

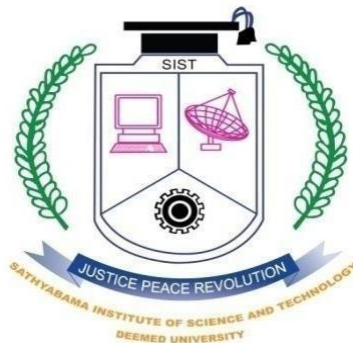
A RELIABLE & FAST AUTOMATIC COMBINATION OF DEEP FEATURES AND SPECIES CATEGORIZATION USING UNIFIED ENSEMBLE LAYER

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in
Computer Science and Engineering

By

PARTH KUMAR PATEL (REG.NO 37110546)

P.VENKATA NITISH (REG.NO 37110830)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119**

MARCH 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC | 12B Status
by UGC | Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING **BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **PARTH KUMAR PATEL (REG.NO 37110546)**, **P.VENKATA NITISH (REG.NO 37110830)** who carried out the project entitled “**A RELIABLE & FAST AUTOMATIC COMBINATION OF DEEP FEATURES AND SPECIES CATEGORIZATION USING UNIFIED ENSEMBLE LAYER**” under my supervision from NOVEMBER 2020 to MARCH 2021.

Internal Guide Name

Dr. G. MATHIVANAN, M.E., Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **PARTH KUMAR PATEL (REG.NO 37110546), P.VENKATA NITISH (REG.NO 37110830)** hereby declare that the Project Report entitled “**A RELIABLE & FAST AUTOMATIC COMBINATION OF DEEP FEATURES AND SPECIES CATEGORIZATION USING UNIFIED ENSEMBLE LAYER**” done by me under the guidance of **Dr. G. MATHIVANAN, M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneshwari M.E., Ph.D., and Dr. L. Lakshmanan M.E., Ph.D., Heads** of the Department of Computer Science & Engineering for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Dr. G. MATHIVANAN, M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

we wish to express our thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project

ABSTRACT

Plant species detection aims at the automatic identification of plants. Although a lot of aspects like leaf, flowers, fruits, seeds could contribute to the decision, but leaf features are the most significant. As a plant leaf is always more accessible as compared to other parts of the plants, it is obvious to study it for plant identification. The present paper introduced a novel plant species classifier based on the extraction of morphological features .First, a two-layer plant taxonomy is constructed to organize large numbers of plant species and their genus hierarchically in a coarse-to-fine fashion. Second, a deep learning framework is developed to enable path-based tree classifier training, where a tree classifier over the plant taxonomy is used to replace the flat softmax layer in traditional deep CNNs. A path-based error function is defined to optimize the joint process for learning deep CNN and tree classifier, where back propagation is used to update both the classifier parameters and the network weights simultaneously.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	V
	LIST OF FIGURES	VIII
1	INTRODUCTION	1
	1.1 INTRODUCTION	1
	1.2 OVERVIEW	1
	1.3 SCOPE OF THE PROJECT	1
	1.4 DOMAIN OVERVIEW	2
	1.4.1 DEEP LEARNING	2
2	LITERATURE SURVEY	5
	2.1 LITERATURE REVIEW	5
3	AIM AND SCOPE OF THE PRESENT INVESTIGATION	8
	3.1 AIM	8
	3.2 SCOPE	8
4	EXPERIMENT MATERIALS AND METHODS	10
	4.1 SYSTEM ANALYSIS	10
	4.1.1 EXISTING SYSTEM	10
	4.1.2 DRAWBACKS OF EXISTING SYSTEM	10
	4.1.3 PROPOSED SYSTEM	10
	4.1.4 ADVANTAGES OF PROPOSED SYSTEM	11
	4.2 SYSTEM REQUIEMENTS	12
	4.2.1 HARDWARE SPECIFICATION	12
	4.2.2 SOFTWARE SPECIFICATION	12
	4.3 SOFTWARE DESCRIPTION	12
	4.3.1 PYTHON	12
	4.3.1.1 PYTHON LINE STRUCTURE	14
	4.3.1.2 PYTHON DATA TYPES	16
	4.3.2 PANDAS	17

	4.3.3 NUMPY	20
	4.3.4 MATHPLOTLIN	22
	4.3.5 ABOUT IDE	23
	4.4 ARCHITECTURE DIAGRAM	27
	4.5 SYSTEM MODULES	27
	4.5.1 EXPLORATORY DATA EVALUATION	27
	4.5.2 PRE-PROCESSING	28
	4.5.3 MODEL CREATION	28
	4.5.4 PREDICTION	29
5	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	30
	5.1 RESULT	30
6	SUMMARY AND CONCLUSION	33
	6.1 SUMMARY	33
	6.2 CONCLUSION	33
	REFERENCES	34
	APPENDIX	36
	A. SCREENSHOT	38
	B. SOURCE CODE	38

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
1.4.1.1	PROCESS OF SOLVING PROBLEM USING DEEP LEARNING	2
1.4.1.2	NEURAL NETWORK	3
4.3.1.1	PYTHON SUPPORT	13
4.3.1.2	PYTHON LIBRARIES	13
4.4	ARCHITECTURE DIAGRAM	27
5.1	ANACONDA PROMPT	30
5.2	TRANING AND VALIDATION ACCURACY	31
5.3	TRANING AND VALIDATION LOSS	31
5.4	FINAL OUTPUT	32

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In this system, we present multi-path multi deep convolutional network for the identification of plant species which feeds different versions of plant images, thus resultant model has better image presentation than traditional Ensemble Learning. Comprehensive experimental evaluation on benchmark plant datasets showed that without using any pre-trained models, our proposed shallow network demonstrate very competitive performance for plant species identification.

1.2 OVERVIEW

Shape is the most common feature that have been used to develop plant identification systems. The features that are commonly used to evaluate shape are, ratio between slimness, roundness, compactness, rectangularity, and other aspects. Texture is one of the important features of the plant identification system, which can be used to characterize the leaves based on the surface structure of the leaves. It is a non-consistent spatial distribution pattern of different image intensities.

1.3 SCOPE OF THE PROJECT

The main contributions of this project are:

1. Data Analysis
2. Dataset Preprocessing
3. Training the Model
4. Testing of Dataset

1.4 DOMAIN OVERVIEW

1.4.1 Deep Learning

Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks.

Deep learning algorithms are constructed with connected layers.

- The first layer is called the Input Layer
- The last layer is called the Output Layer
- All layers in between are called Hidden Layers. The word deep means the network join neurons in more than two layers.

A deep neural network provides state-of-the-art accuracy in many tasks, from object detection to speech recognition. They can learn automatically, without predefined knowledge explicitly coded by the programmers

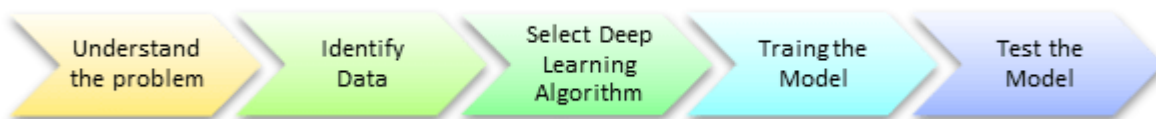


Fig.1.4.1.1 PROCESS OF SOLVING PROBLEM USING DEEP LEARNING

To grasp the idea of deep learning, imagine a family, with an infant and parents. The toddler points objects with his little finger and always says the word 'cat.' As its parents are concerned about his education, they keep telling him 'Yes, that is a cat' or 'No, that is not a cat.' The infant persists in pointing objects but becomes more accurate with 'cats.' The little kid, deep down, does not know why he can say it is a cat or not. He has just learned how to hierarchies complex features coming up with a cat by looking at the pet overall and continue to focus on details such as the tails or the nose before to make up his mind.

A neural network works quite the same. Each layer represents a deeper level of knowledge, i.e., the hierarchy of knowledge. A neural network with four layers will learn more complex feature than with that with two layers.

The learning occurs in two phases.

1. The first phase consists of applying a nonlinear transformation of the input and create a statistical model as output.
2. The second phase aims at improving the model with a mathematical method known as derivative.

The neural network repeats these two phases hundreds to thousands of time until it has reached a tolerable level of accuracy. The repeat of this two-phase is called an iteration

Classification of Neural Networks

1. Shallow neural network: The Shallow neural network has only one hidden layer between the input and output.
2. Deep neural network: Deep neural networks have more than one layer. For instance, Google LeNet model for image recognition counts 22 layers.

The computational models in Deep Learning are loosely inspired by the human brain. The multiple layers of training are called Artificial Neural Networks (ANN).

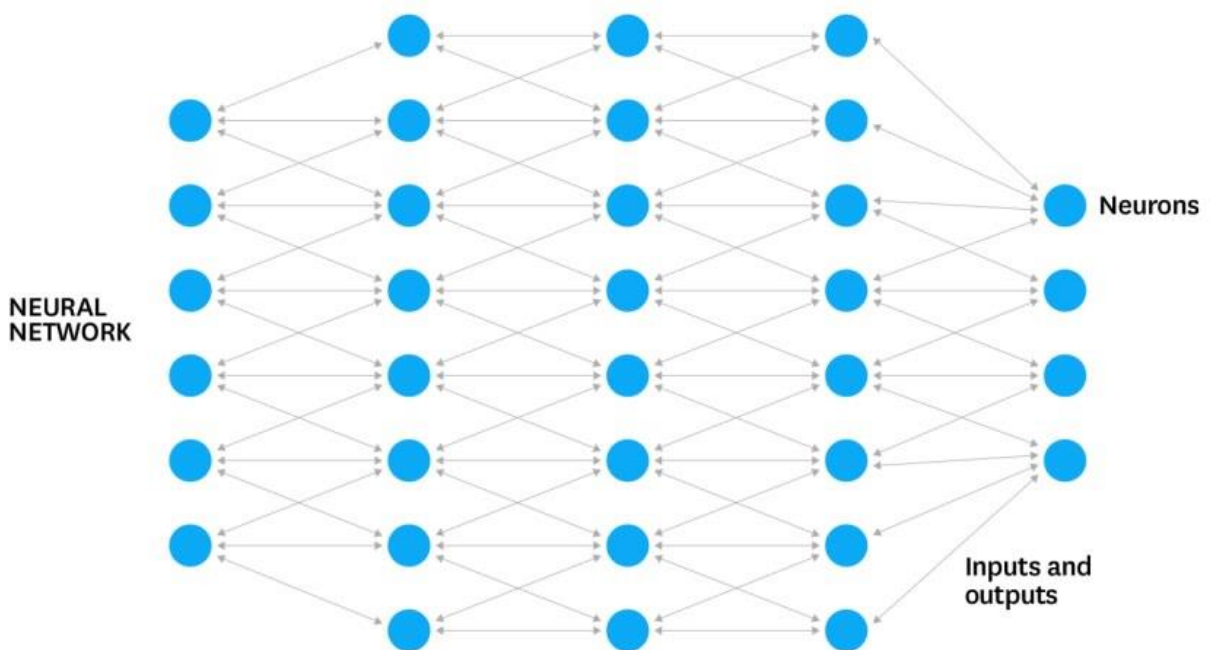


Fig.1.4.1.2 NEURAL NETWORK

Neuron: Artificial Neural Networks contain layers of neurons. A neuron is a computational unit that calculates a piece of information based on weighted input parameters. Inputs accepted by the neuron are separately weighted.

Inputs are summed and passed through a non-linear function to produce output. Each layer of neurons detects some additional information, such as edges of things in a picture or tumors in a human body. Multiple layers of neurons can be used to detect additional information about input parameters.

Nodes: Artificial Neural Network is an interconnected group of nodes akin to the vast network of layers of neurons in a brain. Each circular node represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another.

Inputs: Inputs are passed into the first layer. Individual neurons receive the inputs, with each of them receiving a specific value. After this, an output is produced based on these values.

Outputs: The outputs from the first layer are then passed into the second layer to be processed. This continues until the final output is produced. The assumption is that the correct output is predefined.

Each time data is passed through the network, the end result is compared with the correct one, and tweaks are made to their values until the network creates the correct final output each time.

Some of the commonly used neural networks are as follows:

1. Artificial Neural Network (ANN)
2. Convolutional Neural Network (CNN)
3. Recurrent Neural Network (RNN)
4. Deep Neural Network (DNN)
5. Deep Belief Network (DBN)

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE REVIEW

REVIEW NO 1:

AUTHOR : Fadzilah Siraj, Muhammad Ashraq Salahuddin and Shahrul Azmi Mohd Yusof

They proposed the system for classification of Malaysian blooming flower. In this paper they presents the application of NN and on image processing particularly for understanding flower image features. For predictive analysis, they have used two techniques namely, Neural Network (NN) and Logistic regression. The study shows that NN obtains the higher percentage of accuracy among two techniques. The Otsu's method was applied in order to compute a global threshold. The image is then converted to RGB color space again. In color extraction, the images were transformed from RGB color space to HSV color space the image texture is calculated based on gray-level co-occurrence matrix (GLCM) to obtain the contrast, correlation, energy and homogeneity of the image. The prediction accuracy of logistic regression is 26.8%. Therefore based on 1800 samples of Malaysian flower images, NN has shown a higher average prediction results vs. logistic regression. However this paper cannot present recognition of flower type, its only recognize flower features so in future studies can be focused on developed flower model system which can recognize Malaysian blooming flower or extending the dataset built and Verities sample of images can be captured for different flowers and recognize their types.

REVIEW NO 2:

AUTHOR : Pavan Kumar Mishral, Sanjay Kumar Maurya, Ravindra Kumar Singh and Arun Kumar Misral

They present a semi automatic plant identification based on digital leaf and flower images. They proposed an algorithm for identification using multiclass classification

based on color, shape volume and cell feature. Each stage further also divided into three steps. First stage comparison based on extracted features from RGB component. Second stage based on shape feature Area Convexity, Perimeter Convexity, sphericity and Circulatory. And last stage based on cell and volume fraction feature. Experiment is performed on a sample of diverse collection of 1000 leaf and flower and recognition rate is up to 85% on an average. In proposed system entire feature cannot be taken at a time because it will take lot of time for computation and space. So multi stage comparisons are used for identification of image. Its multi stage comparison so required more tables to stored results and its long process. They used Unsupervised learning algorithm which has less accuracy as compared to supervised classification algorithm.

REVIEW NO 3:

AUTHOR : Tanakorn Tiay, Pipimphorn Benyaphaichit, and Panomkhawn Riyamongkol

They proposed flower Recognition System Based on Image Processing. This system uses edge and color characteristics of flower images to classify flowers. Hu's seven moment algorithm is applied to acquire edge characteristics. Red, green, blue, hue, and saturation characteristics are derived from histograms. K nearest neighbor is used to classify flowers. The system returns the top three most similar flower images. The Canny edge detection algorithm is applied to the cropped image to receive edge data. The edge data will be the input into Hu's seven-moment algorithm. Classification: All characteristic values will be classified by the K-nearest neighbor algorithm. The three most nearest flower characteristics are selected; the most nearest flower information is displayed. This system is based on color model so the accuracy is high if their color are distinct. But if colors are same then it may mislead to classify the image. So this system can be further improved to yield more accuracy by combining other features, such as numbers of petals and flower texture. The accuracy of this system is more than 80%.

REVIEW NO 4:

AUTHOR : C. H. Arun, W. R. Sam Emmanuel and D. Christopher Durairaj

They propose Texture Feature Extraction for Identification of Medicinal Plants and Comparison of Different Classifiers. Texture analyses of the leaf images have been done in this work using the feature computation. The features include grey textures, grey tone spatial dependency matrices (GTSDM) and Local Binary Pattern (LBP) operators. For each leaf image, a feature vector is generated from the statistical values. 70% of the images in the dataset are the training dataset and the rest are included in the test set. Six different classifiers are used to classify the plant leaves based on feature values. It is observed that without preprocessing (NP) the best performance obtained is 88.0% with kNN for categories C1 and C2, 94.7% with DT for C3, 94.7% with kNN for C4, 94.7% with SGD for category C5. In this case, the best performance of 94.7% is achieved using the categories C3, C4 and C5. The best classification gives very small error rate for the medicinal plant leaf identification. It is necessary to reduce the error rate for better result. The reduction of error rate may depend on the selected features and the dataset. The method of classifying without preprocessing performed better. The classification performance of 94.7% is obtained using Stochastic Gradient Descent, Decision Tree and kNN classifiers. Therefore it is concluded that the preprocessing approach is not suitable for the medicinal leaf identification. Instead of using preprocessing, the direct application of feature extraction with different categories produced better performance.

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1 AIM

The aim of the project is that to develop a program where it specify the species of the plant ,soil and weed by using deep learning method on image .As current rates of species loss triggered numerous attempts to protect and conserve biodiversity. Species conservation, however, requires species identification skills, a competence obtained through intensive training and experience. Field researchers, land managers, educators, civil servants, and the interested public would greatly benefit from accessible, up-to-date tools automating the process of species identification. Currently, relevant technologies, such as digital cameras, mobile devices, and remote access to databases, are ubiquitously available, accompanied by significant advances in image processing and pattern recognition. The idea of automated species identification is approaching reality. We review the technical status quo on computer vision approaches for plant species identification, highlight the main research challenges to overcome in providing applicable tools, and conclude with a discussion of open and future research thrusts.

3.2 SCOPE

The scope of the project is that system could help botanists and layman in identifying plant species rapidly. Deep learning is robust for feature extraction as it is superior in providing deeper information of images. In the existing system, a new CNN-based method named D-Leaf was proposed. The leaf images were pre-processed and the features were extracted by using three different Convolutional Neural Network (CNN) models. Leaves are commonly used in plant species recognition due to their availability throughout the year, especially, in the tropical areas. Many useful features can be acquired from a single leaf; such as, shape, texture, venation pattern, and colour. Each of these features could be extracted by different approaches; either through traditional morphometric measurement or machine learning methods. However, some approaches can be applied to obtain more than one feature, for

example, Zernike Moment can be applied to obtain shape and texture features. And in proposed system, the data acquisition and digitization phase the samples of plant leaves are collected and digital images of these samples are produced. The current work employed a public dataset of leaf images, which consists of the acquired digital images of different plant leaves. In the pre-processing phase, digital images of leaves are converted into a grayscale format. In the feature extraction phase, the morphological features, namely, major axis, minor axis, centroid, solidity, perimeter, and orientation are extracted. The major axis is the line connecting at one end called a base point to the tip of the leaf. For drawing the major axis two points are selected. Then the line will be drawn on to the selected points that represent the major perpendicular axis of the image.

A tree classifier over the plant taxonomy is used to replace the N-way flat softmax classifier, and the inter-species visual similarities are leveraged for training the inter-related classifiers for the sibling plant species under the same genus. In this paper, a two-layer plant taxonomy (genus layer and species layer) is constructed to organize large numbers of plant species and their most relevant genus hierarchically in a coarse-to-fine fashion. A path-based back propagation method is used to achieve joint learning of deep network and tree classifier in an end-to-end fashion, so that the critical issue of inter-error propagation can be addressed effectively.

CHAPTER 4

EXPERIMENT MATERIALS AND METHODS

4.1 SYSTEM ANALYSIS

4.1.1 Existing System

In the existing system, Automated plant species identification system could help botanists and layman in identifying plant species rapidly. Deep learning is robust for feature extraction as it is superior in providing deeper information of images. In the existing system, a new CNN-based method named D-Leaf was proposed. The leaf images were pre-processed and the features were extracted by using three different Convolutional Neural Network (CNN) models.

Leaves are commonly used in plant species recognition due to their availability throughout the year, especially, in the tropical areas. Many useful features can be acquired from a single leaf; such as, shape, texture, venation pattern, and colour. Each of these features could be extracted by different approaches; either through traditional morphometric measurement or machine learning methods. However, some approaches can be applied to obtain more than one feature, for example, Zernike Moment can be applied to obtain shape and texture features.

4.1.2 Drawbacks of Existing System

- Takes extremely longer time to reach the stationary distribution.
- It does not produce an exact answer.
- Do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities).
- Prohibitively Expensive in terms of Time and Memory Usage.

4.1.3 Proposed System

In data acquisition and digitization phase the samples of plant leaves are collected and digital images of these samples are produced. The current work employed a

public dataset of leaf images, which consists of the acquired digital images of different plant leaves.

In the pre-processing phase, digital images of leaves are converted into a grayscale format. In the feature extraction phase, the morphological features, namely, major axis, minor axis, centroid, solidity, perimeter, and orientation are extracted. The major axis is the line connecting at one end called a base point to the tip of the leaf. For drawing the major axis two points are selected. Then the line will be drawn on to the selected points that represent the major perpendicular axis of the image.

A tree classifier over the plant taxonomy is used to replace the N-way flat softmax classifier, and the inter-species visual similarities are leveraged for training the inter-related classifiers for the sibling plant species under the same genus. In this paper, a two-layer plant taxonomy (genus layer and species layer) is constructed to organize large numbers of plant species and their most relevant genus hierarchically in a coarse-to-fine fashion. A path-based back propagation method is used to achieve joint learning of deep network and tree classifier in an end-to-end fashion, so that the critical issue of inter-error propagation can be addressed effectively

The Performance of the proposed framework for plant leaf recognition is analyzed based on precision rate, RMSE (Root Mean Square Error) and FAR (False Acceptance Rate).

4.1.4 Advantages of Proposed System

- Boost the Performance.
- Fast response time.
- Low Deployment Cost.
- Quick Calculation Time.
- Very easy to implement for multi-class problem.

4.2 SYSTEM REQUIREMENTS

4.2.1 Hardware Specification

Hardware Requirement

- Hardware Minimum Requirement
- Disk Space 32 GB or more,10 GB or more for Foundation Edition
- Processor 1.4 GHz 64 bit
- Memory 512 MB
- Display (800 × 600) Capable video adapter and monitor

4.2.2 Software Specification

Backend Technologies

- Python
- Numpy
- Sci-learn
- Eclipse IDE

Frontend Technologies

- Web Technologies
- Bootstrap

4.3 SOFTWARE DESCRIPTION

4.3.1 PYTHON

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assure that Python will work on all these operating systems.

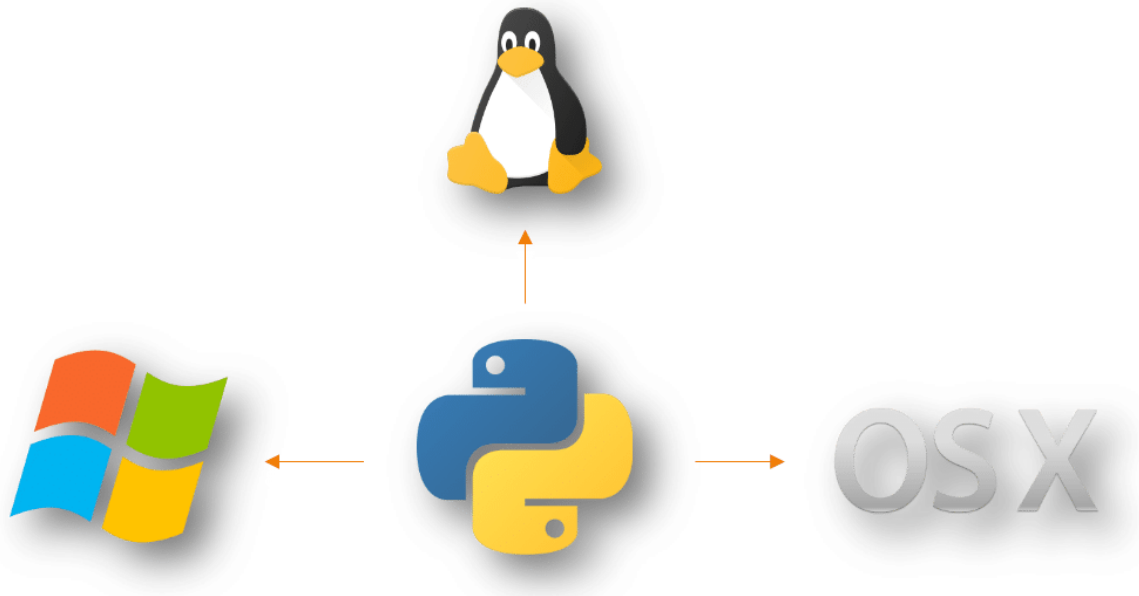


Fig.4.3.1.1 PYTHON SUPPORT

Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations.



Fig.4.3.1.2 PYTHON LIBRARIES

In addition, Python is the most popular language for machine learning and deep learning. As a matter of fact, today, all top organizations are investing in Python to implement machine learning in the back-end.

4.3.1.1 Python Line Structure

Python coding style comprises physical lines as well as logical lines or statements. A physical line in a Python program is a sequence of characters, and the end of the line terminates the line sequence as opposed to some other languages, such as C and C++ where a semi-colon is used to mark the end of the statement. A logical line, on the other hand, is composed of one or more physical lines. The use of a semi-colon is not prohibited in Python, although it's not mandatory. The NEWLINE token denotes the end of the logical line. A logical line that only contains spaces, comments, or tabs are called blank lines and they are ignored by the interpreter.

As we saw that in Python, a new line simply means that a new statement has started. Although, Python does provide a way to split a statement into a multiline statement or to join multiple statements into one logical line. This can be helpful to increase the readability of the statement. Following are the two ways to split a line into two or more lines:

Explicit Line Joining

In explicit line joining, we use a backward slash to split a statement into a multiline statement.

Implicit Line Joining

Statements that reside inside [], {}, or () parentheses can be broken down into two or more physical lines without using a back slash.

Multiple Statements on a Single Line

In Python, it is possible to club multiple statements in the same line using a semi-colon; however, most programmers do not consider this to be a good practice as it reduces the readability of the code.

Whitespaces and Indentation

Unlike most of the programming languages, Python uses indentation to mark a block of code. According to Python coding style guideline or PEP8, we should keep an indent size of four.

Most of the programming languages provide indentation for better code formatting and don't enforce to have it. But in Python it is mandatory. This is why indentation is so crucial in Python.

Comments in any programming language are used to increase the readability of the code. Similarly, in Python, when the program starts getting complicated, one of the best ways to maintain the readability of the code is to use Python comments. It is considered a good practice to include documentations and notes in the python syntax since it makes the code way more readable and understandable to other programmers as well, which comes in handy when multiple programmers are simultaneously working on the same project.

The code can only explain how it does something and not why it does that, but Python comments can do that. With Python comments, we can make documentations for various explanations in our code itself. Comments are nothing but tagged lines of codes which increase the readability of a code and make it self-explanatory. There are different ways of creating comments depending on the type of comment we want to include in our code. Following are different kinds of comments that can be included in our Python program:

1. Single Line Comments
2. Docstring Comments
3. Multiline Comments

Single line Python comments are marked with `#` character. These comments end at the end of the physical line, which means that all characters starting after the `#` character (and lasts till the end of the line) are part of the comment.

Python has the documentation strings (or docstrings) feature which is usually the first statement included in functions and modules. Rather than being ignored by the Python Interpreter like regular comments, docstrings can actually be accessed at the run time using the dot operator.

It gives programmers an easy way of adding quick notes with every Python module, function, class, and method. To use this feature, we use triple quotes in the beginning

of the documentation string or comment and the closing triple quotes at the end of the documentation comment. Docstrings can be one-liners as well as multi-liners.

Unlike some programming languages that support multiline comments, such as C, Java, and more, there is no specific feature for multiline comments in Python. But that does not mean that it is totally impossible to make multiline comments in Python. There are two ways we can include comments that can span across multiple lines in our Python code.

Python Block Comments: We can use several single line comments for a whole block. This type of comment is usually created to explain the block of code that follows the Block comment. Python Block comment is the only way of writing a real comment that can span across multiple lines. It is supported and preferred by Python's PEP8 style guide since Block comments are ignored by Python interpreter or parser. However, nothing is stopping programmers from using the second 'non-real' way of writing multiline comments in Python which is explained below.

Using Docstrings: Docstrings are largely used as multiline comments in Python by many programmers since it is the closest thing to having a multiline comment feature in Python. While it is not wrong to use docstrings when we need to make multiline comments, it is important to keep in mind that there is a significant difference between docstrings and comments. Comments in Python are totally ignored by the Python Interpreter, while docstrings, when used inside the Python function, can be accessed at the run time.

4.3.1.2 Python Data Types

One of the most crucial part of learning any programming language is to understand how data is stored and manipulated in that language. Users are often inclined toward Python because of its ease of use and the number of versatile features it provides. One of those features is dynamic typing.

In Python, unlike statically typed languages like C or Java, there is no need to specifically declare the data type of the variable. In dynamically typed languages such as Python, the interpreter itself predicts the data type of the Python Variable based on the type of value assigned to that variable.

Advantages of Python

- Universal Language Construct
- Support both High Level and Low Level Programming
- Language Interoperability
- Fastest Development life cycle therefore more productive coding environment
- Less memory used because a single container hold multiple data types and each type doesn't require its own function
- Learning Ease and open source development
- Speed and user-friendly data structure
- Extensive and extensible libraries.
- Simple & support IoT

4.3.2 PANDAS

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures. Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables. Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily. Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas'

DataFrames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and DataFrames.

1.Series: a pandas Series is a one dimensional data structure that can store values — and for every value it holds a unique index, too.

2.DataFrame: a pandas DataFrame is a two (or more) dimensional data structure — basically a table with rows and columns. The columns have names and the rows have indexes.

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc. Now, DataFrames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of potentially different types.

In general, you could say that the Pandas DataFrame consists of three main components: the data, the index, and the columns.

Firstly, the DataFrame can contain data that is:

- A Pandas DataFrame
- A Pandas Series: a one-dimensional labeled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a DataFrame.
- A NumPy ndarray, which can be a record or structured
- A two-dimensional ndarray
- Dictionaries of one-dimensional ndarray's, lists, dictionaries or Series.

Some of the key features of Python Pandas are as follows:

- It provides DataFrame objects with default and customized indexing which is very fast and efficient.
- There are tools available for loading data of different file formats into in-memory data objects.
- It is easy to perform data alignment and integrated handling of missing data in Python Pandas.
- It is very simple to perform pivoting and reshaping of data sets in Pandas.
- It also provides indexing, label-based slicing, and sub-setting of large data sets.
- We can easily insert and delete columns from a data structure.
- Data aggregation and transformations can be done using group by.
- High-performance merging and joining of data can be done using Pandas.
- It also provides time series functionality.
- Inserting and deleting columns in data structures.
- Merging and joining data sets.
- Reshaping and pivoting data sets.
- Aligning data and dealing with missing data.
- Manipulating data using integrated indexing for DataFrame objects.
- Performing split-apply-combine on data sets using the group by engine.
- Manipulating high-dimensional data in a data structure with a lower dimension using hierarchical axis indexing.
- Subsetting, fancy indexing, and label-based slicing data sets that are large in size.
- Generating data range, converting frequency, date shifting, lagging, and other time-series functionality.
- Reading from files with CSV, XLSX, TXT, among other formats.
- Arranging data in an order ascending or descending.
- Filtering data around a condition.
- Analyzing time series.

- Iterating over a data set.

4.3.3 Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy. A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

an array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

- The data pointer indicates the memory address of the first byte in the array.
- The data type or dtype pointer describes the kind of elements that are contained within the array.
- The shape indicates the shape of the array.

- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

With NumPy, we work with multidimensional arrays. We'll dive into all of the possible types of multidimensional arrays later on, but for now, we'll focus on 2-dimensional arrays. A 2-dimensional array is also known as a matrix, and is something you should be familiar with. In fact, it's just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we're able to extract an element from a matrix.

We can create a NumPy array using the `numpy.array` function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we'll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to be of the same type, so if we include the header row, all the elements in the array will be read in as strings. Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats.

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

1. NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
2. Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

4.3.4 Mathplotlib

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data virtualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an object oriented way. Furthermore it can be used with general-purpose GUI toolkits like wxPython, Qt, and GTK+. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things

possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

4.3.5 About IDE

IDE – Integrated Development Environment. It is software suite. It contains different instruments for web application software engineers to compose and test applications. Programming Web application software engineers utilize wide scope of devices habitually all through application source code creation, building and application testing. Programming Development apparatuses frequently incorporate word processors for source code creation, code libraries for reuse, compilers and test stages.

An Integrated Development Environment (IDE) is a product application that gives a programming domain to streamline creating and troubleshooting programming. Instead of playing out every one of the means required to make an executable program as disconnected individual assignments, it brings every one of the devices required into one application and workspace. Every one of the devices has a familiarity with the earth, and they cooperate to display a consistent improvement set for the designer.

An IDE typically has an editor and a compiler window. There, we can write our code, run it, and compile it. Hence, it becomes much easier to develop applications using an IDE. The major reasons for using an IDE are given below:

- ▶ We can easily write our code in its text editor window.
- ▶ To store the resource files of a project, we have a project editor window.
- ▶ It makes debugging of the code easier.
- ▶ An IDE's autosuggest feature helps us in writing efficient code.
- ▶ We can import other projects for integrating with the existing one.

Without an IDE, a product web application software engineer must choose, send, incorporate and deal with these instruments independently. An IDE brings huge numbers of those improvement related instruments all together system, application or administration. The incorporated toolset is intended to streamline programming advancement and can distinguish and limit coding missteps and grammatical errors.

- ✓ Komodo
- ✓ NetBeans
- ✓ PyCharm
- ✓ Python for Visual Studio.NET
- ✓ PyDev
- ✓ PyStudio
- ✓ LiClipse (Eclipse)

LiClipse is at the core for all Python based application development activities. It's been available for many years. LiClipse is a set of plugins to enhance Eclipse. It improves the overall Eclipse experience. It is best IDE tool to develop Python Applications.

Advantages of LiClipse

- Perfect Debugger
- Error checking features
- An enhanced ability to analyze the code for quality and security concerns
- Auto complete feature

Based on the various IDE that are described above, we decided to use LiClipse as IDE (Integrated Development Environment) for developing this e-Voting web based application.

Important Features That Make LiClipse our IDE of Choice

- Rich Reusable Libraries are available
- Model Driven Development
- Enterprise Python Tooling
- Powerful Debugging options

IDE is a program that is used to write applications (other programs) in a variety of programming languages. It allows you to complete many distinct program development steps in one place (program window). A program which allows you to do this is known as an integrated development environment, or IDE. The program is free and easy to import onto your home computer.

LiClipse is used for writing programs in Python. There are several steps that must be completed to set up LiClipse and further steps to start up a new Python project (set of Python components which work together to make up an application).

PyCharm is an Integrated Development Environment (IDE) that provides tools for writing efficient code. It supports various frameworks such as Django, Flask, and Pyramid. PyCharm is majorly used by developers for application development with Python language.

Let's suppose, we are making a project using Python programming language. We know that Python is extensively popular among the developer's community, as well as in industries, due to its following features:

- ▶ Simple to code: Python is a very simple language compared to other programming languages. Also, it is a developer-friendly language.
- ▶ Object-oriented language: Python is an object-oriented language that uses concepts of classes and objects.
- ▶ GUI programming support: We can create graphical user interfaces using Python Tkinter library. Besides, we can use modules such as PyQt5, PyQt4, and wxPython to create a GUI application.
- ▶ High-level language: Python is a high-level programming language. We do not need to take care of managing memory or the system architecture while writing Python codes.
- ▶ Portability: Python is a portable programming language, i.e., we can run Python codes on any platform. For example, we can run a code written for Windows in Linux, UNIX, and Mac, without changing it.
- ▶ Integrated language: Python is an integrated programming language. We can easily integrate C, C++, etc. with Python.

- ▶ Interpreted language: Python is an interpreted programming language as well. Python's source code is changed to bytecode. Due to this, it is directly executed without compiling.
- ▶ Large set of standard libraries: Python consists of a rich set of modules, packages, and libraries. It helps in various stages of application development.

"notebook" or "notebook documents" denote documents that contain both code and rich text elements, such as figures, links, equations, ... Because of the mix of code and text elements, these documents are the ideal place to bring together an analysis description, and its results, as well as, they can be executed perform the data analysis in real time.

The Jupyter Notebook App produces these documents.

As a server-client application, the Jupyter Notebook App allows you to edit and run your notebooks via a web browser. The application can be executed on a PC without Internet access, or it can be installed on a remote server, where you can access it through the Internet.

Its two main components are the kernels and a dashboard.

A kernel is a program that runs and introspects the user's code. The Jupyter Notebook App has a kernel for Python code, but there are also kernels available for other programming languages.

The dashboard of the application not only shows you the notebook documents that you have made and can reopen but can also be used to manage the kernels: you can which ones are running and shut them down if necessary.

4.4 ARCHITECTURE DIAGRAM

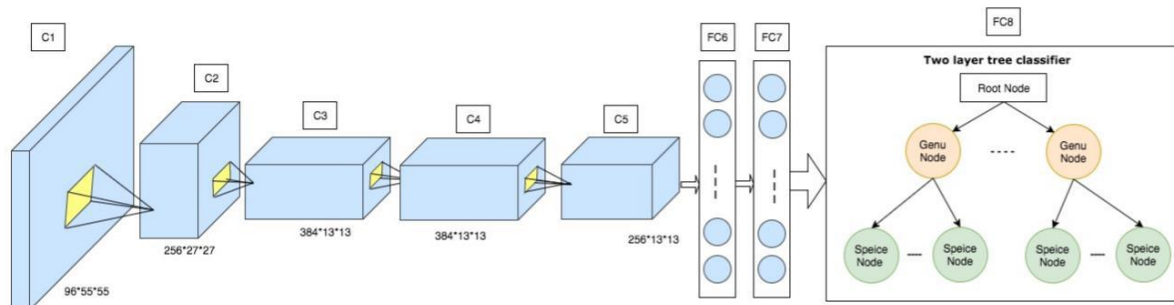


Fig.4.4 ARCHITECTURE DIAGRAM

4.5 SYSTEM MODULES

4.5.1 Exploratory Data Evaluation

Exploratory data analysis is a process of sifting through data in search of interesting information or patterns. Analysts' current tools for exploring data include database management systems, statistical analysis packages, data mining tools, visualization tools, and report generators. Since the exploration process seeks the unexpected in a data-driven manner, it is crucial that these tools are seamlessly integrated so analysts can flexibly select and compose tools to use at each stage of analysis. Few systems have integrated all these capabilities either architecturally or at the user interface level. Visage's information-centric approach allows coordination among multiple application user interfaces. It uses an architecture that keeps track of the mapping of visual objects to information in shared databases.

Graphs generated through EDA are distinct from final graphs. You will typically generate dozens, if not hundreds, of exploratory graphs in the course of analyzing a dataset. Of these graphs, you may end up publishing one or two in a final format. One purpose of EDA is to develop a personal understanding of the data, so all your code and graphs should be geared towards that purpose. Important details that you might add if you were to publish a graph² are not necessary in an exploratory graph.

4.5.2 Pre-processing

Pre-processing literally means that after one algorithm is applied to your image, the output of that algorithm is fed to the input of some other algorithms, while an enhanced image should be the final result for a particular algorithm. Of course, an image enhancement algorithm can also have pre-processing steps, while an enhanced image can still be used as an input for other algorithms.

Convert color images to grayscale to reduce computation complexity: in certain problems you'll find it useful to lose unnecessary information from your images to reduce space or computational complexity.

For example, converting your colored images to grayscale images. This is because in many objects, color isn't necessary to recognize and interpret an image. Grayscale can be good enough for recognizing certain objects. Because color images contain more information than black and white images, they can add unnecessary complexity and take up more space in memory (Remember how color images are represented in three channels, which means that converting it to grayscale reduces the number of pixels that need to be processed).

One important constraint that exists in some machine learning algorithms, such as CNN, is the need to resize the images in your dataset to a unified dimension. This implies that our images must be preprocessed and scaled to have identical widths and heights before fed to the learning algorithm.

Another common pre-processing technique involves augmenting the existing dataset with perturbed versions of the existing images. Scaling, rotations and other affine transformations are typical. This is done to enlarge your dataset and expose the neural network to a wide variety of variations of your images. This makes it more likely that your model recognizes objects when they appear in any form and shape.

4.5.3 Model Creation

It is also possible to standardize pixel values across the entire dataset. This is called feature standardization and mirrors the type of standardization often performed for each column in a tabular dataset.

You can perform feature standardization by setting the `featurewise_center` and `featurewise_std_normalization` arguments on the `ImageDataGenerator` class. These

are in fact set to True by default and creating an instance of ImageDataGenerator with no arguments will have the same effect.

A whitening transform of an image is a linear algebra operation that reduces the redundancy in the matrix of pixel images.

Less redundancy in the image is intended to better highlight the structures and features in the image to the learning algorithm.

Typically, image whitening is performed using the Principal Component Analysis (PCA) technique. More recently, an alternative called ZCA (learn more in Appendix A of this tech report) shows better results and results in transformed images that keeps all of the original dimensions and unlike PCA, resulting transformed images still look like their originals.

Image Augmentations techniques are methods of artificially increasing the variations of images in our data-set by using horizontal/vertical flips, rotations, variations in brightness of images, horizontal/vertical shifts etc.

4.5.4 Prediction

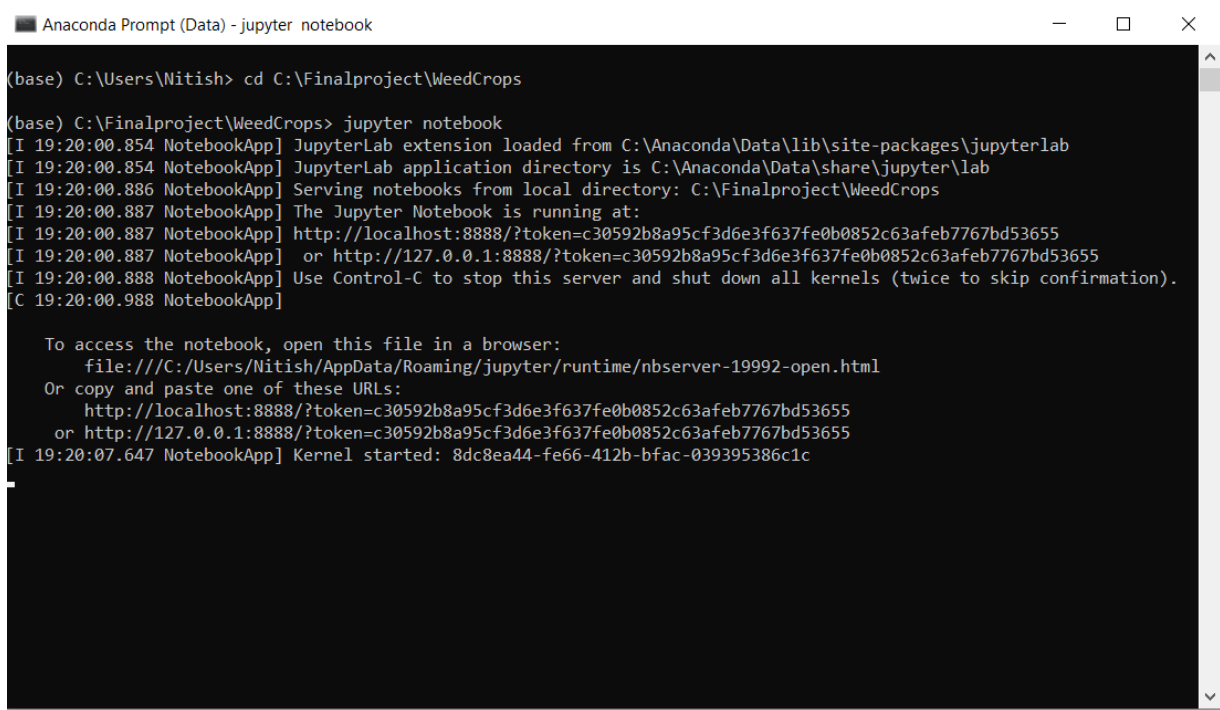
This algorithmic approach is based on distinctive characteristics analysis by analyzing the expected error minimization. This approach considered the empirical risk to improve the training procedure. The risk estimation is here based on the structural analysis so that the generalization error will be reduced. The error margin is analyzed under class deviation and based on it nearest training patterns are obtained. . This model is also based on the polynomial kernel representation so that the effective learning to the elements will be done and more accuracy will be obtained.

It is a recurrent neural network that is trained using Backpropagation Through Time and overcomes the vanishing gradient problem. A block has components that make it smarter than a classical neuron and a memory for recent sequences. A block contains gates that manage the block's state and output. A block operates upon an input sequence and each gate within a block uses the sigmoid activation units to control whether they are triggered or not, making the change of state and addition of information flowing through the block condition.

CHAPTER 5

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

5.1 RESULT

A screenshot of the Anaconda Prompt (Data) - jupyter notebook window. The terminal shows the command 'cd C:\Finalproject\WeedCrops' and 'jupyter notebook'. It displays various status messages from the JupyterLab application, including the extension loaded, application directory, and the URL to access the notebook. The terminal text is as follows:

```
(base) C:\Users\Nitish> cd C:\Finalproject\WeedCrops
(base) C:\Finalproject\WeedCrops> jupyter notebook
[I 19:20:00.854 NotebookApp] JupyterLab extension loaded from C:\Anaconda\Data\lib\site-packages\jupyterlab
[I 19:20:00.854 NotebookApp] JupyterLab application directory is C:\Anaconda\Data\share\jupyter\lab
[I 19:20:00.886 NotebookApp] Serving notebooks from local directory: C:\Finalproject\WeedCrops
[I 19:20:00.887 NotebookApp] The Jupyter Notebook is running at:
[I 19:20:00.887 NotebookApp] http://localhost:8888/?token=c30592b8a95cf3d6e3f637fe0b0852c63afeb7767bd53655
[I 19:20:00.887 NotebookApp] or http://127.0.0.1:8888/?token=c30592b8a95cf3d6e3f637fe0b0852c63afeb7767bd53655
[I 19:20:00.888 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 19:20:00.988 NotebookApp]

To access the notebook, open this file in a browser:
file:///C:/Users/Nitish/AppData/Roaming/jupyter/runtime/nbserver-19992-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=c30592b8a95cf3d6e3f637fe0b0852c63afeb7767bd53655
or http://127.0.0.1:8888/?token=c30592b8a95cf3d6e3f637fe0b0852c63afeb7767bd53655
[I 19:20:07.647 NotebookApp] Kernel started: 8dc8ea44-fe66-412b-bfac-039395386c1c
```

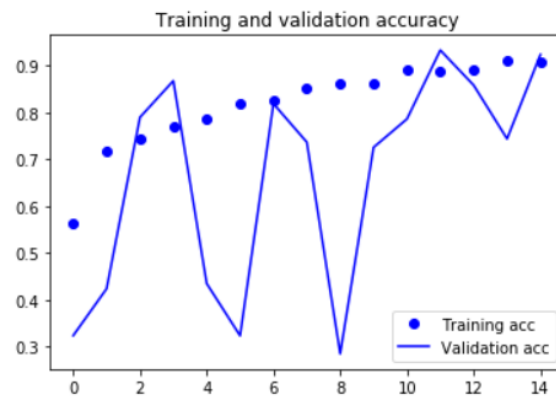
Fig.5.1 ANACONDA PROMPT

```
In [47]: epochs = range(len(acc))
```

```
In [48]: plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()
```

Out[48]: <Figure size 432x288 with 0 Axes>

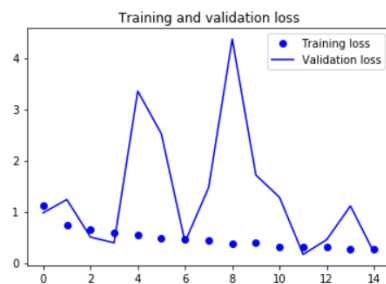


<Figure size 432x288 with 0 Axes>

Fig.5.2 TRAINING AND VALIDATION ACCURACY

```
In [49]: plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```



```
In [50]: test_samples = 3067
batch_size_test = 31
```

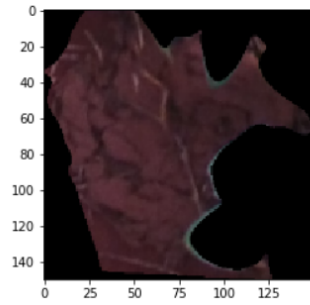
```
In [51]: score = model.evaluate_generator(test_generator, steps = math.ceil(test_samples/batch_size_test))
```

```
In [52]: print("\nTest accuracy: %.1f%%" % (100.0 * score[1]))
```

Test accuracy: 93.0%

Fig.5.3 TRAINING AND VALIDATION LOSS

```
In [58]: from matplotlib import pyplot as plt
plt.imshow(image_to_predict, interpolation='nearest')
plt.show()
```



```
In [59]: data = result[0][3]
print(data)
if data < 5:
    print("It's Soil")
elif data < 26:
    print("It's Soya Bean")
elif data < 33:
    print("It's Grass")
else:
    print("It's Weed")
```

```
0.0
It's Soil
```

Fig.5.4 FINAL OUTPUT

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 SUMMARY

Let us now discuss about the summary in this section, from this project it can accurately predict the plant species and the different part of the plant like weed and soil. This project can save a lot of time for botanist and people. From this I can understand that deep learning is best for image related problems and it can give accurate results.

6.2 CONCLUSION

A path-based deep ensemble learning algorithm is developed to jointly training the deep neural network and the tree classifier over a fixed plant taxonomy. We also construct a big plant dataset suit for large-scale plant identification task. Experiments on our new datasets provide that our path-based deep network can achieve very competitive results on both the accuracy rates and the computational efficiency for largescale plant identification.

REFERENCES

1. Ali Beikmohammadi and Karim Faez. Leaf classification for plant recognition with deep transfer learning. In 2018 4th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), pages 21–26. IEEE, 2018.
2. Vinit Bodhwani, DP Acharjya, and Umesh Bodhwani. Deep residual networks for plant identification. *Procedia Computer Science*, 152:186–194, 2019.
3. Sruti Das Choudhury, Jin-Gang Yu, and Ashok Samal. Leaf recognition using contour unwrapping and apex alignment with tuned random subspace method. *Biosystems Engineering*, 170:72–84, 2018.
4. Mostafa Mehdipour Ghazi, Berrin Yanikoglu, and Erchan Aptoula. Plant identification using deep neural networks via optimization of transfer learning parameters. *Neurocomputing*, 235:228–235, 2017.
5. Guillermo L Grinblat, Lucas C Uzal, Monica G Larese, and Pablo M ´ Granitto. Deep learning for plant identification using vein morphological patterns. *Computers and Electronics in Agriculture*, 127:418–424, 2016.
6. Jing Hu, Zhibo Chen, Meng Yang, Rongguo Zhang, and Yaji Cui. A multiscale fusion convolutional neural network for plant leaf recognition. *IEEE Signal Processing Letters*, 25(6):853–857, 2018.
7. Yan, Zhicheng, et al. HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition. *Proceedings of the IEEE International Conference on Computer Vision*. 2015.
8. Kuang Z, Li Z, Zhao T, et al. Deep Multi-task Learning for Large-Scale Image Classification[C]//Multimedia Big Data (BigMM), 2017 IEEE Third International Conference on. IEEE, 2017: 310-317.
9. Kotschieder, Peter, et al. "Deep neural decision forests." *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

10. Wang, Jiang, et al. "Cnn-rnn: A unified framework for multi-label image classification." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

APPENDIX

A.SCREENSHOT

```
Anaconda Prompt (Data) - jupyter notebook

(base) C:\Users\Nitish> cd C:\Finalproject\WeedCrops

(base) C:\Finalproject\WeedCrops> jupyter notebook
[I 19:20:00.854 NotebookApp] JupyterLab extension loaded from C:\Anaconda\Data\lib\site-packages\jupyterlab
[I 19:20:00.854 NotebookApp] JupyterLab application directory is C:\Anaconda\Data\share\jupyter\lab
[I 19:20:00.886 NotebookApp] Serving notebooks from local directory: C:\Finalproject\WeedCrops
[I 19:20:00.887 NotebookApp] The Jupyter Notebook is running at:
[I 19:20:00.887 NotebookApp] http://localhost:8888/?token=c30592b8a95cf3d6e3f637fe0b0852c63afeb7767bd53655
[I 19:20:00.887 NotebookApp] or http://127.0.0.1:8888/?token=c30592b8a95cf3d6e3f637fe0b0852c63afeb7767bd53655
[I 19:20:00.888 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 19:20:00.988 NotebookApp]

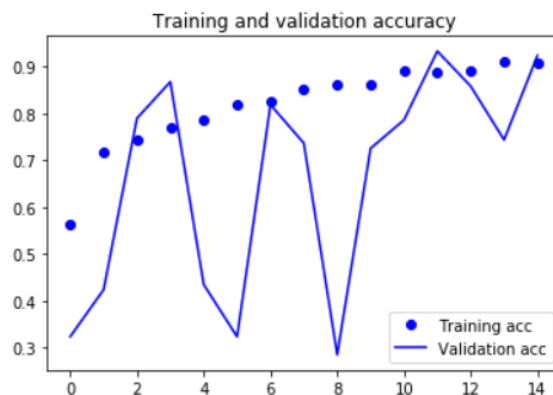
To access the notebook, open this file in a browser:
    file:///C:/Users/Nitish/AppData/Roaming/jupyter/runtime/nbserver-19992-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=c30592b8a95cf3d6e3f637fe0b0852c63afeb7767bd53655
    or http://127.0.0.1:8888/?token=c30592b8a95cf3d6e3f637fe0b0852c63afeb7767bd53655
[I 19:20:07.647 NotebookApp] Kernel started: 8dc8ea44-fe66-412b-bfac-039395386c1c
```

```
In [47]: epochs = range(len(acc))
```

```
In [48]: plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()
```

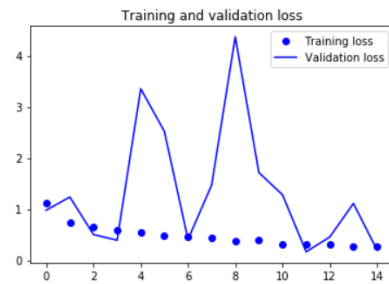
```
Out[48]: <Figure size 432x288 with 0 Axes>
```



```
<Figure size 432x288 with 0 Axes>
```

```
In [49]: plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```



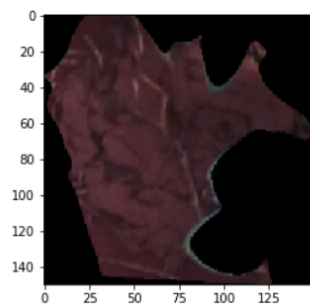
```
In [50]: test_samples = 3067
batch_size_test = 31
```

```
In [51]: score = model.evaluate_generator(test_generator, steps = math.ceil(test_samples/batch_size_test))
```

```
In [52]: print("\nTest accuracy: %.1f%%" % (100.0 * score[1]))
```

Test accuracy: 93.0%

```
In [58]: from matplotlib import pyplot as plt
plt.imshow(image_to_predict, interpolation='nearest')
plt.show()
```



```
In [59]: data = result[0][3]
print(data)
if data < 5:
    print("It's Soil")
elif data < 26:
    print("It's Soya Bean")
elif data < 33:
    print("It's Grass")
else:
    print("It's Weed")
```

0.0
It's Soil

B.SOURCE CODE

```
import os
import shutil
import math
import keras
keras.__version__
from keras import layers
from keras import models
from keras import optimizers
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
import os
base_dir = './WeedCrops'
os.makedirs(base_dir, exist_ok=True)
train_dir = os.path.join(base_dir, 'train')
os.makedirs(train_dir, exist_ok=True)
validation_dir = os.path.join(base_dir, 'validation')
os.makedirs(validation_dir, exist_ok=True)
test_dir = os.path.join(base_dir, 'test')
os.makedirs(test_dir, exist_ok=True)
train_grass_dir = os.path.join(train_dir, 'grass')
os.makedirs(train_grass_dir, exist_ok=True)
train_soil_dir = os.path.join(train_dir, 'soil')
os.makedirs(train_soil_dir, exist_ok=True)
train_soybean_dir = os.path.join(train_dir, 'soybean')
os.makedirs(train_soybean_dir, exist_ok=True)
train_weed_dir = os.path.join(train_dir, 'weed')
os.makedirs(train_weed_dir, exist_ok=True)
validation_grass_dir = os.path.join(validation_dir, 'grass')
os.makedirs(validation_grass_dir, exist_ok=True)
validation_soil_dir = os.path.join(validation_dir, 'soil')
os.makedirs(validation_soil_dir, exist_ok=True)
```

```

validation_soybean_dir = os.path.join(validation_dir, 'soybean')
os.makedirs(validation_soybean_dir, exist_ok=True)
validation_weed_dir = os.path.join(validation_dir, 'weed')
os.makedirs(validation_weed_dir, exist_ok=True)
test_grass_dir = os.path.join(test_dir, 'grass')
os.makedirs(test_grass_dir, exist_ok=True)
test_soil_dir = os.path.join(test_dir, 'soil')
os.makedirs(test_soil_dir, exist_ok=True)
test_soybean_dir = os.path.join(test_dir, 'soybean')
os.makedirs(test_soybean_dir, exist_ok=True)
test_weed_dir = os.path.join(test_dir, 'weed')
os.makedirs(test_weed_dir, exist_ok=True)
original_dataset_dir_soil = './dataset/soil'
original_dataset_dir_soybean = './dataset/soybean'
original_dataset_dir_grass = './dataset/grass'
original_dataset_dir_weed = './dataset/broadleaf'
fnames = ['{}.tif'.format(i) for i in range(1,1950)]
for fname in fnames:
    src = os.path.join(original_dataset_dir_soil, fname)
    dst = os.path.join(train_soil_dir, fname)
    shutil.copyfile(src, dst)
fnames = ['{}.tif'.format(i) for i in range(1950, 2600)]
for fname in fnames:
    src = os.path.join(original_dataset_dir_soil, fname)
    dst = os.path.join(validation_soil_dir, fname)
    shutil.copyfile(src, dst)
fnames = ['{}.tif'.format(i) for i in range(2600, 3250)]
for fname in fnames:
    src = os.path.join(original_dataset_dir_soil, fname)
    dst = os.path.join(test_soil_dir, fname)
    shutil.copyfile(src, dst)

```

```

fnames = ['{}.tif'.format(i) for i in range(1,4427)]
for fname in fnames:
    src = os.path.join(original_dataset_dir_soybean, fname)
    dst = os.path.join(train_soybean_dir, fname)
    shutil.copyfile(src, dst)
fnames = ['{}.tif'.format(i) for i in range(4427, 5902)]
for fname in fnames:
    src = os.path.join(original_dataset_dir_soybean, fname)
    dst = os.path.join(validation_soybean_dir, fname)
    shutil.copyfile(src, dst)
fnames = ['{}.tif'.format(i) for i in range(5902, 7377)]
for fname in fnames:
    src = os.path.join(original_dataset_dir_soybean, fname)
    dst = os.path.join(test_soybean_dir, fname)
    shutil.copyfile(src, dst)
fnames = ['{}.tif'.format(i) for i in range(1,2113)]
for fname in fnames:
    src = os.path.join(original_dataset_dir_grass, fname)
    dst = os.path.join(train_grass_dir, fname)
    shutil.copyfile(src, dst)
fnames = ['{}.tif'.format(i) for i in range(2113, 2817)]
for fname in fnames:
    src = os.path.join(original_dataset_dir_grass, fname)
    dst = os.path.join(validation_grass_dir, fname)
    shutil.copyfile(src, dst)
fnames = ['{}.tif'.format(i) for i in range(2817, 3521)]
for fname in fnames:
    src = os.path.join(original_dataset_dir_grass, fname)
    dst = os.path.join(test_grass_dir, fname)
    shutil.copyfile(src, dst)
fnames = ['{}.tif'.format(i) for i in range(1,716)]

```

```

for fname in fnames:
    src = os.path.join(original_dataset_dir_weed, fname)
    dst = os.path.join(train_weed_dir, fname)
    shutil.copyfile(src, dst)
fnames = ['{}.tif'.format(i) for i in range(716, 954)]
for fname in fnames:
    src = os.path.join(original_dataset_dir_weed, fname)
    dst = os.path.join(validation_weed_dir, fname)
    shutil.copyfile(src, dst)
fnames = ['{}.tif'.format(i) for i in range(954, 1192)]
for fname in fnames:
    src = os.path.join(original_dataset_dir_weed, fname)
    dst = os.path.join(test_weed_dir, fname)
    shutil.copyfile(src, dst)
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(150,150,3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(4, activation='softmax'))
model.summary()
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',

```

```

        metrics=['acc'])
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,)
test_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(train_dir, target_size=(150, 150),
    color_mode="rgb", batch_size=92,
    class_mode='categorical')
validation_generator = test_datagen.flow_from_directory(
    validation_dir,
    target_size=(150, 150),
    color_mode="rgb",
    batch_size=31,
    class_mode='categorical')
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(150, 150),
    color_mode="rgb",
    batch_size=31,
    class_mode='categorical')
training_samples = 9202
batch_size_training_generator = 92
validation_samples = 3067
batch_size_validation_generator = 31
history = model.fit_generator(
    train_generator,

```

```

steps_per_epoch=math.ceil(training_samples/batch_size_training_generator),
epochs=15,
validation_data=validation_generator,
validation_steps=math.ceil(validation_samples/batch_size_validation_generator))
model.save('model_weedcrops.h5')
import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(acc))
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
test_samples =3067
batch_size_test=31
score=          model.evaluate_generator(test_generator,          steps          =
math.ceil(test_samples/batch_size_test))
print("\nTest accuracy: %.1f%%" % (100.0 * score[1]))
from PIL import Image
import numpy as np

```

```

from skimage import transform
def load(filename):
    np_image = Image.open(filename)
    np_image = np.array(np_image).astype('float32')/255
    np_image = transform.resize(np_image, (150, 150, 3))
    np_image = np.expand_dims(np_image, axis=0)
    return np_image
label_map = (test_generator.class_indices)
print (label_map)
image_to_predict = load('./WeedCrops/test/soil/2603.tif')
result = model.predict(image_to_predict)
result= np.around(result,decimals=3)
result=result*100
image_to_predict= np.squeeze(image_to_predict,axis=0)
from matplotlib import pyplot as plt
plt.imshow(image_to_predict, interpolation='nearest')
plt.show()
data = result[0][3]
print(data)
if data < 5:
    print("It's Soil")
elif data < 26:
    print("It's Soya Bean")
elif data < 33:
    print("It's Grass")
else:
    print("It's Weed")

```