

## TABLE OF CONTENTS

S.No.	TITLE	PAGE No.
	<b>ABSTRACT</b>	<b>vi</b>
	<b>LIST OF FIGURES</b>	<b>vii</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Brief Information	1
	1.2 Motivation	2
	1.3 Objective	2
	1.4 Problem Statement	2
<b>2</b>	<b>SYSTEM ANALYSIS</b>	<b>3</b>
	2.1 Existing System	3
	2.2 Proposed System	3
	2.3 Feasibility Study	4
	2.3.1 Economical Feasibility	4
	2.3.2 Technical Feasibility	4
	2.3.3 Social Feasibility	4
	2.4 System Requirement Specification	5
	2.4.1 Functional Requirements	5
	2.4.2 Non-Functional Requirements	5
	2.5 Hardware Requirements	7
	2.6 Software Requirements	7
<b>3</b>	<b>SYSTEM DESIGN</b>	<b>8</b>
	3.1 System Architecture	8

	3.2 Modules	9
	3.3 UML Diagrams	10
	3.3.1 Use Case Diagram	11
	3.3.2 Class Diagram	12
	3.3.3 Sequence Diagram	13
	3.3.4 Activity Diagram	14
	3.3.5 Component Diagram	15
	3.3.6 Deployment Diagram	15
	3.4 Database Design	16
	3.4.1 Normalization	16
<b>4</b>	<b>SYSTEM IMPLEMENTATION</b>	<b>18</b>
	4.1 Front End Implementation	18
	4.2 Back End Implementation	27
	4.3 Source Code	30
	4.4 Output Screens	40
<b>5</b>	<b>SYSTEM TESTING</b>	<b>51</b>
	5.1 Testing Concepts	51
	5.2 Testing Strategies	51
	5.3 Test Cases	53
<b>6</b>	<b>CONCLUSION</b>	<b>55</b>
<b>7</b>	<b>REFERENCES</b>	<b>56</b>

## **ABSTRACT**

Automatic identification and recognition of medicinal plant species in environments such as forests, mountains and dense regions is necessary to know about their existence. In recent years, plant species recognition is carried out based on the shape, geometry and texture of various plant parts such as leaves, stem, flowers etc. Flower based plant species identification systems are widely used. While modern search engines provide methods to visually search for a query image that contains a flower, it lacks in robustness because of the intra-class variation among millions of flower species around the world. Hence in this proposed research work, a Deep learning approach using Convolutional Neural Networks (CNN) is used to recognize flower species with high accuracy. Images of the plant species are acquired using the built-in camera module of a mobile phone. Feature extraction of flower images is performed using a Transfer Learning approach (i.e. extraction of complex features from a pre-trained network). A machine learning classifier such as Logistic Regression or Random Forest is used on top of it to yield a higher accuracy rate. This approach helps in minimizing the hardware requirement needed to perform the computationally intensive task of training a CNN. It is observed that, CNN combined with Transfer Learning approach as feature extractor outperforms all the handcrafted feature extraction methods such as Local Binary Pattern (LBP) by using VGG19 pre-trained model Architecture we are obtaining 88.77% accuracy for the project.

## LIST OF FIGURES

<b>Fig.No.</b>	<b>FIGURE NAME</b>	<b>PAGE No.</b>
3.1	System Architecture	8
3.2	Flowchart of the Technique	9
3.3	UseCase Diagram Admin/User	11
3.4	Class Diagram	12
3.5	Sequence Diagram	13
3.6	Activity Diagram	14
3.7	Component Diagram	15
3.8	Deployment Diagram	15
4.1	Python Installation	20
4.2	Python Setup	21
4.3	Python Setup Progress	22
4.4	Python Path Setup	24

# 1. INTRODUCTION

## 1.1 BREIF INFORMATION

Classification is a systematic arrangement in groups and categories based on its features. Image classification came into existence for decreasing the gap between the computer vision and human vision by training the computer with the data. The image classification is achieved by differentiating the image into the prescribed category based on the content of the vision. Motivation by, in this paper, we explore the study of image classification using deep learning. The conventional methods used for image classifying is part and piece of the field of artificial intelligence (AI) formally called as machine learning. The machine learning consists of feature extraction module that extracts the important features such as edges, textures etc and a classification module that classify based on the features extracted. The main limitation of machine learning is, while separating, it can only extract certain set of features on images and unable to extract differentiating features from the training set of data. This disadvantage is rectified by using the deep learning. Deep learning (DL) is a sub field to the machine learning, capable of learning through its own method of computing.

A deep learning model is introduced to persistently break down information with a homogeneous structure like how a human would make determinations. To accomplish this, deep learning utilizes a layered structure of several algorithms expressed as an artificial neural system (ANN). The architecture of an ANN is simulated with the help of the biological neural network of the human brain. This makes the deep learning most capable than the standard machine learning models. In deep learning, we consider the neural networks that identify the image based on its features. This is accomplished for the building of a complete feature extraction model which is capable of solving the difficulties faced due to the conventional methods. The extractor of the integrated model should be able to learn extracting the differentiating features from the training set of images accurately. Many methods like GIST, histogram of gradient oriented and Local Binary Patterns, SIFT are used to classify the feature descriptors from the image.

## **1.2 MOTIATION**

Flowers are everywhere around us. They can feed insects, birds, animals and humans. They are also used as medicines for humans and some animals. A good understanding of flowers is essential to help in identifying new or rare species when came across. This will help the medicinal industry to improve. The system proposed in the paper can be used by botanists, campers and doctors alike. This can be extended as an image search solution where photo can be taken as an input instead of text in order to get more information about the subject and search accordingly for best matching results.

## **1.3 OBJECTIVE**

In general it is very complex and voluminous to process the huge amount of data which is generated for the classification of images. Hence this motivated me to design the current application using a deep learning network for classification of different flowers. With this we can able to easily classify the different types of flowers.

## **1.4 PROBLEM STATEMENTS**

In this system we try to use deep learning model to classify different type of images and predict the accurate images from that set of images. Once the prediction is done then the report is generated for the end users.

## 2. SYSTEM ANALYSIS

### 2.1 EXISTING SYSTEM

In the existing system there was no proper method to identify the species of medicinal plants based on images and then categorize into individual categories based on their image quality like size, shape, color, species and lot more. The following are the main limitations in the existing system.

#### DISADVANTAGES OF EXISTING SYSTEM

1. More Time Delay in finding the Type of species from a vast number of images.
2. There is no description for every set of matched flower
3. There is no technique which can predict the accuracy of flower once they are identified.
4. There is no method to classify the flowers or plants and find out the name of that flower automatically by using any technique.

### 2.2 PROPOSED SYSTEM

We have developed a deep learning network for classification of different flowers. For this, we have used kaggle category flower data-set having 4242 images of flowers. The data collection is based on the data flicr, google images, yandex images. You can use this dataset to recognize plants from the photo. The pictures are divided into five classes: **chamomile, tulip, rose, sunflower, dandelion**. For each class there are about 800 photos. Photos are not high resolution, about 320x240 pixels. Photos are not reduced to a single size, they have different proportions. This method for classification of flowers can be implemented in real-time applications and can be used to help botanists for their research as well as camping enthusiasts.

#### ADVANTAGES OF PROPOSED SYSTEM

- 1) By using data mining techniques it takes less time for the classification of the flower or plants with more accuracy.
- 2) There is high accuracy and efficiency in predicting the type of flower

3) This will help botanist for their research for finding new species from different flowers.

## **2.3 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are

### **2.3.1 Economical Feasibility**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **2.3.2 Technical Feasibility**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

### **2.3.3 Social Feasibility**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.



## 2.4 SYSTEM REQUIREMENT SPECIFICATION

### 2.4.1 Functional Requirements

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define *what* a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Generally, functional requirements are expressed in the form “system shall do <requirement>”. The plan for implementing functional requirements is detailed in the system *design*. In requirements engineering, functional requirements specify particular results of a system. Functional requirements drive the application architecture of a system. A requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements is: user/stakeholder request -> feature -> use case -> business rule.

Functional requirements drive the application architecture of a system. A requirements analyst generates use cases after gathering and validating a set of functional requirements. Functional requirements may be technical details, data manipulation and other specific functionality of the project is to provide the information to the user. The following are the Functional requirements of our system:

- We are providing all the information related to several flowers and its species information.
- The user can give sample flower image as input to the system and can find out type of flower and its corresponding species name.
- We are using CNN as best model to predict the accurately.

### 2.4.2 Non Functional Requirements

Non-functional requirements define the overall qualities or attributes of the resulting System Non-functional requirements place restrictions on the product being developed, the development process, and specify external constraints that the product must meet. Examples of NFR include safety, security, usability, reliability and performance Requirements. Project

management issues (costs, time, and schedule) are often considered as non-functional requirements.

### **Performance requirements**

Requirements about resources required, response time, transaction rates, throughput, benchmark specifications or anything else having to do with performance. In this project, the user will try to gather all the information from KAGGLE website and then try to train the system with that dataset.

### **Modifiability**

Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person- months).

### **Portability**

The effort required to move the software to a different target platform. The measurement is most commonly person-months or % of modules that need changing.

### **Reliability**

Requirements about how often the software fails. The measurement is often expressed in MTBF (mean time between failures). The definition of a failure must be clear. Also, don't confuse reliability with availability which is quite a different kind of requirement. Be sure to specify the consequences of software failure, how to protect from failure, a strategy for error detection, and a strategy for correction.

### **Security**

One or more requirements about protection of your system and its data. The measurement can be expressed in a variety of ways (effort, skill level, time) to break into the system. Do not discuss solutions (e.g. passwords) in a requirements document.

### **Usability**

Requirements about how difficult it will be to learn and operate the system. The requirements are often expressed in learning time or similar metrics.

### **Legal**

There may be legal issues involving privacy of information, intellectual property rights, export of restricted technologies, etc.

## **2.5 HARDWARE REQUIREMENTS**

- Processor : Core I3
- RAM : 4 GB(min)
- Hard Disk : 100 GB

## **2.6 SOFTWARE REQUIREMENTS**

- ❖ Operating system : Windows 7 Ultimate.
- ❖ Coding Language : Python.
- ❖ Front-End : Python.
- ❖ Data Base : ML and Deep Learning Model using KAGGLE Dataset

### 3. SYSTEM DESIGN

#### 3.1 SYSTEM ARCHITECTURE

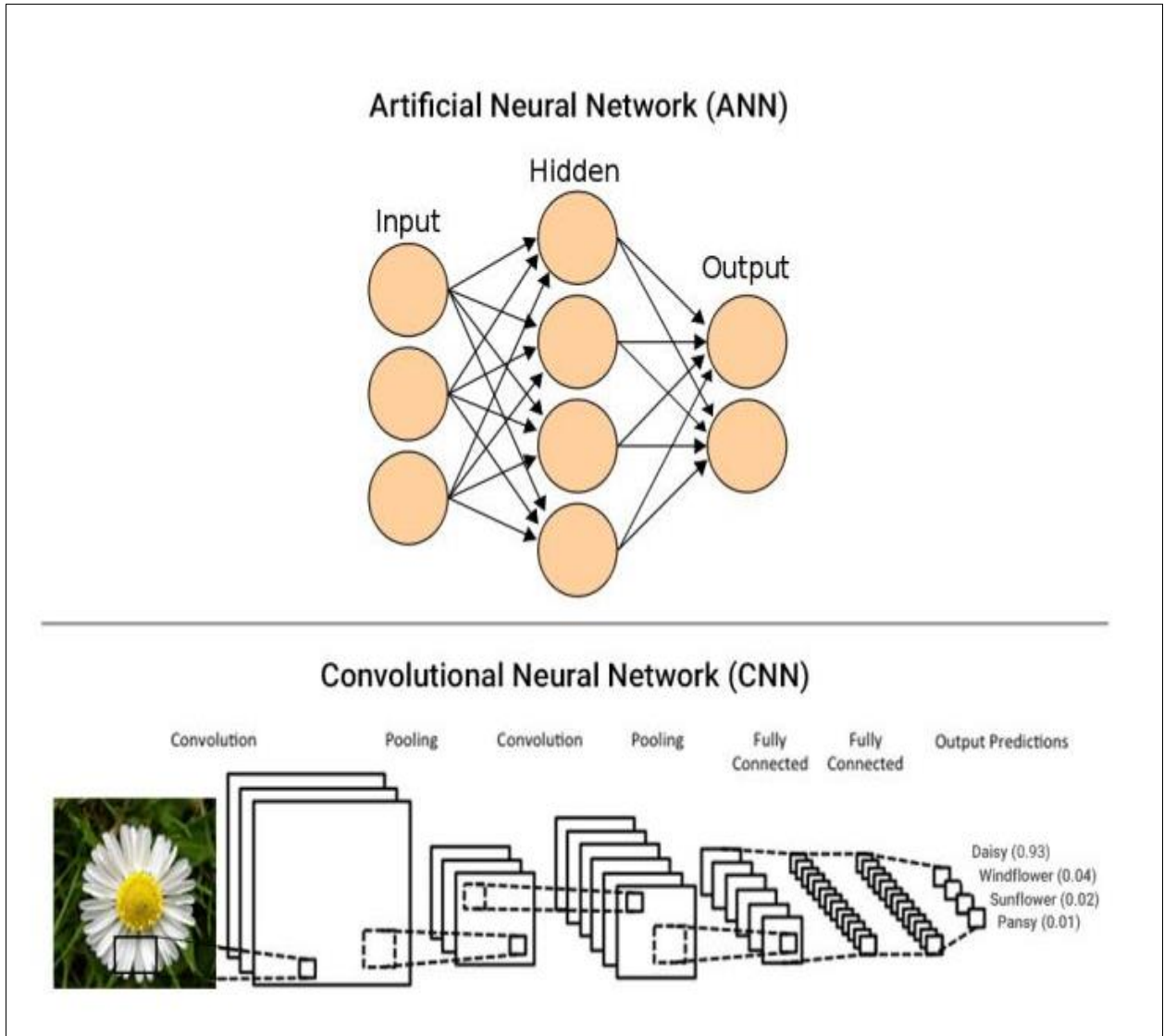


Fig. 3.1 System Architecture

## 3.2 MODULES

The modules are Gathering data, Pre-Processing, Processing, Interpretation.

The whole approach is depicted by the following flowchart.

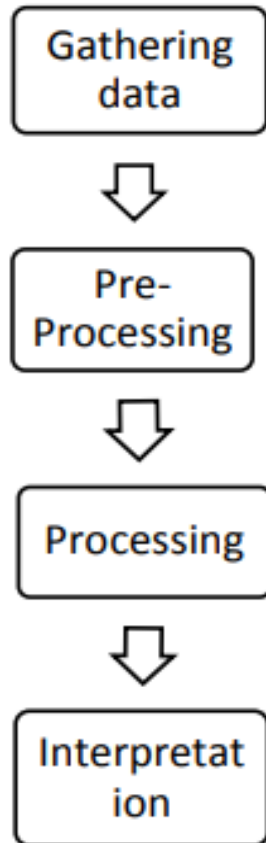


Fig. 3.2 Flowchart of the Technique

### 3.2.1 DATA GATHERING

Here we try to load the dataset from kaggle website using Json file. Here the dataset what we use for our application is flowers-recognition dataset. This dataset contains 4242 images of flowers. The data collection is based on the data flicr, google images, yandex images. You can use this dataset to recognize plants from the photo.

### **3.2.2 PRE-PROCESSING**

Data pre-processing is a technique that is used to convert raw data into a clean dataset. A general rule of the thumb is to assign 80% of the dataset to training set and therefore the remaining 20% to test set.

### **3.2.3 IMPORT LIBRARIES**

Here we try to import all the necessary libraries which are present to design VGG-19 CNN model for FLOWER recognition & species identification.

### **3.2.4 FLOWER PREDICTION USING CNN**

Here we try to apply VGG-19 CNN model on given input data and then we can able to perform the FLOWER identification based on sample images which we upload as input.

## **3.3 UML DIAGRAMS**

The underlying premise of UML is that no one diagram can capture the different elements of a System in its entirety. Hence, UML is made up of nine diagrams that can be used to model a System at different points of time in the software life cycle of a system.

A software system can be said to have two distinct characteristics: a structural, "static" part and a behavioral, "dynamic" part. In addition to these two characteristics, an additional characteristic that a software system possesses is related to implementation. Before we categorize UML diagrams into each of these three characteristics, let us take a quick look at exactly what these characteristics are.

- Use case diagram
- Class diagram
- Object diagram
- State diagram
- Activity diagram
- Sequence diagram
- Collaboration diagram
- Component diagram
- Deployment diagram

### 3.3.1 Use case Diagram

The use case diagram is used to identify the primary elements and processes that form the System. The primary elements are termed as "actors" and the processes are called "use cases." The use case diagram shows which actors interact with each use case.

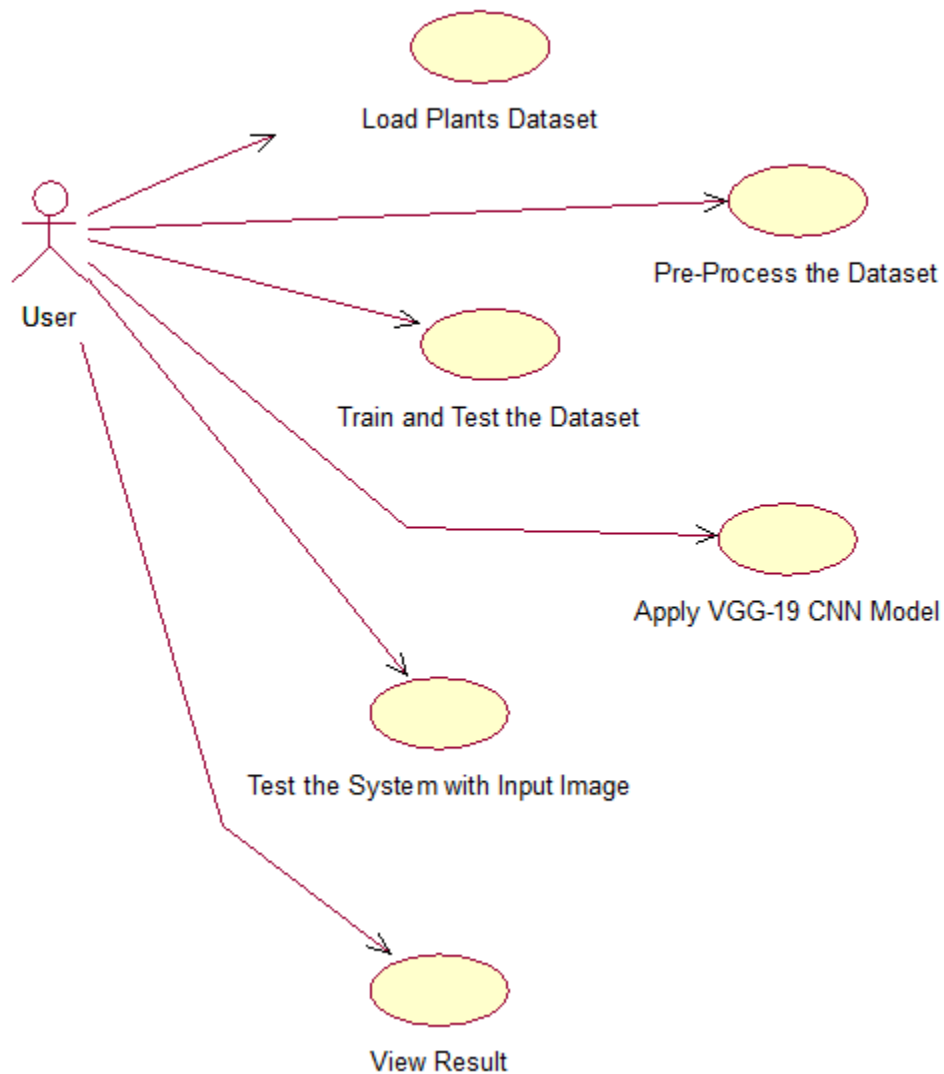


Fig 3.3 Usecase Diagram Admin/User

**Use cases** A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

**Actors** An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

### 3.3.2 Class Diagram

The class diagram is used to refine the use case diagram and define a detailed design of the System. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" Or "has-a" relationship.

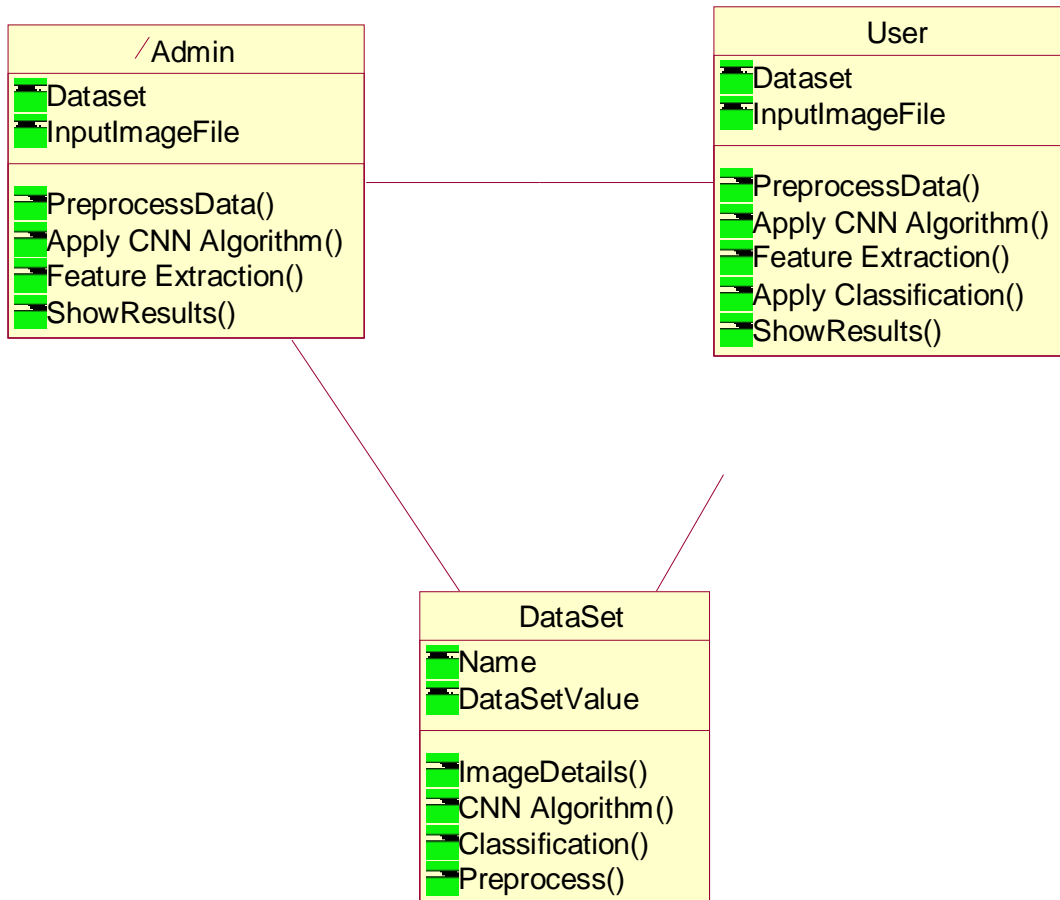


Fig 3.4 Class Diagram



### 3.3.3 Sequence Diagram

A sequence diagram represents the interaction between different objects in the system. The Important aspect of a sequence diagram is that it is time-ordered. Different objects In the sequence diagram interact with each other by passing "messages".

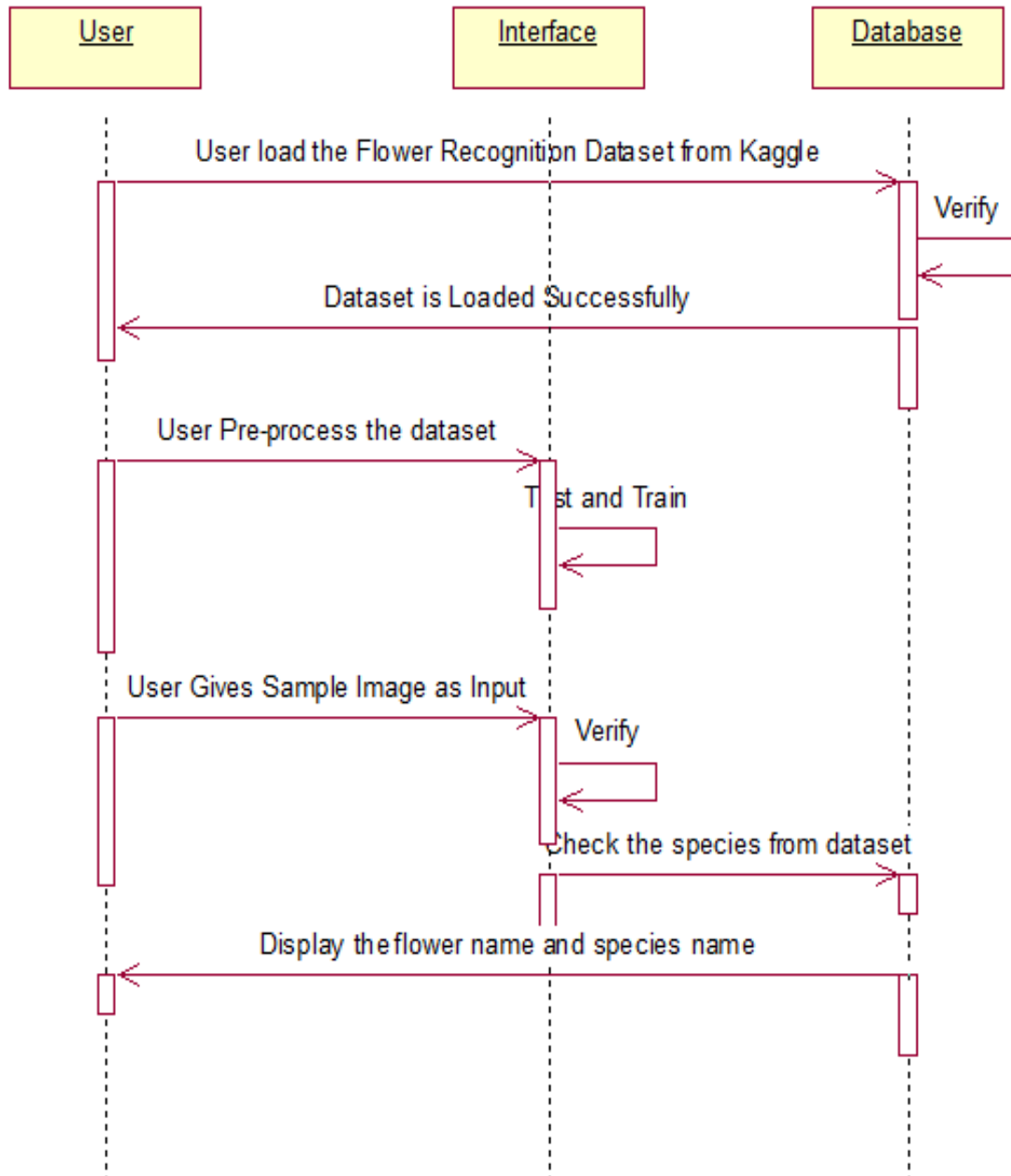


Fig 3.5 Sequence Diagram

### 3.3.4 Activity Diagram

The process flows in the system are captured in the activity diagram. Similar to a state Diagram, an activity diagram also consists of activities, actions, transitions, initial and final States, and guard conditions.

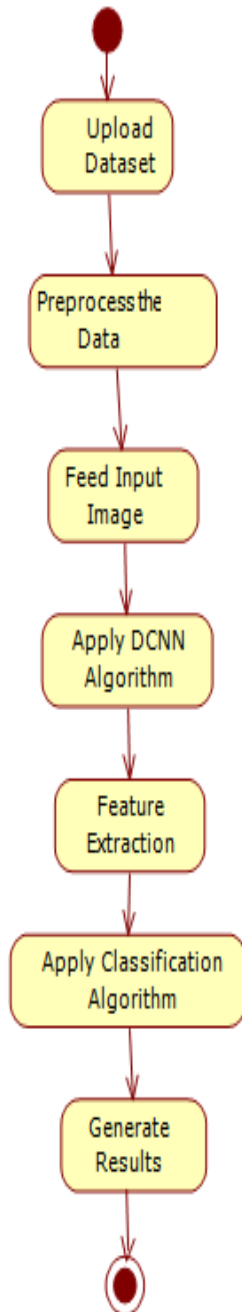


Fig 3.6 Activity Diagram

### 3.3.5 Component Diagram

The process of this diagram shows the organizations and dependencies among a set of components. It represents the static implementation view of a system.

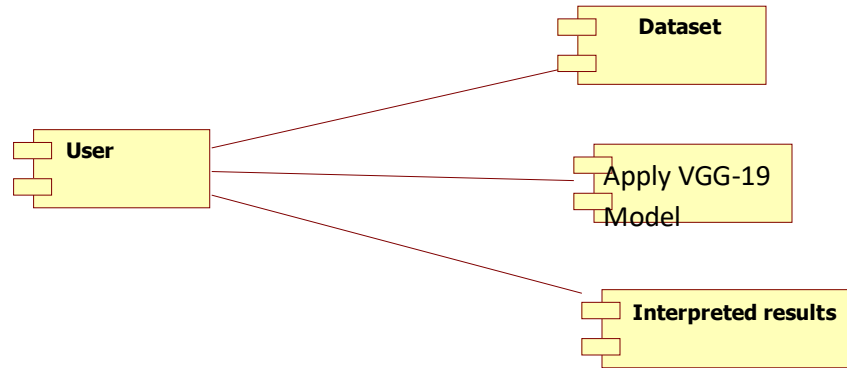


Fig 3.7 Component Diagram

### 3.3.6 Deployment Diagram

The deployment diagram captures the configuration of the runtime elements of the Application. This diagram is by far most useful when a system is built and ready to be Deployed. The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related. The purpose of deployment diagrams can be described as:

- Visualize hardware topology of a system.
- Describe the hardware components used to deploy software components.
- Describe runtime processing nodes.

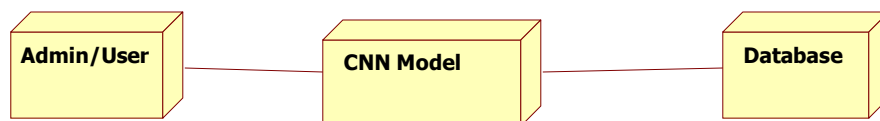


Fig. 3.8 Deployment Diagram

## 3.4 DATABASE DESIGN

The data pertaining to proposed system is voluminous that a careful design of the database must proceed before storing the data in the database. A database management system Provides flexibility in the storage and retrieval of data and production of information. The DBMS is a bridge between the application programs, which determines what data are needed and how they are processed, and the operating system of the computer, which is Responsible for placing data on the magnetic storage devices.

### 3.4.1 Normalization

Normalization theory is built around the concept of normal forms. A relation is said to be in particular normal form if it satisfies a certain specified set of constraints.

#### First Normal form

A relation R is in first normal form if and only if all underlying domains contained atomic values only

#### Second Normal form

A relation R is said to be in second normal form if and only if it is in first normal form and every non-key attribute is fully dependent on the primary key.

#### Third Normal form

A relation R is said to be in third normal form if and only if it is in second normal form and every non key attribute is non transitively depend on the primary key.

#### Boyce and Codd Normal Form (BCNF)

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

- R must be in 3rd Normal Form
- For each functional dependency (  $X \rightarrow Y$  ), X should be a super Key.

#### Fourth Normal Form (4NF)

Fourth Normal Form comes into picture when Multi-valued Dependency occurs in any relation.

For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

- It should be in the Boyce-Codd Normal Form.
- The table should not have any Multi-valued Dependency.

## 4. SYSTEM IMPLEMENTATION

### 4.1 FRONT END IMPLEMENTATION

#### Python Introduction

**Python** is a general purpose, dynamic, high level, and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development. Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development. It supports multiple programming pattern, including object-oriented, imperative, and functional or procedural programming styles. Python is not intended to work in a particular area, such as web programming. That is why it is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD, etc. We don't need to use data types to declare variable because it is dynamically typed so we can write `a=10` to assign an integer value in an integer variable. It makes the development and debugging fast because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

#### Python Applications

Python is known for its general purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development. Here, we are specifying applications areas where python can be applied.

- **Web Applications:** We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, BeautifulSoup, Feedparser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and develop web based applications. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc. The useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering.
- **Desktop GUI Applications:** Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

- **Software Development:** Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.
- **Scientific and Numeric:** Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.
- **Business Application:** Python is used to build Bussiness applications like ERP and e-commerce systems. Tryton is a high level application platform.
- **Console Based Application:** We can use Python to develop console based applications. For example: IPython.
- **Audio or Video based Applications:** Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.\
- **3D CAD Applications:** To create CAD application Fandango is a real application which provides full features of CAD.
- **Enterprise Applications:** Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.
- **Applications for Images:** Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc. Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development. It supports multiple programming pattern, including object-oriented, imperative, and functional or procedural programming styles.

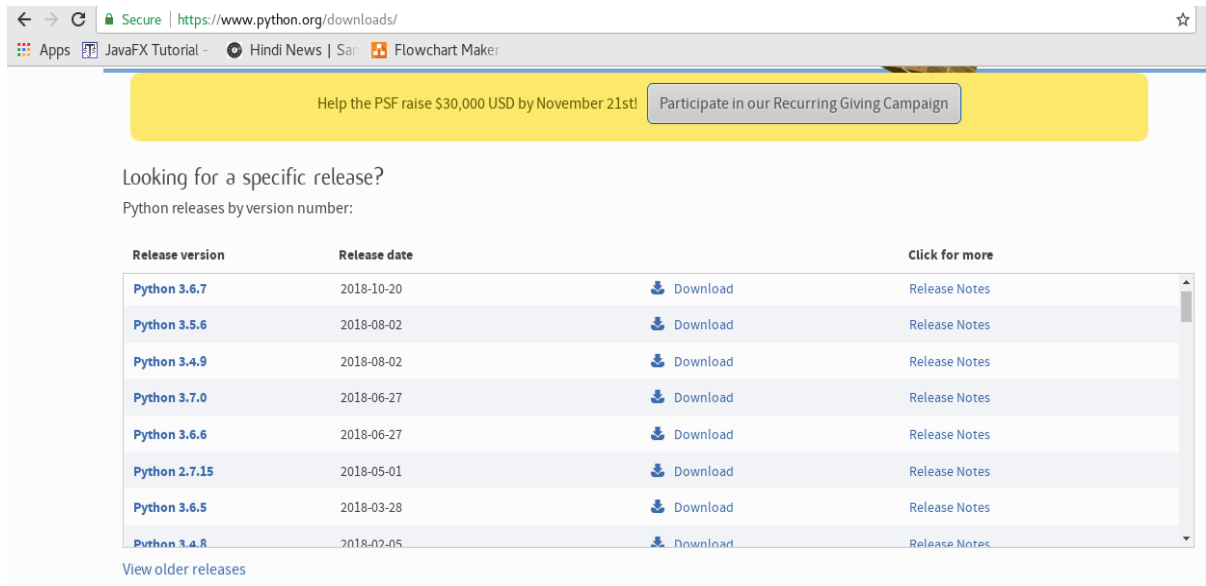
There are several such applications which can be developed using Python

### **How to Install Python (Environment Set-up)**

In this section of the tutorial, we will discuss the installation of python on various operating systems.

#### **Installation on Windows:**

Visit the link [\*\*https://www.python.org/downloads/\*\*](https://www.python.org/downloads/) to download the latest release of Python. In this process, we will install Python 3.6.7 on our Windows operating system.



Double-click the executable file which is downloaded; the following window will open. Select Customize installation and proceed.



Fig. 4.1 Python Installation

The following window shows all the optional features. All the features need to be installed and are checked by default; we need to click next to continue.



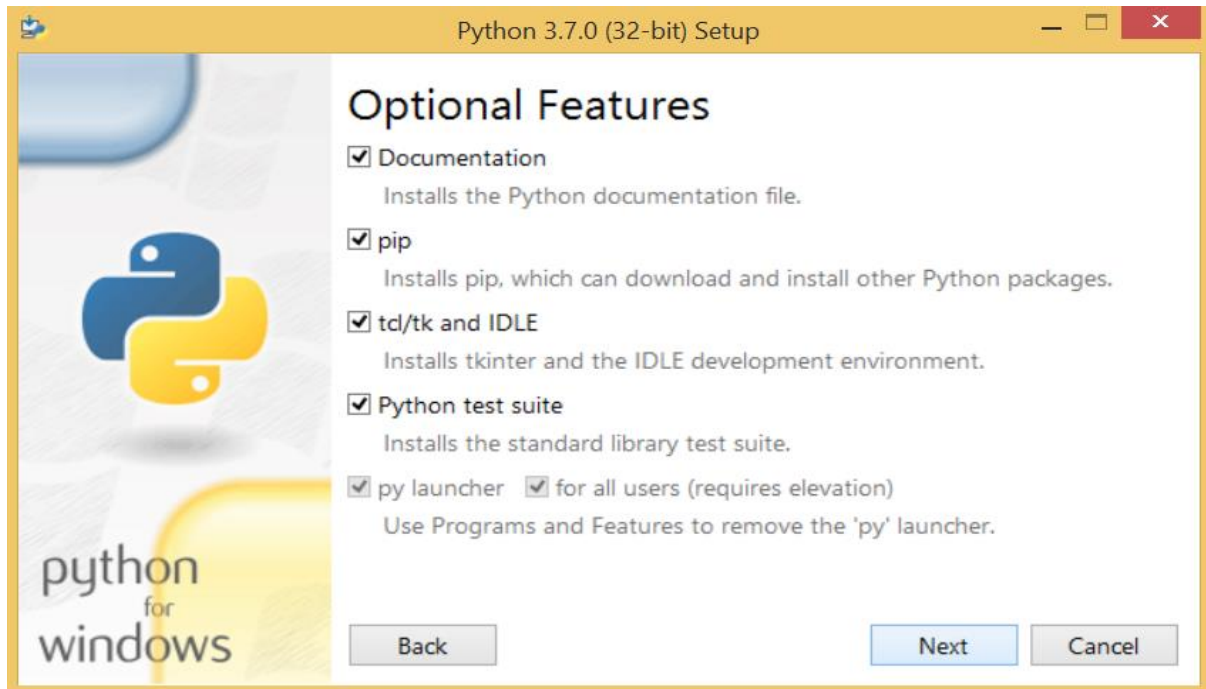
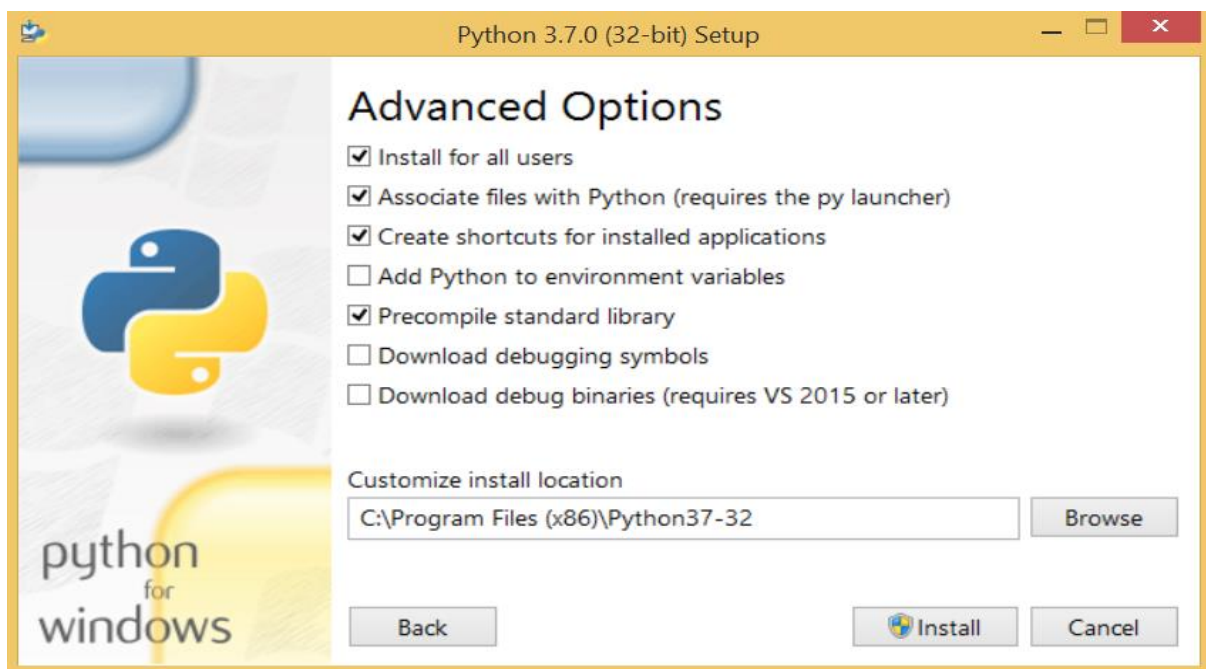


Fig. 4.2 Python Setup

The following window shows a list of advanced options. Check all the options which you want to install and click next. Here, we must notice that the first check-box (install for all users) must be checked.



Now, we are ready to install python-3.6.7. Let's install it.

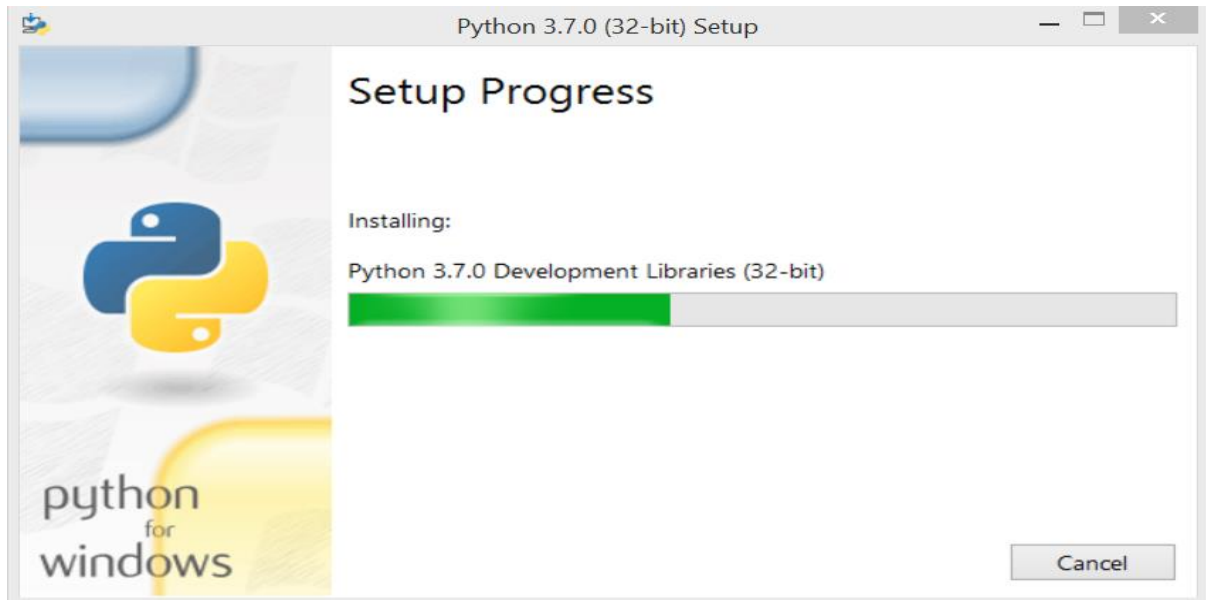
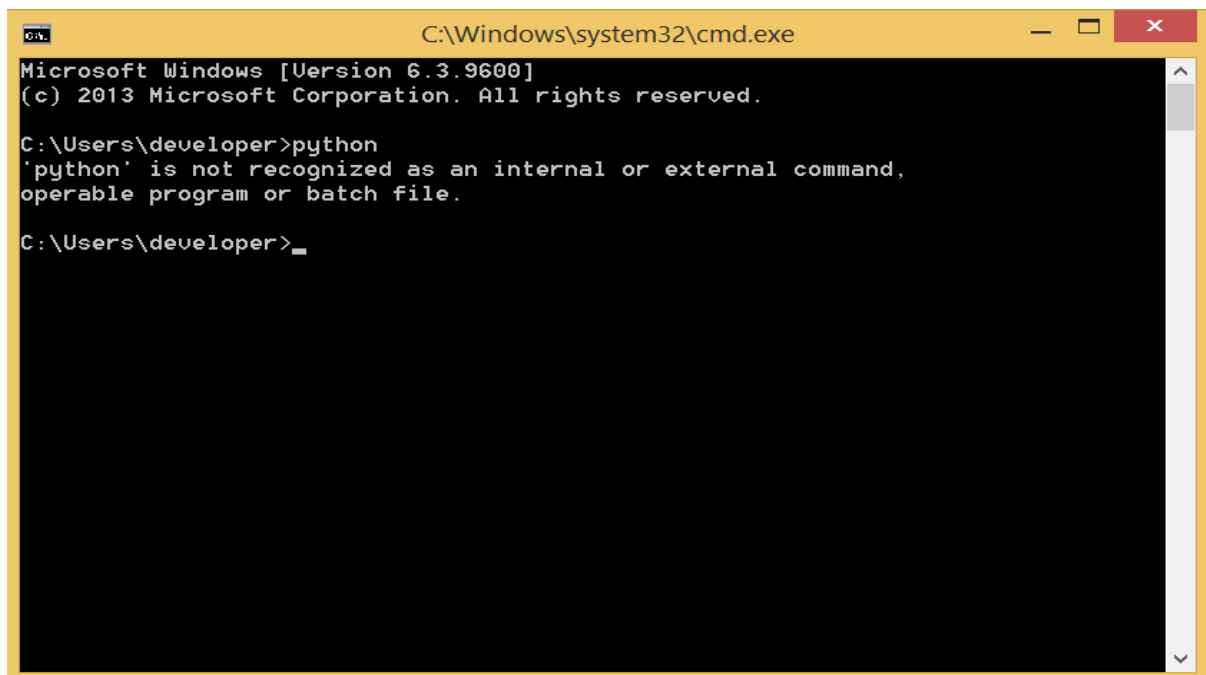
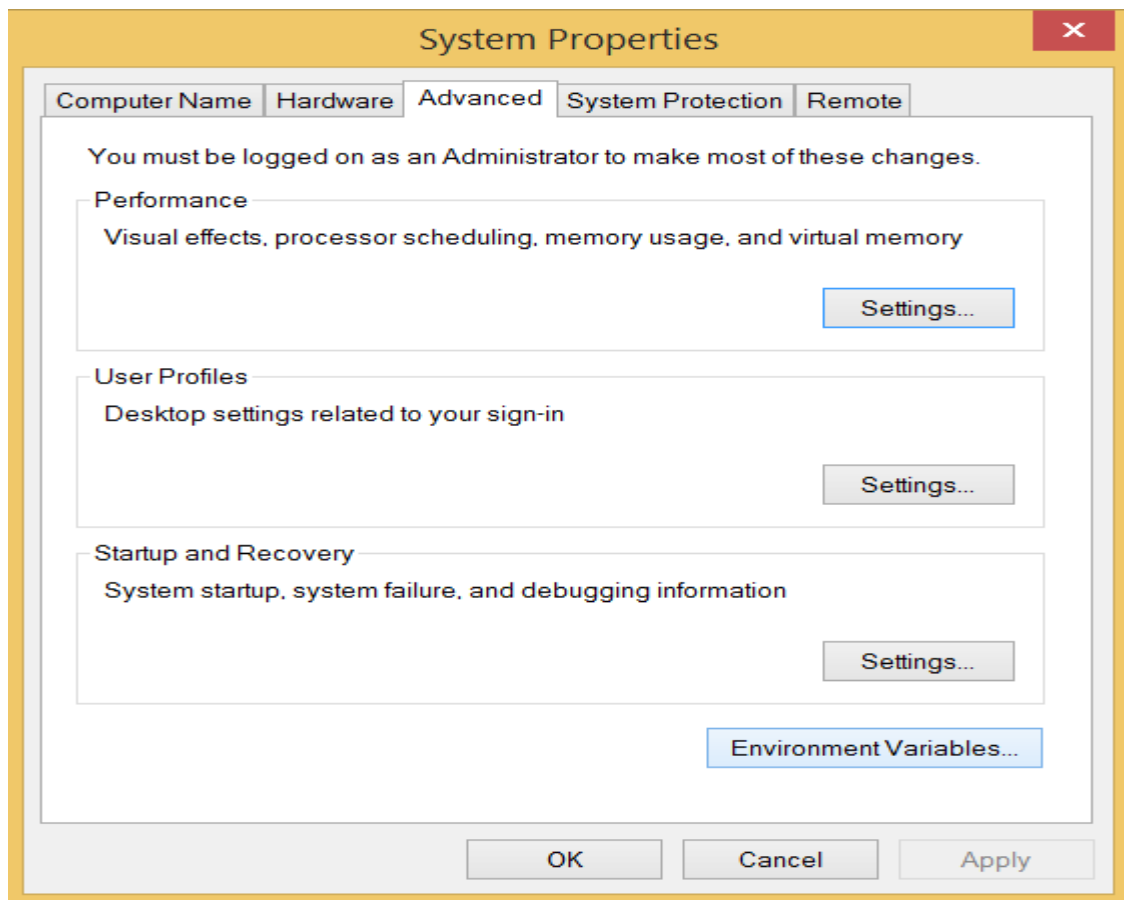
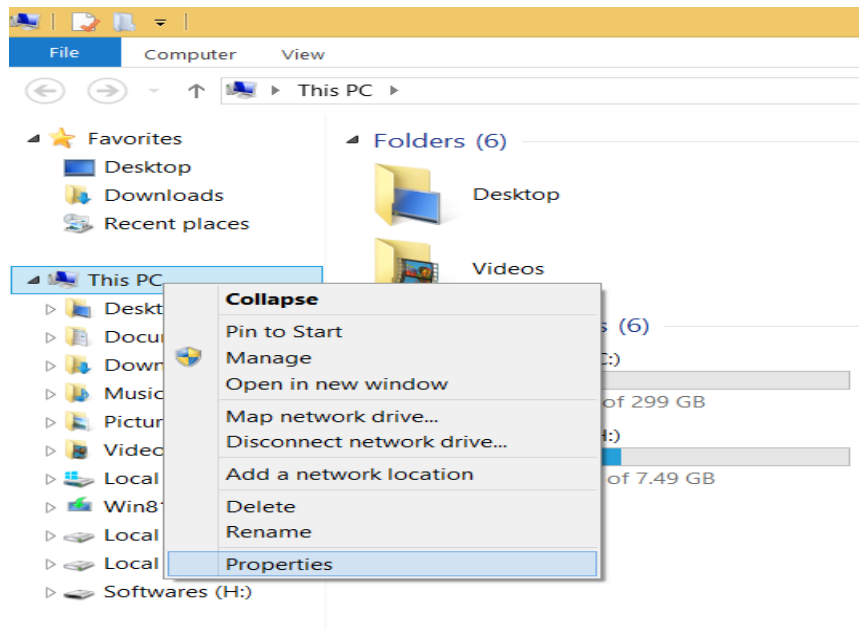


Fig. 4.3 Python Setup Progress

Now, try to run python on the command prompt. Type the command **python** in case of python2 or python3 in case of **python3**. It will show an error as given in the below image. It is because we haven't set the path.



To set the path of python, we need to right click on "my computer" and go to Properties  
→ Advanced → Environment Variables.



Add the new path variable in the user variable section.

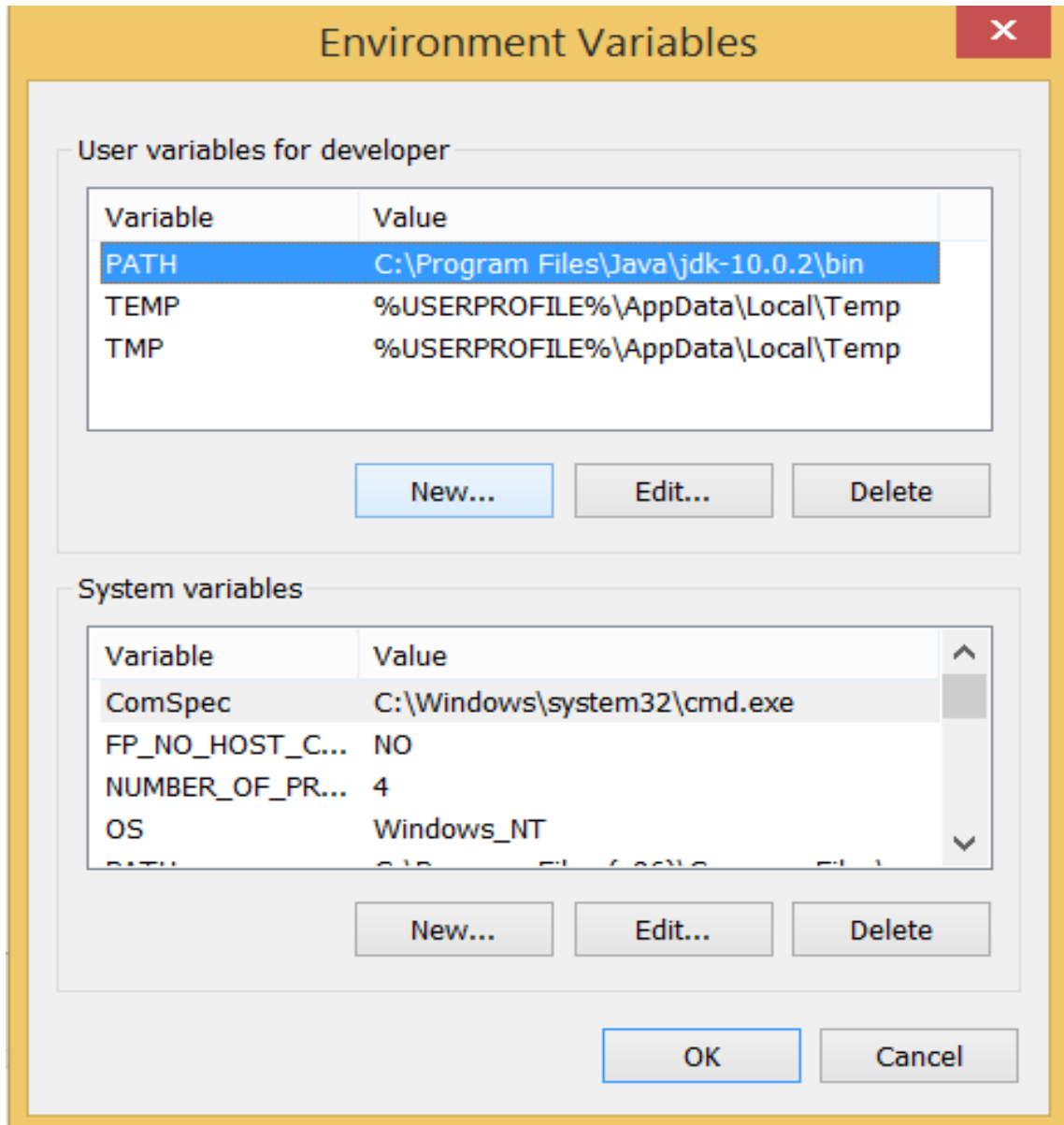
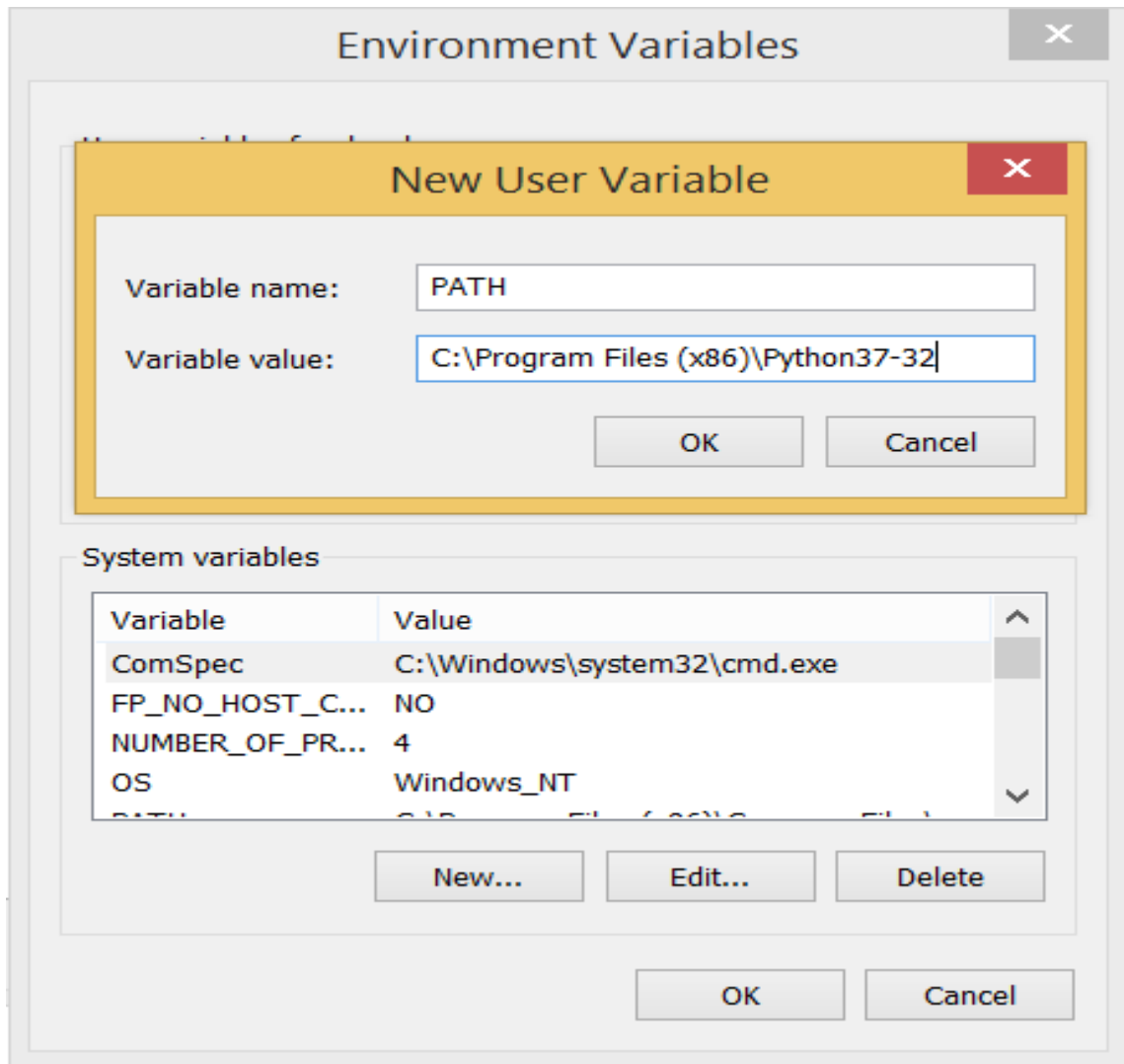


Fig. 4.4 Python Path Setup

Type **PATH** as the variable name and set the path to the installation directory of the python shown in the below image.



Now, the path is set, we are ready to run python on our local system. Restart CMD, and type **Python** again. It will open the python interpreter shell where we can execute the python statements.

### First Python Program

In this Section, we will discuss the basic syntax of python by using which, we will run a simple program to print hello world on the console.

Python provides us the two ways to run a program:

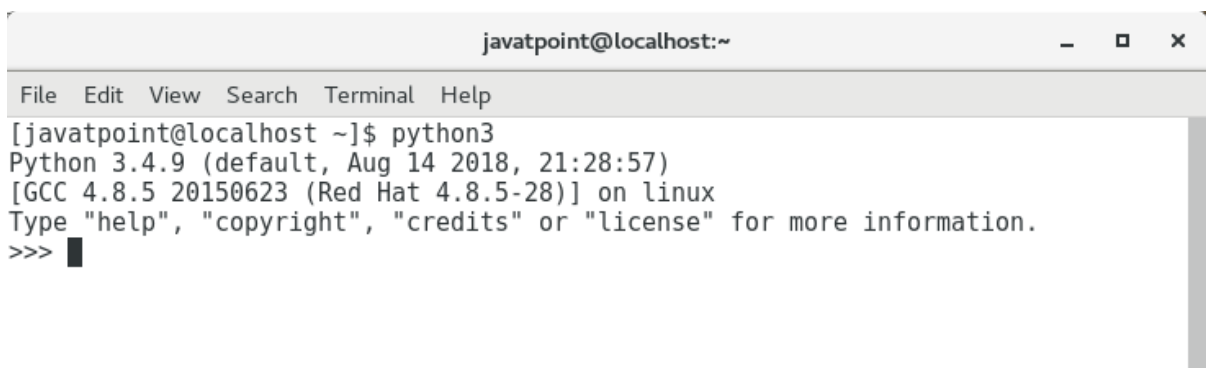
- Using Interactive interpreter prompt
- Using a script file

Let's discuss each one of them in detail.

### Interactive interpreter prompt

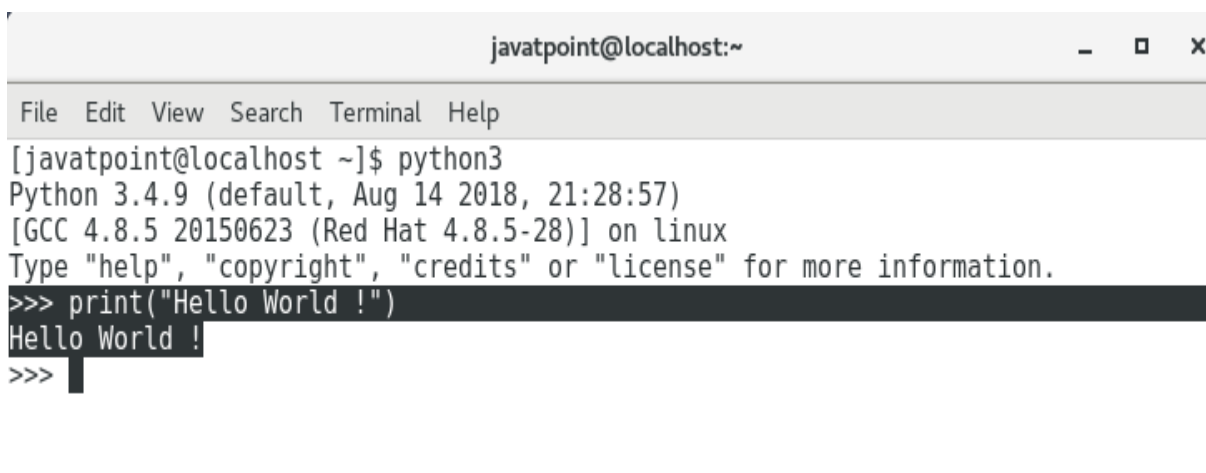
Python provides us the feature to execute the python statement one by one at the interactive prompt. It is preferable in the case where we are concerned about the output of each line of our python program. To open the interactive mode, open the terminal (or command prompt) and type python (python3 in case if you have python2 and python3 both installed on your system).

It will open the following prompt where we can execute the python statement and check their impact on the console.



```
javatpoint@localhost:~
File Edit View Search Terminal Help
[javatpoint@localhost ~]$ python3
Python 3.4.9 (default, Aug 14 2018, 21:28:57)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Let's run a python statement to print the traditional hello world on the console. Python3 provides print() function to print some message on the console. We can pass the message as a string into this function. Consider the following image.



```
javatpoint@localhost:~
File Edit View Search Terminal Help
[javatpoint@localhost ~]$ python3
Python 3.4.9 (default, Aug 14 2018, 21:28:57)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World !")
Hello World !
>>> █
```

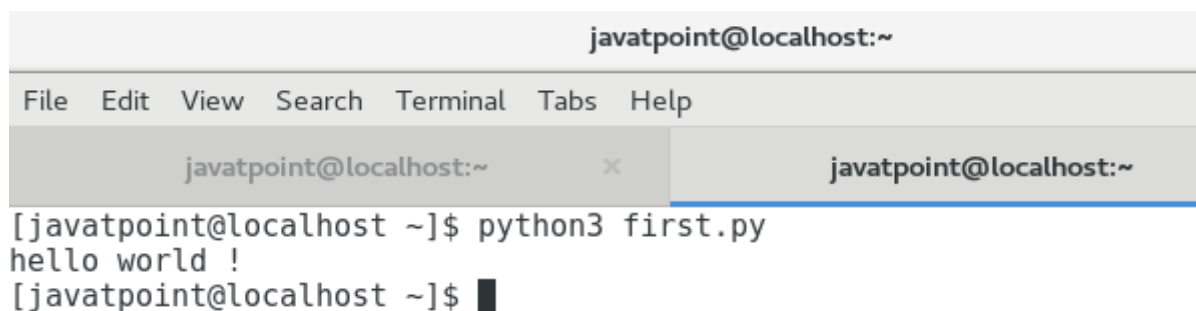
Here, we get the message **"Hello World !"** printed on the console.

### Using a script file

Interpreter prompt is good to run the individual statements of the code. However, we can not write the code every-time on the terminal. We need to write our code into a file which can be executed later. For this purpose, open an editor like notepad, create a file named first.py (python used .py extension) and write the following code in it.

- Print ("hello world"); #here, we have used print() function to print the message on the console. To run this file named as first.py, we need to run the following command on the terminal.

**\$ python3 first.py**



The screenshot shows a terminal window titled 'javatpoint@localhost:~'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', 'Tabs', and 'Help'. Below the menu bar, there are two tabs, both labeled 'javatpoint@localhost:~'. The terminal content shows the command '[javatpoint@localhost ~]\$ python3 first.py' being entered, followed by the output 'hello world !' and a new prompt '[javatpoint@localhost ~]\$' with a cursor.

Hence, we get our output as the message **Hello World !** is printed on the console.

## 4.2 Back End Implementation

### Machine Learning

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range.

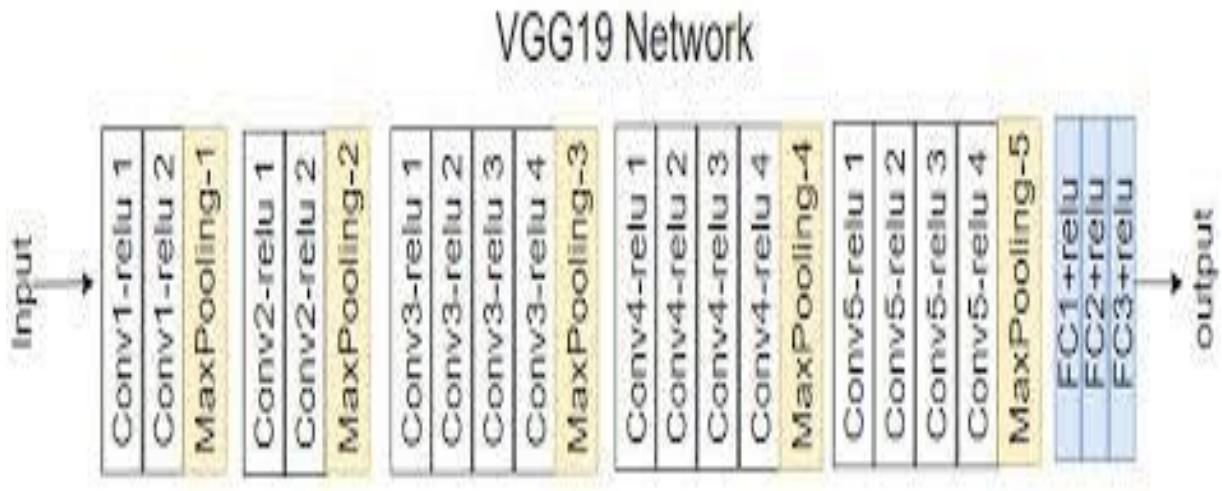
Machine learning is the scientific field dealing with the ways in which machines learn from experience. For many scientists, the term “machine learning” is identical to the term “artificial intelligence”, given that the possibility of learning is the main characteristic of an entity called intelligent in the broadest sense of the word. The purpose of machine learning is the construction of computer systems that can adapt and learn from their experience. There are two types

- **Supervised Learning** : In supervised learning, the system must “learn” inductively a function called target function, which is an expression of a model describing the data. The objective function is used to predict the value of a variable, called dependent variable or output variable, from a set of variables, called independent variables or input variables or characteristics or features. The set of possible input values of the function, i.e. its domain, are called instances. Each case is described by a set of characteristics (attributes or features). A subset of all cases, for which the output variable value is known, is called training data or examples. In order to infer the best target function, the learning system, given a training set, takes into consideration alternative functions, called hypothesis and denoted by  $h$ . In supervised learning, there are two kinds of learning tasks: classification and regression.
- **Unsupervised Learning** : In unsupervised learning, the system tries to discover the hidden structure of data or associations between variables. In that case, training data consists of instances without any corresponding labels. Association Rule Mining appeared much later than machine learning and is subject to greater influence from the research area of databases.

## VGG-19 MODEL

**VGG19** is a variant of VGG model which in short consists of 19 layers (16 convolution layers, 3 fully connected layer, 5 MaxPool layers and 1 SoftMax layer). In general VGG16 and VGG 19 is quite similar except the layers difference, actually VGG 19 gives more accuracy due to increase of some hidden layers compared with VGG 16. In this proposed application we applied VGG 19 on given dataset, so that we got accuracy of 78 percent on 50 epochs.





CNN(Convolutional Neural Network) is a powerful algorithm for image processing. These algorithms are currently the best algorithms we have for the automated processing of images. Many companies use these algorithms to do things like identifying the objects in an image. CNNs have broken the mold and ascended the throne to become the state-of-the-art computer vision technique. Among the different types of neural networks (others include recurrent neural networks (RNN), long short term memory (LSTM), artificial neural networks (ANN), etc.), CNNs are easily the most popular.

There are three types of layers in a Convolutional Neural Network:

1. Convolutional Layer: In a typical neural network each input neuron is connected to the next hidden layer. In CNN, only a small region of the input layer neurons connect to the neuron hidden layer.
2. Pooling Layer: The pooling layer is used to reduce the dimensionality of the feature map. There will be multiple activation & pooling layers inside the hidden layer of the CNN.
3. Fully-Connected layer: Fully Connected Layers form the last few layers in the network. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.

## 4.3 SOURCE CODE

```
# -*- coding: utf-8 -*-
```

```
"""flowers_recognition.ipynb
```

Automatically generated by Colaboratory.

Original file is located at

<https://colab.research.google.com/drive/1236AoFtB-Hkci3HRpX4KnIA1ljwbGpmF>

Directly download the dataset from kaggle using api

```
"""
```

```
from google.colab import files
```

```
files.upload()
```

```
! pip install -q kaggle
```

```
!mkdir -p ~/.kaggle
```

```
!cp kaggle.json ~/.kaggle/
```

```
! chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d alxmamaev/flowers-recognition
```

```
!ls -lrth
```

```
"""!unzipping the flower dataset"""
```

```
!unzip flowers-recognition.zip
```

```
!ls -lrth flowers/
```

```
# !mv flowers/flowers/ content/
```

```
# !ls -lrth content/flower
```

""The pictures are divided into five classes: chamomile, tulip, rose, sunflower, dandelion.

For each class there are about 800 photos. Photos are not high resolution, about 320x240 pixels. Photos are not reduced to a single size, they have different proportions!

""

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import cv2
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
from tqdm.notebook import tqdm
```

```
import os
```

```
import random
```

```
from sklearn.model_selection import train_test_split
```

```
import seaborn as sns
```

```
from sklearn.metrics import classification_report, confusion_matrix
```

```
# from
```

```
""Declaring local variables to be used in this process""
```

```
imgs_path='flowers/'
```

```
img_size=224
```

```
batch_size=32
```

```
""**Data preprocessing -1:-This way we can get better accuracy as we r converting images into array and preprocessing images and feeding to the neural networks as numerical data**""
```

```

labels = ['dandelion', 'daisy','tulip','sunflower','rose']

def get_data(data_dir):

    data = []

    for label in labels:

        path = os.path.join(data_dir, label)

        class_num = labels.index(label)

        for img in os.listdir(path):

            try:

                img_arr = cv2.imread(os.path.join(path, img), cv2.IMREAD_COLOR)

                resized_arr = cv2.resize(img_arr, (img_size, img_size)) # Reshaping images to
preferred size

                data.append([resized_arr, class_num])

            except Exception as e:

                print(e)

        return np.array(data)

data = get_data(imgs_path)

l = []

for i in data:

    l.append(labels[i[1]])

sns.set_style('dark')

sns.countplot(l)

```

""The dataset seems balanced as for each training label , enough training examples exist

Randomly Previewing first 10 images

""

```
fig,ax=plt.subplots(5,2)
```

```
fig.set_size_inches(15,15)
```

```
for i in range(5):
```

```
    for j in range (2):
```

```
        l=random.randint(0,len(data))
```

```
        ax[i,j].imshow(data[l][0])
```

```
        ax[i,j].set_title('Flower: '+labels[data[l][1]])
```

```
plt.tight_layout()
```

```
x = []
```

```
y = []
```

```
for feature, label in data:
```

```
    x.append(feature)
```

```
    y.append(label)
```

```
# Normalize the data
```

```
# from keras.applications.vgg16 import preprocess
```

```
from tensorflow.keras.applications.xception import Xception,preprocess_input
```

```
x = np.array(x)
```

```
x=preprocess_input(x)
```

```

x[0]

x.shape

from sklearn.preprocessing import LabelBinarizer

label_binarizer = LabelBinarizer()

y = label_binarizer.fit_transform(y)

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0,stratify=y)

print(f' X_Train shape is {x_train.shape}, y_train shape is {y_train.shape} ')

print(f' X_test shape is {x_test.shape}, y_test shape is {y_test.shape} ')

del x,y,data

from tensorflow.keras.applications.vgg19 import VGG19

from tensorflow.keras import Model

from tensorflow.keras.layers import MaxPool2D,Dense,Flatten,Dropout

from tensorflow import keras

from tensorflow.keras.models import Sequential

pre_trained_model= VGG19(input_shape=(224,224,3), include_top=False, weights="imagenet")

for layer in pre_trained_model.layers[:19]:

    layer.trainable = False

model=Sequential()

model.add(pre_trained_model)

model.add(MaxPool2D((2,2),strides=(2,2)))

model.add(Flatten())

```

```

model.add(Dense(5,activation='softmax'))

"""model = Sequential([

    pre_trained_model,

    MaxPool2D((2,2) , strides = 2),

    Flatten(),

    Dense(5 , activation='softmax')])"""

model.summary()

model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

from tensorflow.keras.callbacks import ReduceLROnPlateau,EarlyStopping

learning_rate=ReduceLROnPlateau(monitor='val_accuracy',

                                patience=3,

                                verbose=1,

                                factor=0.5,

                                min_lr=0.0001)

cb_early_stop=EarlyStopping(monitor = 'val_loss', patience = 3)

callbacks_list=[learning_rate,cb_early_stop]

history=model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=8,batch_size=64,callbacks=[callbacks_list])

print("Accuracy of the model is - " , model.evaluate(x_test,y_test)[1]*100 , "%")

plt.figure(figsize=[8,6])

plt.plot(history.history['accuracy'],'r',linewidth=2.0)

plt.plot(history.history['val_accuracy'],'b',linewidth=2.0)

```

```

plt.legend(['Training accuracy', 'Validation accuracy'],fontsize=18)

plt.xlabel('Epochs ',fontsize=16)

plt.ylabel('accuracy',fontsize=16)

plt.title('val accuracy',fontsize=16)

y_pred=model.predict(x_test)

y_pred_classes=np.argmax(y_pred,axis=1)

y_test_labels=np.argmax(y_test,axis=1)

class_names=os.listdir(imgs_path)

from sklearn.metrics import confusion_matrix,accuracy_score

# print(confusion_matrix(y_test_labels,y_pred_classes))

sns.heatmap(confusion_matrix(y_test_labels,y_pred_classes),yticklabels=class_names,xticklabels=class_names,annot=True,fmt='.1f') # calculating the confusion matrix

accuracy=accuracy_score(y_test_labels,y_pred_classes)*100 # calculating the accuracy

print(f'Accuracy : {accuracy}')
```

```

# now storing some properly as well as misclassified indexes'.

i=0

prop_class=[]

mis_class=[]

for i in range(len(y_test_labels)):

    if(y_test_labels[i] == y_pred_classes[i]):

        prop_class.append(i)

    if(len(prop_class)==8):
```



```

        break

i=0

for i in range(len(y_test_labels)):

    if(y_test_labels[i] != y_pred_classes[i]):

        mis_class.append(i)

    if(len(mis_class)==8):

        break

"""correctly classified images"""

y_test_labels

count=0

fig,ax=plt.subplots(4,2)

fig.set_size_inches(15,15)

for i in range (4):

    for j in range (2):

        ax[i,j].imshow(x_test[prop_class[count]])

        ax[i,j].set_title("Predicted Flower : "+ labels[y_pred_classes[prop_class[count]]]
+"\\n"+"Actual Flower : "+ labels[y_test_labels[prop_class[count]]])

    plt.tight_layout()

    count+=1

"""mis-classified images"""

count=0

fig,ax=plt.subplots(4,2)

```

```

fig.set_size_inches(15,15)

for i in range (4):

    for j in range (2):

        ax[i,j].imshow(x_test[mis_class[count]])

        ax[i,j].set_title("Predicted          Flower          :
"+labels[y_pred_classes[mis_class[count]]]+"\\n"+"Actual          Flower          :
"+labels[y_test_labels[mis_class[count]]])

        plt.tight_layout()

        count+=1

"""Now testing on external images"""

from google.colab import files

files.upload()

image_name='tulip.jpg' # enter image to test

x1=[]

img_1 = cv2.imread(image_name, cv2.IMREAD_COLOR) #

resized_arr_1 = cv2.resize(img_1, (img_size, img_size)) # Reshaping images to preferred size

plt.imshow(resized_arr_1)

x1.append([resized_arr_1])

x1=np.array(x1)

# plt.imshow(x)

# Normalize the data

x1 = preprocess_input(x1)

```

```

x1 = x1.reshape(-1, img_size, img_size, 3)

x1.shape

"""**getting predictions**"""

pred=model.predict(x1) # for predicting class

# print(pred)

# prob=model.predict_proba(x1) # predicting probability

labels_pred=np.argmax(pred,axis=0)

# labels=get_labels(labels_pred)

flowers = ["dandelion","daisy","tulip","sunflower","rose"]

pred_results=pd.DataFrame(data=pred,columns=flowers)

# print(pred_results.head())

import seaborn as sns

# sns.set_theme(style="darkgrid")

ax=sns.barplot(data=pred_results)

plt.show()

# pred_results.head()


```

## 4.4 OUTPUT SCREENS

### LOAD DATASET

Directly download the dataset from kaggle using api

```
from google.colab import files
files.upload()
```

 Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving kaggle.json to kaggle.json

```
{'kaggle.json': b'{"username": "b131258", "key": "5cc292bc58798bdf958636f8f9ed5bd"}'}
```

```
[ ] ! pip install -q kaggle
```

```
[ ] !mkdir -p ~/.kaggle
    !cp kaggle.json ~/.kaggle/
```

```
[ ] ! chmod 600 ~/.kaggle/kaggle.json
```

### UNZIP THE DATASET

!unzipping the flower dataset

```
!unzip flowers-recognition.zip
```

```
inflating: flowers/daisy/15029936576_8d6f96c72c_n.jpg
inflating: flowers/daisy/15100730728_a450c5f422_n.jpg
inflating: flowers/daisy/15207766_fc2f1d692c_n.jpg
inflating: flowers/daisy/15306268004_4680ba95e1.jpg
inflating: flowers/daisy/153210866_03cc9f2f36.jpg
inflating: flowers/daisy/15327813273_06cdf42210.jpg
inflating: flowers/daisy/154332674_453cea64f4.jpg
inflating: flowers/daisy/15760153042_a2a90e9da5_m.jpg
inflating: flowers/daisy/15760811380_4d686c892b_n.jpg
inflating: flowers/daisy/15784493690_b1858cdb2b_n.jpg
inflating: flowers/daisy/15813862117_dedcd1c56f_m.jpg
inflating: flowers/daisy/15853110333_229c439e7f.jpg
inflating: flowers/daisy/158869618_f1a6704236_n.jpg
inflating: flowers/daisy/16020253176_60f2a6a5ca_n.jpg
inflating: flowers/daisy/16025261368_911703a536_n.jpg
```

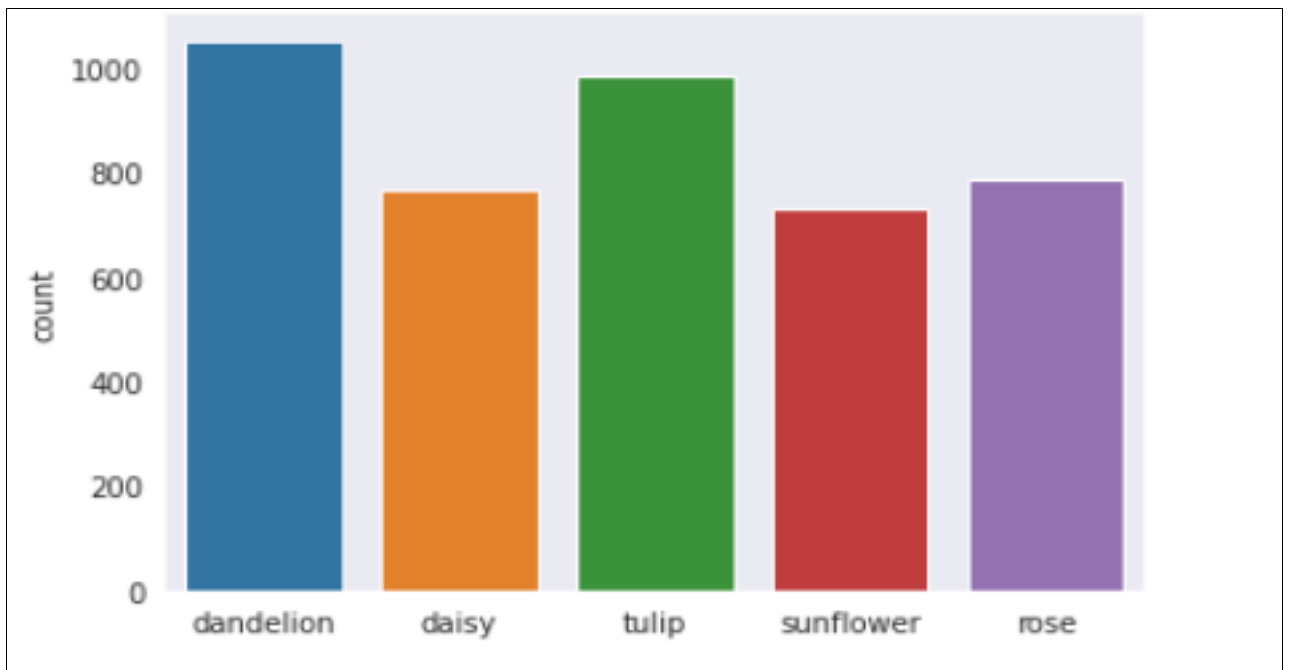
## DATA PRE-PROCESSING

Data preprocessing -1 this way we can get better accuracy as we r converting images into array and preprocessing images and feeding to the neural networks as numerical data

```
▶ labels = ['dandelion', 'daisy', 'tulip', 'sunflower', 'rose']

def get_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img), cv2.IMREAD_COLOR)
                resized_arr = cv2.resize(img_arr, (img_size, img_size)) # Reshaping images to preferred size
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)
```

## PLOT SAMPLE IMAGE



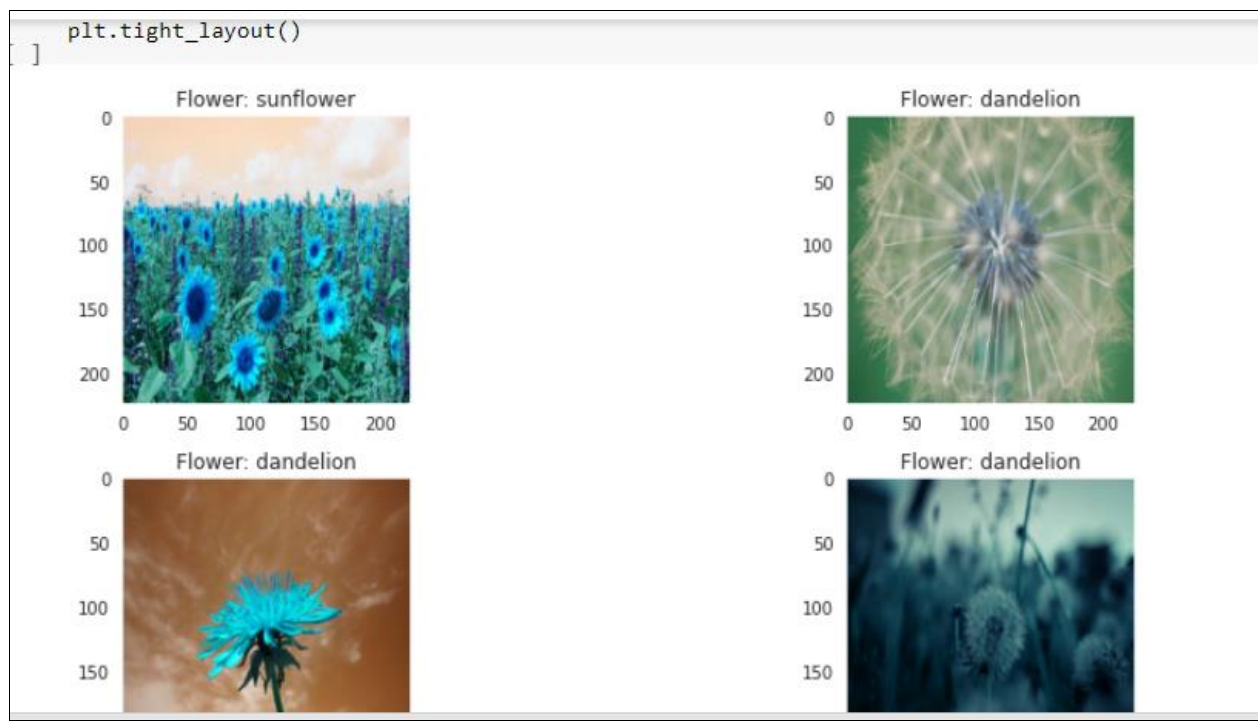
## APPLY CNN MODEL

```
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras import Model
from tensorflow.keras.layers import MaxPool2D,Dense,Flatten,Dropout
from tensorflow import keras
from tensorflow.keras.models import Sequential

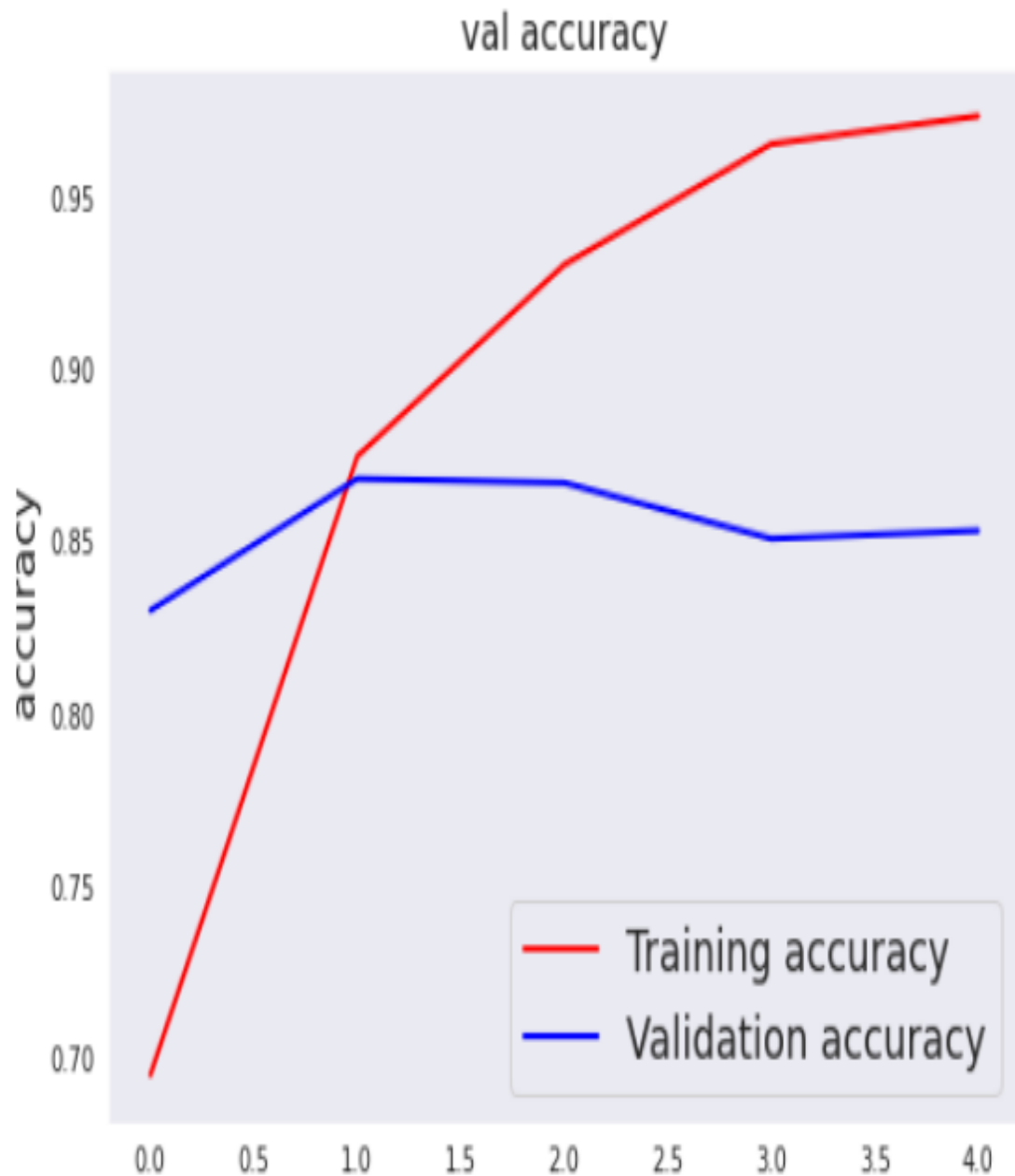
pre_trained_model = VGG19(input_shape=(224,224,3), include_top=False, weights="imagenet")

for layer in pre_trained_model.layers[:19]:
    layer.trainable = False

model=Sequential()
model.add(pre_trained_model)
model.add(MaxPool2D((2,2),strides=(2,2)))
model.add(Flatten())
model.add(Dense(5,activation='softmax'))
```

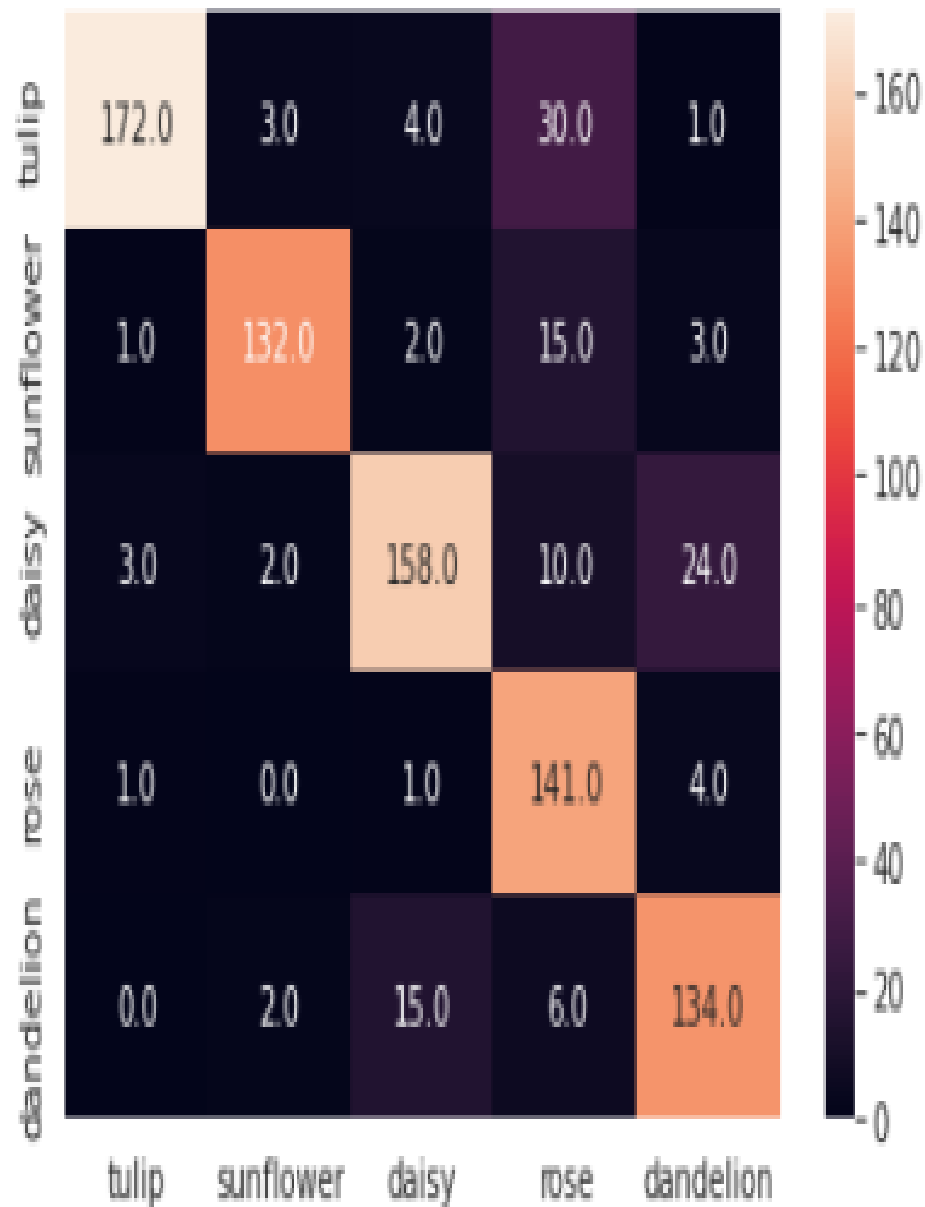


## TRAIN AND TEST VALIDATION



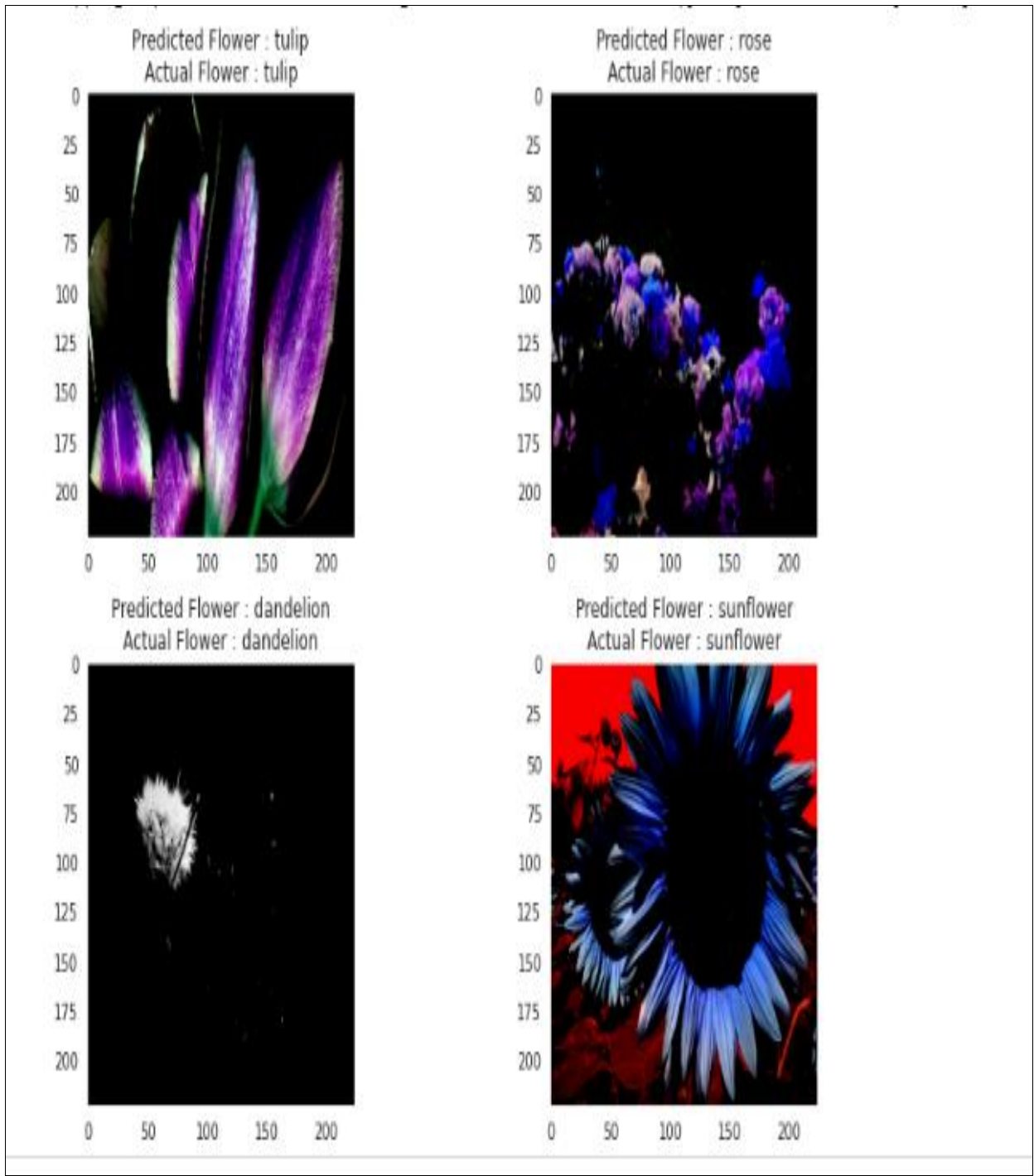


Accuracy : 85.30092592592592



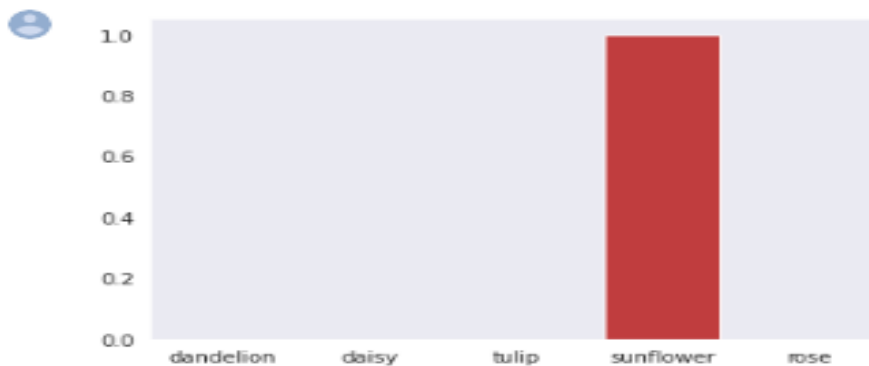


## PREDICT SPECIES NAME FROM SAMPLE INPUT IMAGE



## GETTING PREDICTIONS

```
▶ pred=model.predict(x1) # for predicting class
# print(pred)
# prob=model.predict_proba(x1) # predicting probability
labels_pred=np.argmax(pred,axis=0)
# labels=get_labels(labels_pred)
flowers = ["dandelion","daisy","tulip","sunflower","rose"]
pred_results=pd.DataFrame(data=pred,columns=flowers)
# print(pred_results.head())
import seaborn as sns
# sns.set_theme(style="darkgrid")
ax=sns.barplot(data=pred_results)
plt.show()
# pred_results.head()
```



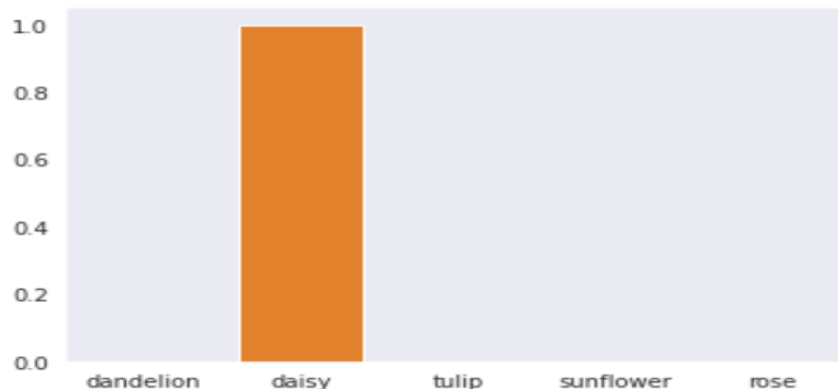
```
pred=model.predict(x1) # for predicting class
# print(pred)
# prob=model.predict_proba(x1) # predicting probability
labels_pred=np.argmax(pred,axis=0)
# labels=get_labels(labels_pred)
flowers = ["dandelion","daisy","tulip","sunflower","rose"]
pred_results=pd.DataFrame(data=pred,columns=flowers)
# print(pred_results.head())
import seaborn as sns
# sns.set_theme(style="darkgrid")
ax=sns.barplot(data=pred_results)
plt.show()
# pred_results.head()
```



```

pred=model.predict(x1) # for predicting class
# print(pred)
# prob=model.predict_proba(x1) # predicting probability
labels_pred=np.argmax(pred,axis=0)
# labels=get_labels(labels_pred)
flowers = ["dandelion","daisy","tulip","sunflower","rose"]
pred_results=pd.DataFrame(data=pred,columns=flowers)
# print(pred_results.head())
import seaborn as sns
# sns.set_theme(style="darkgrid")
ax=sns.barplot(data=pred_results)
plt.show()
# pred_results.head()

```



```

pred=model.predict(x1) # for predicting class
# print(pred)
# prob=model.predict_proba(x1) # predicting probability
labels_pred=np.argmax(pred,axis=0)
# labels=get_labels(labels_pred)
flowers = ["dandelion","daisy","tulip","sunflower","rose"]
pred_results=pd.DataFrame(data=pred,columns=flowers)
# print(pred_results.head())
import seaborn as sns
# sns.set_theme(style="darkgrid")
ax=sns.barplot(data=pred_results)
plt.show()
# pred_results.head()

```



## OUTPUT:

```
image_name='sunflower.jpg' # enter image to test
x1=[]
img_1 = cv2.imread(image_name, cv2.IMREAD_COLOR) #
resized_arr_1 = cv2.resize(img_1, (img_size, img_size)) # Reshaping images to preferred size
plt.imshow(resized_arr_1)
x1.append([resized_arr_1])
x1=np.array(x1)
# plt.imshow(x)
# Normalize the data
x1 = preprocess_input(x1)
x1 = x1.reshape(-1, img_size, img_size, 3)
x1.shape
```

(1, 224, 224, 3)



```
image_name='rose.jpg' # enter image to test
x1=[]
img_1 = cv2.imread(image_name, cv2.IMREAD_COLOR) #
resized_arr_1 = cv2.resize(img_1, (img_size, img_size)) # Reshaping images to preferred size
plt.imshow(resized_arr_1)
x1.append([resized_arr_1])
x1=np.array(x1)
# plt.imshow(x)
# Normalize the data
x1 = preprocess_input(x1)
x1 = x1.reshape(-1, img_size, img_size, 3)
x1.shape
```

(1, 224, 224, 3)



```

image_name='dandi.jpg' # enter image to test
x1=[]
img_1 = cv2.imread(image_name, cv2.IMREAD_COLOR) #
resized_arr_1 = cv2.resize(img_1, (img_size, img_size)) # Reshaping images to preferred size
plt.imshow(resized_arr_1)
x1.append([resized_arr_1])
x1=np.array(x1)
# plt.imshow(x)
# Normalize the data
x1 = preprocess_input(x1)
x1 = x1.reshape(-1, img_size, img_size, 3)
x1.shape

```

(1, 224, 224, 3)

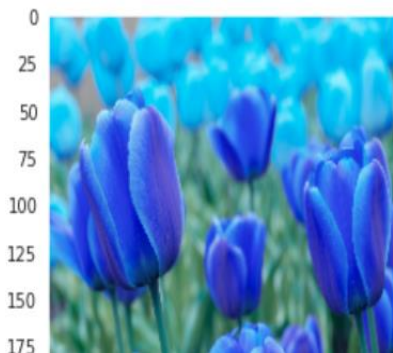


```

image_name='tulip.jpg' # enter image to test
x1=[]
img_1 = cv2.imread(image_name, cv2.IMREAD_COLOR) #
resized_arr_1 = cv2.resize(img_1, (img_size, img_size)) # Reshaping images to preferred size
plt.imshow(resized_arr_1)
x1.append([resized_arr_1])
x1=np.array(x1)
# plt.imshow(x)
# Normalize the data
x1 = preprocess_input(x1)
x1 = x1.reshape(-1, img_size, img_size, 3)
x1.shape

```

(1, 224, 224, 3)



```

image_name='daisy.jpg' # enter image to test
x1=[]
img_1 = cv2.imread(image_name, cv2.IMREAD_COLOR) #
resized_arr_1 = cv2.resize(img_1, (img_size, img_size)) # Reshaping images to preferred size
plt.imshow(resized_arr_1)
x1.append([resized_arr_1])
x1=np.array(x1)
# plt.imshow(x)
# Normalize the data
x1 = preprocess_input(x1)
x1 = x1.reshape(-1, img_size, img_size, 3)
x1.shape

```

(1, 224, 224, 3)





## **5. SYSTEM TESTING**

### **5.1 TESTING CONCEPTS**

The purpose of 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.

### **5.2 TESTING STRATEGIES**

#### **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 satisfactory, as shown by successful 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.



## 5.3 TEST CASES

### Negative Test Cases

<b>Test Case 1:</b> Dataset Selection		Priority(H.L):High
<b>Test Objective:</b> to check Dataset Selection success or fail		
<b>Test Description:</b> In this HOME screen, when the user selects the flower-recognition dataset as input		
<b>Requirement Verified:</b> Yes		
<b>Test Environment:</b> System connected with the dataset.		
Actions		Expected Results
<ul style="list-style-type: none"> <li>when the user selects the dataset by clicking upload button.</li> </ul>		<ul style="list-style-type: none"> <li>Uploading dataset fail</li> <li>Select valid Dataset</li> </ul>
<b>Pass:</b> no <b>Condition Pass:</b> No <b>Fail:</b> Yes		
<b>Problems/Issues:</b> Yes		
<b>Notes:</b> Selection is fail		

<b>Test Case 2:</b> Execute VGG-19 Algorithms		Priority(H.L):High
<b>Test Objective:</b> has to check whether algorithms are working		
<b>Test Description:</b> In this home page user dataset and after that he will check VGG-19 algorithm which can find the flower species		
<b>Requirement Verified:</b> No		
<b>Test Environment:</b> System connected with the dataset.		
Actions		Expected Results
<ul style="list-style-type: none"> <li>Run VGG-19 by clicking Run button</li> </ul>		<ul style="list-style-type: none"> <li>Comparing fail</li> <li>Select valid dataset and algorithms</li> </ul>
<b>Pass:</b> no <b>Condition Pass:</b> No <b>Fail:</b> Yes		
<b>Problems/Issues:</b> Yes		
<b>Notes:</b> Algorithm fail		

## Positive Test Cases

<b>Test Case 1:</b> Dataset Selection		Priority(H.L):High
<b>Test Objective:</b> to check dataset Selection success or fail		
<b>22252Test Description:</b> In this HOME screen, when the user selects the input dataset it display path label		
<b>Requirement Verified:</b> Yes		
<b>Test Environment:</b> System connected with the dataset.		
Actions		Expected Results
<ul style="list-style-type: none"> <li>when the user selects the input by clicking upload button.</li> </ul>		<ul style="list-style-type: none"> <li>Dataset path Label can be shown</li> </ul>
<b>Pass:</b> yes <b>Condition Pass:</b> No <b>Fail:</b> No		
<b>Problems/Issues:</b> No		
<b>Notes:</b> Dataset Selection successfully completed		

<b>Test Case 2:</b> Running VGG-19 Algorithm		Priority(H.L):High
<b>Test Objective:</b> has to check ML algorithms related to trained Dataset		
<b>22252Test Description:</b> In this home page user will choose VGG-19 for flower-classification		
<b>Requirement Verified:</b> No		
<b>Test Environment:</b> System connected with the dataset.		
Actions		Expected Results
<ul style="list-style-type: none"> <li>Run VGG-19 algorithm by clicking Run button</li> </ul>		<ul style="list-style-type: none"> <li>Display flower name and its species name</li> </ul>
<b>Pass:</b> yes <b>Condition Pass:</b> No <b>Fail:</b> Yes		
<b>Problems/Issues:</b> No		
<b>Notes:</b> CNN Model is successful.		

## 6. CONCLUSION

The proposed work is a faster way to train a Convolutional Neural Network (CNN) with a smaller dataset and limited computational resource such as CPU. As there are millions of flower species around the world, this system could easily be adapted by training more number of flower species images to recognize different species around the world. Thus, the future work would be to construct a larger database with not only flower images, but also with leaves, fruits, bark etc., collected from different sources around different parts of the world. This system would also be useful to identify plants for medicinal purposes such as in case of first aid. The user can quickly take an image of the plant species and get information about it to decide whether or not it can be used for first aid. The crucial part in building such a system is the training dataset which needs to be prepared either by manually taking pictures of the plants around the city or by using public datasets.

## 7. REFERENCES

- [1] Saitoh, T.; Kaneko, T., “Automatic recognition of wild Flowers”, Pattern Recognition, Proceedings. 15th International Conference on, vol.2, no., pp.507-510 vol.2, 2000.
- [2] Kumar, N., Belhumeur, P.N., Biswas, A., Jacobs, D.W., Kress, W.J., Lopez, I.C., Soares, J.V.B. “Leafsnap: A computer vision system for automatic plant species identification”, European Conference on Computer Vision. pp. 502-516, 2012.
- [3] D. Barthelemy. “The pl@ntnet project: A computational plant identification and collaborative information system”, Technical report, XIII World Forestry Congress, 2009.
- [4] G. Cerutti, V. Antoine, L. Tougne, J. Mille, L. Valet, D. Coquin, and A. Vacavant, “Reves participation-tree species classification using random forests and botanical features”, in Conference and Labs of the Evaluation Forum, 2012.
- [5] Y. Nam, E. Hwang, and D. Kim, “Clover: A mobile content-based leaf image retrieval system”, In Digital Libraries: Implementing Strategies and Sharing Experiences, Lecture Notes in Computer Science, pages 139-148, 2005.
- [6] J.-X. Du, X.-F. Wang and G.-J. Zhang, “Leaf shape based plant species recognition”, Applied Mathematics and Computation, vol. 185, 2007.
- [7] A. H. Kulkarni, H. M. Rai, K. A. Jahagirdar and P. S. Upparamani, “A Leaf Recognition Technique for Plant Classification Using RBPNN and Zernike Moments”, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, Issue 1, pp. 984-988, 2013.
- [8] Nilsback and Andrew Zisserman, “A Visual Vocabulary for Flower Classification”, Computer Vision and Pattern Recognition, IEEE Computer Society Conference on. Vol.2, 2006.

- [9] Nilsback, M.E., Zisserman, A, "Automated flower classification over a large number of classes", Indian Conference on Computer Vision, Graphics and Image Processing. pp. 722-729, 2008.
- [10] Siraj, Fadzilah, Muhammad Ashraq Salahuddin, and Shahrul Azmi Mohd Yusof, "Digital Image Classification for Malaysian Blooming Flower", Computational Intelligence, Modelling and Simulation (CIMSIM), IEEE, 2010.
- [11] Robert M. Haralick, K. Shanmugam, Its'hak Dinstein, "Textural Features for Image Classification", IEEE Transactions on Systems, Man and Cybernetics, Vol.SMC-3, No. 6, November 1973, pp.610-621, 1973.
- [12] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks", ICLR, 2014.
- [13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.
- [14] François Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions", arXiv:1610.02357 [cs.CV], 2016.