Major Project Report

Recognition Model of Crop Diseases and Insect Pests Based on Deep Learning in Harsh Environments

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

By

AHMED ALI SYED 17RT1A1245

Under the guidance of

Dr. G. Sambasiva Rao
(Professor & HOD of I.T DEPT)

Department of Information Technology
NAWAB SHAH ALAM KHAN COLLEGE OF ENGINEERING & TECHNOLOGY
(Affiliated to JNTU, Hyderabad)
MALAKPET, HYDERABAD, TELENGANA

Chapter 1: INTRODUCTION

1.1 Overview of Project

Human population steadily continues to grow, and along with it the need for food production increases. According to the UN projections, human population is expected to reach 9.7 billion in 2050, 2 billion more than today. Considering that most of the population growth is to occur in the least developed countries (around 80% increase in the next 30 years), where the food scarcity is the main problem, it is easy to conclude that minimizing food loss in those countries is a primary concern. It is estimated that the yield loss worldwide is between 20 and 40 percent, with many farms suffering a total loss.

Traditional methods for detecting diseases require manual inspection of plants by experts. This process needs to be continuous, and can be very expensive in large farms, or even completely unavailable to many small farm holders living in rural areas. This is why many attempts to automate disease detection have been made in the last few decades. One of the notable approaches is the use of hyper spectral imaging. Hyper spectral images are usually taken by satellites or airborne imaging devices and used for monitoring large areas. A downside of this approach is extremely high equipment cost, as well as high dimensionality and small number of samples which make them unsuitable for machine learning (ML) analysis.

Because of the recent breakthroughs in computer vision and the availability of cheap hardware, currently the most popular approach is the analysis of RGB images. The other motive for analysing RGB images is that with the current smart phone iniquitousness these solutions have potential to reach even the most rural areas. RGB images can be analysed by classical ML algorithms or the deep learning (DL) approach.

Classical methods rely on image pre-processing and the extraction of features which are then fed into one of the ML algorithms. Popular algorithm choices are Support Vector Machines (SVM), k-Nearest Neighbours (k-NN), Fully Connected Neural Networks (FCNN), Decision Trees, Random Forests etc. In the last few years, the researchers shifted almost exclusively to the DL methods for image classification tasks. The reason is that they almost always outperform classical algorithms when given reasonably sized dataset, and can be implemented without the need for hand-engineered features. In this paper, we compare the DL approach with classical ML algorithms for the study case of plant disease classification.

Chapter 2: LITERATURE SURVEY

1) The global burden of pathogens and pests on major food crops

AUTHORS: Savary, Serge, et al.

Crop pathogens and pests reduce the yield and quality of agricultural production. They cause substantial economic losses and reduce food security at household, national and global levels. Quantitative, standardized information on crop losses is difficult to compile and compare across crops, agro ecosystems and regions. Here, we report on an expert-based assessment of crop health, and provide numerical estimates of yield losses on an individual pathogen and pest basis for five major crops globally and in food security hotspots. Our results document losses associated with 137 pathogens and pests associated with wheat, rice, maize, potato and soybean worldwide. Our yield loss (range) estimates at a global level and per hotspot for wheat (21.5% (10.1–28.1%)), rice (30.0% (24.6–40.9%)), maize (22.5% (19.5–41.1%)), potato (17.2% (8.1–21.0%)) and soybean (21.4% (11.0–32.4%)) suggest that the highest losses are associated with food-deficit regions with fast-growing populations, and frequently with emerging or re-emerging pests and diseases. Our assessment highlights differences in impacts among crop pathogens and pests and among food security hotspots. This analysis contributes critical information to prioritize crop health management to improve the sustainability of agro ecosystems in delivering services to societies.

2) Using deep learning for image-based plant disease detection

AUTHORS: Mohanty, Sharada P., David P. Hughes, and Marcel Salathé

Crop diseases are a major threat to food security, but their rapid identification remains difficult in many parts of the world due to the lack of the necessary infrastructure. The combination of increasing global Smartphone penetration and recent advances in computer vision made possible by deep learning has paved the way for smart phone-assisted disease diagnosis. Using a public dataset of 54,306 images of diseased and healthy plant leaves collected under controlled conditions, we train a deep convolutional neural network to identify 14 crop species and 26 diseases (or absence thereof). The trained model achieves an accuracy of 99.35% on a held-out test set, demonstrating the feasibility of this approach. Overall, the approach of training deep learning models on increasingly large and publicly available image datasets presents a clear path toward smart phone-assisted crop disease diagnosis on a massive global scale.

3) A practical plant diagnosis system for field leaf images and feature visualization

AUTHORS: Fujita, E., et al.

An accurate, fast and low-cost automated plant diagnosis system has been called for. While several studies utilizing machine learning techniques have been conducted, significant issues remain in most cases where the dataset is not composed of field images and often includes a substantial number of

inappropriate labels. In this paper, we propose a practical automated plant diagnosis system. We first build a highly reliable dataset by cultivating plants in a strictly controlled setting. We then develop a robust classifier capable of analyzing a wide variety of field images. We use a total of 9,000 original cucumber field leaf images to identify seven typical viral diseases, Downy mildew and healthy plants including initial symptoms. We also visualize the key regions of diagnostic evidence. Our system attains 93.6% average accuracy, and we confirm that our system captures important features for the diagnosis of Downy mildew.

4) **Textural features for image classification**

AUTHORS: Haralick, Robert M., Karthikeyan Shanmugam, and Its' Hak Dinstein

Texture is one of the important characteristics used in identifying objects or regions of interest in an image, whether the image is a photomicrograph, an aerial photograph, or a satellite image. This paper describes some easily computable textural features based on gray-tone spatial dependencies, and illustrates their application in category-identification tasks of three different kinds of image data: photomicrographs of five kinds of sandstones, 1:20 000 panchromatic aerial photographs of eight land-use categories, and Earth Resources Technology Satellite (ERTS) multispecialty imagery containing seven land-use categories. We use two kinds of decision rules: one for which the decision regions are convex polyhedra (a piecewise linear decision rule), and one for which the decision regions are rectangular parallelepipeds (a min-max decision rule). In each experiment the data set was divided into two parts, a training set and a test set. Test set identification accuracy is 89 percent for the photomicrographs, 82 percent for the aerial photographic imagery, and 83 percent for the satellite imagery. These results indicate that the easily computable textural features probably have a general applicability for a wide variety of image-classification applications.

5) **Support-vector networks**

AUTHORS: Cortes, Corinne, and Vladimir Vapnik

The support-vector network is a new learning machine for two-group classification problems. The machine conceptually implements the following idea: input vectors are non-linearly mapped to a very high-dimension feature space. In this feature space a linear decision surface is constructed. Special properties of the decision surface ensure high generalization ability of the learning machine. The idea behind the support-vector network was previously implemented for the restricted case where the training data can be separated without errors. We here extend this result to non-separable training data. High generalization ability of support-vector networks utilizing polynomial input transformations is demonstrated. We also compare the performance of the support-vector network to various classical learning algorithms that all took part in a benchmark study of Optical Character Recognition.

2.1 Overview of Domain

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called **artificial neural networks**.

If you are just starting out in the field of deep learning or you had some experience with neural networks some time ago, you may be confused. I know I was confused initially and so were many of my colleagues and friends who learned and used neural networks in the 1990s and early 2000s.

The leaders and experts in the field have ideas of what deep learning is and these specific and nuanced perspectives shed a lot of light on what deep learning is all about.

In this post, you will discover exactly what deep learning is by hearing from a range of experts and leaders in the field.

Kick-start your project with my new book Deep Learning With Python, including step-by-step tutorials and the Python source code files for all examples.
Let's dive in.

Deep Learning is Large Neural Networks

Andrew Ng from Coursera and Chief Scientist at Baidu Research formally founded Google Brain that eventually resulted in the productization of deep learning technologies across a large number of Google services.

He has spoken and written a lot about what deep learning is and is a good place to start.In early talks on deep learning, Andrew described deep learning in the context of traditional artificial neural networks.

In the 2013 talk titled "Deep Learning, Self-Taught Learning and Unsupervised Feature Learning" he described the idea of deep learning as:

Using brain simulations, hope to:

– Make learning algorithms much better and easier to use.
– Make revolutionary advances in machine learning and AI.

In a co-authored article in Science titled "I believe this is our best shot at progress towards real AI Later his comments became more nuanced.

The core of deep learning according to Andrew is that we now have fast enough computers and enough data to actually train large neural networks. When discussing why now is the time that deep learning is taking off at Extract Conf 2015 in a talk titled "What data scientists should know about deep learning", he commented: Very large neural networks we can now have and … huge amounts of data that we have access to He also commented on the important point that it is all about scale.

That as we construct larger neural networks and train them with more and more data, their performance continues to increase. This is generally different to other machine learning techniques that reach a plateau in performance.

For most flavors of the old generations of learning algorithms … performance will plateau. … Deep learning … is the first class of algorithms … that is scalable. … performance just keeps getting better as you feed them more data

He provides a nice cartoon of this in his slides: Finally, he is clear to point out that the benefits from deep learning that we are seeing in practice come from supervised learning. From the 2015 Extract Conf talk, he commented: almost all the value today of deep learning is through supervised learning or learning from labeled data Earlier at a talk to Stanford University titled "Deep Learning" in 2014 he made a similar comment: one reason that deep learning has taken off like crazy is because it is fantastic at supervised learning Andrew often mentions that we should and will see more benefits coming from the unsupervised side of the tracks as the field matures to deal with the **abun dance of unlabeled data available**.

Jeff is a Wizard and Google Senior Fellow in the Systems and Infrastructure Group at Google and has been involved and perhaps partially responsible for the scaling and adoption of deep learning within Google. Jeff was involved in the Google Brain project and the development of large-scale deep learning software Disbelieve and later Tensor Flow.

In a 2016 talk titled "Deep Learning for Building Intelligent Computer Systems" he made a comment in the similar vein, that deep learning is really all about large neural networks.

When you hear the term deep learning, just think of a large deep neural net. Deep refers to the number of layers typically and so this kind of the popular term that's been adopted in the press. I think of them as deep neural networks generally.

He has given this talk a few times, and in a modified set of slides for the same talk, he highlights the **scalability of neural networks** indicating that results get better with more data and larger models, that in turn require more computation to train.

Deep Learning is Hierarchical Feature Learning

In addition to scalability, another often cited benefit of deep learning models is their ability to perform automatic feature extraction from raw data, also called feature learning.

Yoshua Bengio is another leader in deep learning although began with a strong interest in the automatic feature learning that large neural networks are capable of achieving.

He describes deep learning in terms of the algorithms ability to discover and learn good representations using feature learning. In his 2012 paper titled "Deep Learning of Representations for Unsupervised and Transfer Learning" he commented:

Deep learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features

An elaborated perspective of deep learning along these lines is provided in his 2009 technical report titled "Learning deep architectures for AI" where he emphasizes the importance the hierarchy in feature learning.

Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features.

In the soon to be published book titled "Deep Learning" co-authored with Ian Goodfellow and Aaron Courville, they define deep learning in terms of the depth of the architecture of the models.

The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these concepts are built on top of each other, the graph is deep, with many layers. For this reason, we call this approach to AI deep learning.

This is an important book and will likely become the definitive resource for the field for some time. The book goes on to describe multilayer perceptions as an algorithm used in the field of deep learning, giving the idea that deep learning has subsumed artificial neural networks.

The quintessential example of a deep learning model is the feed forward deep network or multilayer perception (MLP).

Peter Norvig is the Director of Research at Google and famous for his textbook on AI titled "Artificial Intelligence: A Modern Approach".
In a 2016 talk he gave titled "Deep Learning and Understandability versus Software Engineering and Verification" he defined deep learning in a very similar way to Joshua, focusing on the power of abstraction permitted by using a deeper network structure.

A kind of learning where the representation you form have several levels of abstraction, rather than a direct input to output

Why call it "Deep Learning"?
Why Not Just "Artificial Neural Networks"?

Geoffrey Hinton is a pioneer in the field of artificial neural networks and co-published the first paper on the back propagation algorithm for training multilayer perceptron networks.
He may have started the introduction of the phrasing "deep" to describe the development of large artificial neural networks.

He co-authored a paper in 2006 titled "A Fast Learning Algorithm for Deep Belief Nets" in which they describe an approach to training "deep" (as in a many layered network) of restricted Boltzmann machines.

Using complementary priors, we derive a fast, greedy algorithm that can learn deep, directed belief networks one layer at a time, provided the top two layers form an undirected associative memory.

Recognition Model of Crop Diseases and Insect Pests Based on Deep Learning in Harsh Environments

This paper and the related paper Geoff co-authored titled “Deep Boltzmann Machines” on an undirected deep network were well received by the community (now cited many hundreds of times) because they were successful examples of greedy layer-wise training of networks, allowing many more layers in feed forward networks.

Reducing the Dimensionality of Data with Neural Networks” they stuck with the same description of “deep” to describe their approach to developing networks with many more layers than was previously typical.

We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

In the same article, they make an interesting comment that meshes with Andrew Ng’s comment about the recent increase in compute power and access to large datasets that has unleashed the untapped capability of neural networks when used at larger scale.

It has been obvious since the 1980s that backpropagation through deep autoencoders would be very effective for nonlinear dimensionality reduction, provided that computers were fast enough, data sets were big enough, and the initial weights were close enough to a good solution. All three conditions are now satisfied.

In a talk to the Royal Society in 2016 titled “Deep Learning“, Geoff commented that Deep Belief Networks were the start of deep learning in 2006 and that the first successful application of this new wave of deep learning was to speech recognition in 2009 titled “Acoustic Modeling using Deep Belief Networks“, achieving state of the art results.

It was the results that made the speech recognition and the neural network communities take notice, the use “deep” as a differentiator on previous neural network techniques that probably resulted in the name change.

The descriptions of deep learning in the Royal Society talk are very backpropagation centric as you would expect. Interesting, he gives 4 reasons why backpropagation (read “deep learning”) did not take off last time around in the 1990s. The first two points match comments by Andrew Ng above about datasets being too small and computers being too slow.

Deep Learning as Scalable Learning Across Domains

Deep learning excels on problem domains where the inputs (and even output) are analog. Meaning, they are not a few quantities in a tabular format but instead are **images of pixel data, documents of text data or files of audio data**.

Yann LeCun is the director of Facebook Research and is the father of the network architecture that excels at object recognition in image data called the Convolutional Neural Network (CNN). This technique is seeing great success because like multilayer perceptron feedforward neural networks, the technique scales with data and model size and can be trained with backpropagation.
This biases his definition of deep learning as the development of very large CNNs, which have had great success on object recognition in photographs.

In a 2016 talk at Lawrence Livermore National Laboratory titled "Accelerating Understanding: Deep Learning, Intelligent Applications, and GPUs" he described deep learning generally as learning hierarchical representations and defines it as a scalable approach to building object recognition systems:
deep learning [is] … a pipeline of modules all of which are trainable. … deep because [has] multiple stages in the process of recognizing an object and all of those stages are part of the training"

Jurgen Schmidhuber is the father of another popular algorithm that like MLPs and CNNs also scales with model size and dataset size and can be trained with backpropagation, but is instead tailored to learning sequence data, called the Long Short-Term Memory Network (LSTM), a type of recurrent neural network.

We do see some confusion in the phrasing of the field as "deep learning". In his 2014 paper titled "Deep Learning in Neural Networks: An Overview" he does comment on the problematic naming of the field and the differentiation of deep from shallow learning. He also interestingly describes depth in terms of the complexity of the problem rather than the model used to solve the problem.

At which problem depth does Shallow Learning end, and Deep Learning begin? Discussions with DL experts have not yet yielded a conclusive response to this question. let me just define for the purposes of this overview: problems of depth 10 require Very Deep Learning.

Demis Hassabis is the founder of Deep Mind, later acquired by Google. Deep Mind made the breakthrough of combining deep learning techniques with reinforcement learning to handle complex learning problems like game playing, famously demonstrated in playing Atari games and the game Go with Alpha Go.

In keeping with the naming, they called their new technique a Deep Q-Network, combining Deep Learning with Q-Learning. They also name the broader field of study "Deep Reinforcement Learning".

In their 2015 nature paper titled "Human-level control through deep reinforcement learning" they comment on the important role of deep neural networks in their breakthrough and highlight the need for hierarchical abstraction.

To achieve this, we developed a novel agent, a deep Q-network (DQN), which is able to combine reinforcement learning with a class of artificial neural network known as deep neural networks. Notably, recent advances in deep neural networks, in which several layers of nodes are used to build up progressively more abstract representations of the data, have made it possible for artificial neural networks to learn concepts such as object categories directly from raw sensory data.

Finally, in what may be considered a defining paper in the field, Yann LeCun, Yoshua Bengio and Geoffrey Hinton published a paper in Nature titled simply "Deep Learning". In it, they open with a clean definition of deep learning highlighting the multi-layered approach.
Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.

Later the multi-layered approach is described in terms of **representation learning and abstraction**.
Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level.The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure.

This is a nice and generic a description, and could easily describe most artificial neural network algorithms. It is also a good note to end on.

Summary

In this post you discovered that deep learning is just very big neural networks on a lot more data, requiring bigger computers.

Although early approaches published by Hinton and collaborators focus on greedy layer wise training and unsupervised methods like auto encoders, modern state-of-the-art deep learning is focused on training deep (many layered) neural network models using the back propagation algorithm. The most popular techniques are:

- Multilayer Perceptron Networks.
- Convolutional Neural Networks.
- Long Short-Term Memory Recurrent Neural Networks.

I hope this has cleared up what deep learning is and how leading definitions fit together under the one umbrella.

2.2 Present System

A method of detection of specific disease in grapes is proposed in. Downy mildew pest/disease can be detected by the real time system with weather data. The central sever provide forecast service of weather condition and disease. Another kind of solution related of monitoring traps which are used to capture pest is with the help of image sensors. In, he authors designed and implemented a low power consumed system which is based on wireless image sensors and powered by battery. The frequency of capturing and transferring trap images of sensors can be set and remote adjusted by trapping application.

2.3 Overview of Language (PYTHON)

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython,

the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

```
$ Python
Python 2.4.3 (#1, Nov 11 2010, 13:34:43)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type the following text at the Python prompt and press the Enter −

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");. However in Python version 2.4.3, this produces the following result −

```
Hello, Python!
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

$ python test.py

This produces the following result −

Hello, Python!

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo

```
#!/usr/bin/python

print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

```
$ chmod +x test.py     # This is to make file executable
$./test.py
```

This produces the following result −

Hello, Python!

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

Reserved Words

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

```
and      exec     not
assert   finally  or
break    for      pass
class    from     print
continue          global  raise
def      if       return
del      import   try
elif     in       while
else     is       with
except   lambdayield
```

Lines and Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

```
if True:
   print "True"
else:
   print "False"
```

However, the following block generates an error −

```
if True:
print "Answer"
```

```
print "True"
else:
print "Answer"
print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python

import sys

try:
   # open file stream
   file = open(file_name, "w")
except IOError:
   print "There was an error writing to", file_name
   sys.exit()
print "Enter '", file_finish,
print "' When finished"
while file_text != file_finish:
   file_text = raw_input("Enter text: ")
   if file_text == file_finish:
      # close the file
      file.close
      break
   file.write(file_text)
   file.write("\n")
file.close()
file_name = raw_input("Enter filename: ")
if len(file_name) == 0:
```

```
   print "Next time please enter something"
   sys.exit()
try:
   file = open(file_name, "r")
except IOError:
   print "There was an error reading file"
   sys.exit()
file_text = file.read()
file.close()
print file_text
```

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

```
total = item_one + \
        item_two + \
        item_three
```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

Quotation in Python

Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal –

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

```
#!/usr/bin/python

# First comment
print "Hello, Python!" # second comment
```

This produces the following result −

```
Hello, Python!
```

You can type a comment on the same line after a statement or expression −

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows −

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
# I said that already.
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

```
'''
This is a multiline
comment.
'''
```

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action −

```
#!/usr/bin/python

raw_input("\n\nPress the enter key to exit.")
```

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example −

```
if expression :
   suite
elif expression :
   suite
else :
   suite
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python -h
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
```

Options and arguments (and corresponding environment variables):

-c cmd : program passed in as string (terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5);
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing −

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value −

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

Live Demo

```
#!/usr/bin/python

tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );
print "tup1[0]: ", tup1[0];
print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result −

```
tup1[0]:  physics
tup2[1:5]:  [2, 3, 4, 5]
```

Updating Tuples

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Zara
dict['Age']:  7
```

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Alice']: ", dict['Alice']
```

When the above code is executed, it produces the following result −

```
dict['Alice']:
Traceback (most recent call last):
  File "test.py", line 4, in <module>
    print "dict['Alice']: ", dict['Alice'];
KeyError: 'Alice'
```

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8; # update existing entry
dict['School'] = "DPS School"; # Add new entry

print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

When the above code is executed, it produces the following result −

```
dict['Age']:  8
dict['School']:  DPS School
```

Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
del dict['Name']; # remove entry with key 'Name'
dict.clear();     # remove all entries in dict
del dict ;        # delete entire dictionary

print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist any more −

```
dict['Age']:
Traceback (most recent call last):
  File "test.py", line 8, in <module>
    print "dict['Age']: ", dict['Age'];
TypeError: 'type' object is unsubscriptable
```

Note − del() method is discussed in subsequent section.

Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys −

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Manni
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example −

Live Demo

```
#!/usr/bin/python

dict = {['Name']: 'Zara', 'Age': 7}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
Traceback (most recent call last):
  File "test.py", line 3, in <module>
    dict = {['Name']: 'Zara', 'Age': 7};
TypeError: unhashable type: 'list'
```

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates −

Live Demo

```
#!/usr/bin/python
```

```
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');

# Following action is not valid for tuples
# tup1[0] = 100;

# So let's create a new tuple as follows
tup3 = tup1 + tup2;
print tup3;
```

When the above code is executed, it produces the following result −

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example −

Live Demo

```
#!/usr/bin/python

tup = ('physics', 'chemistry', 1997, 2000);
print tup;
del tup;
print "After deleting tup : ";
print tup;
```

This produces the following result. Note an exception raised, this is because after del tup tuple does not exist anymore −

```
('physics', 'chemistry', 1997, 2000)
After deleting tup :
Traceback (most recent call last):
  File "test.py", line 9, in <module>
```

```
print tup;
NameError: name 'tup' is not defined
```

DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusabilityand "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

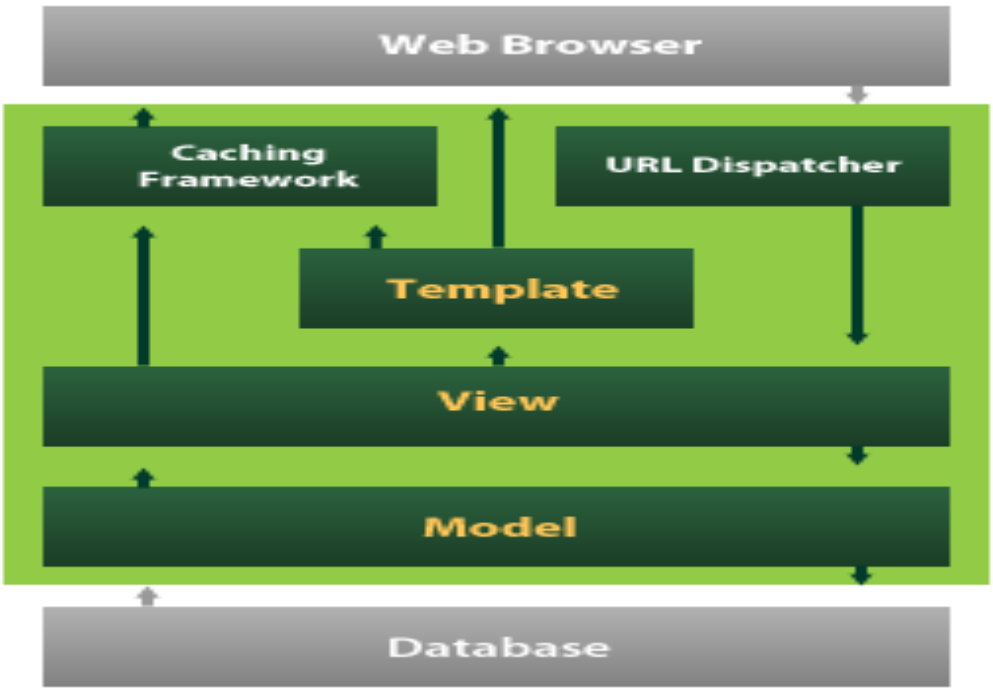


Fig.no.2.3.1: Django

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models

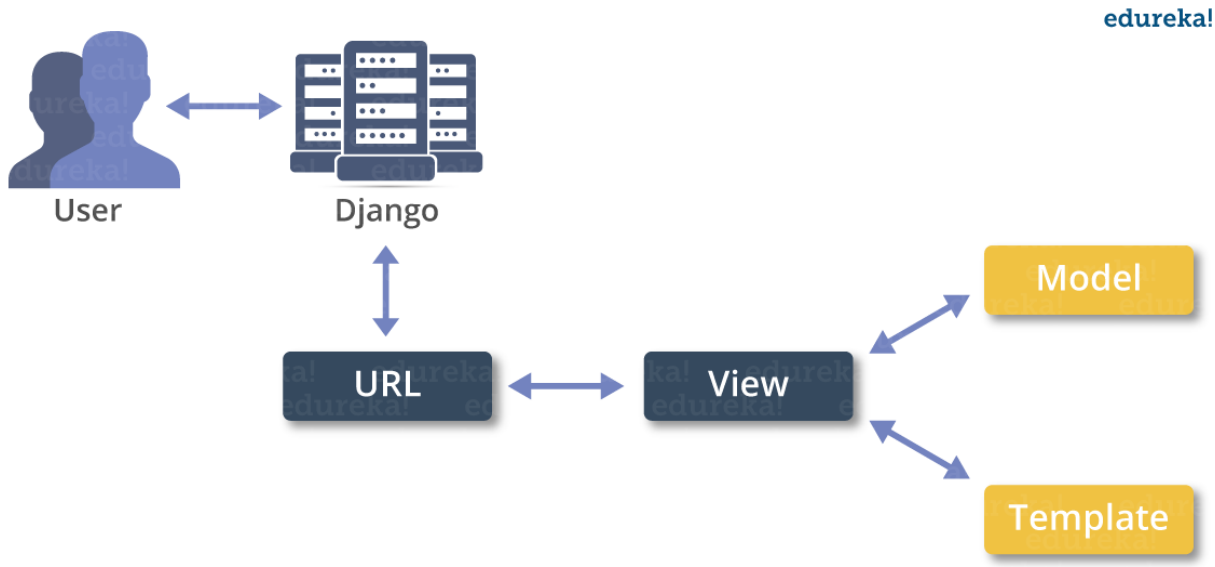


Fig.no.2.3.1: User Django

Create a Project

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code −

$ Django-admin start projects my project

This will create a "my project" folder with the following structure −

```
myproject/
  manage.py
  myproject/
    __init__.py
    settings.py
    urls.py
    wsgi.py
```

The Project Structure

The “myproject” folder is just your project container, it actually contains two elements −

manage.py − This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code −

```
$ python manage.py help
```

The “myproject” subfolder − This folder is the actual python package of your project. It contains four files −

__init__.py − Just for python, treat this folder as package.

settings.py − As the name indicates, your project settings.

urls.py − All links of your project and the function to call. A kind of ToC of your project.

wsgi.py − If you need to deploy your project over WSGI.

Setting Up Your Project

Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set −

DEBUG = True

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to ‘True’ for a live project. However, this has to be set to ‘True’ if you want the Django light server to serve static files. Do it only in the development mode.

```
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': 'database.sql',
    'USER': '',
    'PASSWORD': '',
    'HOST': '',
    'PORT': '',
```

```
  }
}
```

Database is set in the 'Database' dictionary. The example above is for SQLite engine. As stated earlier, Django also supports −

MySQL (django.db.backends.mysql)
PostGreSQL (django.db.backends.postgresql_psycopg2)
Oracle (django.db.backends.oracle) and NoSQL DB
MongoDB (django_mongodb_engine)
Before setting any new engine, make sure you have the correct db driver installed.

You can also set others options like: TIME_ZONE, LANGUAGE_CODE, TEMPLATE…

Now that your project is created and configured make sure it's working −

```
$ python manage.py runserver
```

You will get something like the following on running the above code −

```
Validating models...

0 errors found
September 03, 2015 - 11:41:50
Django version 1.6.11, using settings 'myproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others. See it as a module of your project.

Create an Application

We assume you are in your project folder. In our main "myproject" folder, the same folder then manage.py −

$ python manage.py startapp myapp

You just created myapp application and like project, Django create a “myapp” folder with the application structure −

```
myapp/
  __init__.py
  admin.py
  models.py
  tests.py
  views.py
```

__init__.py − Just to make sure python handles this folder as a package.

admin.py − This file helps you make the app modifiable in the admin interface.

models.py − This is where all the application models are stored.

tests.py − This is where your unit tests are.

views.py − This is where your application views are.

Get the Project to Know About Your Application

At this stage we have our "myapp" application, now we need to register it with our Django project "myproject". To do so, update INSTALLED_APPS tuple in the settings.py file of your project (add your app name) −

```
INSTALLED_APPS = (
  'django.contrib.admin',
  'django.contrib.auth',
  'django.contrib.contenttypes',
  'django.contrib.sessions',
  'django.contrib.messages',
  'django.contrib.staticfiles',
```

```
  'myapp',
)
```

Creating forms in Django, is really similar to creating a model. Here again, we just need to inherit from Django class and the class attributes will be the form fields. Let's add a forms.py file in myapp folder to contain our app forms. We will create a login form.

myapp/forms.py

```
#-*- coding: utf-8 -*-
from django import forms

class LoginForm(forms.Form):
  user = forms.CharField(max_length = 100)
  password = forms.CharField(widget = forms.PasswordInput())
```

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django: DateInput for dates, CheckboxInput for checkboxes, etc.

Using Form in a View

There are two kinds of HTTP requests, GET and POST. In Django, the request object passed as parameter to your view has an attribute called "method" where the type of the request is set, and all data passed via POST can be accessed via the request.POST dictionary.

Let's create a login view in our myapp/views.py −

```
#-*- coding: utf-8 -*-
from myapp.forms import LoginForm

def login(request):
  username = "not logged in"

  if request.method == "POST":
    #Get the posted form
```

```
    MyLoginForm = LoginForm(request.POST)

    if MyLoginForm.is_valid():
      username = MyLoginForm.cleaned_data['username']
  else:
    MyLoginForm = Loginform()

  return render(request, 'loggedin.html', {"username" : username})
```

The view will display the result of the login form posted through the loggedin.html. To test it, we will first need the login form template. Let's call it login.html.

```
<html>
  <body>

    <form name = "form" action = "{% url "myapp.views.login" %}"
      method = "POST" >{% csrf_token %}

      <div style = "max-width:470px;">
        <center>
          <input type = "text" style = "margin-left:20%;"
            placeholder = "Identifiant" name = "username" />
        </center>
      </div>

      <br>

      <div style = "max-width:470px;">
        <center>
          <input type = "password" style = "margin-left:20%;"
            placeholder = "password" name = "password" />
        </center>
      </div>
```

```
    <br>

    <div style = "max-width:470px;">
      <center>

        <button style = "border:0px; background-color:#4285F4; margin-top:8%;
          height:35px; width:80%;margin-left:19%;" type = "submit"
          value = "Login" >
          <strong>Login</strong>
        </button>

      </center>
    </div>

  </form>

 </body>
</html>
```

The template will display a login form and post the result to our login view above. You have probably noticed the tag in the template, which is just to prevent Cross-site Request Forgery (CSRF) attack on your site.

```
{% csrf_token %}
```

Once we have the login template, we need the loggedin.html template that will be rendered after form treatment.

```
<html>

 <body>
   You are : <strong>{{username}}</strong>
 </body>

</html>
```

Now, we just need our pair of URLs to get started: myapp/urls.py

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views',
  url(r'^connection/',TemplateView.as_view(template_name = 'login.html')),
  url(r'^login/', 'login', name = 'login'))
```

When accessing "/myapp/connection", we will get the following login.html template rendered −

Setting Up Sessions

In Django, enabling session is done in your project settings.py, by adding some lines to the MIDDLEWARE_CLASSES and the INSTALLED_APPS options. This should be done while creating the project, but it's always good to know, so MIDDLEWARE_CLASSES should have −

```
'django.contrib.sessions.middleware.SessionMiddleware'
```

And INSTALLED_APPS should have −

```
'django.contrib.sessions'
```

By default, Django saves session information in database (django_session table or collection), but you can configure the engine to store information using other ways like: in file or in cache.

When session is enabled, every request (first argument of any view in Django) has a session (dict) attribute.

Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing our login page you won't see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.

For this, first lets change our login view to save our username cookie server side −

```
def login(request):
```

```
  username = 'not logged in'
  if request.method == 'POST':
    MyLoginForm = LoginForm(request.POST)

    if MyLoginForm.is_valid():
      username = MyLoginForm.cleaned_data['username']
      request.session['username'] = username
    else:
      MyLoginForm = LoginForm()

  return render(request, 'loggedin.html', {"username" : username}
```

Then let us create formView view for the login form, where we won't display the form if cookie is set −

```
def formView(request):
  if request.session.has_key('username'):
    username = request.session['username']
    return render(request, 'loggedin.html', {"username" : username})
  else:
    return render(request, 'login.html', {})
```

Now let us change the url.py file to change the url so it pairs with our new view −

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView
urlpatterns = patterns('myapp.views',
  url(r'^connection/','formView', name = 'loginform'),
  url(r'^login/', 'login', name = 'login'))
```

When accessing /myapp/connection, you will get to see the following page

Chapter 3 PROPOSED SYSTEMS

In this project, a complex Internet of Things environment of crop diseases and insect pests identification model is established. Through the deployment of sensors and cameras in complex mountainous environment, the environmental information and image information of the scene are collected, and the basic database of crop pest identification is established. Through the deep learning network model, the image information is learned and recognized, which is used to identify and collect leaf images, and then identify pests and diseases.

This project contains the following modules:

MODULES:

3.1 Data preprocessing

3.2 Support Vector Machines

3.3 k-Nearest Neighbours

3.4 Fully Connected Neural Network.

MODULE DESCRIPTION

3.1 Data preprocessing:-

Data is stored in colab. We can download the data and laad the datasets , clean the data then after processes the data

3.2 Support Vector Machines:-

SVM is a supervised learning algorithm used for classification or regression problems. Classification is done by defining a separating hyperplane in the feature space. In the original form, it performs linear classification on two classes. By using kernels, it can also perform nonlinear classification. Kernels are used for an efficient transformation of the original feature space into high dimensional or infinite dimensional feature space, allowing for highly non-linear hyperplanes. SVM can fit highly complex datasets and at the same time exhibit good generalization propertie.

3.3 k-Nearest Neighbours:-

k-NN is a very simple algorithm often used for classification problems.

It is both non-parametric (doesn't have a fixed number of parameters) and lazy learning (doesn't have a training phase). k-NN works under the assumption that most samples from the same class are close to each other in the feature space.

When determining the class of the sample, k-NN will look at its k closest neighbours and decide to which class it belongs by the simple majority rule. Small values of k will allow for higher non-linearity but will be sensitive to outliers. High values of k achieve good generalization but fail to fit complex boundaries. The best value for parameter k is determined experimentally For this dataset, small values of k were shown to give the best results. Varying k from 1 to 9 doesn't change the accuracy much, with best result being 78.06% much lower than the SVM. We used k=5 in this work

3.4 Fully Connected Neural Network:-

FCNN is the simplest type of artificial neural networks. It is a supervised learning algorithm able to model highly non-linear functions. As opposed to SVM and k-NN, it does not converge to the global optimum, but when properly configured, it usually gives good enough results.we used an FCNN with four hidden layers with 300, 200, 100 and 50 neurons per layer, respectively. Activation function in hidden layers is a rectified linear unit (ReLU), with a soft ax in the output layer [8]. We used L2 regularization with regularization parameter equal to 0.3. Adam optimizer with default parameters was used this configuration gave us the accuracy of 91.46% on the test set.

Chapter 4. SYSTEM DESIGN

SYSTEM ARCHITECTURE:

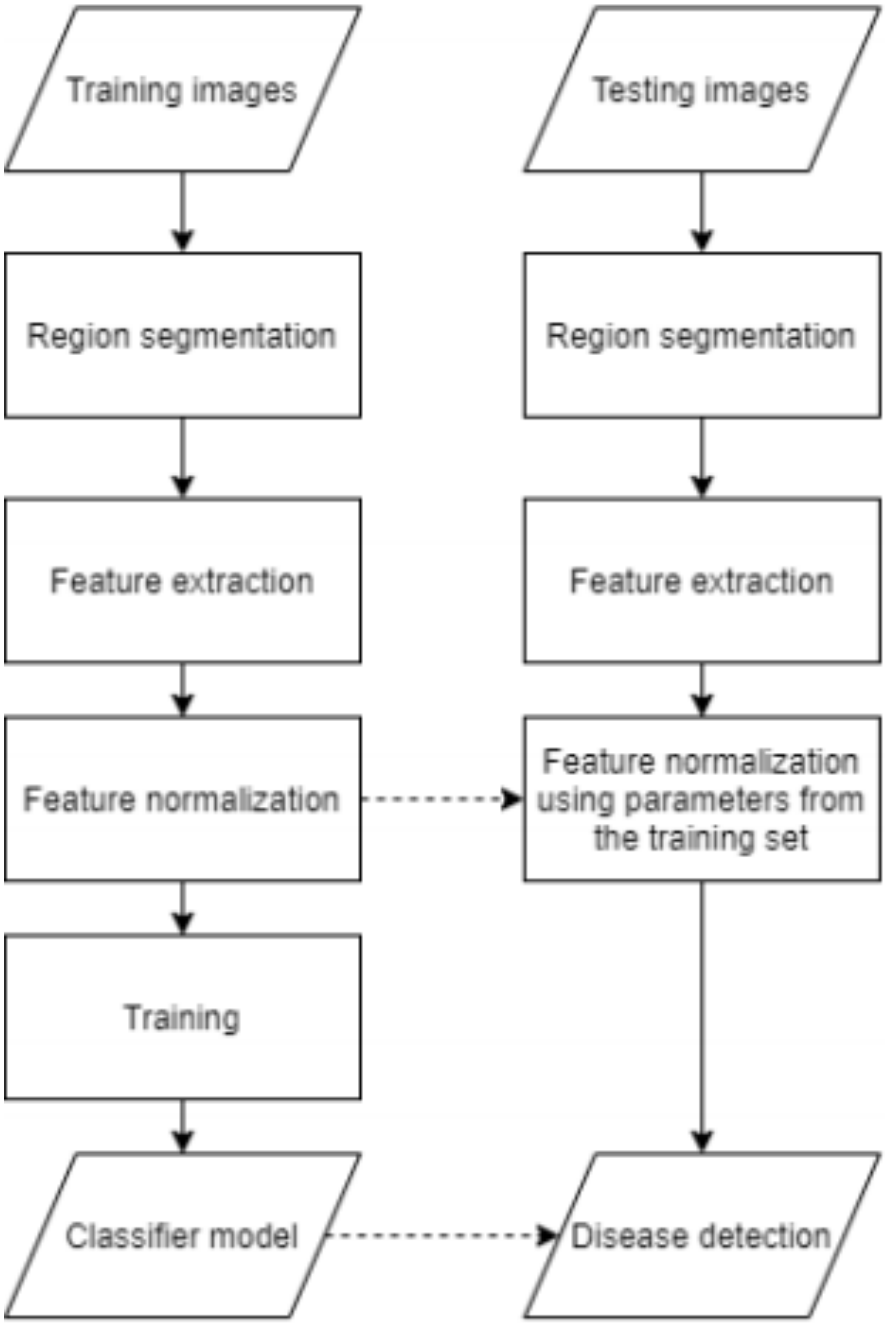


Fig 4.1.1: System Architecture

UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modelling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

USE CASE DIAGRAM:

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

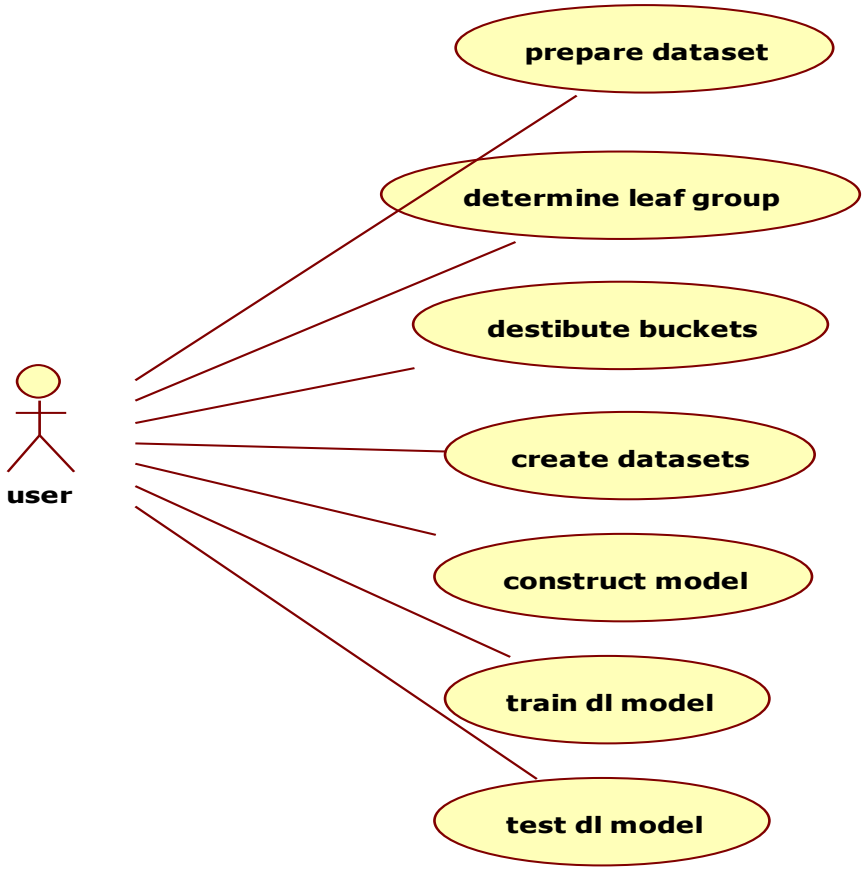


Fig 4.1.2: **Use case Diagram**

CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

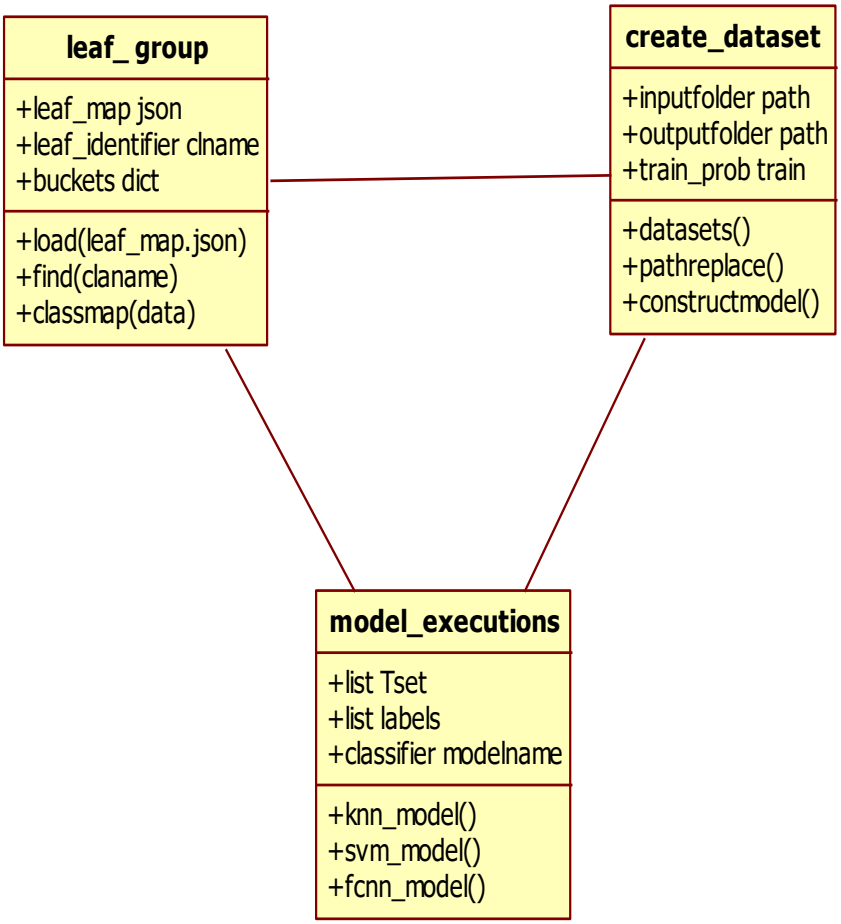


Fig 4.1.3: Class Diagram

SEQUENCE DIAGRAM:

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

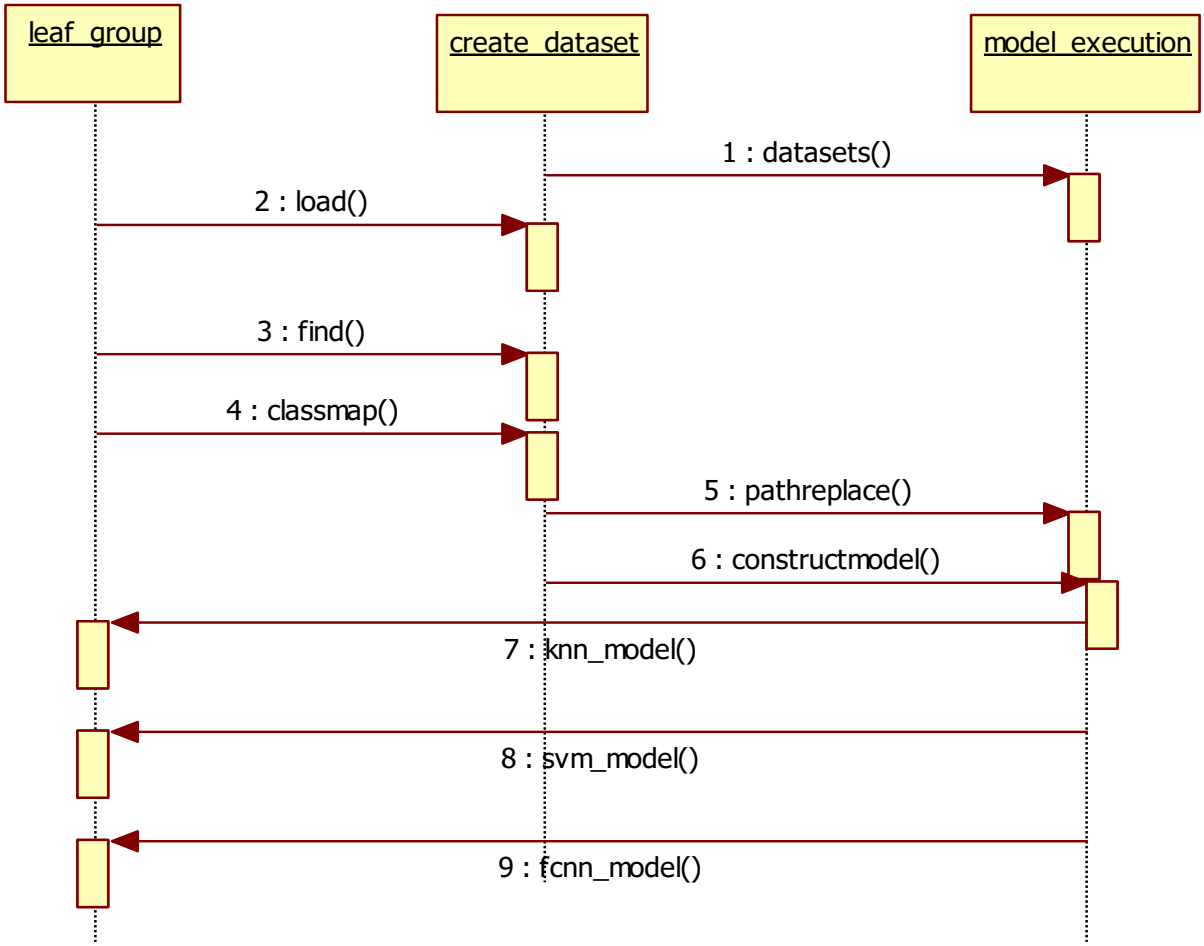


Fig 4.1.3 Sequence Diagram

ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

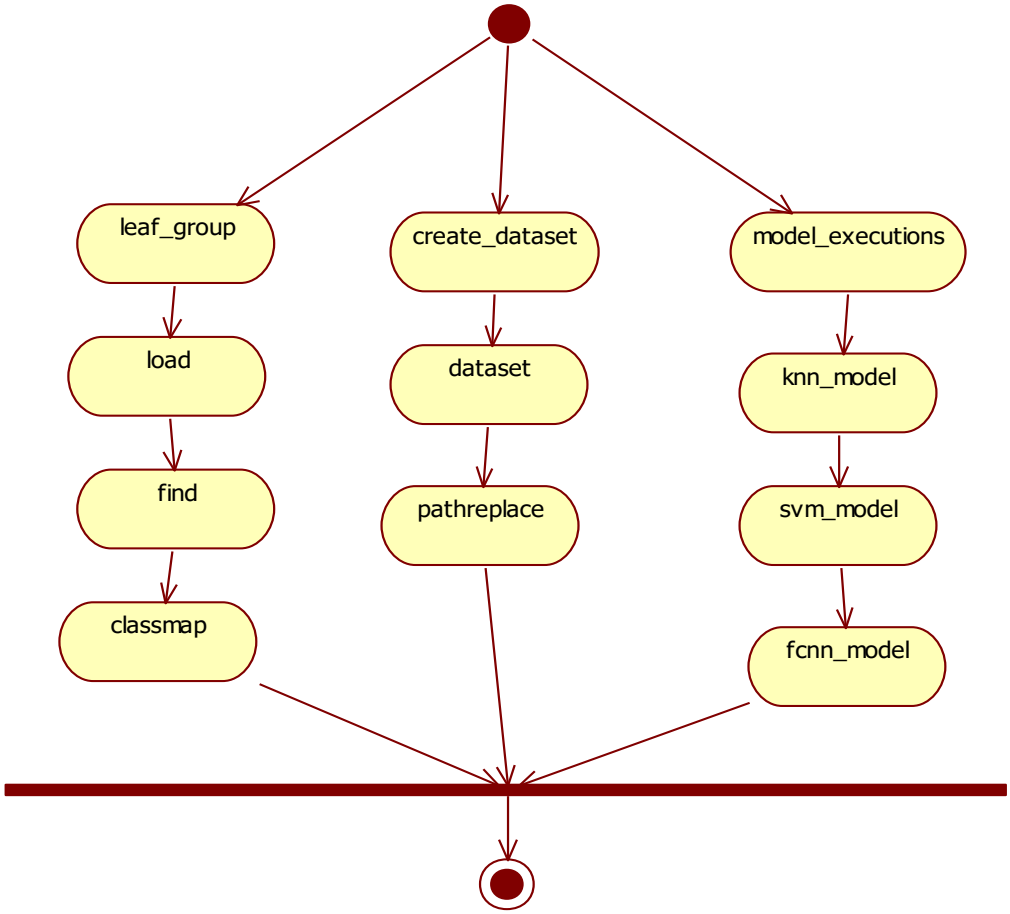


Fig 4.1.4: Activity Diagram

Chapter 5. IMPLEMENTATION

In order to enable farmers to identify and detect pests and diseases conveniently and quickly, this paper establishes a system based on Wechat applet.

The program can identify the disease on the leaves of crops with diseases, which is convenient for farmers to understand the situation of diseasesand insect pests and to obtain expert guidance.

The system first uploads the image, and then transmits the image data to the back-end for processing through the network front- end. Image preprocessing is mainly to optimize the incoming image. First of all, the image is zoomed to meet the require-ments of the model input, too large image will seriously affect the efficiency of recognition.

Secondly, in order to achieve higher recognition efficiency, the image is cut randomly and the pixels are optimized. Finally, the name and status of the crop with the highest matching degree will be given after therecognition is completed. If the crop is in an unhealthy state,the corresponding guidance will be given and returned to thecell phone.

The detection result of the system is shown in Fig. 8,the identification result is peach scab, which is a common disease of peach trees, and the identification is accurate afterverification.

Then we also identified healthy leaves. Fig. 9 showsthat the recognition result is a healthy cherry leaf, and the recognition result is accurate.

Fig 5.1.1 Recognition result of peach scab.

Fig 5.1.2 Recognition result of puccinia polysra.

Fig 5.1.3 Recognition result of healthy cherry leaf.

Finally, we recognize the corn leaves. Fig. 10 shows the results of identification of corn leaf rust. The occurrence of rust disease in maize will produce a pile of rust-colored powder on the leaves, which is harmful to maize crops. The recognition results show that the system can achieve the desired effect.

5.1 H/W and S/W Requirements:

HARDWARE REQUIREMENTS:

- Processor : Intel Core i7.
- Hard Disk : 1TB.
- Monitor : 15'' LED
- Input Devices : Keyboard, Mouse
- Ram : 16GB.

SOFTWARE REQUIREMENTS:

- Operating system : Windows 10.
- Coding Language : Python
- Tool : PyCharm, Visual Studio Code
- Database : SQLit

5.2 CODE:

Script.py:

```
'''input
test2.jpg
'''

#~ @author = dwij28 (Abhinav Jha) ~#

import warnings
warnings.filterwarnings('ignore') # suppress import warnings

import os
import sys
import cv2

import numpy as np

import tensorflow as tf
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
import tflearn
''' <global actions> '''

IMG_SIZE = 50
LR = 1e-3
MODEL_NAME = 'dwij28leafdiseasedetection-{}-{}.model'.format(LR, '2conv-basic')
tf.logging.set_verbosity(tf.logging.ERROR) # suppress keep_dims warnings
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3' # suppress tensorflow gpu logs

''' </global actions> '''
```

```
def process_verify_data(filepath):

    verifying_data = []

    img_name = filepath.split('.')[0]
    img = cv2.imread(filepath, cv2.IMREAD_COLOR)
    img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    verifying_data = [np.array(img), img_name]

    np.save('verify_data.npy', verifying_data)

    return verifying_data

def analysis(filepath):

    verify_data = process_verify_data(filepath)

    str_label = "Cannot make a prediction."
    status = "Error"

    tf.reset_default_graph()

    convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], name='input')

    '''
    # relu:

    Relu is used in the middle / hidden layers of the network to regularize the activation.
    It is essentialy the function: max(0, x)
    Activation should not be in negative, either it should be zero or more than that.

    # softmax:
```

```
Softmax is used for the output layer in multi class classification problems.
It is essentialy the function: log(1 + e^x)
It outputs a vector of probabilities of each class.

'''

convnet = conv_2d(convnet, 32, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)

convnet = conv_2d(convnet, 64, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)

convnet = conv_2d(convnet, 128, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)

convnet = conv_2d(convnet, 32, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)

convnet = conv_2d(convnet, 64, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 4, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR,
loss='categorical_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard_dir='log')

if os.path.exists('{}.meta'.format(MODEL_NAME)):
    model.load(MODEL_NAME)
    print ('Model loaded successfully.')
```

```
    else:
        print ('Error: Create a model using neural_network.py first.')

    img_data, img_name = verify_data[0], verify_data[1]

    orig = img_data
    data = img_data.reshape(IMG_SIZE, IMG_SIZE, 3)

    model_out = model.predict([data])[0]

    if np.argmax(model_out) == 0: str_label = 'Healthy'
    elif np.argmax(model_out) == 1: str_label = 'Bacterial'
    elif np.argmax(model_out) == 2: str_label = 'Viral'
    elif np.argmax(model_out) == 3: str_label = 'Lateblight'

    if str_label =='Healthy': status = 'Healthy'
    else: status = 'Unhealthy'

    result = 'Status: ' + status + '.'

    if (str_label != 'Healthy'): result += '\nDisease: ' + str_label + '.'

    return result

def main():
    filepath = input("Enter Image File Name:\n")
    print (analysis(filepath))
    import webbrowser
    webbrowser.open("https://colab.research.google.com/drive/1L6s--5ZASj-
pq_xoan9EhMmiMQd60L7v")

if __name__ == '__main__': main()
```

Image_process.py:

```
#~ @author = dwij28 (Abhinav Jha) ~#

import cv2
import numpy as np

def image_masking(filepath):

        BLUR = 21
        CANNY_THRESH_1 = 10
        CANNY_THRESH_2 = 200
        MASK_DILATE_ITER = 10
        MASK_ERODE_ITER = 10
        MASK_COLOR = (0.0,0.0,0.0) # In BGR format

        img = cv2.imread(filepath)
        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

        edges = cv2.Canny(gray, CANNY_THRESH_1, CANNY_THRESH_2)
        edges = cv2.dilate(edges, None)
        edges = cv2.erode(edges, None)

        contour_info = []
        _, contours, __ = cv2.findContours(edges, cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)

        for c in contours:
            contour_info.append((c, cv2.isContourConvex(c), cv2.contourArea(c),))
        contour_info = sorted(contour_info, key=lambda c: c[2], reverse=True)

        max_contour = contour_info[0]
        mask = np.zeros(edges.shape)
        cv2.fillConvexPoly(mask, max_contour[0], (255))
```

```
    mask = cv2.dilate(mask, None, iterations=MASK_DILATE_ITER)
    mask = cv2.erode(mask, None, iterations=MASK_ERODE_ITER)
    mask = cv2.GaussianBlur(mask, (BLUR, BLUR), 0)

    mask_stack = np.dstack([mask]*3)
    mask_stack  = mask_stack.astype('float32') / 255.0
    img = img.astype('float32') / 255.0

    masked = (mask_stack * img) + ((1-mask_stack) * MASK_COLOR)
    masked = (masked * 255).astype('uint8')

    fileName, fileExtension = filepath.split('.')
    fileName += '-masked.'
    filepath = fileName + fileExtension
    print( filepath)

    cv2.imwrite(filepath, masked)

if __name__ == '__main__':
    filepath = input("Enter Image File Name:\n")
    image_masking(filepath)
```

5.3 INPUT AND OUTPUT DESIGN

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system.

The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

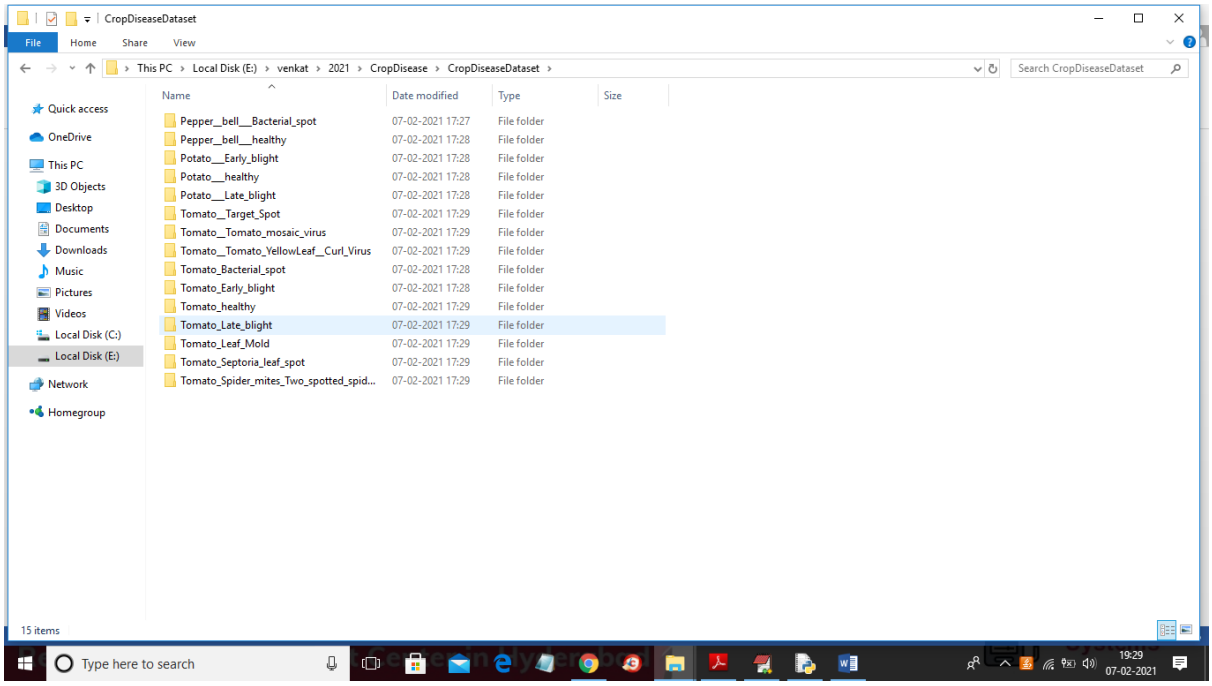
N681fyResearch on Recognition Model of Crop Diseases and Insect Pests Based on Deep Learning in Harsh Environments

In this paper author is applying deep learning convolution neural network (CNN) to predict crop disease and its pests to reduce economical loss in crop business. To build disease recognition model author is applying RESNET CNN model which consists of 3 parts

1) Feature Extraction: CNN compose of multiple layers and first layer define for feature extraction and this features will be extracted from given input image dataset or any other multidimensional dataset.
2) Feature Selection: Using this layer features will be selected by applying a layer called pooling or max polling.

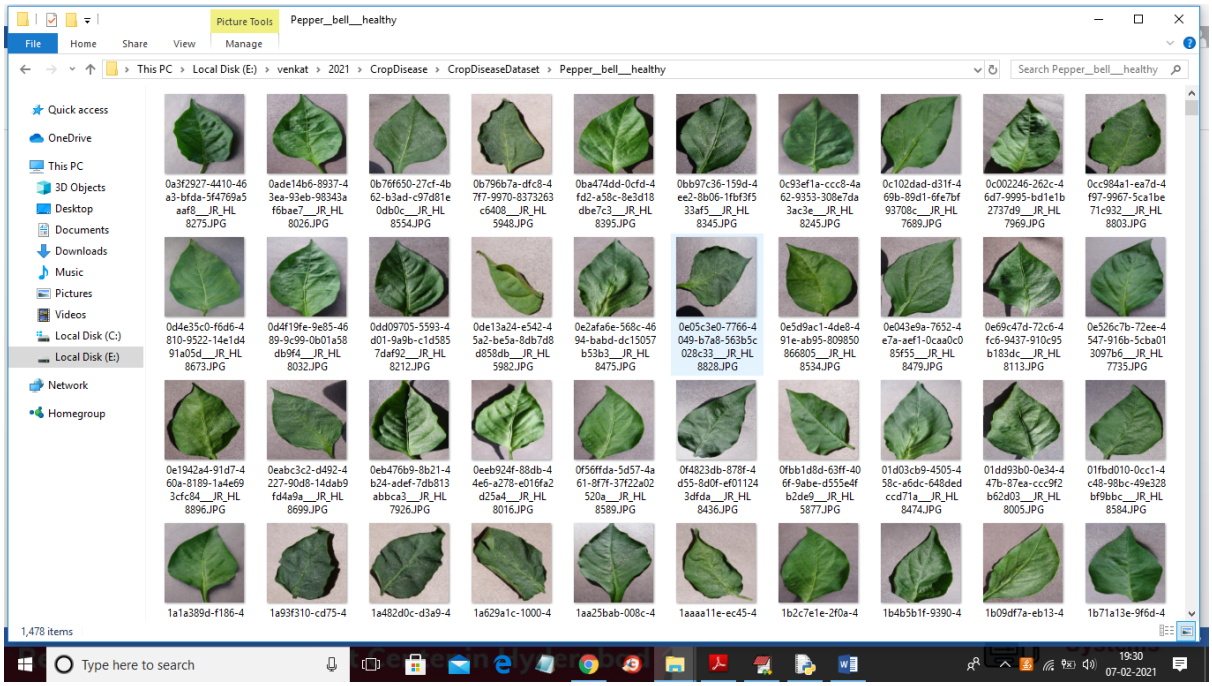
3) Activation module: using this module RELU will be applied on input features to remove out unimportant features and hold only relevant important features
4) Flatten: This layer will be define to convert multidimensional input features into single dimensional input array
5) Dense: This layer can be used to connect one layer to other layer to receive input features from previous layer to new layer to further filter input features in next layer to get most important features from dataset to have best prediction result.

To implement this project we have used crop disease recognition dataset and this dataset saved inside 'CropDiseaseDataset' folder and below screen shots showing various type of crop disease images



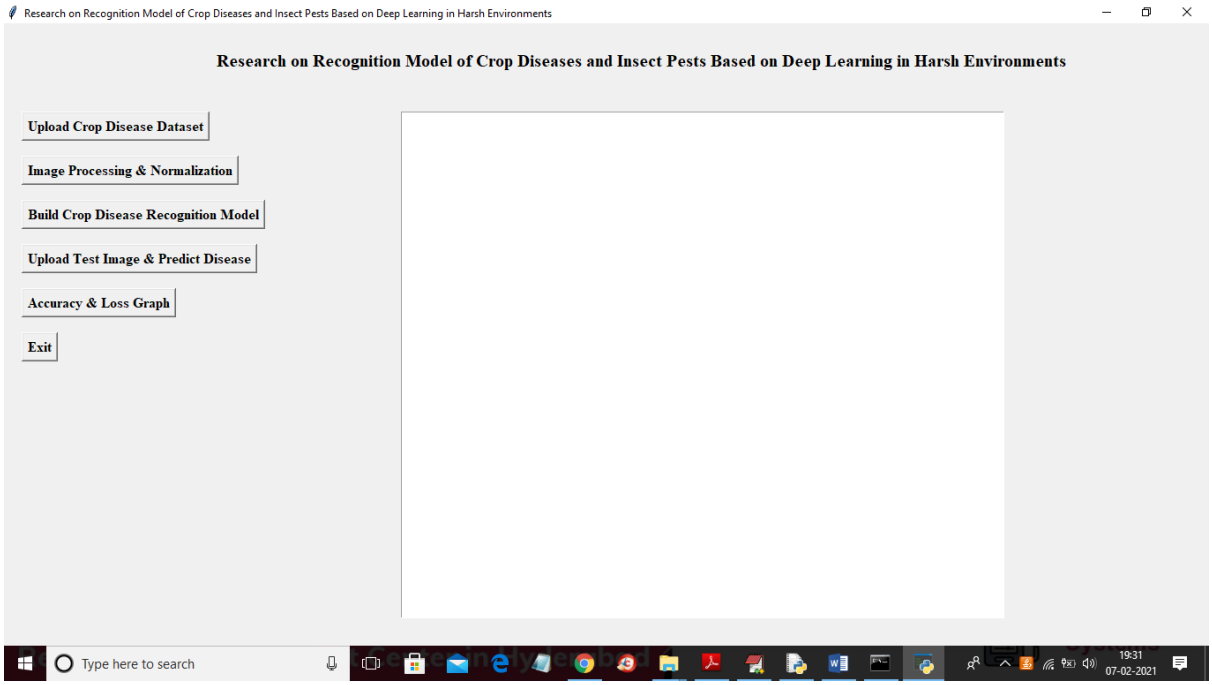
Screenshot 1: Crop Disease Dataset

In above screen we are using 15 different type of crop images and each folder contains images of own leaf and in below screen you can see those image. You too just go inside any above folder to see images



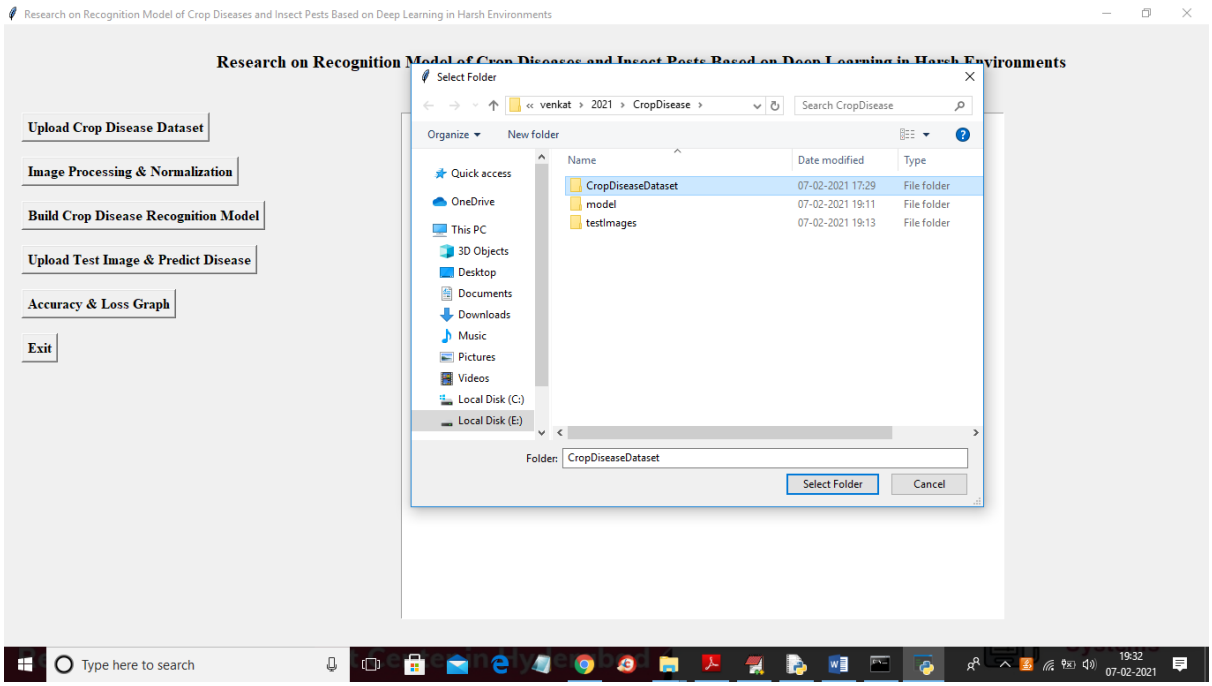
Screenshot 2:Picture Tools

To run project double click on ‘run.bat’ file to get below screen



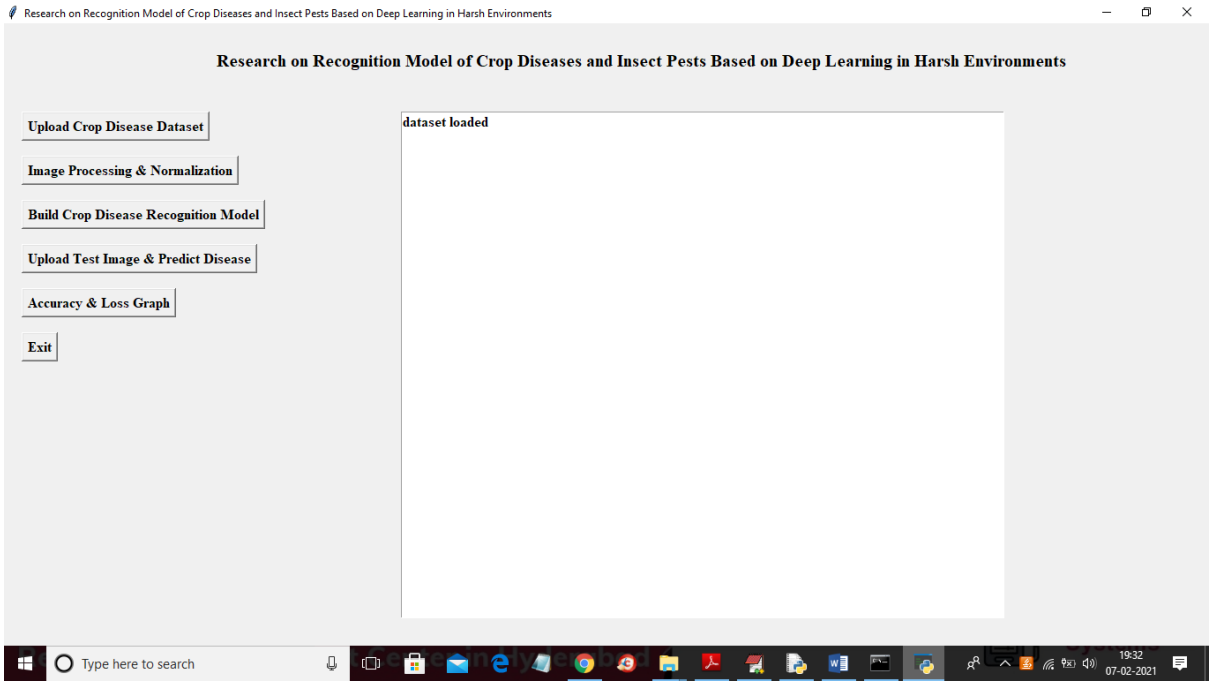
Screenshot 3: Upload Crop Disease Dataset

In above screen click on ‘Upload Crop Disease Dataset’ button to upload dataset images



Screenshot 4: screen selecting and uploading

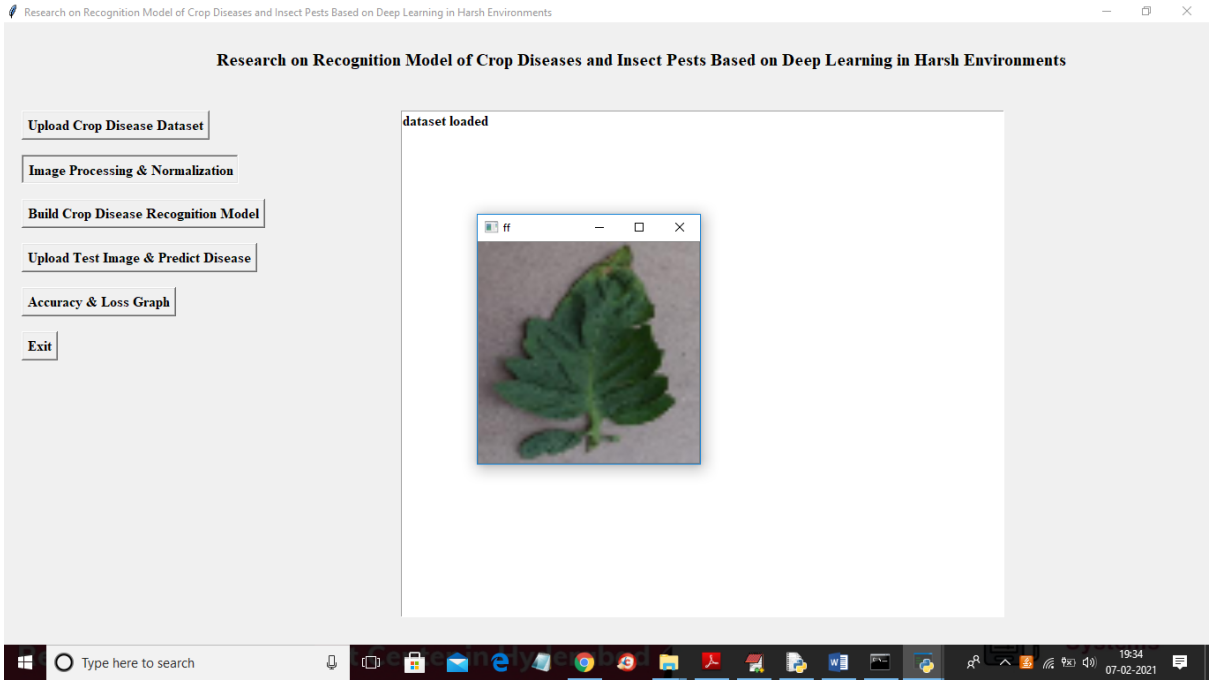
In above screen selecting and uploading 'CropDiseaseDataset' folder and then click on 'SelectFolder' button to load dataset and to get below screen



Screenshot 5: Image Processing & Normalization

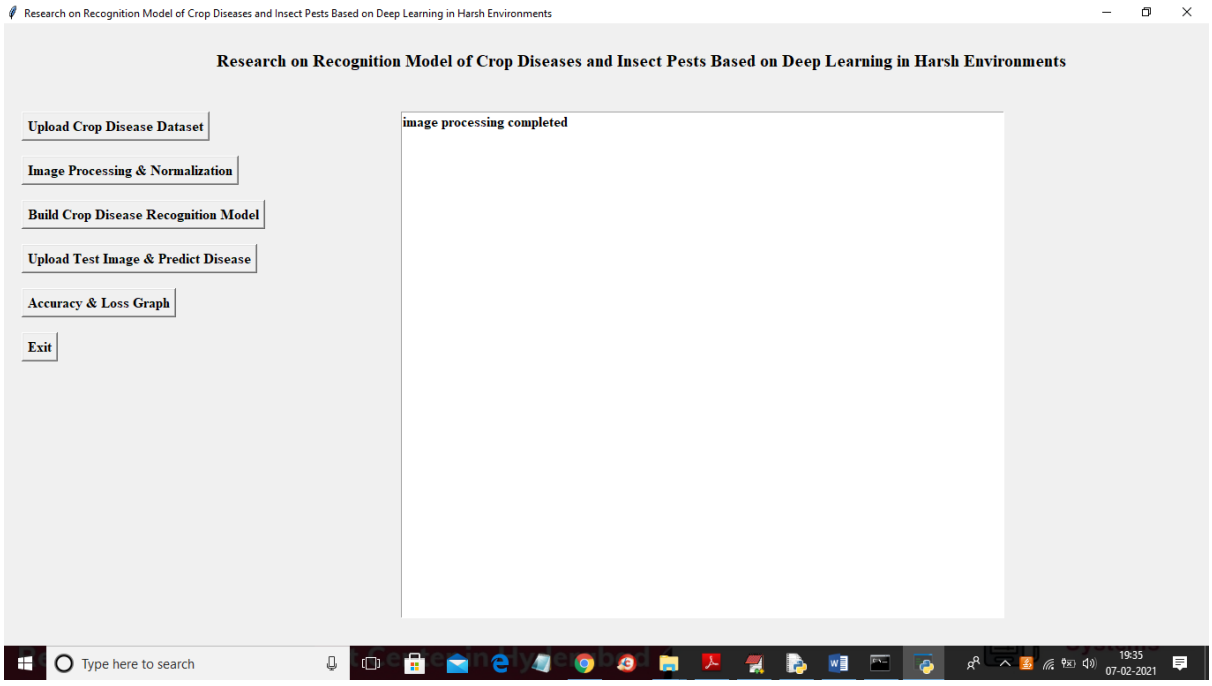
In above screen dataset loaded and now click on 'Image Processing & Normalization' button to read all images and then process images to normalize by converting each image pixel value between 0 and 1 and

for that normalization we will divide image pixels with 255 and then get value as 0 or 1 as all images pixel value will be between 0 to 255.



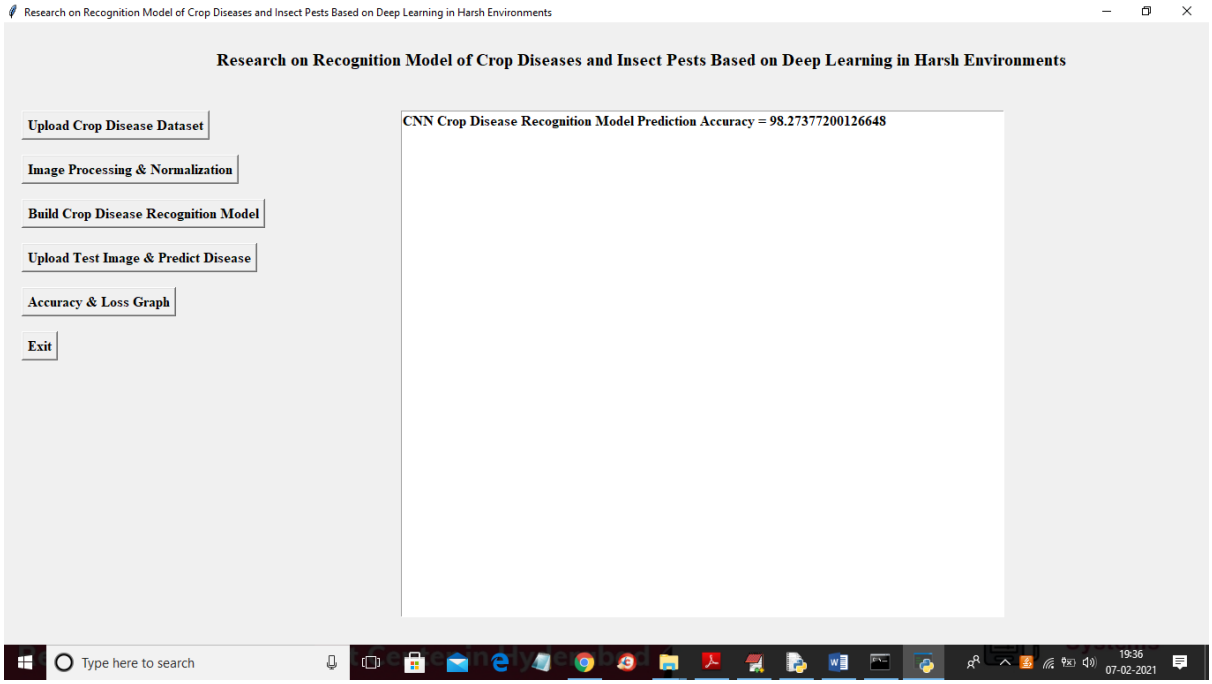
Screenshot 6: check whether images loaded

In above screen after applying normalization we are just displaying one random image from dataset to check whether images loaded and process properly or not and now you close above image to get below screen



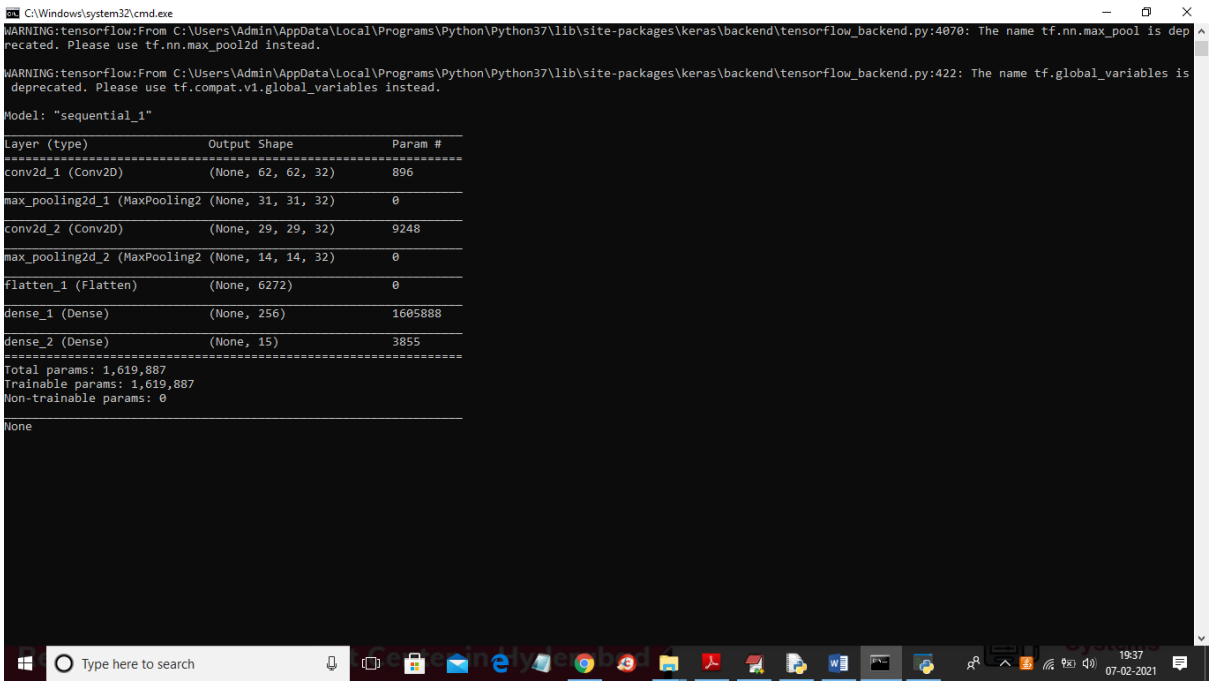
Screenshot 7: Build Crop Disease Recognition Model

In above screen all images process successfully and now dataset images are ready and now click on 'Build Crop Disease Recognition Model' button to build CNN model



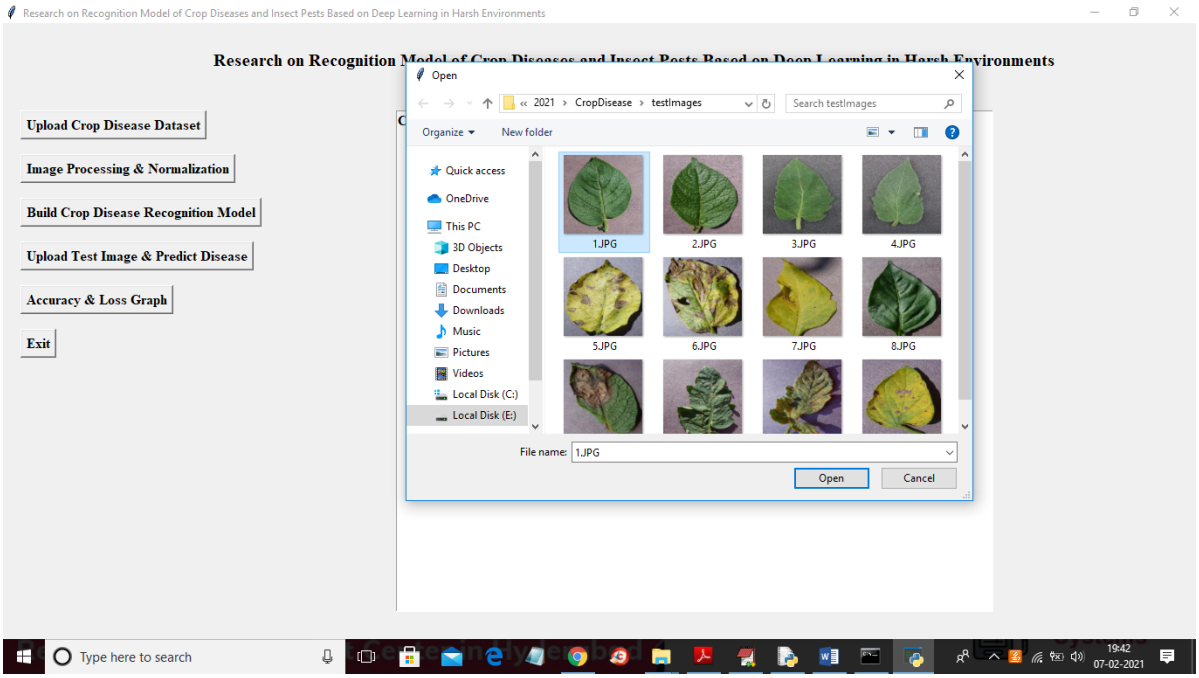
Screenshot 8: CNN layers details

In above screen CNN model generated and its prediction accuracy is 98% and in below console screen we Can see all CNN layers details



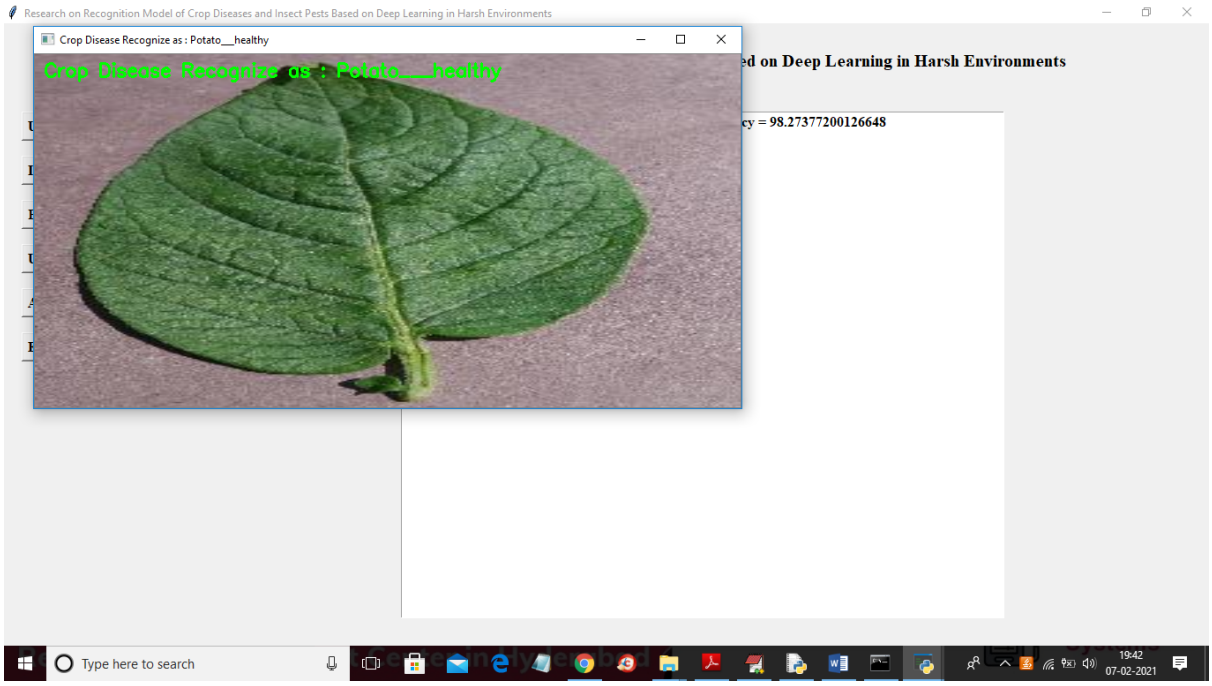
Screenshot 9: crop disease recognition model

In above screen we can see we have used CONV2D, MAXPOOLING, FLATTEN and DENSE layer to build crop disease recognition model and RELU details you can see in code. Now model is ready and now click on 'Upload Test Image & Predict Disease' button to upload any test image and then application will predict disease or healthy from that image



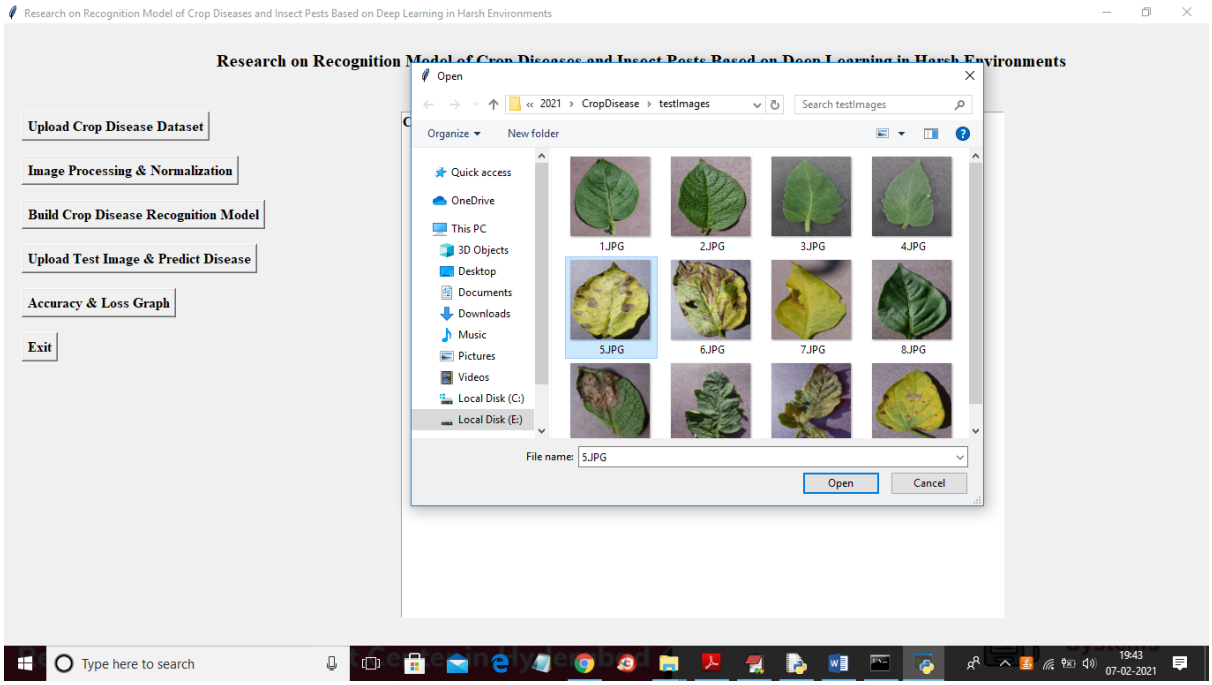
Screenshot 10: screen selecting and uploading '1.JPG' image file

In above screen selecting and uploading '1.JPG' image file and then click on 'Open' button to get below prediction result



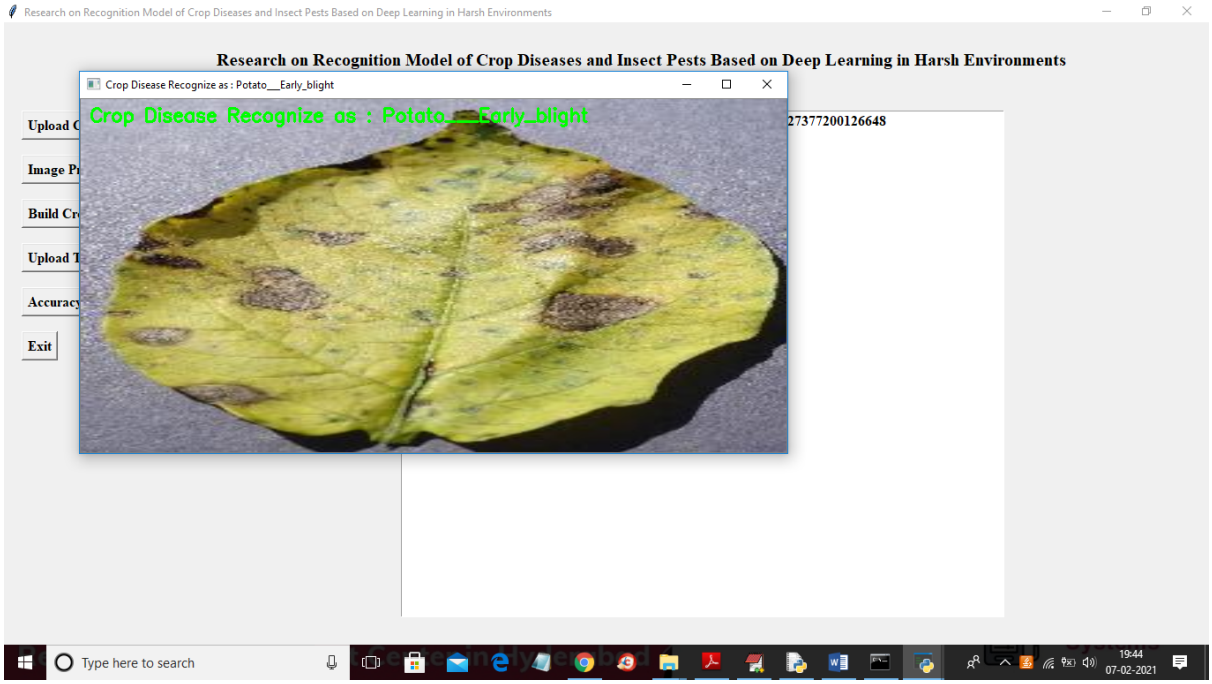
Screenshot 11: potato leaf predicted as healthy

In above screen potato leaf predicted as healthy and now try with other image



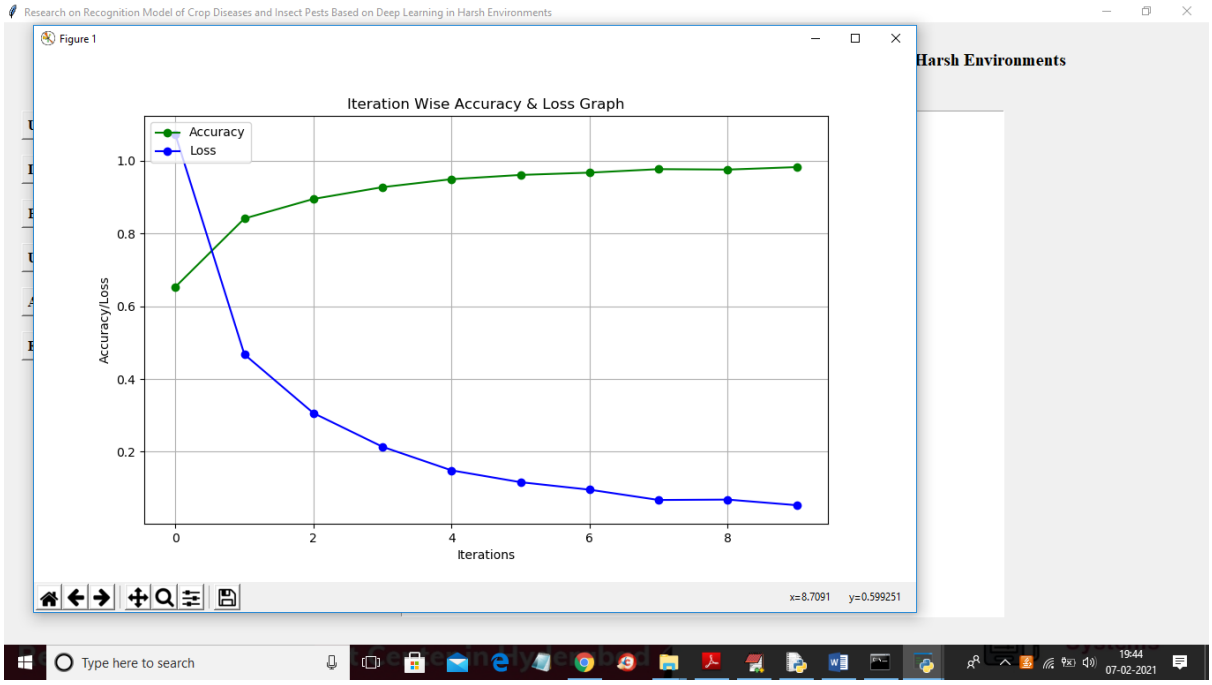
Screenshot 12: screen selecting and uploading '5.JPG' file

In above screen selecting and uploading '5.JPG' file and click 'Open' button to get below result



Screenshot 13: image potato EARLY BLIGHT disease is detected

In above image potato EARLY BLIGHT disease is detected or recognize and similarly you can upload any other image and get result and now click on 'Accuracy & Loss Graph' button to get below graph

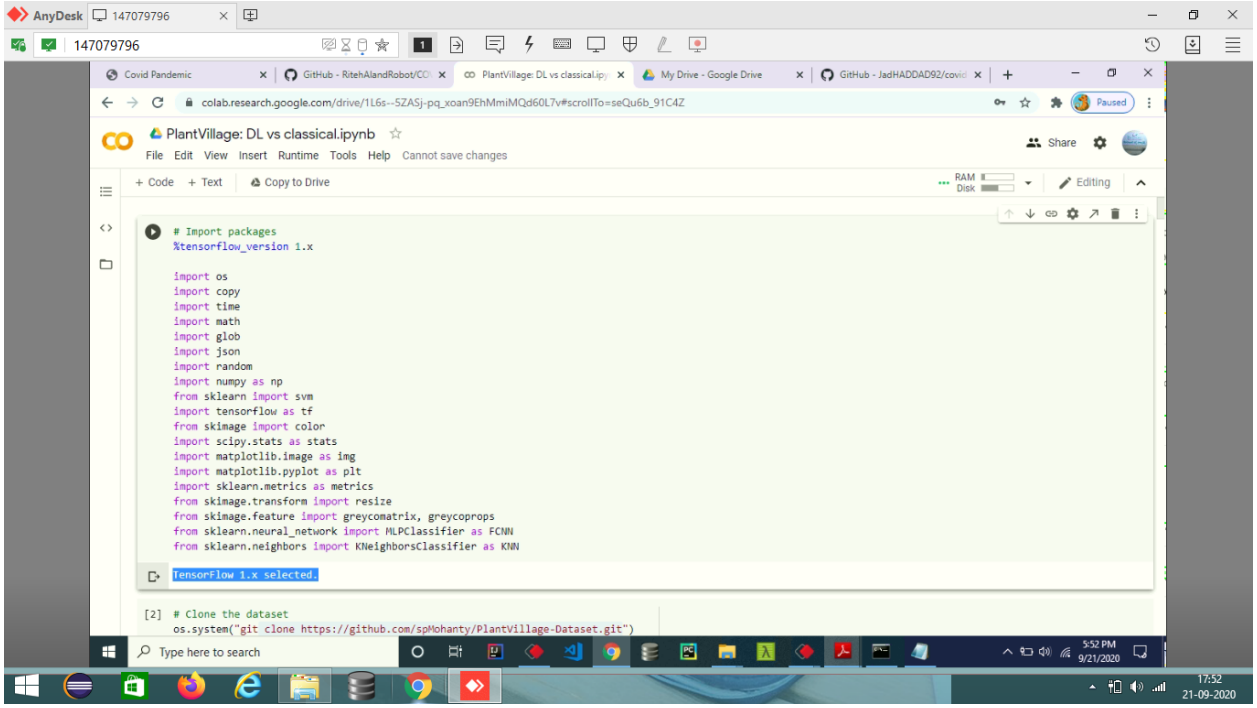


Screenshot 14: graph x-axis represents epoch/iterations and y-axis represents accuracy/loss

In above graph x-axis represents epoch/iterations and y-axis represents accuracy/loss and green line represents accuracy and blue line represents loss and from above graph we can see with each increasing iteration accuracy is getting better and better and loss getting decrease.

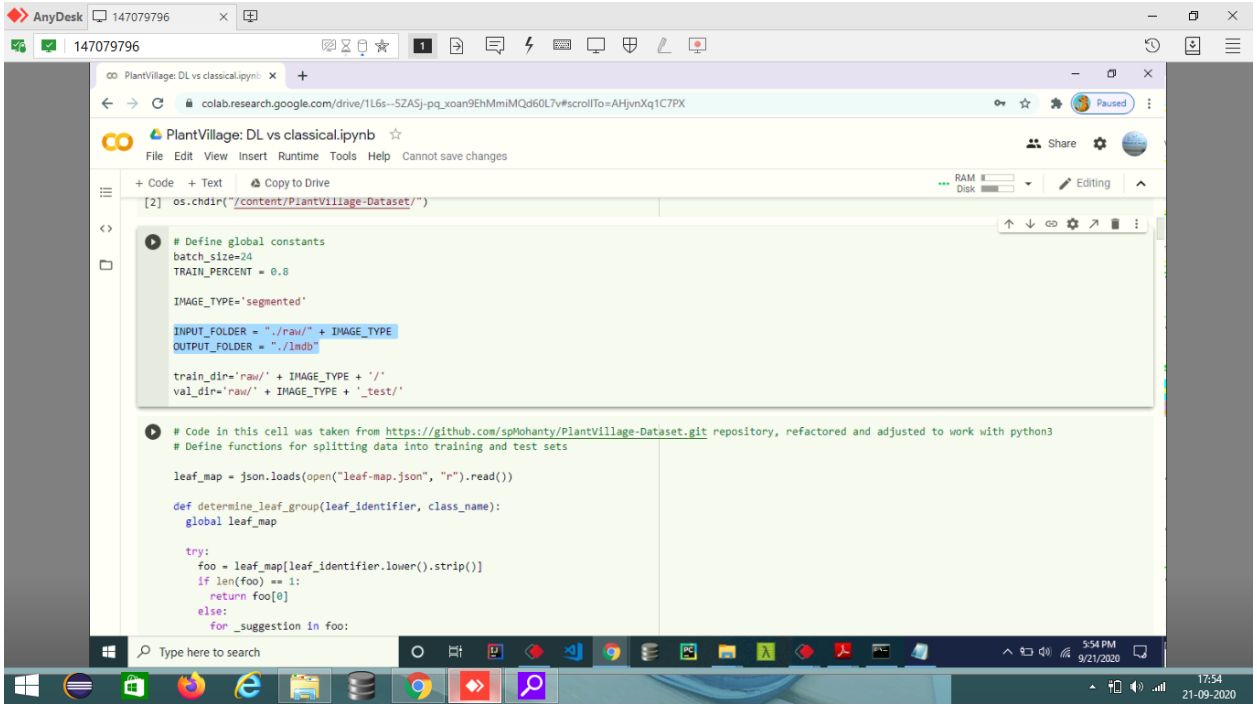
Screenshots

Parent Dataset:

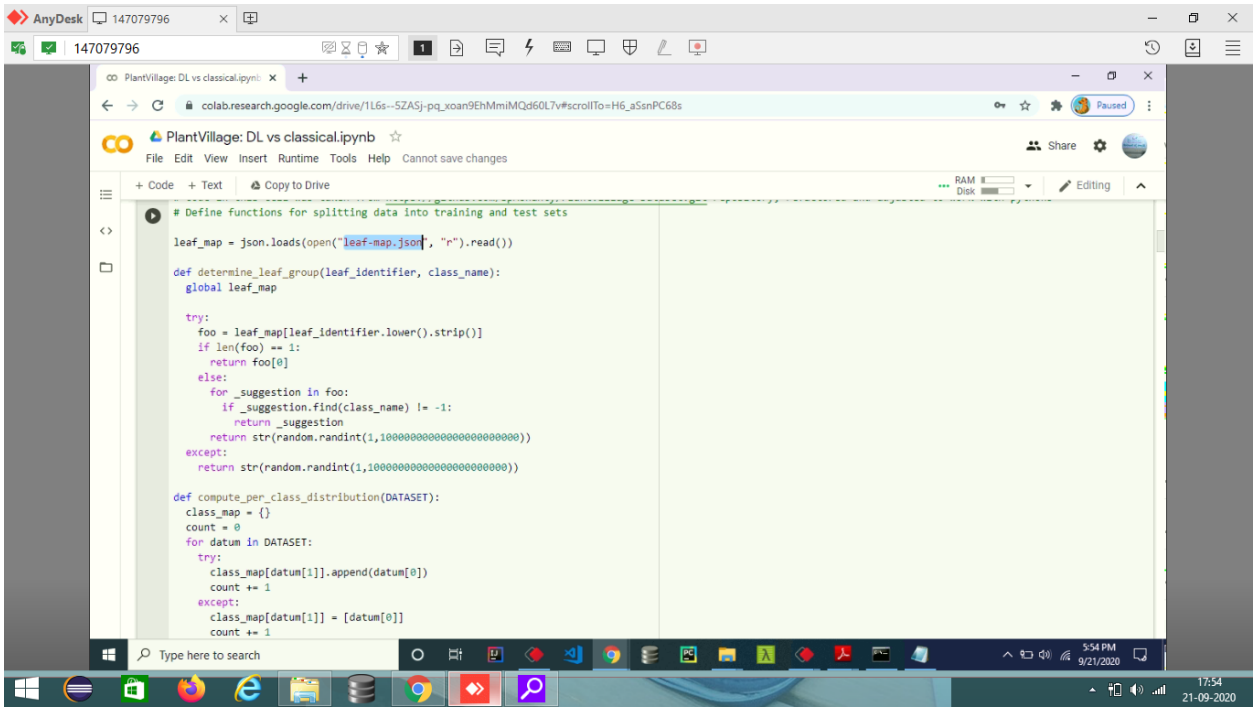


Screenshot 15: Parent Dataset

Input/Output folder:

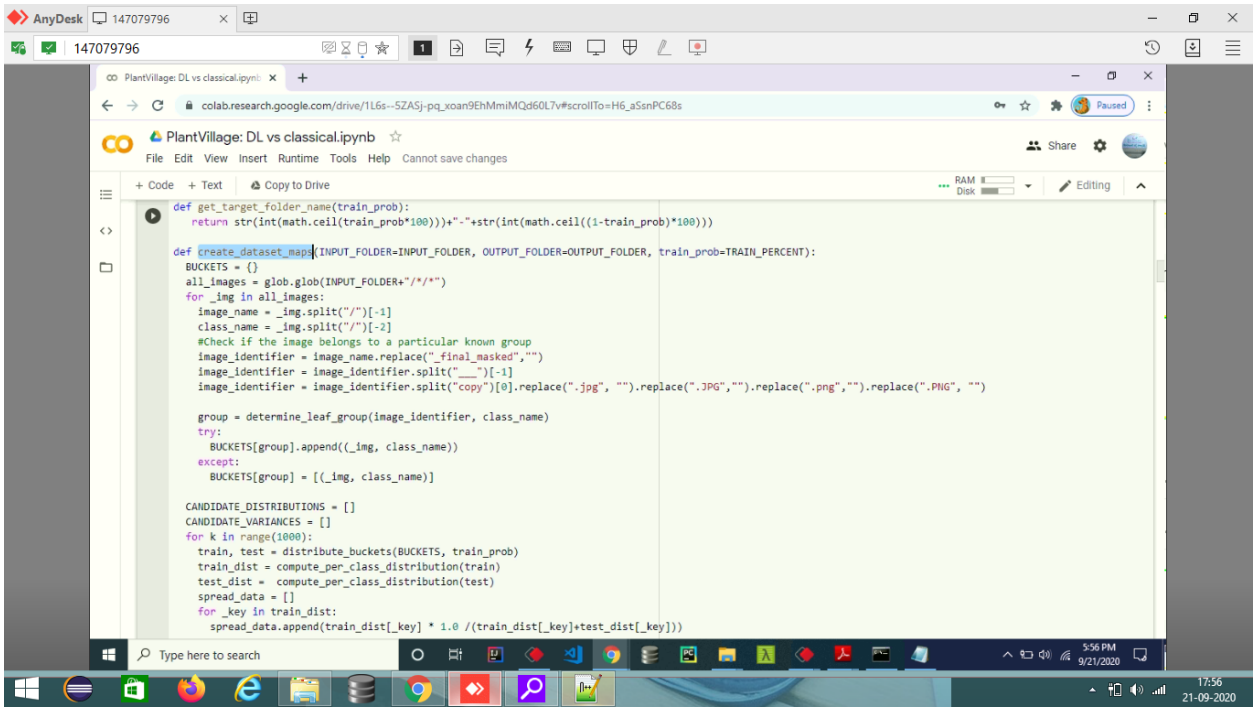


Screenshot 16: Input/Output folder



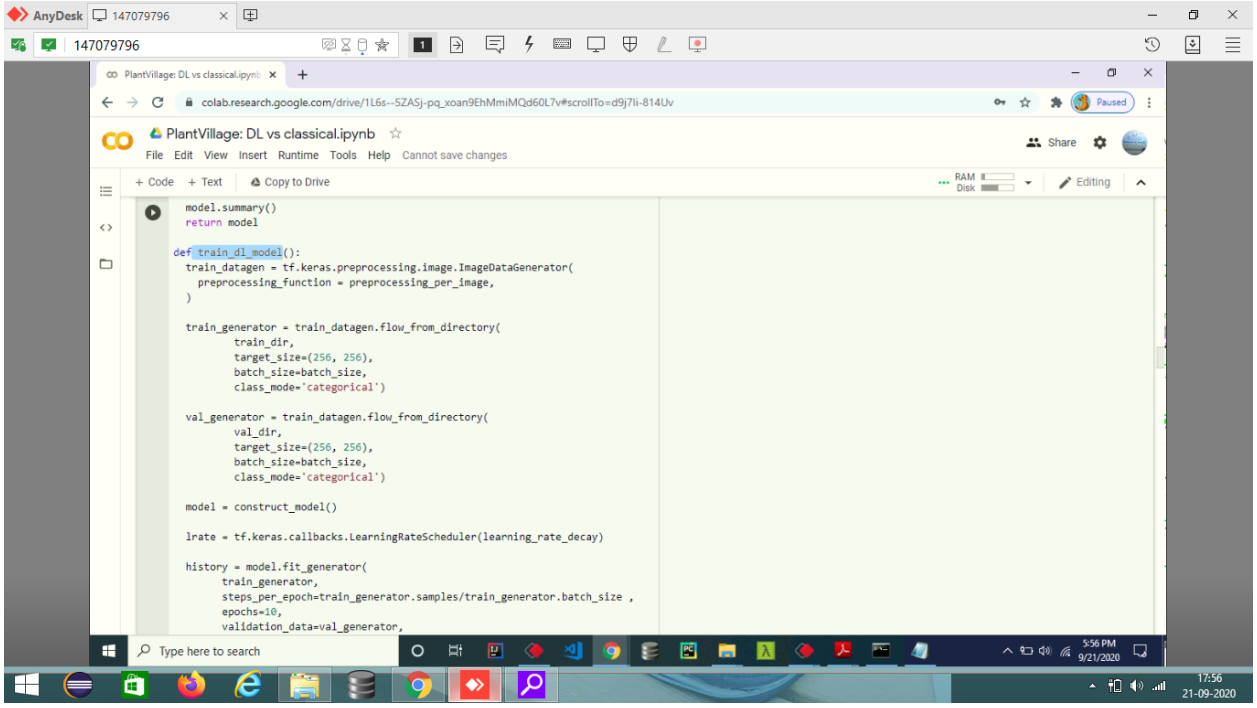
Screenshot 17: Input/Output folder

Dataset:



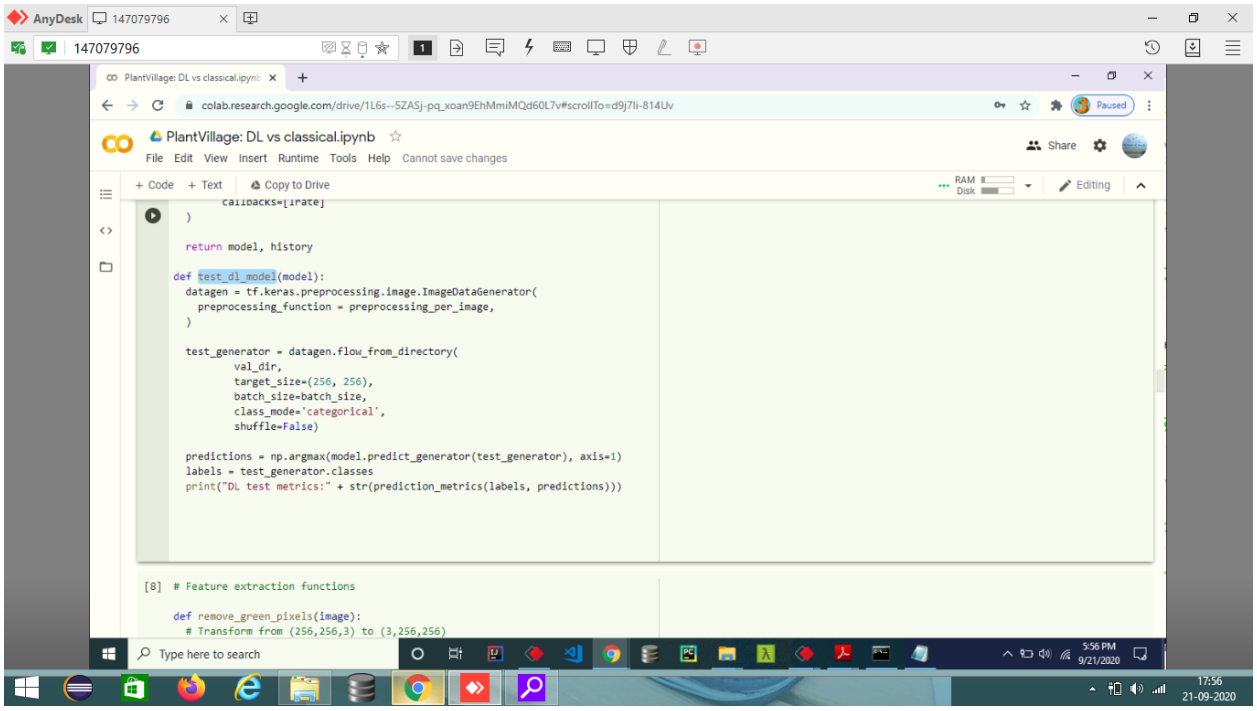
Screenshot 18: Dataset

Training model:



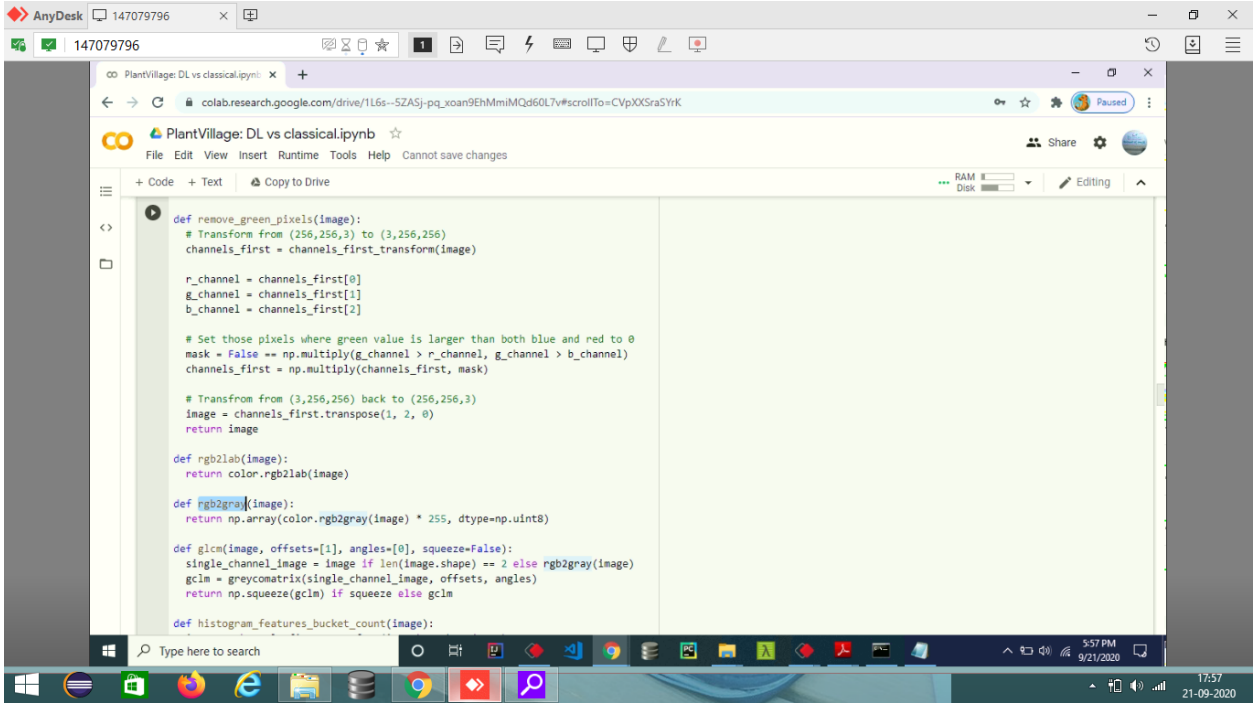
Screenshot 19: Training model

Test model:



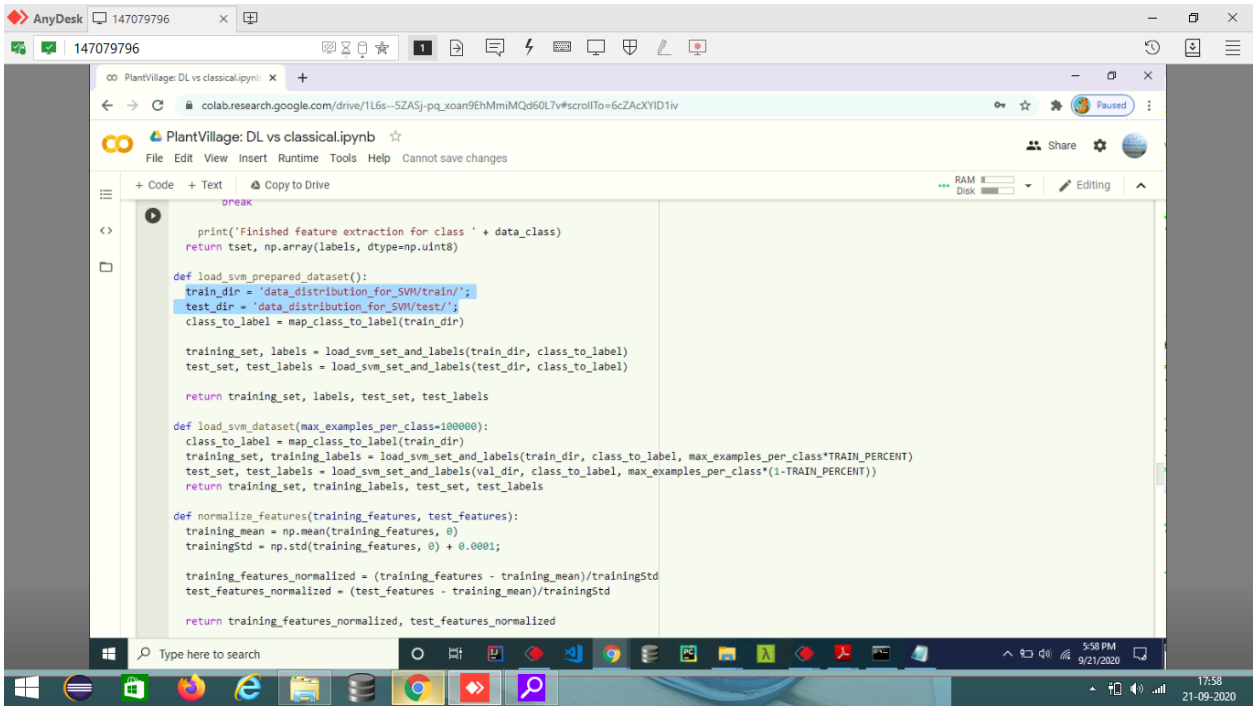
Screenshot 20: Test model

Gracl Image processing:



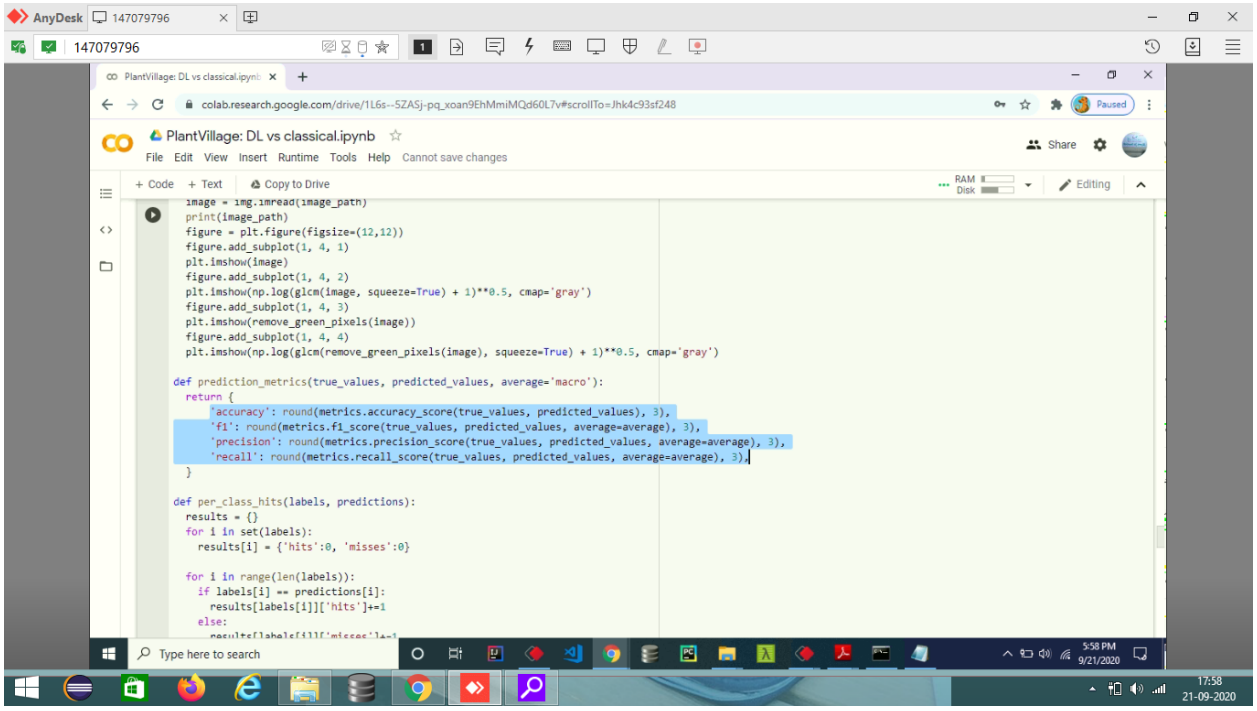
Screenshot 21: Gracl Image processing:

SVM:



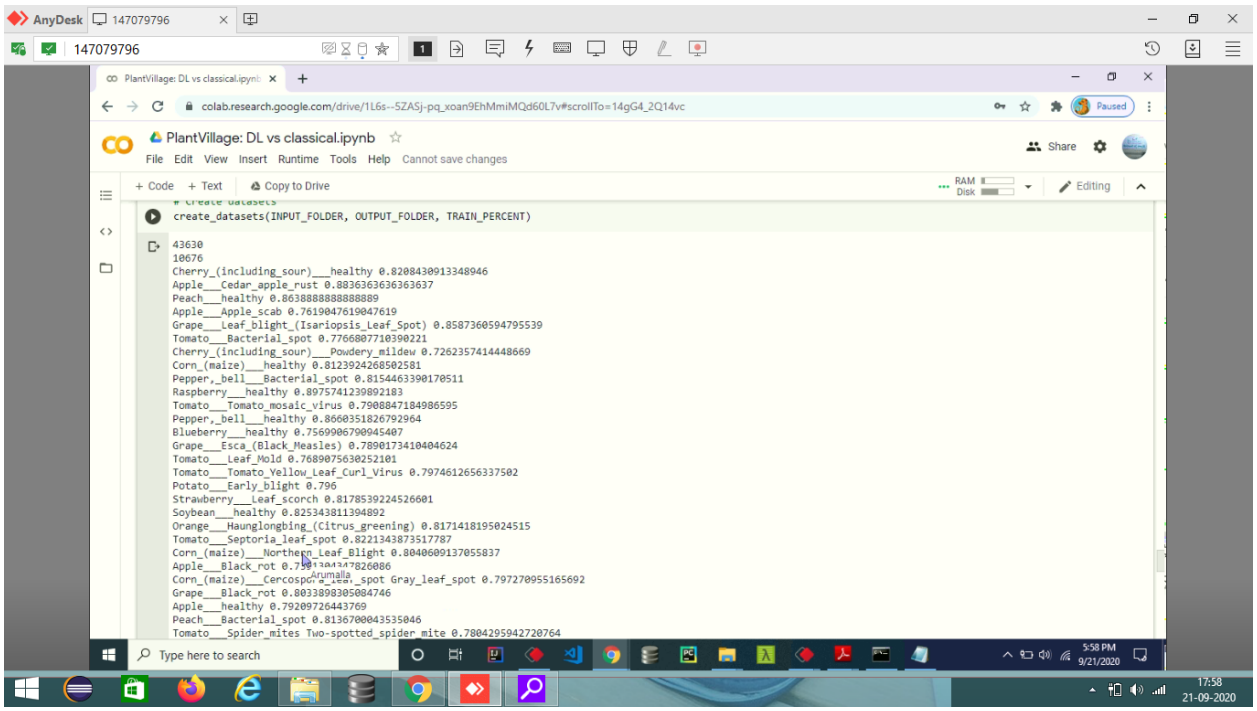
Screenshot 22: SVM

Accuracy:



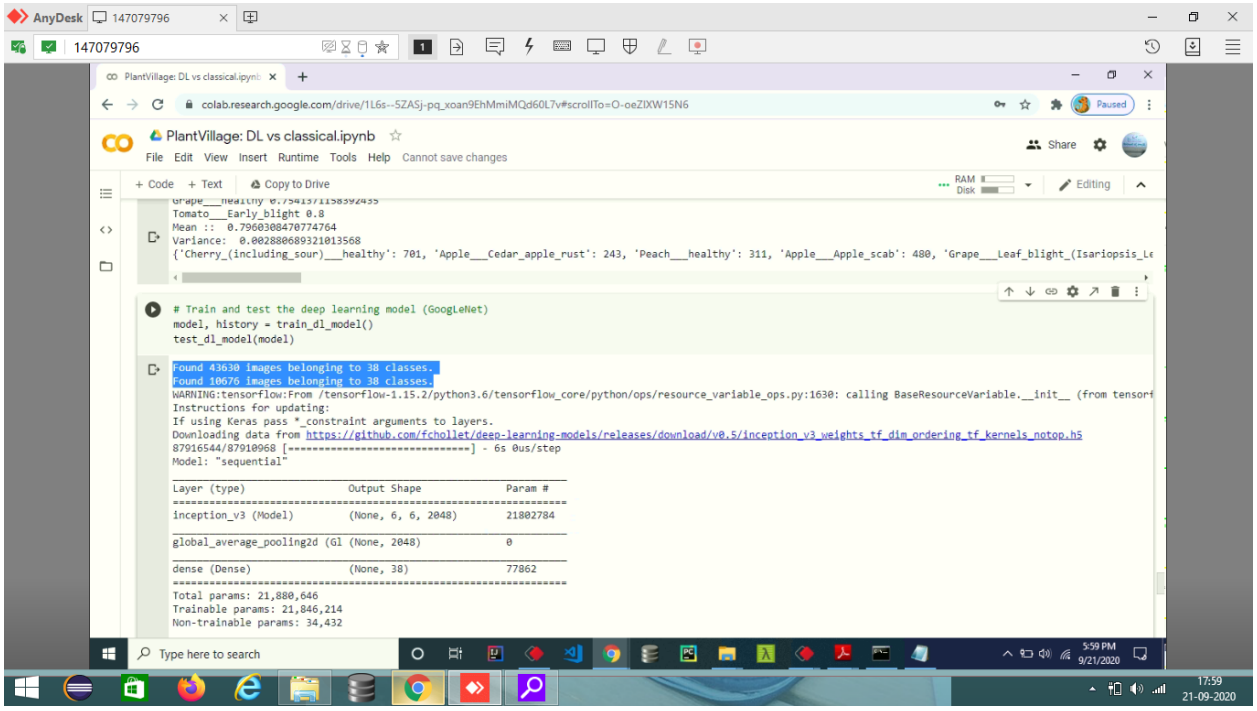
Screenshot 23: Accuracy

Datasets:



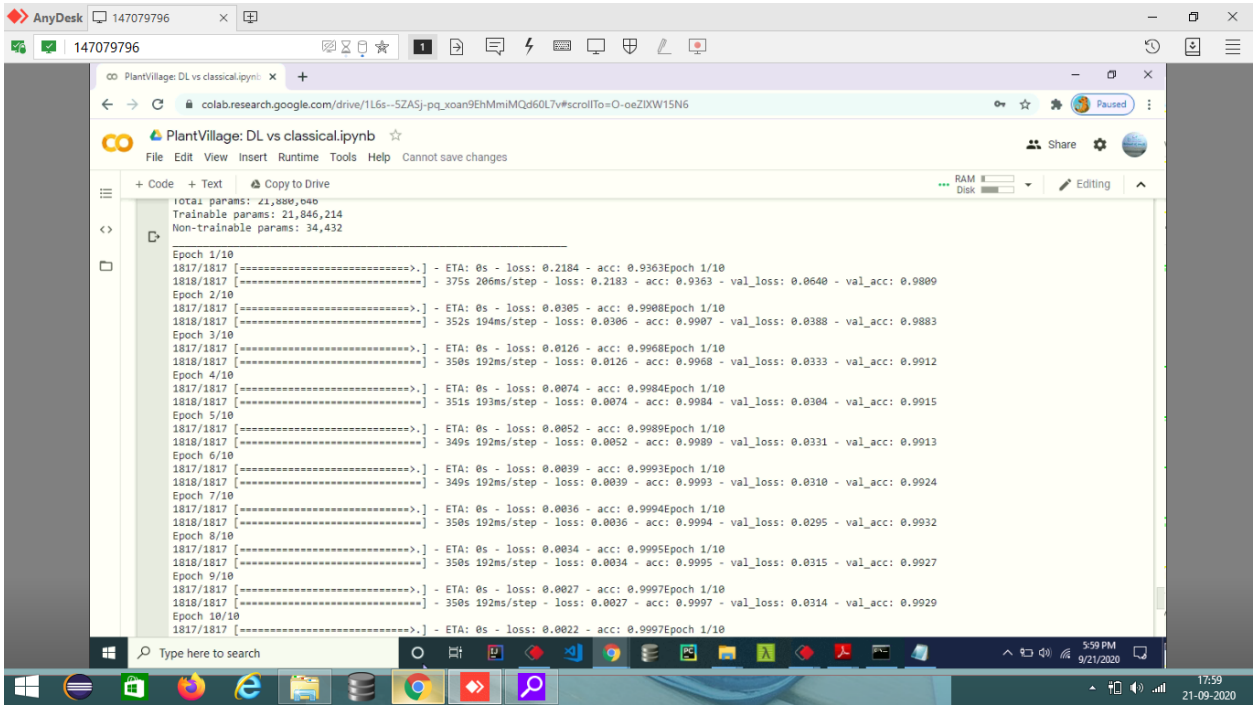
Screenshot 24: Datasets

Epoch starting:



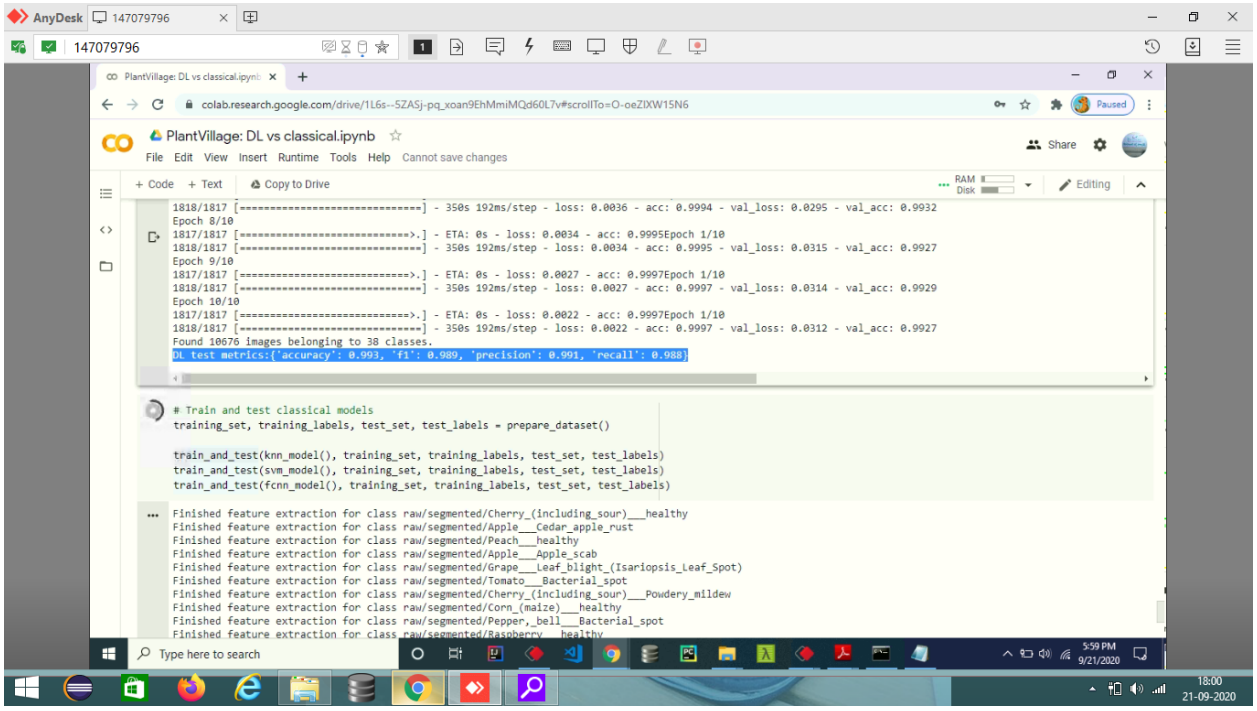
Screenshot 25: Epoch starting

Epoch executing and result:



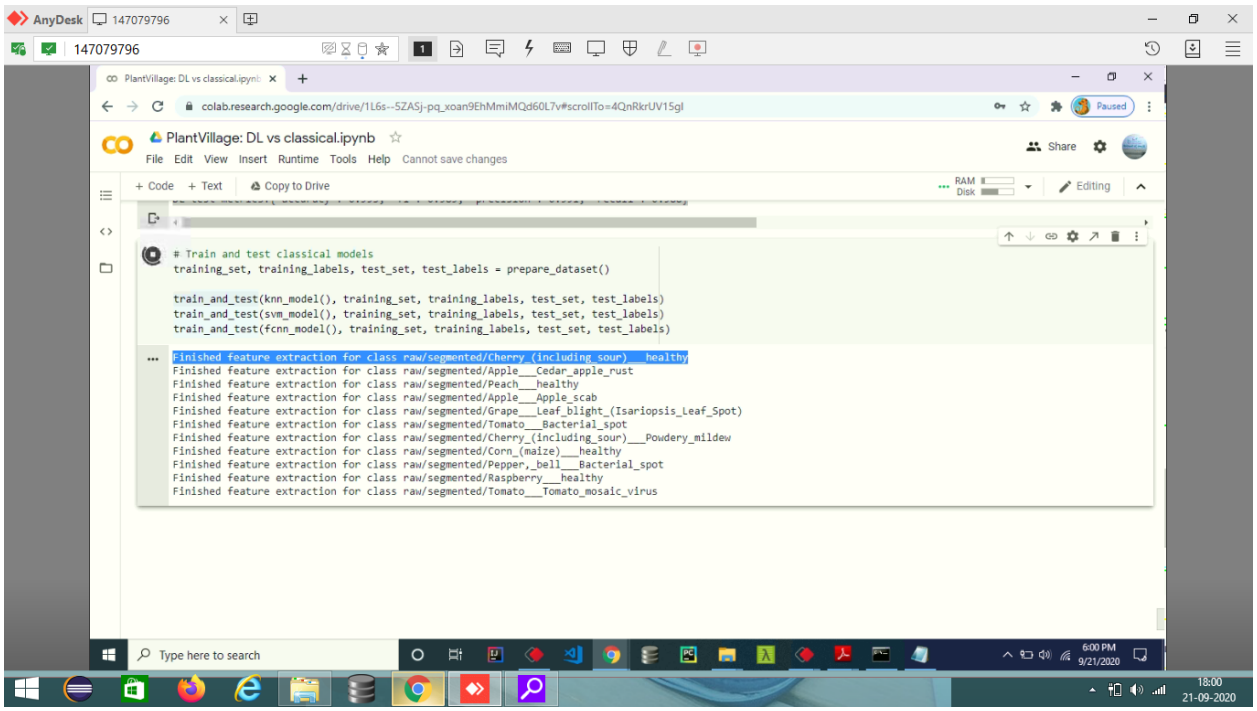
Screenshot 26: Epoch executing and result

Deep learning test result:

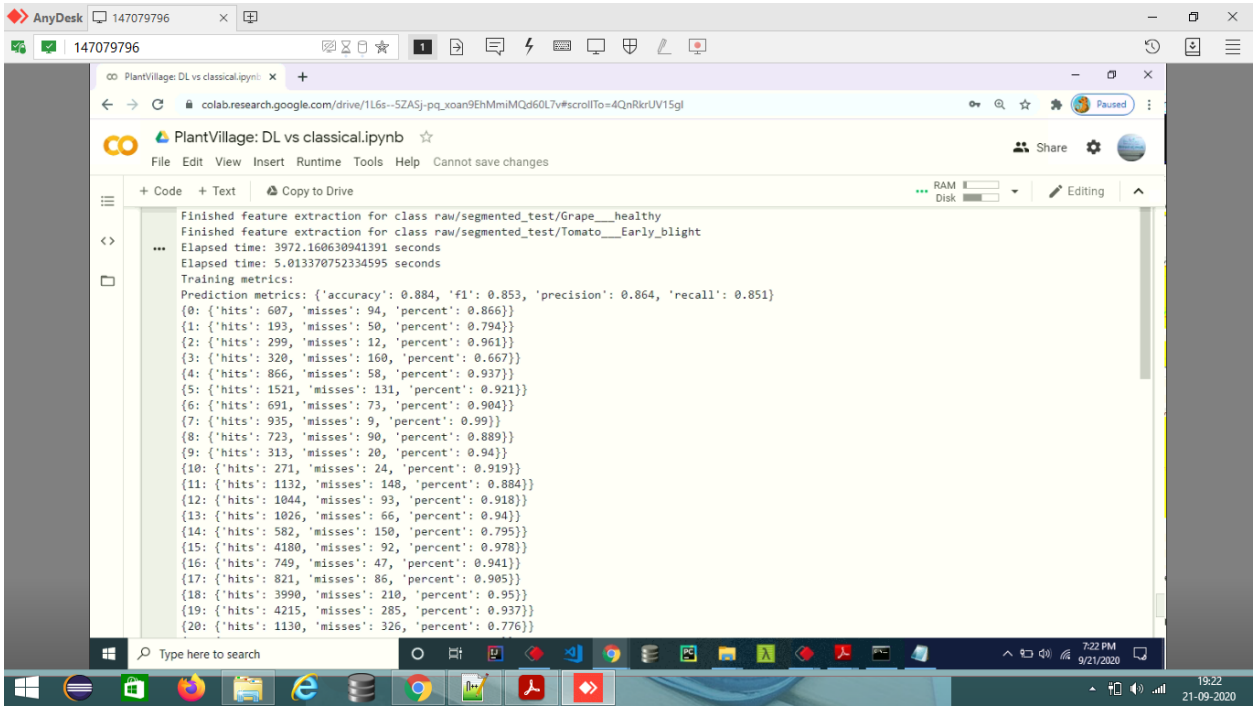


Screenshot 27: Deep learning test result

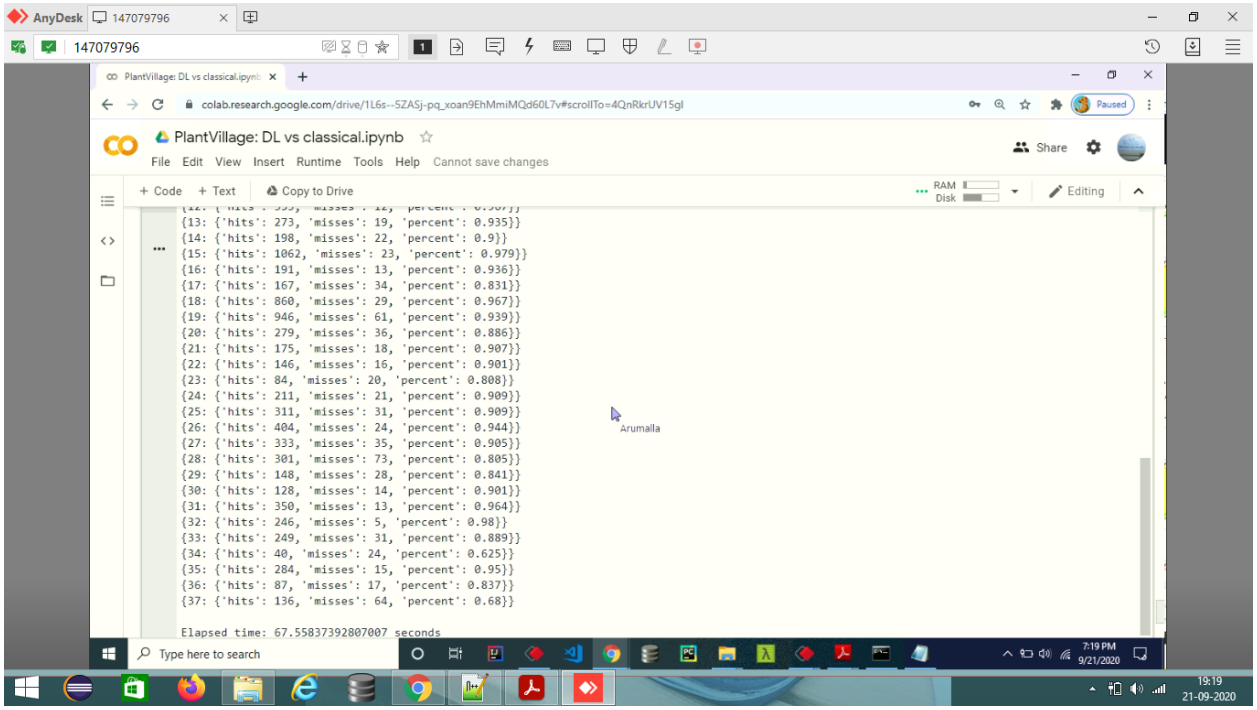
Extraction:



Screenshot 28: Extraction
Training/Testing result:

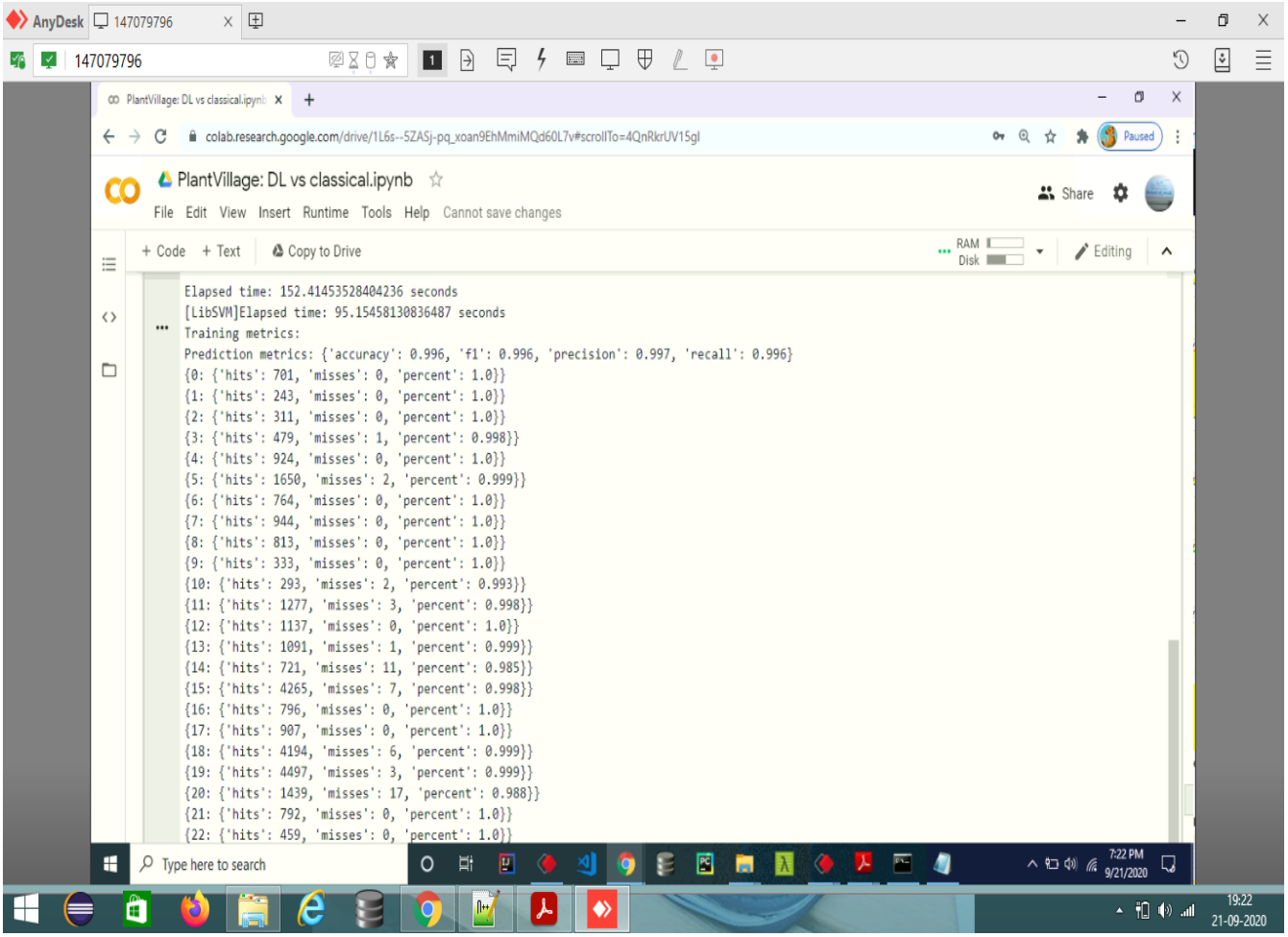


Screenshot 29: Training/Testing result



Screenshot 30: Training/Testing result

SVM Model:



`Screenshot 31: SVM Model

Chapter 6: TESTING

SYSTEM TEST

The purpose of this testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product it is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive.

Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and

consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	:	identified classes of valid input must be accepted.
Invalid Input	:	identified classes of invalid input must be rejected.
Functions	:	identified functions must be exercised.
Output	:	identified classes of application outputs must be exercised.
Systems/Procedures	:	interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing.

Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be

written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Chapter 7: CONCLUSION

7.1 Conclusion

In this project, 3 kinds of disease recognition of 10 kinds of crops were studied. The Inception-ResNet-v2 model is constructed by using deep learning theory and convolution neural network technology. Experiments show that the model can effectively identify the data set, and the overall recognition accuracy is as high as 86.1%. The results show that the recognition accuracy of this hybrid network model is relatively higher than the traditional model, and it can be effectively applied to the identification and detection of plant diseases and insect pests.

7.2 Future Enhancement

This code is implemented from scratch for extracting disease affected portions from images of Jute Plant Stems. Different image processing methods like Hue-based Segmentation, Morphological Analysis (i.e. erosion, dilation etc.), Blob Detection, Largest Connected Component, Color co-occurrence methodology, Texture Analysis etc. are implemented and applied. Extracted features are used for Jute Plant Disease Detection using.

REFERENCES

[1]United Nations, Department of Economic and Social Affairs, Population Division (2019). World Population Prospects 2019: Highlights (ST/ESA/SER.A/423).

[2] Savary, Serge, et al. "The global burden of pathogens and pests on major food crops." Nature ecology & evolution 3.3 (2019): 430.

[3] Mohanty, Sharada P., David P. Hughes, and Marcel Salathé. "Using deep learning for image-based plant disease detection." Frontiers in plant science 7 (2016): 1419.

[4] Fujita, E., et al. "A practical plant diagnosis system for field leaf images and feature visualization." International Journal of Engineering & Technology 7.4.11 (2018): 49-54.

[5] Haralick, Robert M., Karthikeyan Shanmugam, and Its' Hak Dinstein. "Textural features for image classification." IEEE Transactions on systems, man, and cybernetics 6 (1973): 610-621.

[6] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

[7] Cunningham, Padraig, and Sarah Jane Delany. "k-Nearest neighbour classifiers." Multiple Classifier Systems 34.8 (2007): 1-17.

[8] Haykin, Simon. Neural networks: a comprehensive foundation. Prentice Hall PTR, 1994.

[9] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[10] Duan, Kai-Bo, and S. Sathiya Keerthi. "Which is the best multiclass SVM method? An empirical study." International workshop on multiple classifier systems. Springer, Berlin, Heidelberg, 2005.

[11] Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009.