithun Kumar(16241A1287)**, students of **B.Tech** in the Department of Information Technology, Gokaraju Rangaraju Institute of Engineering and Technology during the period 2016-2020 in the partial fulfillment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from GRIET, Hyderabad.

Dr. N. Rajasekhar
(Internal Project Guide)

Dr. K Prasanna Lakshmi
(Head of Department)

(Project External)

ACKNOWLEDGEMENT

We take the immense pleasure in expressing gratitude to our internal guide **Dr. N Rajasekhar, Professor, Information Technology, GRIET**. We express our sincere thanks for his encouragement, suggestions and support, which provided the impetus and paved the way for the successful completion of the project work.

We wish to express our gratitude **to Dr. K. Prasanna Lakshmi, G. Vijender Reddy our Project Co-coordinators**, for their constant support in the course of the project.

We express our sincere thanks to **Dr. Jandhyala N Murthy, Director, GRIET, and Dr. J. Praveen, Principal, GRIET**, for providing us the conductive environment for carrying through our academic schedules and project with ease. We also take this opportunity to convey our sincere thanks to the teaching and non-teaching staff of GRIET College, Hyderabad.

Email: saikiran7702059634@gmail.com
Contact No: 8008955312
Address: Ngo's Colony, Hyderabad.

Email: sairoshik.guntha@gmail.com
Contact No: 9963693873
Address: Manikonda, Hyderabad.

Email: mithunkumar.mamillapalli@gmail.com
Contact No: 9701526177
Address: Pragathi Nagar, Hyderabad.

Email: p.harshavardhanreddy1234@gmail.com
Contact No: 9494363854
Address: Miyapur, Hyderabad

DECLARATION

This is to certify that the project entitled "**Classification of Plant Diseases (Leaf) Using Alexnet Architecture**" is a bonafied work done by us in partial fulfillment of the requirements for the award of the degree **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad. We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites, books and paper publications are mentioned in the Bibliography. This work was not submitted earlier at any other University or Institute for the award of any degree.

Guntreddi Sai Kiran	*16241A1270*
Guntha Sai Roshik	*16241A1269*
Pallypati Harshavardhan Reddy	*16241A12A0*
Mamillapalli Mithun Kumar	*16241A1287*

ABSTRACT

Plants are the basic source of food and energy in the environment. Their growth doesn't require any feeding manually unless we damage the ecosystem, which we did and are still doing. Due to increase in population, different techniques/chemicals were employed/used to increase production from plants and at the same time opened doors to some bacteria, fungus and virus that affect plant growth by feeding on it. One kind of bacteria might boost production for a particular plant but it affects the growth of some other plant in the surroundings. This led to outbreak in numerous types of plant diseases. Our Earth hosts nearly 391,000 species of plants and double the varieties of diseases affecting them. Analysis of plants regularly helps in identifying the problem at early stage and controlling the effect of bacteria/fungus/virus.

Our project proposes to predict such abnormalities with the help of plant leaf images. Analysis of diseases affected plant leafs using strong neural networks with the help of machine learning algorithms helps in precise classification of disease. This helps in less human intervention and accurate identification of the problem. Different methods including AlexNet architecture and various machine learning methodologies have been explored to produce fruitful results.

Table of Contents:

1. INTRODUCTION

The growing requirement for food and other basic needs due to increase in population has forced mankind to produce more in less time.

This opened doors to use of chemicals and other resources for meeting our needs which in turn affected the flora and fauna around us giving birth to number of disease causing virus/bacteria/fungi.

There are three types of these diseases based on the part of plant affected

a) Stem Diseases
b) Root diseases
c) Foliage diseases (Leaf)

This project deals with foliage disease or disease affecting leafs of plants that are most common among plants like potato, corn, tomato, apple etc.., We propose improvement of already existing model which involves training a machine learning model to classify and provide a clear idea about the type of disease the plant is affected with.

2. PROJECT OVERVIEW

Our project main aim is to identify the diseases in plants through technology without human intervention. Leafs are the most vulnerable part of the three important building blocks of a plant which include stem and root. The root hides itself in the soil and hence it is safe from external threats and the stems though are exposed to the outside world, are protected with a covering known as bark. Leafs are majorly responsible for the process of producing the fruit or flower or seed, should carry out photosynthesis for this purpose and hence face the adverse effects of the outside world, thus making them more vulnerable.

The change in the global climatic conditions has mainly affected and boosted this issue. The government provides various fertilizers and chemicals to reduce loss due to diseases but the bacteria/fungi/virus has not failed in their purpose to destroy the plant.

The project could only be successful if data is collected time to time, maintained and available through a common database.

The common challenges that one could face in this project would be collection of data from various sources, places and integrating them, maintaining them and it may also result in serious problems if data is not collected properly. Quality of seeds, seed rate at sowing and some other factors also affect the plant growth.

2.1 EXISTING SYSTEM:

Traditional method of identifying the disease through sight and the feel of the affected area is still practiced vastly around the country. Few of the advanced farming communities use drones and other methods to identify and sprinkle fertilizers and chemicals in the foreign countries. But in India, still the old and conventional methods are employed.

2.2 PROPOSED SYSTEM:

Our project opted few features such as using images to identify but the feeding of images is manually done instead of an interconnected system. The images are captured manually and fed to the model to obtain the type of disease.

Our project uses modern methodologies in machine learning such as neural networks and best in class architecture to provide more accurate and less biased results. The model is trained on a vast variety of images.

3. REQUIREMENT ENGINEERING

3.1 SOFTWARE REQUIREMENTS:

Our project requires various software and its components.

3.1.1 Jupyter Notebook

The Jupyter Notebook is an open-source web application that enables you to create and share live code, calculations, visualizations, and narrative text documents. Uses include: data cleaning and transformation, computational simulation, mathematical modeling, machine learning, data visualization and much more.

3.1.2 Python

Python is a language for decoding, high-level, general-purpose programming. Python was developed by Guido van Rossum and first published in 1991, and has a design philosophy that emphasizes code readability, utilizing substantial white space in particular. It provides constructs that allow both small and large scale, simple programming. Van Rossum led the language community until stepping down as leader in July 2018.

3.2 Machine Learning Packages in built in Python

3.2.1 NumPy

NumPy, which stands for Numerical Python, is a library of multidimensional array objects and a set of processing routines for those arrays.

3.2.2 SKLEARN:

Scikit-learn is a free library for the Python programming language to learn machine software. It features various classification, regression, and clustering algorithms including supporting vector machines, random forests, gradient boosting, k-means, and DBSCAN, and is designed to interface with the NumPy and SciPy numerical and scientific libraries in Python.

3.2.3 Pandas

Python Data Analysis Library pandas is an open source providing the Python programming language with high performance, easy-to-use data structures and data analysis tools.

3.2.4 KERAS

Keras is an open source library written in Python for the neural networks. It can run atop TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Built to enable rapid experimentation with deep neural networks, it is built to be user-friendly, modular, and extensible.

3.2.5 OpenCV

It is a library of functions that helps in performing computer vision operations. It supports some models such as TensorFlow, PyTorch deep learning frameworks.

4. LITERATURE SURVEY

4.1 AlexNet Architecture

AlexNet is an extremely versatile model that can provide high accuracy on very complex datasets. It is a leading architecture for any function of object detection, and can have major applications of artificial intelligence problems in the computer vision field. For image tasks AlexNet that be implemented in the future rather than CNNs.

4.1.1 Understanding AlexNet:

- These are some of the features used that are new approaches to convolutional neural networks by AlexNet architecture:

    1. ReLU Non-Linearity
    2. Multiple GPU's (Optional)
    3. Overlapping Pooling

4.1.2 The Overfitting Problem

AlexNet has 60 million parameters (all combined), a major issue in terms of overfitting. Two methods were employed to reduce overfitting:

- Data Augmentation

  This technique involves the images to turned, sheared, zoomed and transformed into different possible looks to train the model for it to be robust and adjust to the real world problem. It also reduces the bias and increases the training time to increase in the data.

- Dropout

  This method consists of a fixed chance of "turning off" neurons (e.g. 50 per cent). It means that each iteration uses a different set of parameters of the model, which allows each neuron to have more robust features that can be combined with other neurons at random. Dropout however also increases the training time required for convergence of the model.

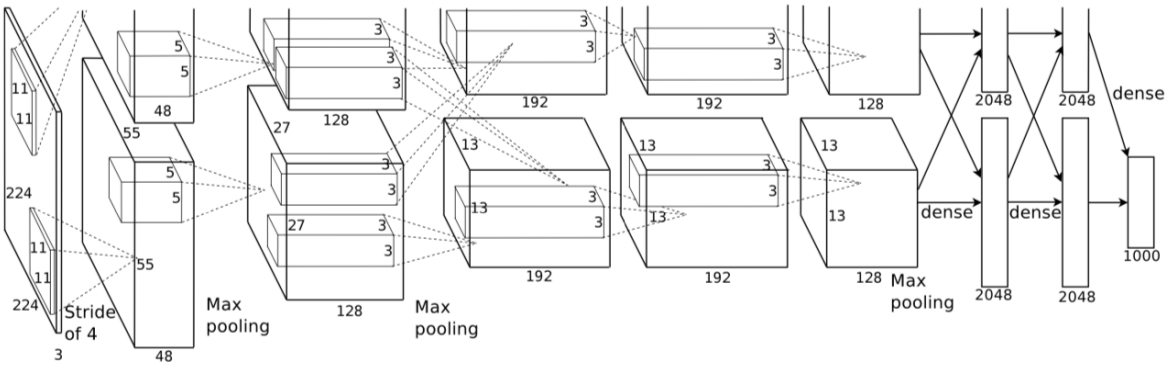


Fig: AlexNet Architecture

5. TECHNOLOGY

5.1 CNN (Convolutional Neural Network)

A Convolution Neural Network (CNN / ConvNet) is a Deep Learning algorithm capable of capturing an input image, assigning importance (learnable weights and biases) to various aspects / objects in the image, and being able to distinguish one from another.

5.1.1 Convolution

It's a method in which we take a small number matrix (called a kernel or filter), move it over our image and transform it based on the filter values. The aim of the Convolution Operation is to extract from the input image the high-level features such as edges. ConvNets should not be limited to a single layer. The first ConvLayer is responsible for capturing the low-level features like edges, color, orientation of gradients etc. The architecture also adapts with added layers to the High-Level functionality, giving us a network that has a wholesome understanding of images in the dataset, close to how we would. It requires the same padding and true padding of two techniques.

5.1.2 Pooling Layer

As with the Convolutional Layer, the Pooling layer is responsible for raising the Convolved Feature's spatial dimension. It is to reduce the computing power needed for the data processing by reducing the dimensionality. There are two different forms of Pooling:

- **Max Pooling:** It returns the maximum value from the portion of the image covered by the Kernel.
- **Average Pooling:** It returns the average of all the values from the portion of the image covered by the Kernel.

Even Max Pooling serves as a Noise Suppressant. It absolutely discards the noisy activations and also conducts de-noise along with reduction of the dimensions. At the other hand, Average Pooling actually performs the reduction of dimensionality as a method to eliminate noise. And we can assume that Max Pooling does much better than Average Pooling.

5.1.3 Classification - Fully Connected Layer

Adding a Fully-Connected layer is a cheap way to learn the high-level features of non-linear combinations as defined by the output. Now that our input image has been transformed into a suitable shape for our Multi-Level Perceptron, we must flatten the image into a column vector. The flattened output is supplied to a feed-forward neural network and back-propagation applied to all training iterations.

The model will distinguish between dominant and other low-level features in images over a series of epochs, and classify them using the Softmax Classification technique.

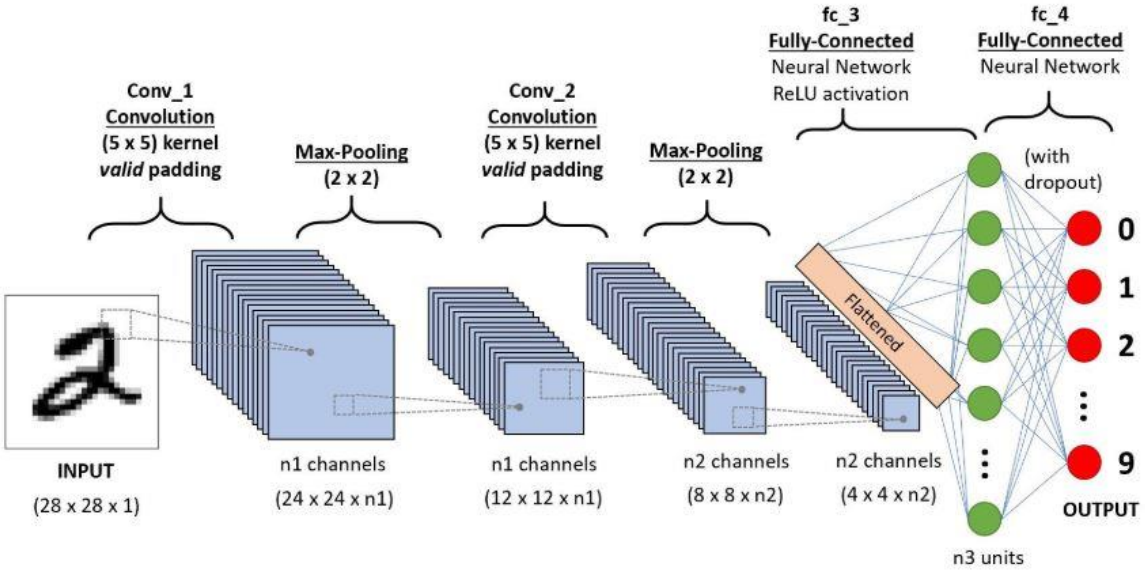


Fig: Convolutional Neural Network

6. DESIGN ANALYSIS

Why Use UML?

As the strategic value of software grows for many companies, the industry is searching for strategies to optimize software development and boost quality and cut costs and time-to-market. Other methods include development of the materials, visual programming, patterns and frameworks. Companies are now searching for strategies to handle the complexity of structures as they expand reach and size. In particular, they acknowledge the need to address recurrent architectural issues such as physical distribution, competition, replication, protection, load balancing, and tolerance of faults. The Unified Modeling Language (UML) was developed to meet these needs.

6.1 UML Diagrams

UML diagram is designed to allow developers and customers to view a software system from a particular perspective and visualize in varying degrees. UML diagrams commonly produced include in the visual modeling resources. A use case can be defined in its simplest form as a particular way of using the program from the viewpoint of a User (actor) viewpoint. One use case may be described by a more detailed definition as:

 a behavior pattern exhibited by system.

 a sequence of connected transactions performed by associate degree actor and therefore the system

 delivering one thing useful to the actor

Use cases provide a way to:

express system necessities

 communicate with the tip users and domain consultants

 Check the system

Cases of use are better found by observing the players and determining what the actor can do with the method. Because all of a system's needs cannot usually be met in one use case, a series of use cases is common. Together this list of use cases defines all ways to use the program.

A UML system is shown using five distinct views that describe the system from clearly different perspective. Every view is outlined by a collection of diagrams that is as follows.

 User Model View

- This view describes the user point of view of the system.
- The analysis illustration describes a usage scenario from the end-user's point of view.

 Structural model view

- In this model the information and functionality are arrived from within the system.
- Static structures are designed by this model view.

 Behavioral Model View

- It represents the dynamic of behavioral as elements of the system, portraying the interactions of assortment between varied structural components delineate within the user model and structural model view.

 Implementation Model View

- In this the structural and behavioral as components of the system are pictured as they are to be designed.

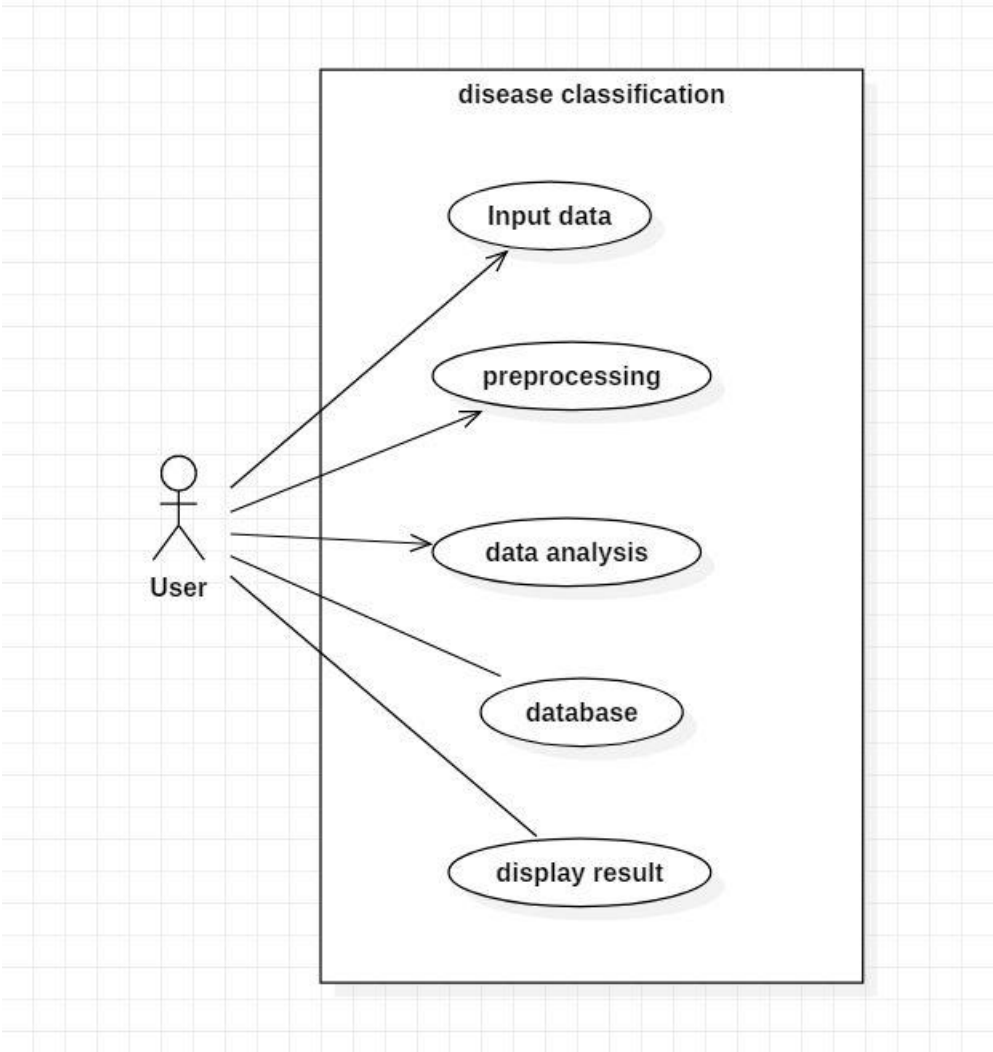
 Environmental Model View

- In this the structural and behavioral side of the surroundings within which the system is to be enforced is depicted or portrayed.

UML is designed through two totally different domains especially:

- UML Analysis modeling, here the entire focus is on structural model as well as user model view of the system.
- UML design modeling that concentrates on the behavioral

6.1.1 USECASE DIAGRAM

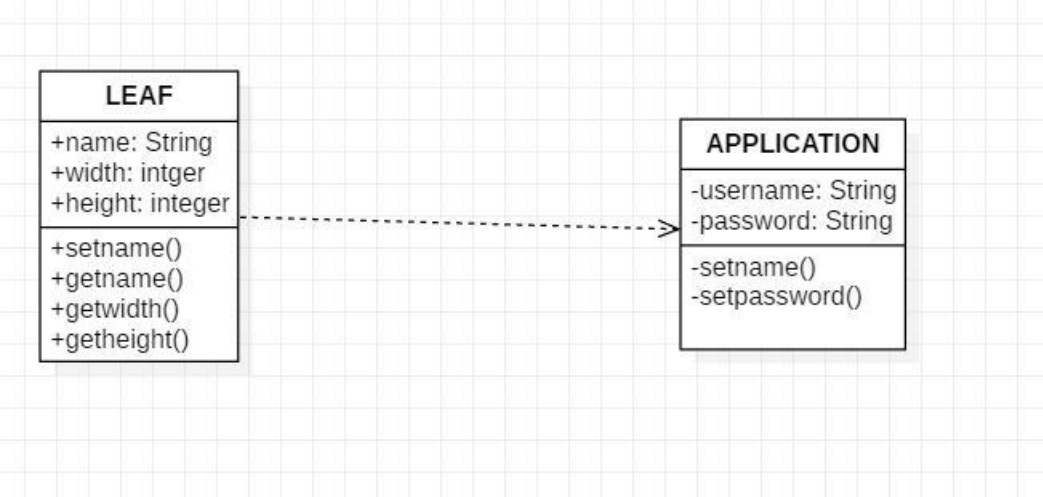


- Usecase diagram shows the interaction of user with system.
- The above use case diagram has two actors and 5 uses cases.

  Actors: - User

  Use cases: - input data, preprocessing, data analysis, database, display result
- We used directed association and simple association relationships.

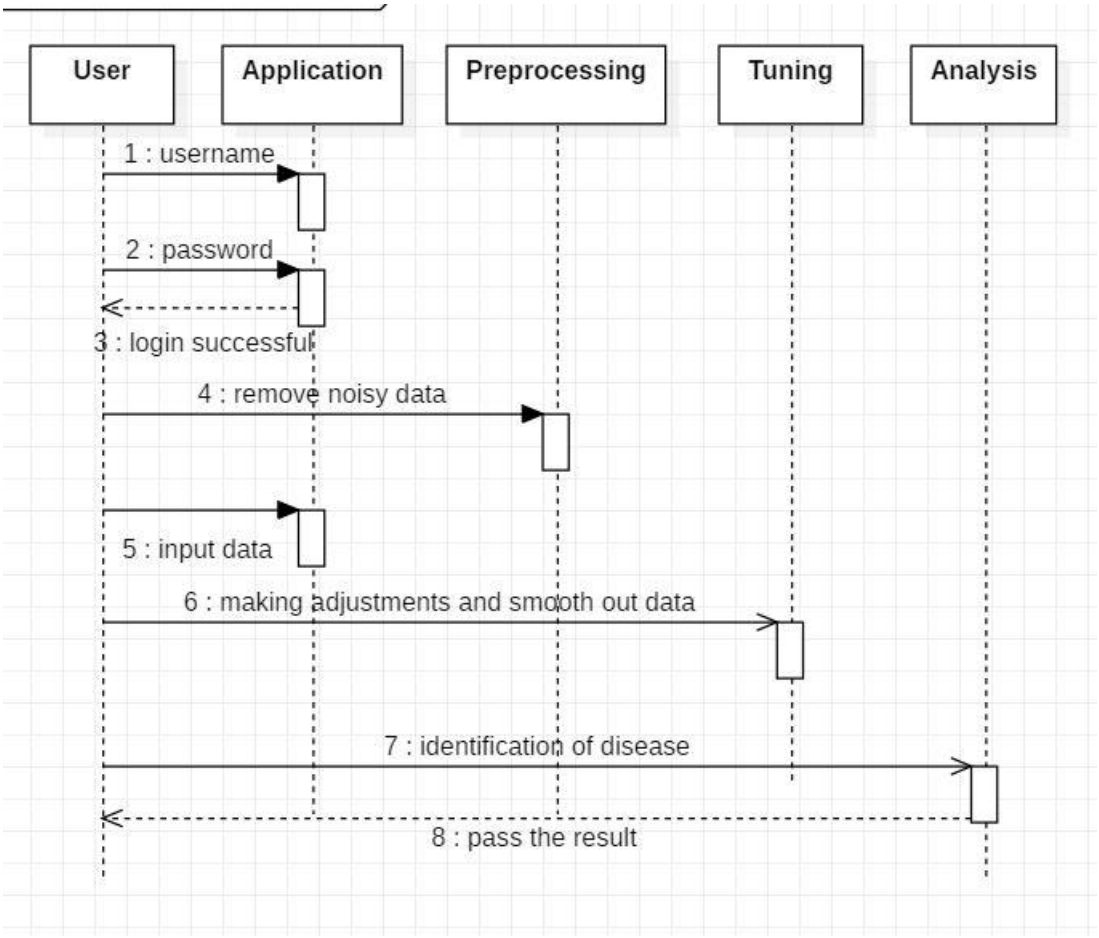
6.1.2 CLASS DIAGRAM



- Class diagram is used to represent structure of system.
- The class diagram contains two classes.

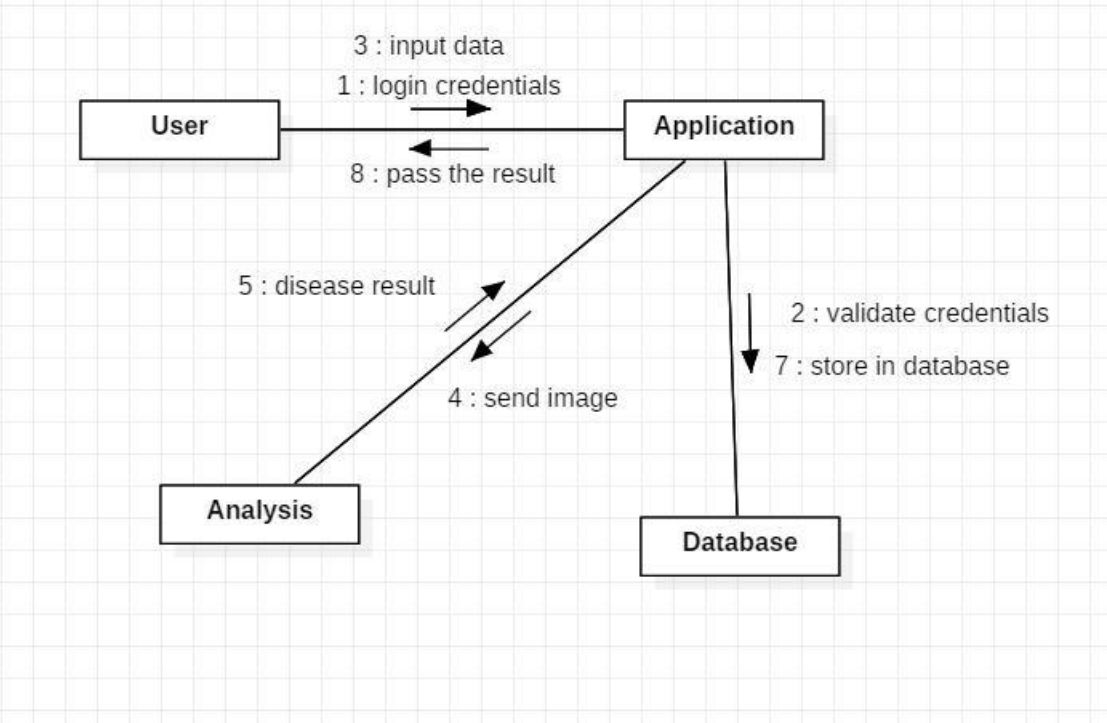
  Class: -Leaf and Application
- We used dependency relationship between (1) Leaf and Application

6.1.3 SEQUENCE DIAGRAM



- This diagram consists of five objects and various interactions (messages) between them.
- We place the important objects from left to right.
- Activation boxes tell the time an object requires completing a task.

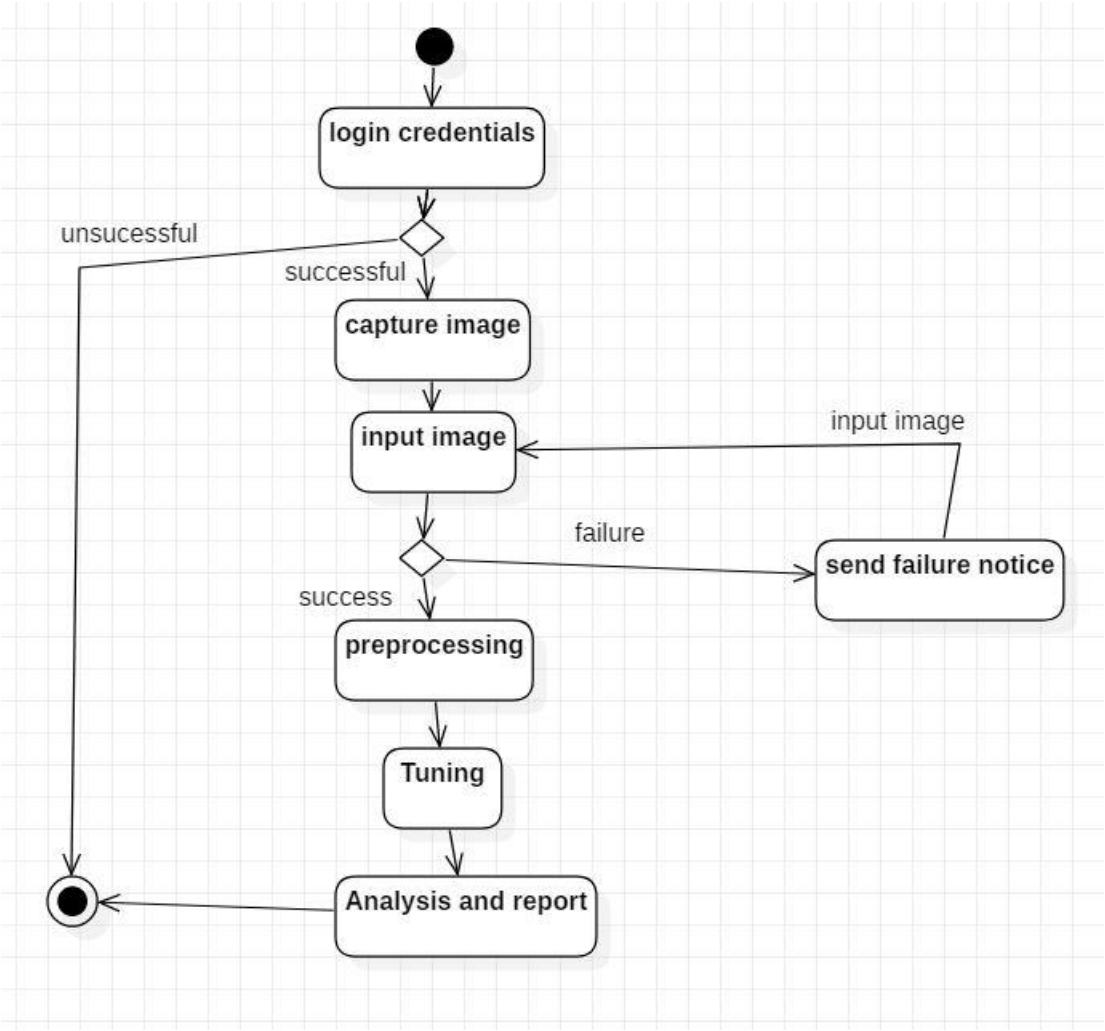
6.1.4 COLLABORATION DIAGRAM



- Communication diagram or interaction diagram are another names of collaboration diagram.

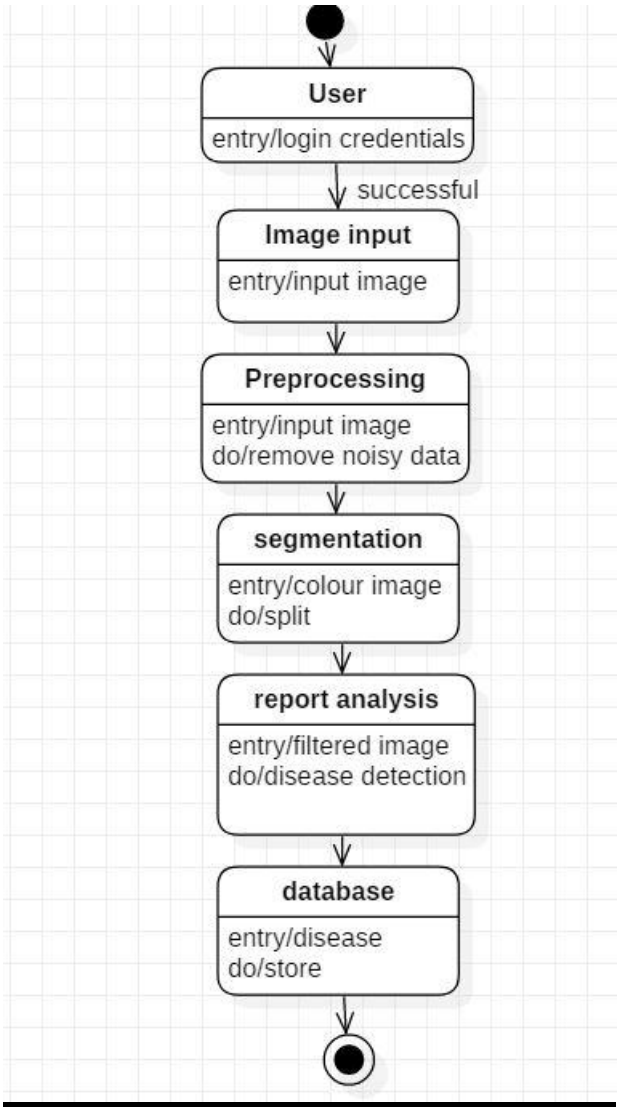
- It describes the relationship and interactions among software objects.

6.1.5 ACTIVITY DIAGRAM



- Activity diagram shows the execution and flow of behavior of system.
- An activity diagram consists of activity states and action states, transactions, objects.
- It starts with initial state and ends with final state symbols.

6.1.6 STATECHART DIAGRAM



- Statechart diagram models the dynamic nature of system.
- It shows the various states of an object in its lifetime.
- It contains elements like
    - Initial State
    - Transition
    - Event and Action
    - Final state

7. IMPLEMENTATION

7.1 MODULES:

7.1.1 COLLECTION OF DATA

In the first and starting module we collected data required through different online free data providers such as GitHub and Kaggle. We collected data related to different varieties of plants such as Potato, Tomato etc., and different types of diseases such as blight, Spider-Mite and Scab.

For example,

i. Apple

Apple Scab

Cedar Apple Rust

ii. Corn

Crecospora

Northern Leaf Blight

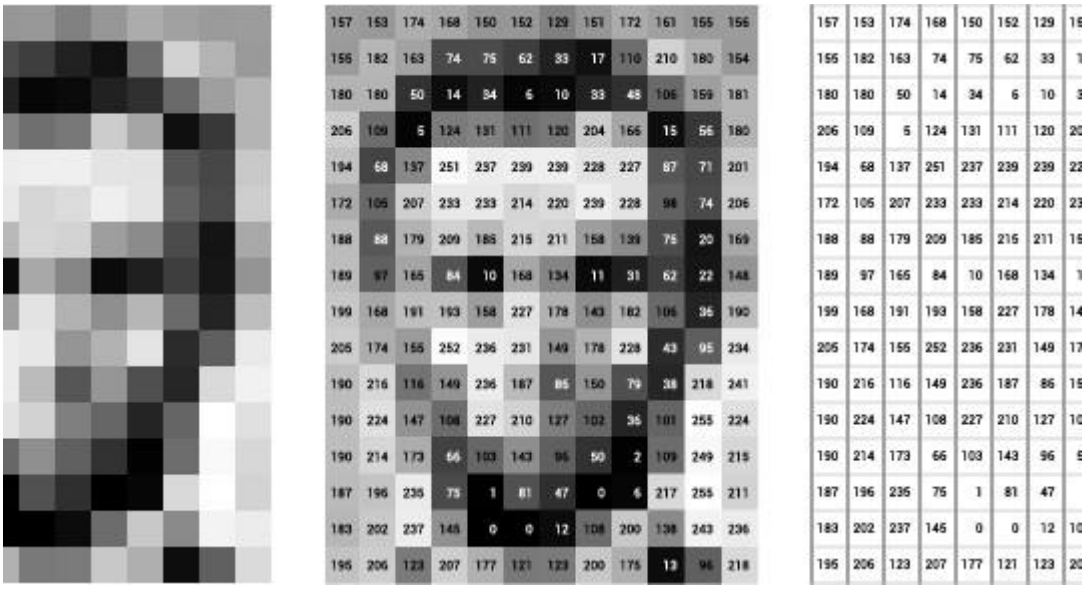
i. Potato

Early Blight

Late Blight

7.1.2 PRE-PROCESSING THE DATA

In this module, data is verified to check for any unwanted occurring's. The images are then scaled to a common height and width using functions in OpenCV module. The data is the extracted from the data (e.g. RGB values) and made ready for training in the next module.



7.1.3 TRAINING AND VALIDATING THE MODEL

In the second module, we used Jupyter notebook software and python language for training. We used various machine learning algorithms for prediction and in this module we did all our coding part. We trained and validated the data in this module using deep learning framework TensorFlow with Keras background. We got fruitful results and high accuracy.

7.1.4 TESTING THE MODEL

In this module, we tested the model using unseen data and verified its performance. The model performed exceptionally well, it was accurate up to 90%. Further we saved the weights and model in local module.

7.2 SOURCE CODE

7.2.1 Constructing the Model

```
import os
os.listdir("new-plant-diseases-dataset")


# In[2]:


# Importing Keras libraries and packages
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers.normalization import BatchNormalization

# Initializing the CNN
classifier = Sequential()

# Convolution Step 1
classifier.add(Convolution2D(96, 11, strides = (4, 4), padding = 'valid', input_shape=(224, 224, 3), activation = 'relu'))

# Max Pooling Step 1
classifier.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2), padding = 'valid'))
classifier.add(BatchNormalization())

# Convolution Step 2
classifier.add(Convolution2D(256, 11, strides = (1, 1), padding='valid', activation = 'relu'))

# Max Pooling Step 2
classifier.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2), padding='valid'))
classifier.add(BatchNormalization())

# Convolution Step 3
classifier.add(Convolution2D(384, 3, strides = (1, 1), padding='valid', activation = 'relu'))
classifier.add(BatchNormalization())

# Convolution Step 4
classifier.add(Convolution2D(384, 3, strides = (1, 1), padding='valid', activation = 'relu'))
classifier.add(BatchNormalization())

# Convolution Step 5
classifier.add(Convolution2D(256, 3, strides=(1,1), padding='valid', activation = 'relu'))

# Convolution Step 5
classifier.add(Convolution2D(256, 3, strides=(1,1), padding='valid', activation = 'relu'))

# Max Pooling Step 3
classifier.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2), padding = 'valid'))
classifier.add(BatchNormalization())

# Flattening Step
classifier.add(Flatten())

# Full Connection Step
classifier.add(Dense(units = 4096, activation = 'relu'))
classifier.add(Dropout(0.4))
classifier.add(BatchNormalization())
classifier.add(Dense(units = 4096, activation = 'relu'))
classifier.add(Dropout(0.4))
classifier.add(BatchNormalization())
classifier.add(Dense(units = 1000, activation = 'relu'))
classifier.add(Dropout(0.2))
classifier.add(BatchNormalization())
classifier.add(Dense(units = 21, activation = 'softmax'))
classifier.summary()


# In[3]:


#trainable parameters decrease after freezing some bottom layers
classifier.summary()


# In[4]:


classifier.load_weights('best_weights_9.hdf5')
```

7.2.2 Compiling the Model

```
# Compiling the Model
from keras import optimizers
classifier.compile(optimizer=optimizers.SGD(lr=0.001, momentum=0.9, decay=0.005),
              loss='categorical_crossentropy',
              metrics=['accuracy'])


# In[6]:


# image preprocessing
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   fill_mode='nearest')

valid_datagen = ImageDataGenerator(rescale=1./255)

batch_size = 128
base_dir = "new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)"

training_set = train_datagen.flow_from_directory(base_dir+'/train',
                                                 target_size=(224, 224),
                                                 batch_size=batch_size,
                                                 class_mode='categorical')

valid_set = valid_datagen.flow_from_directory(base_dir+'/valid',
                                              target_size=(224, 224),
                                              batch_size=batch_size,
                                              class_mode='categorical')
```

7.2.3 Training and Validating the Model

```
class_dict = training_set.class_indices
print(class_dict)


# In[8]:


li = list(class_dict.keys())
print(li)


# In[9]:


train_num = training_set.samples
valid_num = valid_set.samples


# In[ ]:


# checkpoint
from keras.callbacks import ModelCheckpoint
weightpath = "best_weights_9.hdf5"
checkpoint = ModelCheckpoint(weightpath, monitor='val_acc', verbose=1, save_best_only=True, save_weights_only=True, mode='max')
callbacks_list = [checkpoint]

#fitting images to CNN
history = classifier.fit_generator(training_set,
                         steps_per_epoch=25,
                         validation_data=valid_set,
                         epochs=25,
                         validation_steps=valid_num//batch_size,
                         callbacks=callbacks_list)
#saving model
filepath="new-plant-diseases-dataset/AlexNetModel.hdf5"
classifier.save(filepath)
```

8. RESULTS

```
Epoch 21/25
25/25 [==============================] - 770s 31s/step - loss: 0.3567 - acc: 0.8709 - val_loss: 1.2721 - val_acc: 0.6778

Epoch 00021: val_acc did not improve from 0.88441
Epoch 22/25
25/25 [==============================] - 761s 30s/step - loss: 0.3300 - acc: 0.8912 - val_loss: 0.3243 - val_acc: 0.8946

Epoch 00022: val_acc improved from 0.88441 to 0.89456, saving model to best_weights_9.hdf5
Epoch 23/25
25/25 [==============================] - 765s 31s/step - loss: 0.3684 - acc: 0.8731 - val_loss: 0.4518 - val_acc: 0.8521

Epoch 00023: val_acc did not improve from 0.89456
Epoch 24/25
25/25 [==============================] - 765s 31s/step - loss: 0.3341 - acc: 0.8841 - val_loss: 0.2796 - val_acc: 0.9090

Epoch 00024: val_acc improved from 0.89456 to 0.90896, saving model to best_weights_9.hdf5
Epoch 25/25
25/25 [==============================] - 759s 30s/step - loss: 0.3186 - acc: 0.8938 - val_loss: 0.5130 - val_acc: 0.8443

Epoch 00025: val_acc did not improve from 0.90896
```

As we can conclude from the above results the model performed exceptionally well with unseen data and performed with a 90% accuracy. The trained model is saved and can be modified with changing weights to obtain better results.

9. CONCLUSION

We know plants are basic source of food for human beings as well as animals. So, their existence is essential for life existence on earth. Our project deals with the identification of various plant diseases which are not known and cannot be identified by normal people. It identifies the disease that a plant suffering from which helps one to diagnose the plant. It helps in increase of green coverage of earth. Identification of diseases and its diagnosis helps one to save plants and trees on earth which helps in the maintenance of ecological balance.

We used ALEXNET and it uses ReLu which has advantage of training time. It takes less time when compared to a CNN using tanh. Our project deals with Foliage diseases and doesn't deal with root diseases and stem diseases. Our project identifies diseases related potato, corn, tomato and apple. Our project achieved 90 percent accuracy in classification of diseases.

10. FUTURE ENHANCEMENTS

Our project identifies only foliage diseases (leaf) and it doesn't deal with diseases related to stem and roots. We can enhance this project and can increase its scope by extending it to classify stem and root diseases. Our project deals with only diseases related to potato, corn, tomato and apple plants. So, we can include some other plants. We can enhance project by making it to use live images or can use any unmanned air vehicle to capture image and pass it to system to classify the image and can use same UAV to diagnose.

11. BIBLIOGRAPHY

- **"ImageNet Classification with Deep Convolutional Neural Networks"** by Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton.
  https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

- **"Plant Disease Detection and Classification by Deep Learning"** by Muhammad Hammad Saleem, Johan Potgieter, and Khalid Mahmood Arif.
  https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6918394/

- **Tomato Disease Classification using AlexNet** by P. S. Georgantopoulos, C. Constantinopoulos and D. Kosmopoulos.
  https://www.researchgate.net/publication/335992667_Tomato_Disease_Classification_using_AlexNet

- **Understanding AlexNet** by Sunita Nayak
  https://www.learnopencv.com/understanding-alexnet/

- **A Comprehensive Guide to Convolutional Neural Networks** by Sumit Saha
  https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

- **AlexNet: The Architecture that Challenged CNNs** by Jerry Wei
  https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951

- **Convolutional Layers** from Keras Documentation
  https://keras.io/layers/convolutional/

- **Gentle Dive into Math Behind Convolutional Neural Networks** by Piotr Skalski
  https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9