A Project Report On

COTTON PLANT DISEASE PREDICTION USING DEEP LEARNING

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Submitted By

P.BHAVANA	**19P31A1236**
AYESHA MOLLICK	**19P31A1204**
K.PRAVALLIKA	**19P31A1240**
DIPAK KUMAR CHAUDHARY	**19P31A1258**

Under the esteemed supervision of

Mrs K Naga Bhargavi., M. Tech ., (Ph.D) .

Associate Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

Permanently Affiliated to JNTUK, Kakinada * Approved by AICTE New Delhi

Accredited by NBA, Accredited by NAAC (A+) with 3.4 CGPA

Aditya Nagar, ADB Road, Surampalem , Kakinada District, Andhra Pradesh.

2019-2023

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

Permanently Affiliated to JNTUK, Kakinada * Approved by AICTE New Delhi

Accredited by NBA, Accredited by NAAC (A+) with 3.4 CGPA

Aditya Nagar, ADB Road, Surampalem , Kakinada District, Andhra Pradesh

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project work entitled "**Cotton Plant Disease Prediction Using Deep Learning**", is a bonafide work carried out by **P. Bhavana (19P31A1236), Ayesha Mollick (19P31A1204), K.Pravallika (19P31A1240),Dipak Kumar Chaudhary(19P31A1258)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Information Technology** from Aditya College of Engineering & Technology during the academic year 2019-2023.

Project Guide	**Head Of The Department**
Mrs K Naga Bhargavi.,M.Tech., Ph.D.	**Mr. R. V. V. N. Bheema Rao.,M.Tech.,(Ph.D)**
Associate Professor	**Assistant Professor**

EXTERNAL EXAMINER

DECLARATION

We hereby declare that this project entitled "Cotton Plant Disease Prediction Using Deep Learning", has been undertaken by us and this work has been submitted to **Aditya College of Engineering & Technology** affiliated to JNTUK, Kakinada, in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Information Technology**.

We further declare that this project work has not been submitted in full or part for the award of any degree of this or in any other educational institutions.

Project Associates

P.Bhavana	**19P31A1236**
Ayesha Mollick	**19P31A1204**
K.Pravallika	**19P31A1240**
Dipak Kumar Chaudhary	**19P31A1258**

ACKNOWLEDGEMENT

It is with immense pleasure that we would like to express our indebted gratitude to our Project Supervisor, **Mrs K Naga Bhargavi** **M.Tech.,(Ph.D).** who has guided us a lot and encouraged us in every step of the project work, his valuable moral support and guidance throughout the project helped us to a great extent.

We wish to express our sincere thanks to the Head of the Department **Mr. R V V N Bheema Rao** **M.Tech.,(Ph.D)** for his valuable guidance given to us throughout the period of the project work and throughout the program.

We feel elated to thank **Dr. A Rama Krishna** **Ph.D** Dean of Aditya College of Engineering & Technology for his cooperation in completion of our project and throughout the program.

We feel elated to thank **Dr. Dola Sanjay S** **Ph.D** Principal of Aditya College of Engineering & Technology for his cooperation in completion of our project and throughout the program.

We wish to express our sincere thanks to all **faculty members, lab programmers** for their valuable assistance throughout the period of the project.

We avail this opportunity to express our deep sense and heart full thanks to the Management of **Aditya College Of Engineering & Technology** for providing a great support for us in completing our project and also throughout the program.

P.Bhavana	**(19P31A1236)**
Ayesha Mollick	**(19P31A1204)**
K.Pravallika	**(19P31A1240)**
Dipak Kumar Chaudhary	**(19P31A1258)**

Aditya College of Engineering & Technology

Approved by AICTE, New Delhi, * Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA, Accredited by NAAC (A+) with CGPA of 3.4
Recognized by UGC under Section 2(f) and 12(B) of UGC Act 1956
Aditya Nagar, ADB Road, Surampalem

Institute Vision & Mission

Vision

To induce higher planes of learning by imparting technical education with

- International standards
- Applied research
- Creative Ability
- Values based instruction and to emerge as a premiere institute

Mission

Achieving academic excellence by providing globally acceptable technical education by forecasting technology through

- Innovative research and development
- Industry institute interaction
- Empowered manpower

[illegible]

Principal

PRINCIPAL
Aditya College of
Engineering & Technolo
SURAMPALEM

Aditya College of Engineering & Technology

Approved by AICTE, New Delhi, * Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA, Accredited by NAAC (A+) with CGPA of 3.4
Recognized by UGC under Section 2(f) and 12(B) of UGC Act 1956
Aditya Nagar, ADB Road, Surampalem

Department of Information Technology

Vision

To be a department with high repute and focused on quality education

Mission

- To Provide an environment for the development of professionals with knowledge and skills
- To promote innovative learning
- To promote innovative ideas towards society
- To foster trainings with institutional collaborations
- To involve in the development of software applications for societal needs

[illegible]

Head of the Department

Head of the Department
Dept.of IT
Aditya College of Engineering & Technology
SURAMPALEM 533 437

[illegible]

Principal

PRINCIPAL
Aditya College of
Engineering & Technology
SURAMPALEM

Aditya College of Engineering & Technology

Approved by AICTE, New Delhi, * Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA, Accredited by NAAC (A+) with CGPA of 3.4
Recognized by UGC under Section 2(f) and 12(B) of UGC Act 1956
Aditya Nagar, ADB Road, Surampalem

Department of Information Technology

Program Educational Objectives

Program educational objectives are broad statements that describe the career and professional accomplishments that the program is preparing graduates to achieve.

PEO-1:

Graduates will be skilled in Mathematics, Science & modern engineering tools to solve real life problems.

PEO-2:

Excel in the IT industry with the attained knowledge and skills or pursue higher studies to acquire emerging technologies and become an entrepreneur.

PEO-3:

Accomplish a successful career and nurture as a responsible professional with ethics and human values.

[illegible]

Head of the Department

Head of the Department
Dept.of IT
Aditya College of Engineering & Technology
SURAMPALEM - 533 437

[illegible]

Principal

PRINCIPAL
Aditya College of
Engineering & Technology
SURAMPALEM

Aditya College of Engineering & Technology

Approved by AICTE, New Delhi, * Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA, Accredited by NAAC (A+) with CGPA of 3.4
Recognized by UGC under Section 2(f) and 12(B) of UGC Act 1956
Aditya Nagar, ADB Road, Surampalem

Department of Information Technology

Program Outcomes

1. Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design / Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct Investigations of Complex Problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-Long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

[illegible]

Head of the Department

Head of the Department
Dept.of IT
itya/College of Engineering & Technology
SURAMPALEM 533 437

[illegible]

Principal

PRINCIPAL
Aditya College of
Engineering & Techno
SURAMPALEM

Aditya College of Engineering & Technology

Approved by AICTE, New Delhi, * Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA, Accredited by NAAC (A+) with CGPA of 3.4
Recognized by UGC under Section 2(f) and 12(B) of UGC Act 1956
Aditya Nagar, ADB Road, Surampalem

Department of Information Technology

Program Specific Outcomes

PSO-1:

Apply mathematical foundations, algorithmic and latest computing tools and techniques to design computer-based systems to solve engineering problems.

PSO-2:

Apply knowledge of engineering and develop software-based applications for research and development in the areas of relevance under realistic constraints.

PSO-3:

Apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product.

[illegible]

Head of the Department

Head of the Department
Dept.of IT
Aditya College of Engineering & Technology
SURAMPALEM 533 437

[illegible]

Principal

PRINCIPAL
Aditya College of
Engineering & Technology
SURAMPALEM

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

Surampalem, Andhra Pradesh.

DEPARTMENT OF INFORMATION TECHNOLOGY

Course Outcomes (COs)

Course Name:	Project work	**Year**	IV
Faculty Name:	K Naga Bhargavi	**Semester**	II
Academic year:	2022-23	**Regulation**	R19

Course Outcomes

After completing this course, the student will be able to:

CO Number	**CO Statement**	**Taxonomy**
CO1	**Apply** fundamental and disciplinary concepts and methods in ways appropriate to their principal areas of study.	Apply
CO2	**Demonstrate** skill and knowledge of current information and technological tools and techniques specific to the professional field of study.	Apply
CO3	**Effectively** communicate with engineers and the community at large in written, oral and visual forms.	Apply
CO4	**Apply** the theoretical knowledge to identify, analyze and solve industrial problems with team work and multidisciplinary approach.	Apply
CO5	**Demonstrate** professionalism with ethics and relate engineering issues to broader societal context.	Apply
CO6	**Practice** the skills to excel and engage in lifelong learning.	Apply

Signature of the Guide

ABSTRACT

Agriculture is one of the important professions in many countries including India. As most part of the Indian financial system is dependent on agriculture production, the keen attention to the concern of food production is necessary. The taxonomy and identification of crop infection got much importance in technical as wellas economic in the Agricultural Industry. Keeping track of diseases in plants with the help of specialists is very costly in agriculture region. There is a need for a system which can automatically detect the diseases as it can bring revolution in monitoring large fields of crop and then plant leaves can be taken cure as soon as possible after detection of disease.

The use of deep learning models to identify lessions on cotton leaves on the basis of images of the crop in the field. Its cultivation in tropical regions has made it the target of a wide spectrum of agricultural pests and diseases, and efficient solutionsare required. Moreover, the symptoms of the main pests and diseases cannot be differentiated in the initial stages, and the correct identification of a lesion can be difficult for the producer. To help resolve the problem, the present research provides a solution based on deep learning Automatic classifier CNN will be used for classification based on learning with some training samples. The CNN model is used to predict the plant is diseased or not based on spot on the leaves. Finally CNN provides minimum error and better accuracy in identification of cotton plant disease.

INDEX

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 Introduction

India is an agricultural country as per Observation in India Most of the people depend upon agriculture. Farmers have a good range of multiplicity to pick suitable crops for his or her farm. However, the cultivation of those crops for optimum yield and quality produce is usually technical. The disease diagnosis is restricted by human visual capabilities because most of the primary symptoms are microscopic. This process is tedious, time-consuming. Nowadays within the area of research, a major concern is an identification of the symptoms of the disease by means of image processing. The farmers are struggling during their lifestyle for a way to affect the disease of the cotton leaf. There is a requirement for a disease diagnosis system which will support farmers. This technique focuses on disease identification by processing acquired digital images of leaves of the plant.

Plant disease detection is a significant challenge in the agriculture sector. Some of the plants show visible symptoms on the plant leaf. These leaf patterns can be used to identify different diseases and take immediate action to prevent the spread. Most of these plant diseases are difficult to detect through naked eyes, and even experienced persons end up wrong. The accuracy of the manual prediction depends upon the experience and knowledge of the person. There are much research works done for plant leaf disease identification using machine learning and deep learning models. This study proposes a deep learning model to classify different plant diseases. In standard machinelearning models, the features are extracted manually, and the algorithm learns from that data. Thus, it is a two-step process. The deep learning model uses an artificial neural network that can learn features from an input image and make intelligent decisions on its own. Thus, deep learning models are far more capable than the standard machine learning models for image-based classification.

The goal of this application is to develop a system which recognizes crop diseases. In this the user has to upload an image on the system, Image processing starts with the digitized color image of the diseased leaf. Finally, by applying the CNN plantdisease can be predicted. A. Purpose of Proposed System:

1. Developing a user-friendly web-based system for farmers
2. Recognizing Cotton leaf diseases accurately from input images

For this project, I have created deep learning model convolutional neural network(CNN) using keras library for our project cotton disease prediction. First of all, why I am using CNN because CNNs are used for image classification and recognition because of its high accuracy. The CNN follows a hierarchical model which works on building a network, like a funnel, and finally gives out a fully connected layer where all the neurons are connected to each other and the output is processed.

In recent times, the sophisticated emerging technology has attracted many researchers in the field of detection and classification of cotton leaf diseases and pests. In Ethiopia, there are several constraints which reduce the yield and quality of the product. Particularly, identification of potential diseases or pests on Ethiopian cotton isbased on traditional ways. There is a wide area of farm suitable for cotton plantation, but only limited research attention is given to cotton crop production. Traditionally, experts detect and identify such plant diseases and pests on bared eyes. Bared eye determination is considered as a loss of low-level accuracy in order to detect any diseases. On high demand, different advanced technologies were aided for structuring the systems to assist nonautomatic recognition of the plant diseases and pests to increasethe accuracy for any corrective measures. With the help of advanced technologies, the plant diseases were reduced, thus increasing the productivity which helped to raise the economy via boosting the production. For that reason, the implementation of information technology-based solutions in the sector of agriculture had high level of significance for Ethiopia’s development in monetary, community, and eco-friendly developments by increased cotton crops’ productions.

Day by day, all over the world, agriculture land is going to be reduced because the population is increasing rapidly and lack of water resources. Disease in the plant isone of those hazards that have to be examined at this stage. In contrast, the isolation ofplants from their natural environment is being happened, and they are grown in unusual conditions. Many valuable crops and plants are very vulnerable to disease. They would have a great struggle to survive in nature without human involvement. Yield loss in harvests is regularly connected with plant illness or factors, such as climate, water availability, and supplement accessibility. To improve the productivity of the crop, environmental factors or product resources such as temperature and humidity are important. An important role is played by the root exudates of the plant, which helps in improving the nutrients of the soil. Compared to their wild relations, cultivated plants are always more flexible to diseases. This is the large numbers of the same species or different kinds, having a similar gene grown together, sometimes over many kilometers of distances. am using CNN because CNNs are used for image classification and recognition because of its high accuracy. The CNN follows a hierarchical model which works on building anetwork, like a funnel, and finally gives out a fully connected layer where all the neurons are connected to each other and the output is processed.

In recent times, the sophisticated emerging technology has attracted many researchers in the field of detection and classification of cotton leaf diseases and pests. In Ethiopia, there are several constraints which reduce the yield and quality of the product. Traditionally, experts detect and identify such plant diseases and pests on bared eyes. Bared eye determination is considered as a loss of low-level accuracy in order to detect any diseases. On high demand, different advanced technologies were aided for structuring the systems to assist non automatic recognition of the plant diseases and pests to increasethe accuracy for any corrective measures. With the help of advanced technologies, the plant diseases were reduced, thus increasing the productivity which helped to raise the economy via boosting the production. For that reason, the implementation of information technology-based solutions in the sector of agriculture had high level of significance for Ethiopia’s development in monetary, community, and eco-friendly developments by increased cotton crops’ production. This developed model is implemented using python version 3.7.3 and the model is equipped on the deep learning package called Keras, TensorFlow backed, and Jupyterwhich are used as the developmental environment. This model achieved an accuracy of96.2% for identifying leaf disease in cotton plants. This revealed the feasibility of its usage in real-time applications and the potential need for ITbasedsolutions to support traditional or manual disease identification.

1.2 Purpose of the System

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. CNN is a deep learning algorithm, by using this algorithm we are detecting the Cotton Plant diseases. A good DL algorithm results better predictions.

1.3 Scope of the System

Based on the diseases observed in paddy crop a model is generated. The aim is to apply this model and predict the diseases of paddy crop and recommend suitable measures.Code is generated and made the model more accurate for predicting the disease.

1.4 Existing System

In the past days, farmers usually detect crop diseases with their naked eye. It requires detailed knowledge about the types of diseases and lot of experience needed to make sure the actual disease detection. Some of the diseases look almost similar to farmers often leaves them confused. To prevent this situation, we need better and perfect guidance on which fertilizers to use, to make the correct identification of diseases.

In the existing system we are having old mechanisms, so it makes less accuracy on the outcome.

1.5 Proposed System

A Convolutional Neural network (CNN) is a type of Artificial Neural Network, which is widely used for image/object recognition and classification. The goal of this model is to detect paddy crop diseases by using user input images. Here the CNN model is used to detect the diseases in paddy crops.

The proposed system applies the concept of classification learning, that is implemented through deep learning algorithm. Through this CNN algorithm this model produces the output with high accuracy rate.

CHAPTER 2

REQUIREMENTS ANALYSIS

2. REQUIREMENT ANALYSIS

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. Requirements analysis is an important aspect of project management. Requirements can be architectural, structural, behavioural, functional and non-functional.

2.1 Functional Requirements

• Ease of Use: To promote efficiency and productivity in the development process.

• High Performance: To support real-time, large scale simulation.

• Maintainability: To minimize life-cycle cost.

• Scalability: To support new requirements and ever increasing scope.

• Scalability: To support new requirements and ever increasing scope.

• Cross-Platform Support: To support multiple operating systems to enable maximum selection of available hardware.

• pen Standards Support: To gain maximum leverage from existing and futuresimulation of code.

2.2 Non-Functional Requirements

A non-functional requirement is a specification that describes the system's operation capabilities and constraints that enhance its functionality. We've covered different types of software requirements, based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system.

• User Interface and Human Factor

• Software Requirements

• Hardware Requirements

• Usability

• Reliability

• Performance

• Physical Environment

• Resource Requirements

• Security Requirements

2.2.1 User Interface and Human Factor

User interface design focuses on the following key areas:

• The design of interfaces between different software components.

• The design of interfaces between the software and other nonhuman producers and consumers of information.

• The design of the interface between a human and the computer.

2.2.2 Software requirements

The Software requirements of the system are mentioned below:

Operating System: Windows 8 or newer

Software: Goolge Colab

Language Used: Python

Libraries/Packages: Tensorflow,numpy,matplotlib.

2.2.3 Hardware Requirements

The Hardware requirements of the system are mentioned below:

CPU: Intel Core i3-8100

GPU: RX 570 (or GTX 1060 3GB)

RAM: 8GB DDR4 CPU

Cooler: Stock (or Deepcool Gammaxx 400)

Operating System: Windows 10

2.2.4 Usability

This system is used for continuously monitoring the images of the cotton leaves and predict whether the plant is diseased or not which helps to make better decisions for its management.

2.2.5 Reliability

This model produces the high accuracy output than any other model.

2.2.6 Performance

As it makes use of different models that help us to make the system having high performance output.

2.2.7 Physical Environment

It is compatible with any system like a laptop & system.

2.2.8 Resource Requirements

Required software is to be installed like python,Google Colab, and respected libraries are to be installed in our system to execute our project successfully.

2.2.9 Security Requirements

It is responsible for taking data, such as user preferences and descriptions of the items that can be recommended.

CHAPTER 3

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 Introduction

System analysis is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvements on the system. System analysis is a problem-solving activity that requires intensive communication between the system users and system developers. System analysis is an important phase of any system development process. The system is studied to the minutest detail and analysed. The system analyst plays the role of an interrogator and dwells deep into the working of the present system.

The system is viewed as a whole and the inputs to the system are identified. The outputs from the organization are traced through the various processing that the inputs phase through in the organization. A detailed study of these processes is made by various techniques like Interviews, reviews, etc.

The data collected by these sources is scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system.

UML DIAGRAMS

➢ Unified Modeling Language is the language used to visualize, specify, construct and document any component of software engineering.

➢ The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

3.2 Use cases

3.2.1 Actors

➢ Developer

➢ User

3.2.2 List of Use case Developer

➢ Change user interface design

➢ Add, update, delete Interface **User**

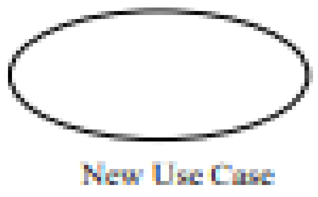
➢ Experience the Virtual Environment.

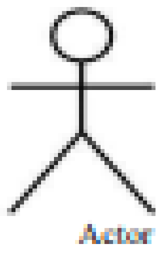
➢ View individual products.

3.2.3 Use case Diagrams

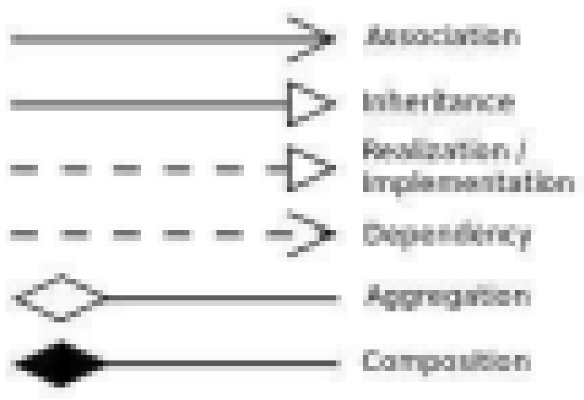
Use case diagram is created to visualize the relationships between actors and use cases. A use case is a pattern of behaviour the system exhibits. Each use case is a sequence of related transactions performed by an actor and the system. Diagrammatically actor and use case are represented by stick figure and oval respectively.

Symbols





RelationShips



A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

UML is the modeling toolkit that you can use to build your diagrams. Use cases are represented with a labeled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modeled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself.

• Use cases: Horizontally shaped ovals that represent the different uses that a user might have.

• Actors: Stick figures that represent the people actually employing the use cases.

• Associations: A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.

• System boundary boxes: A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.

• Packages: A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.

• Actors: The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data

. • System: A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.

• Goals: The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

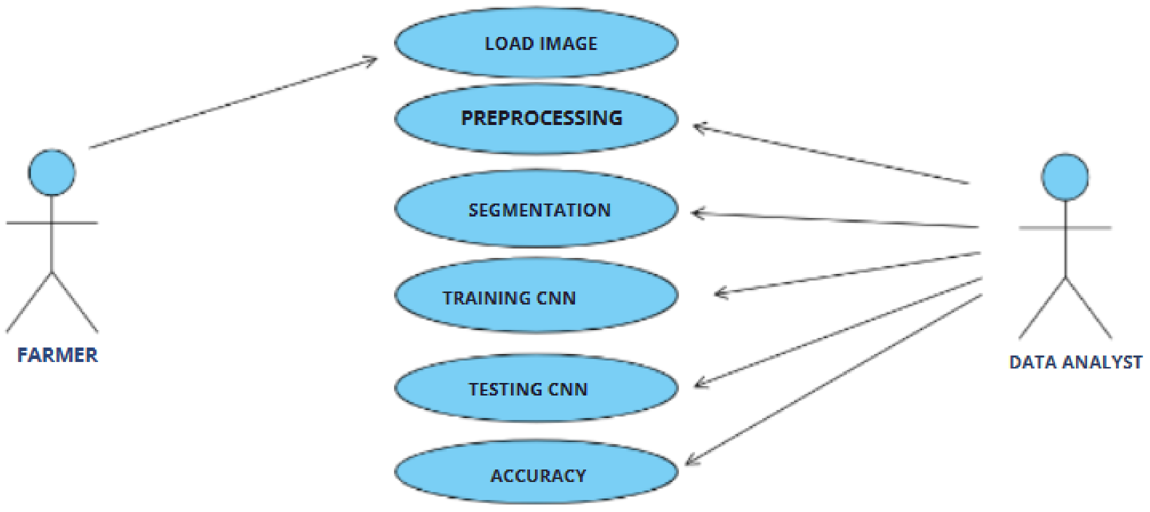


Fig no.3.2.3.1 Use case diagram

CHAPTER 4

SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 Introduction

An effective system development life cycle (SDLC) should result in a high qualitysystem that meets customer expectations, reaches completion within time and cost evaluations, and works effectively and efficiently in the current and planned informationtechnology infrastructure.

System Development Life Cycle (SDLC) is a conceptual model which includes policies and procedures for developing or altering systems throughout their life cycles.

System design is the process of defining the architecture, models, interfaces, anddata for a system to satisfy specified requirements. System design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of system analysis, system architecture and system engineering.

The design phase describes how the system will fulfil the user requirements. To achieve this, we must create both logical and physical design. In this phase the system design functions and operations are described.

4.2 System Architecture

A system architecture or systems architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description and representation of a system, organized in a way that supports reasoning about structures and behaviour of a system.

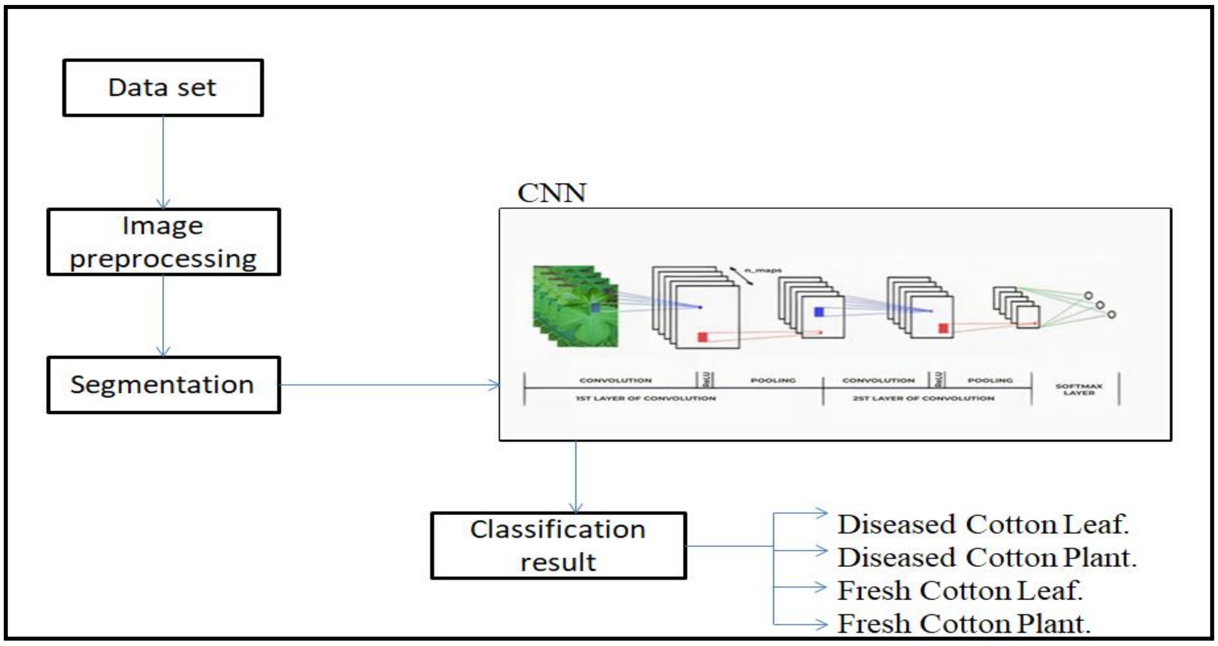


Fig: 4.2.1 System Architecture

4.3 System Object Model

4.3.1 Introduction

System object model (SOM) is an object- oriented library packaging technology developed by IBM that allows various programming languages to share class libraries, regardless of the language in which they were originally written. System object model was intended to be used as a solution to many of the interoperability and reuse problems that occur while sharing class libraries between object-oriented and non-object-oriented languages.

SOM was designed to be used across IBM' s mainframe computers and desktops. It serves as an object-oriented model that can be distinguished from other models contained in object-oriented programming languages. SOM basically includes an interface definition a runtime environment with procedure calls and a set of enabling frameworks.

4.3.2 Subsystems

Modules

- Google Colab: Used to create and run the system application.
- User: To calculate the parametric models.

.4.4 Object Description

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of a system at a particular moment. Object diagrams are used to render a set of objects and their relationships as an instance.

4.4.1 Objects

Object is any entity that can be manipulated by the commands of programming languages such as value, variable, function, or data structure.

4.4.2 Class Diagrams

A class diagram showing the systems classes, their attributes, operations, and collaborations and the relationships among objects. The class diagrams are the main building lock of object-oriented modelling. Class diagrams can also be used for data modelling.

The classes in a class diagram represent both the main elements, interactions in the application and the classes to be programmed. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system.

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with objectoriented languages and thus widely used at the time of construction

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

The purpose of the class diagram can be summarized as –

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

Class diagrams are the most popular UML diagrams used for construction of software applications.

It is very important to learn the drawing procedure of class diagram. Class diagrams have a lot of properties to consider while drawing but here the diagram will be considered from a top-level view.

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams represents the whole system.

The following points should be remembered while drawing a class diagram –

- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified
- For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
- Use notes whenever required to describe some aspect of the diagram. At the end of the drawing, it should be understandable to the developer/coder.
- Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct.

4.5 Dynamic Model

4.5.1 State Chart Diagram

An activity diagram is another important diagram in UML to describe the dynamic aspects of the system. An activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc. The basic purposes of activity diagrams are similar to the other four diagrams. It captures the dynamic behaviour of the system. Other four diagrams are used to show the message flow from one object to another but the activity diagram is used to show message flow from one activity to another.

State Chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State Chart diagrams are useful to model the reactive.

systems. Reactive systems can be defined as a system that responds to external or internal events.

State Chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State Chart diagram is to model lifetime of an object from creation to termination. State Chart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using State Chart diagrams –

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object

State Chart diagram defines the states of a component and these state changes are dynamic in nature. Its specific purpose is to define the state changes triggered by events. Events are internal or external factors influencing the system.

State Chart diagrams are used to model the states and also the events operating on the system. When implementing a system, it is very important to clarify different states of an object during its life time and State Chart diagrams are used for this purpose. When these states and events are identified, they are used to model it and these models are used during the implementation of the system.

If we look into the practical implementation of State Chart diagram, then it is mainly used to analyze the object states influenced by events. This analysis is helpful to understand the system behaviour during its execution.

The main usage can be described as –

- To model the object states of a system.
- To model the reactive system. Reactive system consists of reactive objects.
- To identify the events responsible for state changes.

- Forward and reverse engineering State Chart diagram is used to describe the states of different objects in its life cycle. Emphasis is placed on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately. State Chart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.

Before drawing a State Chart diagram we should clarify the following points –

- Identify the important objects to be analyzed.
- Identify the states.
- Identify the event

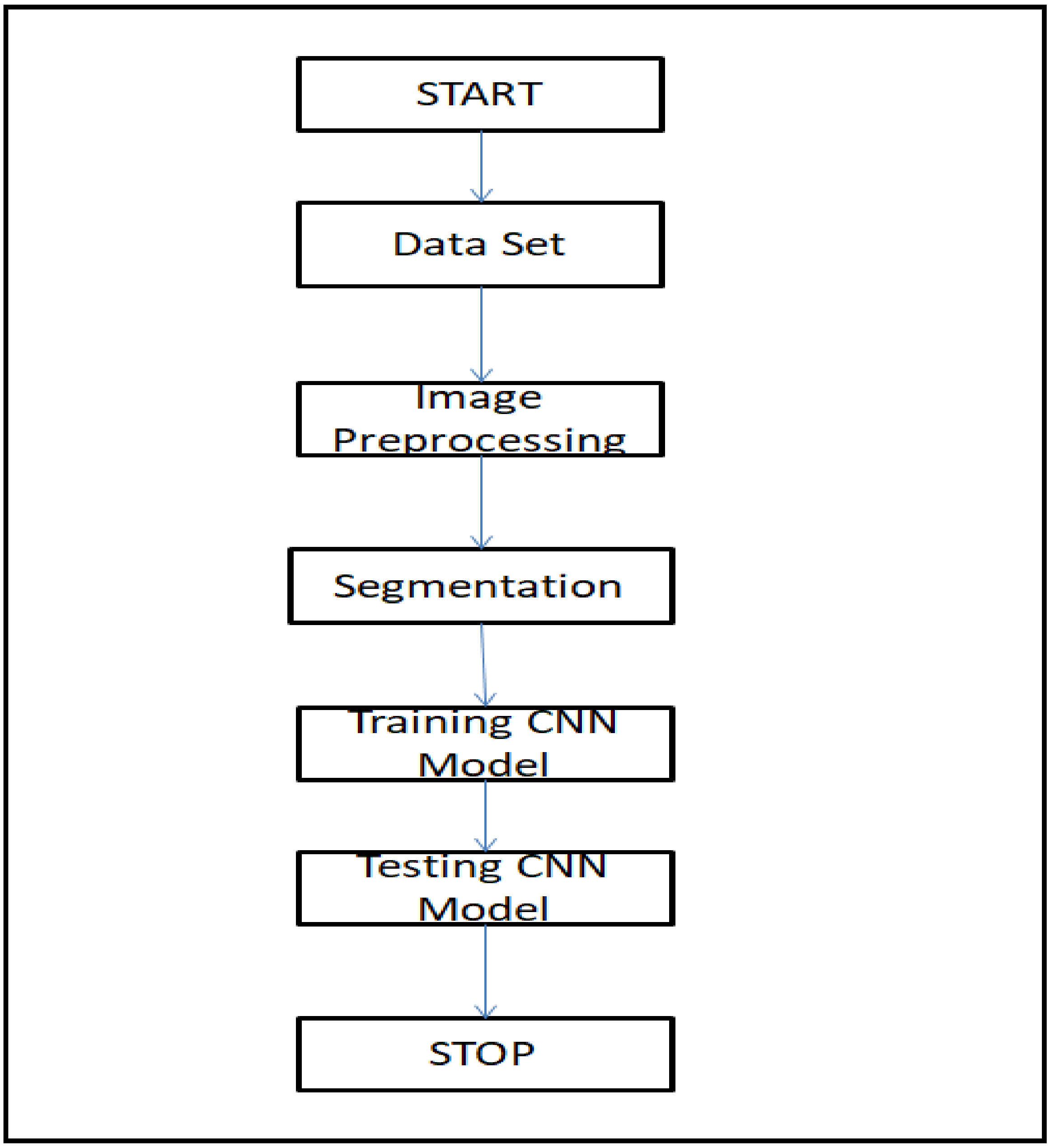


Fig. 4.5.1.1 State Chart Diagram

4.5.2 Sequence diagram

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. A sequence diagram shows object interactions arranged in time sequence. The interaction that takes place in collaboration that either realizes a use case or an operation. Sequence diagrams provide high level interactions between user of the system and the system, between the system and the other systems, or between subsystems.

Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.

The following scenarios are ideal for using a sequence diagram:

- Usage scenario: A usage scenario is a diagram of how your system could potentially be used. It's a great way to make sure that you have worked through the logic of every usage scenario for the system.
- Method logic: Just as you might use a UML sequence diagram to explore the logic of a use case, you can use it to explore the logic of any function, procedure, or complex process.
- Service logic: If you consider a service to be a high-level method used by different clients, a sequence diagram is an ideal way to map that out.
- Sequence diagram Visio - Any sequence diagram that you create with Visio can also be uploaded into Lucidchart. Lucidchart supports .vsd and .vdx file import and is a great Microsoft Visio alternative. Almost all of the images you see in the UML section of this site were generated using Lucidchart.

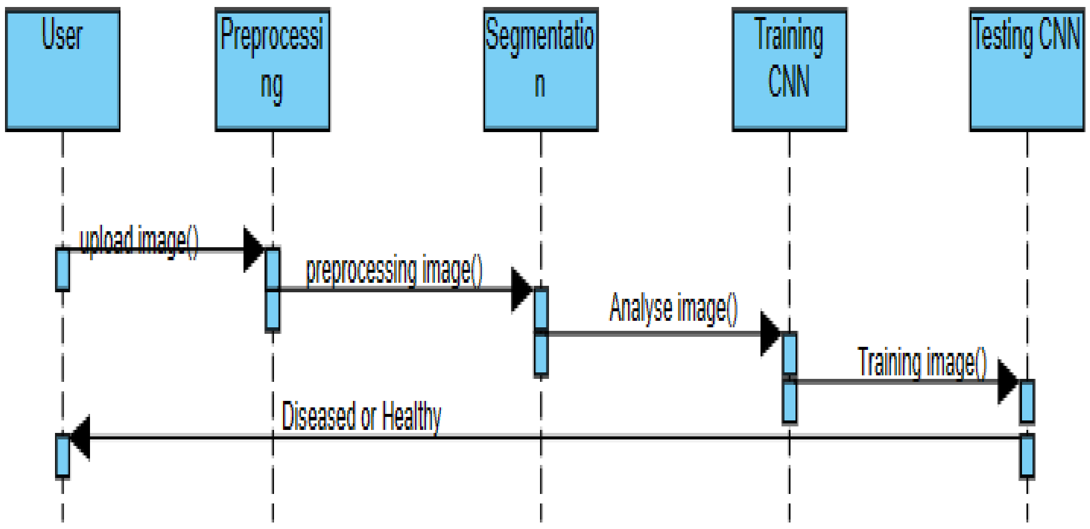


Fig 4.5.2.1 sequence diagram

4.5.3 Communication Diagram

The communication diagram and the sequence diagram are similar. They're semantically equivalent, that is, they present the same information, and you can turn a communication to a sequence diagram and vice versa. The main distinction between them is that the communication diagram arranges elements according to space, the sequence diagram is according to time.

Communication diagrams offer benefits similar to sequence diagrams, but they will offer a better understanding of how components communicate and interact with each other rather than solely emphasizing the sequence of events. They can be a useful reference for businesses, organizations, and engineers who need to visualize and understand the physical communications within a program. Try drawing a sequence diagram to:

- Model the logic of a sophisticated procedure, function, or operation.
- Identify how commands are sent and received between objects or components of a process.
- Visualize the consequences of specific interactions between various components in a process.
- Plan and understand the detailed functionality of an existing or future scenario.

A communication diagram offers the same information as a sequence diagram, but while a sequence diagram emphasizes the time and order of events, a communication diagram emphasizes the messages exchanged between objects in an application. Sequence diagrams can fall short of offering the "big picture.”

This is where communication diagrams come in and offer that broader perspective within a process.

4.6 Static Model

4.6.1 Component Diagram

UML Component diagrams are used in modelling the physical aspects of object oriented systems that are used for visualizing, specifying, and documenting componentbased systems and also for constructing executable systems through forward and reverse engineering. Component diagram is a special kind of diagram in UML

The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities. Thus from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment. A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

The purpose of the component diagram can be summarized as –

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

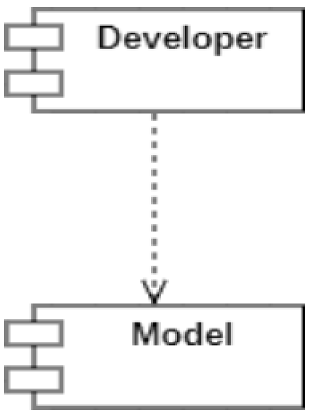


Fig. 4.6.1.1 Component Diagram

4.6.2 Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

Deployment diagrams are important for visualizing, specifying, and documenting embedded, client/server, and distributed systems and also for managing executable systems through forward and reverse engineering.

A deployment diagram is just a special kind of class diagram, which focuses on a system's nodes. Graphically, a deployment diagram is a collection of vertices and arcs. Deployment diagrams commonly contain: 3-D box represents a node, either software or hardware HW node can be signified with <> Connections between nodes are represented with a line, with optional <> Nodes can reside within a node.

A UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modelling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).

Deployment diagrams are important for visualizing, specifying, and documenting embedded, client/server, and distributed systems and also for managing executable systems through forward and reverse engineering. A deployment diagram is just a special kind of class diagram, which focuses on a system's nodes. Graphically, a deployment diagram is a collection of vertices and arcs.

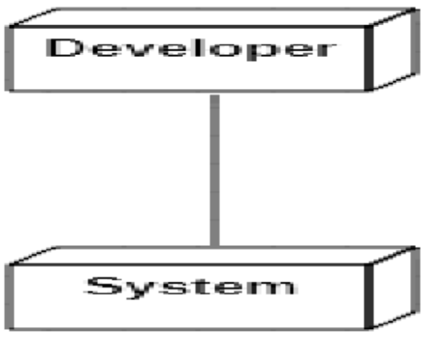


Fig. 4.6.2.1 Deployment Diagram

CHAPTER 5
IMPLEMENTATION

5. IMPLEMENTATION

5.1 Software used

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, makes it an ideal language for scripting and rapid application development.

Python supports multiple programming pattern, including object oriented, imperative and functional or procedural programming styles.

Python is not intended to work on special area such as web programming. That is why it is known as multipurpose 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.

Python 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 History

- Python laid its foundation in the late 1980s.
- The implementation of Python was started in the December 1989 by **Guido Van Rossum** at CWI in Netherland.
- In February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources.
- In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.

- *Python 2.0 added new features like: list comprehensions, garbage collection system.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
- ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
- Python is influenced by following programming languages:
    - ABC language.
    - Modula-3

Python Features

Python provides lots of features that are listed below.

1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

3) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

5) Free and Open Source

Python language is freely available at offical web address.The source-code is also available. Therefore it is open source.

6) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

8) Large Standard Library

Python has a large and broad library and prvides rich set of module and functions for rapid application development.

9) GUI Programming Support

Graphical user interfaces can be developed using Python.

10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

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 specifing applications areas where python can be applied.

1) 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 delelop web based applications. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc.

2) 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.

3) 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.

4) 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.

5) Business Applications

Python is used to build Bussiness applications like ERP and e-commerce systems. Tryton is a high level application platform.

6) Console Based Application

We can use Python to develop console based applications. For example: **IPython**.

7) 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.

8) 3D CAD Applications

To create CAD application Fandango is a real application which provides full features of CAD.

9) 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.

10) Applications for Images

Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc.

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.

Why Python

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Installation on Windows

Visit the link *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.

Virtual Environments and Packages

Introduction

Python applications will often use packages and modules that don't come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library's interface.

This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

The solution for this problem is to create a virtual environment, a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages. Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0. If application B requires a library be upgraded to version 3.0, this will not affect application A's environment.

Creating Virtual Environments

The module used to create and manage virtual environments is called venv. venv will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running python3 or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the venv module as a script with the directory path:

```
python3 -m venv tutorial-env
```

This will create the tutorial-env directory if it doesn’t exist, and also create directories inside it containing a copy of the Python interpreter, the standard library, and various supporting files.

A common directory location for a virtual environment is .venv. This name keeps the directory typically hidden in your shell and thus out of the way while giving it a name that explains why the directory exists. It also prevents clashing with .env environment variable definition files that some tooling supports.

Once you’ve created a virtual environment, you may activate it.

On Windows, run:

```
tutorial-env\Scripts\activate.bat
```

On Unix or MacOS, run:

```
source tutorial-env/bin/activate
```

(This script is written for the bash shell. If you use the csh or fish shells, there are alternate activate.csh and activate.fish scripts you should use instead.)

Activating the virtual environment will change your shell’s prompt to show what virtual environment you’re using, and modify the environment so that running python will get you that particular version and installation of Python. For example:

```
$ source ~/envs/tutorial-env/bin/activate

(tutorial-env) $ python

Python 3.5.1 (default, May  6 2016, 10:59:36)

...
```

```
>>> import sys

>>>sys.path

['', '/usr/local/lib/python35.zip', ...,

'~/envs/tutorial-env/lib/python3.5/site-packages']

>>>
```

Managing Packages with pip

You can install, upgrade, and remove packages using a program called pip. By default pip will install packages from the Python Package Index, <https://pypi.org>. You can browse the Python Package Index by going to it in your web browser, or you can use pip's limited search feature:

```
(tutorial-env) $ pip search astronomy
skyfield             - Elegant astronomy for Python
gary                 - Galactic astronomy and gravitational dynamics.
novas                - The United States Naval Observatory NOVAS astronomy library
astroobs             - Provides astronomy ephemeris to plan telescope observations
PyAstronomy          - A collection of astronomy related tools for Python.
```

pip has a number of subcommands: "search", "install", "uninstall", "freeze", etc. (Consult the Installing Python Modules guide for complete documentation for pip.)

You can install the latest version of a package by specifying a package's name:

```
(tutorial-env) $ pip install novas
Collecting novas
Downloading novas-3.1.1.3.tar.gz (136kB)
Installing collected packages: novas
Running setup.py install for novas
Successfully installed novas-3.1.1.3
```

You can also install a specific version of a package by giving the package name followed by == and the version number:

```
(tutorial-env) $ pip install requests==2.6.0
Collecting requests==2.6.0
Using cached requests-2.6.0-py2.py3-none-any.whl
Installing collected packages: requests
Successfully installed requests-2.6.0
```

If you re-run this command, pip will notice that the requested version is already installed and do nothing. You can supply a different version number to get that version, or you can run pip install --upgrade to upgrade the package to the latest version:

```
(tutorial-env) $ pip install --upgrade requests
Collecting requests
Installing collected packages: requests
Found existing installation: requests 2.6.0
Uninstalling requests-2.6.0:
Successfully uninstalled requests-2.6.0
Successfully installed requests-2.7.0
```

pip uninstall followed by one or more package names will remove the packages from the virtual environment.

pip show will display information about a particular package:

```
(tutorial-env) $ pip show requests
---
Metadata-Version: 2.0
Name: requests
Version: 2.7.0
Summary: Python HTTP for Humans.
Home-page: http://python-requests.org
Author: Kenneth Reitz
Author-email: me@kennethreitz.com
License: Apache 2.0
Location: /Users/akuchling/envs/tutorial-env/lib/python3.4/site-packages
Requires:
```

pip list will display all of the packages installed in the virtual environment:

```
(tutorial-env) $ pip list
novas (3.1.1.3)
numpy (1.9.2)
pip (7.0.3)
requests (2.7.0)
setuptools (16.0)
```

pip freeze will produce a similar list of the installed packages, but the output uses the format that pip install expects. A common convention is to put this list in a requirements.txt file:

```
(tutorial-env) $ pip freeze > requirements.txt
(tutorial-env) $ cat requirements.txt
novas==3.1.1.3
numpy==1.9.2
requests==2.7.0
```

The requirements.txt can then be committed to version control and shipped as part of an application. Users can then install all the necessary packages with install -r:

```
(tutorial-env) $ pip install -r requirements.txt
Collecting novas==3.1.1.3 (from -r requirements.txt (line 1))
  ...
Collecting numpy==1.9.2 (from -r requirements.txt (line 2))
  ...
Collecting requests==2.7.0 (from -r requirements.txt (line 3))
Installing collected packages: novas, numpy, requests
Running setup.py install for novas
Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0
```

pip has many more options. Consult the Installing Python Modules guide for complete documentation for pip. When you’ve written a package and want to make it available on the Python Package Index, consult the Distributing Python Modules guide.

WHAT IS DEEP LEARNING

Deep learning is one of the foundations of artificial intelligence (AI), and the current interest in deep learning is due in part to the buzz surrounding AI. Deep learning techniques have improved the ability to classify, recognize, detect and describe – in one word, understand [13]. For example, deep learning is used to classify images, recognize speech, detect objects and describe content.

Several developments are now advancing deep learning:

Algorithmic improvements have boosted the performance of deep learning methods.

New machine learning approaches have improved accuracy of models.

New classes of neural networks have been developed that fit well for applications like text translation and image classification.

We have a lot more data available to build neural networks with many deep layers, including streaming data from the Internet of Things, textual data from social media, physicians notes and investigative transcripts [14].

Computational advances of distributed cloud computing and graphics processing units have put incredible computing power at our disposal. This level of computing power is necessary to train deep algorithms.

At the same time, human-to-machine interfaces have evolved greatly as well. The mouse and the keyboard are being replaced with gesture, swipe, touch and natural language, ushering in a renewed interest in AI and deep learning .

How Deep Learning Works Deep learning changes how you think about representing the problems that you're solving with analytics. It moves from telling the computer how to solve a problem to training the computer to solve the problem itself.

A traditional approach to analytics is to use the data at hand to engineer features to derive new variables, then select an analytic model and finally estimate the parameters (or the unknowns) of that model. These techniques can yield predictive systems that do not generalize well because completeness and correctness depend on the quality of the model and its features . For example, if you develop a fraud model with feature engineering, you start with a set of variables, and you most likely derive a model from those variables using data transformations. You may end up with 30,000 variables that your model depends on, then you have to shape the model, figure out which variables are meaningful, which ones are not, and so on. Adding more data requires you to do it all over again.

The new approach with deep learning is to replace the formulation and specification of the model with hierarchical characterizations (or layers) that learn to recognize latent features of the data from the regularities in the layers . The paradigm shift with deep learning is a move from feature engineering to feature representation. The promise of deep learning is that it can lead to predictive systems that generalize well, adapt well, continuously improve as new data arrives, and are more dynamic than predictive systems built on hard business rules. You no longer fit a model. Instead, you train the task.

Deep learning is making a big impact across industries. In life sciences, deep learning can be used for advanced image analysis, research, drug discovery, prediction of health problems and disease symptoms, and the acceleration of insights from genomic sequencing. In transportation, it can help autonomous vehicles adapt to changing conditions]. It is also used to protect critical infrastructure and speed response.

How Deep Learning Being Used

To the outside eye, deep learning may appear to be in a research phase as computer science researchers and data scientists continue to test its capabilities. However, deep learning has many practical applications that businesses are using today, and many more that will be used as research continues . Popular uses today include:

Speech Recognition

Both the business and academic worlds have embraced deep learning for speech recognition. Xbox, Skype, Google Now and Apple’s Siri, to name a few, are already employing deep learning technologies in their systems to recognize human speech and voice patterns.

Natural Language Processing

Neural networks, a central component of deep learning, have been used to process and analyse written text for many years. A specialization of text mining, this technique can be used to discover patterns in customer complaints, physician notes or news reports, to name a few.

Image Recognition

One practical application of image recognition is automatic image captioning and scene description. This could be crucial in law enforcement investigations for identifying criminal activity in thousands of photos submitted by bystanders in a crowded area where a crime has occurred. Self-driving cars will also benefit from image recognition through the use of 360- degree camera technology.

Recommendation Systems

Amazon and Netflix have popularized the notion of a recommendation system with a good chance of knowing what you might be interested in next, based on past behaviour. Deep learning can be used to enhance recommendations in complex environments such as music interests or clothing preferences across multiple platforms.

Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks like classifying objects in images [20]. While deep learning was first theorized in the 1980s, there are two main reasons it has only recently become useful:

1. Deep learning requires large amounts of labelled data. For example, driverless car development requires millions of images and thousands of hours of video.

2. Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning.

When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less. When choosing between machine learning and deep learning, consider whether you have a high-performance GPU and lots of labelled data. If you don’t have either of those things, it may make more sense to use machine learning instead of deep learning. Deep learning is generally more complex, so you’ll need at least a few thousand images to get reliable results. Having a high-performance GPU means the model will take less time to analyse all those images [21]

Deep Learning Opportunities and Applications

A lot of computational power is needed to solve deep learning problems because of the iterative nature of deep learning algorithms, their complexity as the number of layers increase, and the large volumes of data needed to train the networks.

The dynamic nature of deep learning methods – their ability to continuously improve and adapt to changes in the underlying information pattern – presents a great opportunity to introduce more dynamic behaviour into analytics [22]. Greater personalization of customer analytics is one possibility. Another great opportunity is to improve accuracy and performance in applications where neural networks have been used for a long time. Through better algorithms and more computing power, we can add greater depth.

While the current market focus of deep learning techniques is in applications of cognitive computing, there is also great potential in more traditional analytics applications, for example, time series analysis. Another opportunity is to simply be more efficient and streamlined in existing analytical operations. Recently, some study showed that with deep neural networks in speech-to-text transcription problems [23]. Compared to the standard techniques, the worderror-rate decreased by more than 10 percent when deep neural networks were applied. They also eliminated about 10 steps of data preprocessing, feature engineering and modelling. The impressive performance gains and the time savings when compared to feature engineering signify a paradigm shift.

Here are some examples of deep learning applications are used in different industries:

Automated Driving:

Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.

Aerospace and Defence:

Deep learning is used to identify objects from satellites that locate areas of interest, and identify safe or unsafe zones for troops.

Medical Research:

Cancer researchers are using deep learning to automatically detect cancer cells. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to accurately identify cancer cells [24].

Industrial Automation:

Deep learning is helping to improve worker safety around heavy machinery by automatically detecting when people or objects are within an unsafe distance of machines. **Electronics:**

Deep learning is being used in automated hearing and speech translation. For example, home assistance devices that respond to your voice and know your preferences are powered by deep learning applications.

5.2 Source Code

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

```
! ls -hl
```

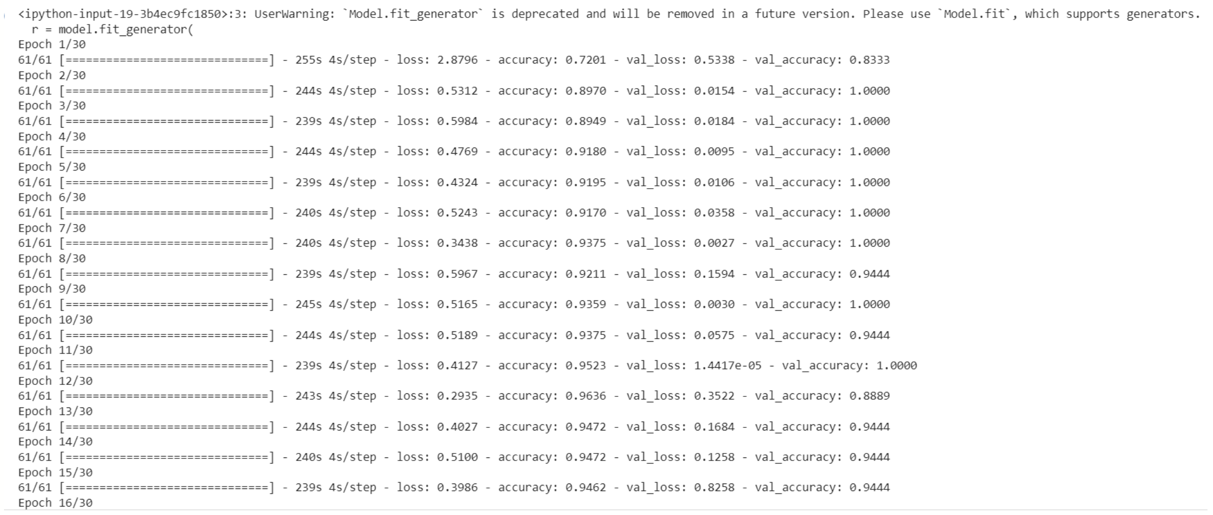
```
! unzip data-20230219T141116Z-001.zip
```

```
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
config = ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.5
config.gpu_options.allow_growth = True
session = InteractiveSession(config=config)


# import the libraries as shown below

from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
#from keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.inception_v3 import preprocess_in
put
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
import matplotlib.pyplot as plt



# re-size all the images to this
IMAGE_SIZE = [224, 224]

train_path = 'data/train'
valid_path = 'data/test
# Use the Image Data Generator to import the images from the dataset
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)




test_datagen = ImageDataGenerator(rescale = 1./255)
```

```
# Make sure you provide the same target size as initialied for the i
mage size
training_set = train_datagen.flow_from_directory('data/train',
                                                 target_size = (224, 224),
                                                  batch_size = 32,


class_mode = 'categorical')

for my_batch in training_set:
  # my_batch is a tuple with images and labels
  images = my_batch[0]
  labels = my_batch[1]
  for i in range(len(labels)):
    # Gives oe image and its corresponding label
    plt.imshow(images[i])
    plt.colorbar()
    plt.show()
    print(labels[i])

  break


test_set = test_datagen.flow_from_directory('data/test',
                                            target_size = (224, 224),
                                             batch_size = 32,
                                          class_mode = 'categorical')


for my_batch in test_set:
  # my_batch is a tuple with images and labels
  images = my_batch[0]
  labels = my_batch[1]
  for i in range(len(labels)):
    # Gives oe image and its corresponding label
    plt.imshow(images[i])
    plt.colorbar()
    plt.show()
    print(labels[i])

  break
```

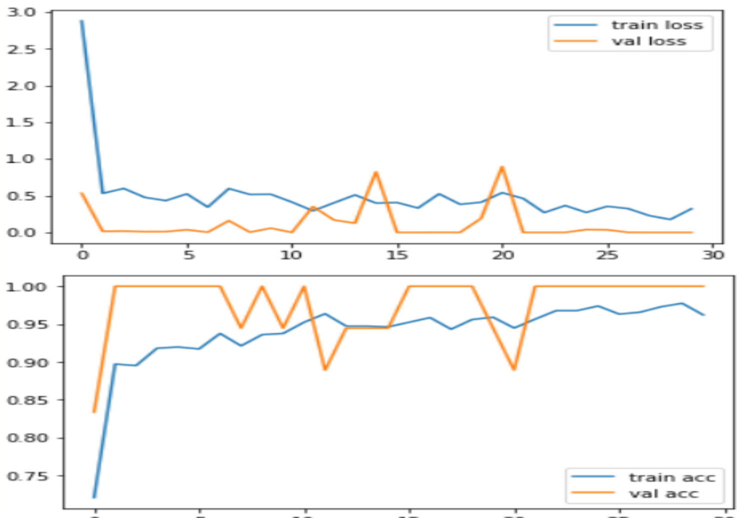
```
# Import the Vgg 16 library as shown below and add preprocessing lay
er to the front of VGG
# Here we will be using imagenet weights

inception = InceptionV3(input_shape=IMAGE_SIZE + [3], weights='image
net', include_top=False)




# don't train existing weights
for layer in inception.layers:
    layer.trainable = False


 # useful for getting number of output classes
folders = glob('data/train/*')


# our layers - you can add more if you want
x = Flatten()(inception.output)


prediction = Dense(len(folders), activation='softmax')(x)

# create a model object
model = Model(inputs=inception.input, outputs=predictio


prediction = Dense(len(folders), activation='softmax')(x)

# create a model object
model = Model(inputs=inception.input, outputs=predictio


# view the structure of the model
model.summary()


# tell the model what cost and optimization method to use
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

```
# fit the model
# Run the cell. It will take some time to execute
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=30,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```



```
<ipython-input-19-3b4ec9fc1850>:3: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  r = model.fit_generator(
Epoch 1/30
61/61 [==============================] - 255s 4s/step - loss: 2.8796 - accuracy: 0.7201 - val_loss: 0.5338 - val_accuracy: 0.8333
Epoch 2/30
61/61 [==============================] - 244s 4s/step - loss: 0.5312 - accuracy: 0.8970 - val_loss: 0.0154 - val_accuracy: 1.0000
Epoch 3/30
61/61 [==============================] - 239s 4s/step - loss: 0.5984 - accuracy: 0.8949 - val_loss: 0.0184 - val_accuracy: 1.0000
Epoch 4/30
61/61 [==============================] - 244s 4s/step - loss: 0.4769 - accuracy: 0.9180 - val_loss: 0.0095 - val_accuracy: 1.0000
Epoch 5/30
61/61 [==============================] - 239s 4s/step - loss: 0.4324 - accuracy: 0.9195 - val_loss: 0.0106 - val_accuracy: 1.0000
Epoch 6/30
61/61 [==============================] - 240s 4s/step - loss: 0.5243 - accuracy: 0.9170 - val_loss: 0.0358 - val_accuracy: 1.0000
Epoch 7/30
61/61 [==============================] - 240s 4s/step - loss: 0.3438 - accuracy: 0.9375 - val_loss: 0.0027 - val_accuracy: 1.0000
Epoch 8/30
61/61 [==============================] - 239s 4s/step - loss: 0.5967 - accuracy: 0.9211 - val_loss: 0.1594 - val_accuracy: 0.9444
Epoch 9/30
61/61 [==============================] - 245s 4s/step - loss: 0.5165 - accuracy: 0.9359 - val_loss: 0.0030 - val_accuracy: 1.0000
Epoch 10/30
61/61 [==============================] - 244s 4s/step - loss: 0.5189 - accuracy: 0.9375 - val_loss: 0.0575 - val_accuracy: 0.9444
Epoch 11/30
61/61 [==============================] - 239s 4s/step - loss: 0.4127 - accuracy: 0.9523 - val_loss: 1.4417e-05 - val_accuracy: 1.0000
Epoch 12/30
61/61 [==============================] - 243s 4s/step - loss: 0.2935 - accuracy: 0.9636 - val_loss: 0.3522 - val_accuracy: 0.8889
Epoch 13/30
61/61 [==============================] - 244s 4s/step - loss: 0.4027 - accuracy: 0.9472 - val_loss: 0.1684 - val_accuracy: 0.9444
Epoch 14/30
61/61 [==============================] - 240s 4s/step - loss: 0.5100 - accuracy: 0.9472 - val_loss: 0.1258 - val_accuracy: 0.9444
Epoch 15/30
61/61 [==============================] - 239s 4s/step - loss: 0.3986 - accuracy: 0.9462 - val_loss: 0.8258 - val_accuracy: 0.9444
Epoch 16/30
```

```
# plot the loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```



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

```
print("train loss:", r.history['loss'][-1])
print("val loss:", r.history['loss'][-1])


print("train accuracy:", r.history['accuracy'][-1])
print("val accuracy:", r.history['accuracy'][-1])


# save it as a h5 file


from tensorflow.keras.models import load_model

model.save('model_inception.h5')


y_pred = model.predict(test_set)


y_pred


import numpy as np
y_pred = np.argmax(y_pred, axis=1)


y_pred
```

CHAPTER 6
TESTING

6.TESTING

6.1 INTRODUCTION

In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information.

System Testing and Implementation

The purpose is to exercise the different parts of the module code to detect coding errors. After this the modules are gradually integrated into subsystems, which are then integrated themselves too eventually forming the entire system. During integration of module integration testing is performed. The goal of this is to detect designing errors, while focusing the interconnection between modules. After the system was put together, system testing is performed. Here the system is tested against the system requirements to see if all requirements were met and the system performs as specified by the requirements. Finally accepting testing is performed to demonstrate to the client for the operation of the system.

For the testing to be successful, proper selection of the test case is essential. There are two different approaches for selecting test case. The software or the module to be tested is treated as a black box, and the test cases are decided based on the specifications of the system or module. For this reason, this form of testing is also called “black box testing”.

The focus here is on testing the external behavior of the system. In structural testing the test cases are decided based on the logic of the module to be tested. A common approach here is to achieve some type of coverage of the statements in the code. The two forms of testing are complementary: one tests the external behavior, the other tests the internal structure.

Testing is an extremely critical and time-consuming activity. It requires proper planning of the overall testing process. Frequently the testing process starts with the test plan.

This plan identifies all testing related activities that must be performed and specifies the schedule, allocates the resources, and specifies guidelines for testing. The test plan specifies conditions that should be tested; different units to be tested, and the manner in which the module will be integrated together. Then for different test unit, a test case specification document is produced, which lists all the different test cases, together with the expected outputs, that will be used for testing. During the testing of the unit the specified test cases are executed and the actual results are compared with the expected outputs. The final output of the testing phase is the testing report and the error report, or a set of such reports. Each test report contains a set of test cases and the result of executing the code with the test cases. The error report describes the errors encountered and the action taken to remove the error.

6.2 Testing Techniques

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of conditions known as test cases and the output is evaluated to determine whether the program is performing as expected. In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

Black Box Testing

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structure or external database access
- Performance errors
- Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

White Box Testing

In this testing, the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

1. Guarantee that all independent paths have been executed.
2. Execute all logical decisions on their true and false sides
3. Execute all loops at their boundaries and within their operational
4. Execute internal data structures to ensure their validity.

6.1.2 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.

This System consists of 3 modules. Those are Reputation module, route discovery module, audit module. Each module is taken as unit and tested. Identified errors are corrected and executable unit are obtained.

Integration testing

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

System Testing

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.

Functional Testing

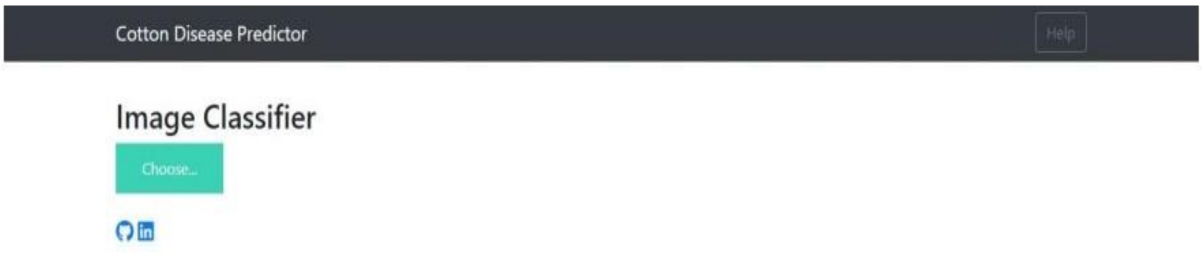
Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

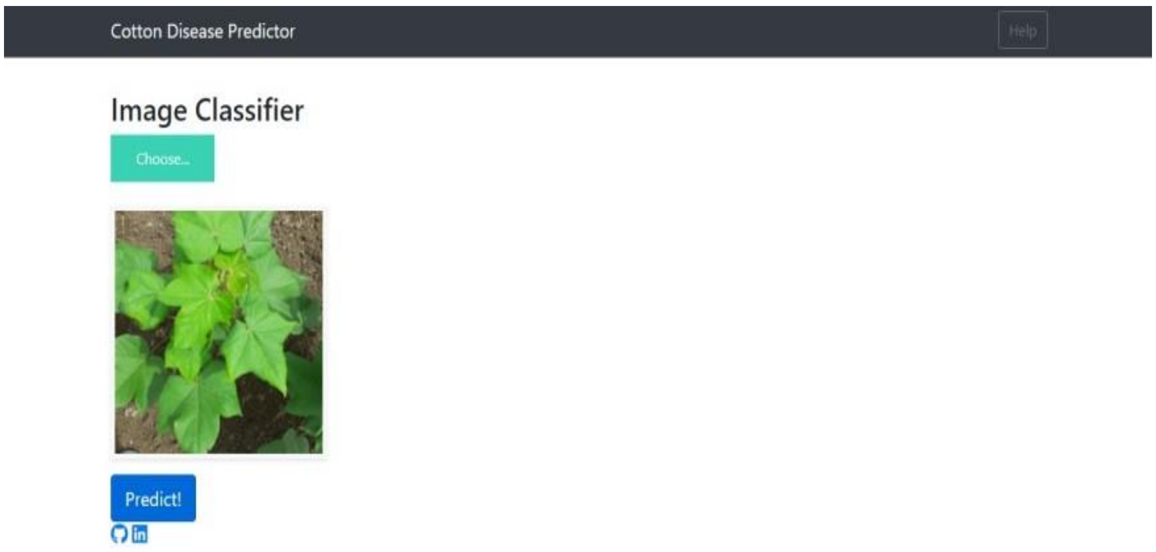
Functional testing is centered on the following items

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.

CHAPTER 7
OUTPUT SCREENS

7. OUTPUT SCREENS





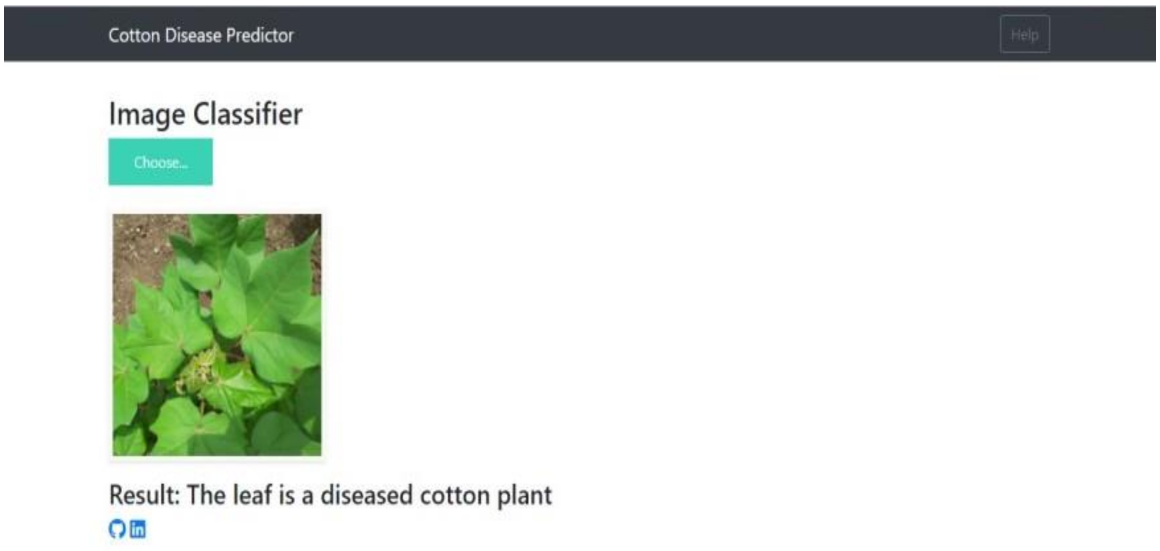


Image Classifier

Result: The leaf is a diseased cotton plant

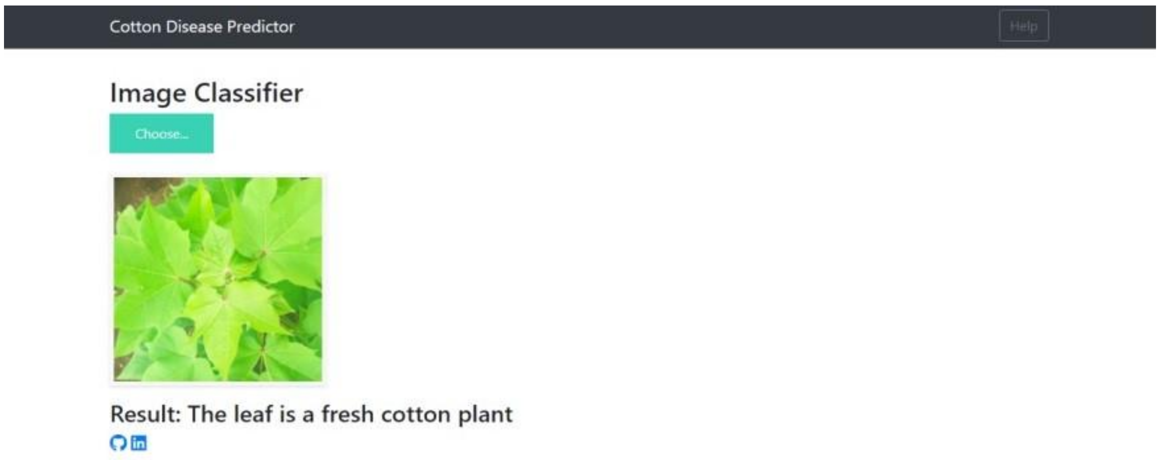


Image Classifier

Result: The leaf is a fresh cotton plant

CHAPTER 8

CONCLUSION

8.CONCLUSION

To improve the production and yield of the cotton plant, it is necessary to detect plant diseases. It is better if disease detection is done at an early stage. Deep learning plays an essential role in image processing to detect infected cotton leaves. We have developed a deep learning model using inception v3 that is used to classify the diseased and healthy leaf of the cotton plant. The model classifies the diseased and healthy leaf with 96% accuracy. With this technique, the plant disease can be recognized at the initial stage and using the pest control tools problem can be solved while minimizing risk to people as well as to the environment.

CHAPTER 9

FUTURE SCOPE

9.FUTURE SCOPE

The plant diseases may vary depending on the crop grown. The future Extension of this work is to further study plant diseases and classify them based on the type of the disease. Analyzing and comparing different models is to be performed for improving the classification performance.

Working on different crop data sets and plant diseases using deep learning models help farmers to monitor large fields of crops efficiently and earn profits. The future enhancements can include upgrading user interface and the accuracy to detect diseases.

CHAPTER 10

BIBLIOGRAPHY

10. BIBLIOGRAPHY

[1] Umit Atilaa, Murat Uçarb, Kemal Akyolc, and Emine Uçar, "Plant leaf disease classification using Efficient Net deep learning model," Ecological Informatics, vol.61, Article 101182, March 2021

[2] M.Akila and P.Deepan, "Detection and Classification of Plant Leaf Diseases by using Deep Learning Algorithm," International Journal of Engineering Research & Technology (IJERT), SSN: 2278-0181, 2018

[3] Pranita P. Gulve, Sharayu S. Tambe, Madhu A. Pandey, Mrs S.S.Kanse, "Leaf Disease Detection of Cotton Plant Using Image Processing Technique", IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) e-ISSN: 2278-2834,p-ISSN: 2278-8735. PP 50-54 ,2015

[4] Vijay S. Bhong, B.V. Pawar,"Study and Analysis of Cotton Leaf Disease Detection Using Image Processing", International Journal of Advanced Research in Science, Engineering and Technology, ISSN: 2350-0328, Vol. 3, Issue 2,February 2016

[5] RakeshChaware, RohitKarpe, PrithviPakhale, Prof.SmitaDesai, "Detection and Recognition of Leaf Disease Using Image Processing", International Journal of Engineering Science and Computing, May 2017

[6] Srdjan Sladojevic, Marko Arsenovic, Andras Anderla, Dubravko Culibrk, and Darko Stefanovic, "Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification," Hindawi Publishing Corporation Computational Intelligence and Neuroscience, vol. 2016, Article ID 3289801, 2016

[7] Ethiopian Institute of Agricultural Research, Cotton Research Strategy, EIAR, Addis Ababa, Ethiopia, 2017.

[8] Dunne, R.; Desai, D.; Sadiku, R.; Jayaramudu, J. A review of natural fibers, their sustainability and automotive applications. J. Reinf. Plast. Compos. 2016, 35, 1041–1050. [CrossRef]

[9] Afzal, M.N.; Tariq, M.; Ahmed, M.; Abbas, G.; Mehmood, Z. Managing Planting Time for Cotton Production; Ahmad, S., Hasanuzzaman, M., Eds.; Cotton Production and Uses; Springer: Singapore, 2020. [Cross]