Plant Disease Detection Using Deep Learning

DISSERTATION

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

of

Bachelor of Technology

in

Computer Science & Engineering

By:

Gurman Singh (111/CSE2/2019)
Harneet Singh (103/CSE2/2019)
Kuldeep Singh (117/CSE2/2019)
Sahibjot Singh (065/CSE2/2019)

Under the guidance of:

Ms. Jyotsna

Department of Computer Science & Engineering
Guru Tegh Bahadur Institute of Technology

Guru Gobind Singh Indraprastha University
Dwarka, New Delhi
Year 2022-2023

DECLARATION

We hereby declare that all the work presented in the dissertation entitled **"Plant Disease Detection using Deep Learning"** in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **Computer Science & Engineering**, Guru Tegh Bahadur Institute of Technology, affiliated to Guru Gobind Singh Indraprastha University Delhi is an authentic record of our work carried out under the guidance of **Ms**. **Jyotsna**

Date: 25 May 2023

Gurman Singh (111/CSE2/2019)
Harneet Singh (103/CSE2/2019)
Kuldeep Singh (117/CSE2/2019)
Sahibjot Singh (065/CSE2/2019)

CERTIFICATE

This is to certify that the dissertation entitled **"Plant Disease Detection using Deep Learning",** which is submitted by **Mr. Gurman Singh, Mr. Harneet Singh, Mr. Kuldeep Singh, and Mr. Sahibjot Singh** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **Computer Science & Engineering**, Guru Tegh Bahadur Institute of Technology, New Delhi is an authentic record of the candidate's work carried out by them under our guidance. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Ms. Jyotsna
(Project Guide)

Dr. Aashish Bhardwaj
(Head of Department)
Computer Science & Engineering

Date: 25 May 2023

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude and appreciation to all those who have contributed to the successful completion of our dissertation. First and foremost, we would like to extend my deepest gratitude to our dissertation guide, **Ms. Jyotsna**. Your expertise, patience, and unwavering commitment to excellence have been invaluable throughout this journey. Your guidance and insightful feedback have helped shape my research, and we are immensely grateful for your mentorship.

We would also like to express our sincere thanks to the Head of the Department, **Dr. Aashish Bhardwaj**, for providing us with the necessary resources and facilities to conduct our research. Your support and encouragement have been instrumental in the successful completion of this dissertation. We also take this opportunity to give thanks to all others who gave us support for the project or in other aspects of our study at Guru Tegh Bahadur Institute of Technology.

Date: 15 May 2023

Gurman Singh
(111/CSE2/2019)
gurman29singh@gmail.com

Harneet Singh
(103/CSE2/2019)
harneetsinghch110@gmail.com

Kuldeep Singh
(117/CSE2/2019)
kuldeepsingh2670@gmail.com

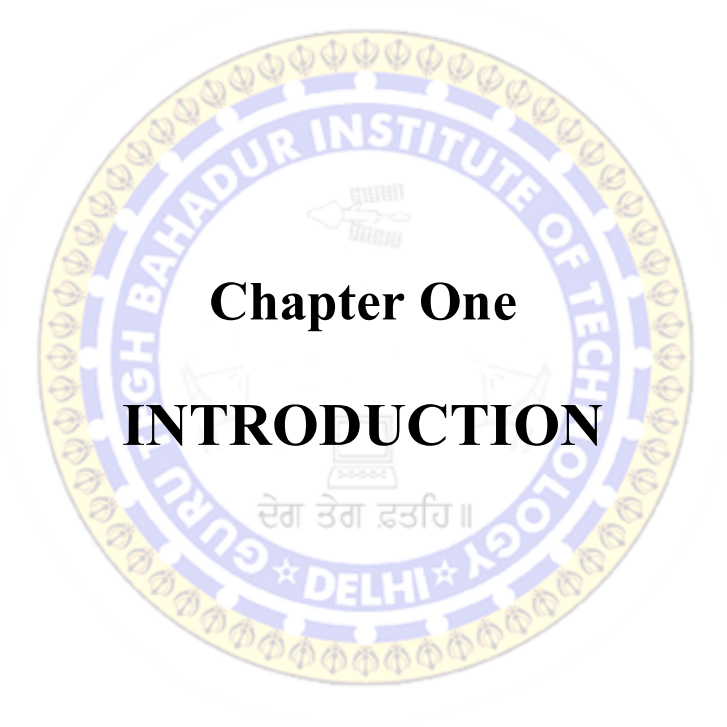
Sahibjot Singh
(065/CSE2/2019)
sahibjot7180@gmail.com

ABSTRACT

Machine learning, Deep learning, and Artificial intelligence are the Future. We use these technologies in almost every field. In the Farming sector, we can also use this technology for the Preparation of soil, adding fertilizers, sowing of seed, Irrigation, weed protection, harvesting, disease prediction, etc. We are using Deep Learning for Plant disease detection based on images of a leaf of a plant. We are using deep learning for this task because here we are working with image data. Deep learning has a Convolution neural network that is used to find features from the leaf of the plant. Researchers have been involved in finding an efficient way to make better predictions od disease, therefore we have discussed a few methods to achieve good results. We have used deep neural networks and machine learning techniques to build a good model. We have used Plant Village Dataset. In this data-set, 39 different classes of plant leaf and background images are available. The data-set containing 61,486 images. We used six different augmentation techniques for increasing the data-set size. The techniques are image flipping, Gamma correction, noise injection, PCA color augmentation, rotation, and Scaling. There is a total of 39 Classes that we have to predict using the CNN Model. Basically, first we Resize every image into 224 x 224. After that this image feed into the Convolutional Neural Network. We feed color image so it has 3 channels RGB. First conv layer we apply 32 filter size or output channels. That means 32 different filters apply to the images and try to find features and after that using 32 features, we create a features map that has channels 32. So from 3 x 224 x 224 it will become 32 x 222 x 222. After that we are applying ReLU activation function to remove non linearity and after that we are applying Batch Normalization to normalize the weights of the neuron. After that this image we feed to the max pool layer which takes only the most relevant features only so that why we get the output image in shape 32 x 112 x 112. After that, we feed this image to the next convolutional layer and its process is the same as mentioned above. At last, we flatten the final max pool layer output and feed to the next linear layer which is also called a fully connected layer, and finally, as a final layer, we predict 39 categories. So as a model output, we get tensor 1x39 size. And from that tensor, we take an index of the maximum value in the tensor. That particular index is our main prediction.

LIST OF FIGURES AND TABLES

CONTENTS



Chapter One

INTRODUCTION

Introduction

The health and productivity of agricultural crops are vital for ensuring global food security and sustainable development. However, the prevalence of plant diseases poses significant threats to crop yields, leading to substantial economic losses for farmers worldwide. Detecting plant diseases in a timely and accurate manner is crucial for effective disease management and minimizing crop damage. Therefore, this project aims to contribute to the field of plant pathology by developing a "Plant Disease Detection" system capable of efficiently identifying diseases affecting various crops.

The proposed Plant Disease Detection system will leverage Convolutional Neural Networks (CNNs) techniques to analyze images of plant leaves or other affected parts. By training the system on the "Plant Village Dataset," which consists of 39 different classes of plant leaf and background images, it will acquire the ability to accurately identify the presence of diseases or abnormalities. The focus of the project will be on detecting common and economically significant plant diseases, including powdery mildew, leaf spot, blight, and rust.

To ensure widespread accessibility and reach to farmers, the project will include the implementation of a user-friendly website and a mobile application. These platforms will enable farmers to easily capture images of plant leaves or affected areas using their smartphones and upload them for disease diagnosis. The website and mobile app will provide a seamless user experience, guiding farmers through the image capture process and providing instant feedback on the presence of diseases. Additionally, the system will offer detailed information on disease management strategies, recommended treatments, and preventive measures, empowering farmers with valuable knowledge to effectively address plant diseases.

By harnessing the power of machine learning, along with the implementation of a website and mobile app, this project seeks to provide an efficient and accurate solution for detecting and diagnosing plant diseases. The successful implementation of the Plant Disease Detection system, utilizing the Plant Village Dataset and user-friendly interfaces, will contribute to improved crop management practices, increased

agricultural productivity, and ultimately, a more sustainable future for the farming community.

The benefits of this project extend beyond disease detection alone. By enabling early intervention and targeted disease management, farmers will be able to mitigate the impact of diseases, reducing crop losses and optimizing resource utilization. This, in turn, will lead to enhanced agricultural productivity and food security on a global scale.

The development of the Plant Disease Detection system, utilizing machine learning techniques, the Plant Village Dataset, and the implementation of a website and mobile app, represents a significant step towards efficient disease management in agriculture. By empowering farmers with timely and accurate disease identification through easily accessible platforms, this project aims to foster sustainable crop management practices, increase agricultural productivity, and pave the way for a more resilient and prosperous future in the farming community.

1.1 Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

The Python 2 language, i.e. Python 2.7.x, is "sunsetting" on January 1, 2020 (after extension; first planned for 2015), and the Python team of volunteers will not fix

security issues or improve it in other ways after that date. With the end-of-life, only Python 3.5.x and later will be supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open-source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

Python's large standard library[120] provides tools suited to many tasks and is commonly cited as one of its greatest strengths. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary-precision decimals,[121] manipulating regular expressions, and unit testing.

Some parts of the standard library are covered by specifications—for example, the Web Server Gateway Interface (WSGI) implementation wsgiref follows PEP 333[122]—but most are specified by their code, internal documentation, and test suites. However, because most of the standard library is cross-platform Python code, only a few modules need altering or rewriting for variant implementations.

As of 14 November 2022, the Python Package Index (PyPI), the official repository for third-party Python software, contains over 415,000[123] packages with a wide range of functionality, including:

- Automation
- Data Analytics
- Databases
- Documentation
- Graphical user interfaces
- Image processing
- Machine learning
- Mobile apps
- Multimedia
- Computer networking
- Scientific computing

- System Administration
- Test frameworks
- Text processing
- Web frameworks
- Web scraping

Most Python implementations (including CPython) include a read–eval–print loop (REPL), permitting them to function as a command line interpreter for which users enter statements sequentially and receive results immediately.

Python also comes with an Integrated development environment (IDE) called IDLE, which is more beginner-oriented.

Other shells, including IDLE and IPython, add further abilities such as improved auto-completion, session state retention, and syntax highlighting. As well as standard desktop integrated development environments, there are Web browser-based IDEs, including SageMath, for developing science- and math-related programs.

1.1.1 FastAPI

FastAPI is a modern, fast (high-performance), a web framework for building APIs with Python 3.7+ based on standard Python type hints. It is built on top of Starlette for the web parts and Pydantic for the request/response parts. One of the key features of FastAPI is its use of type hints to automatically generate web-based API documentation, as well as automatically validate incoming and outgoing data. This means that you can use FastAPI to build an API with a minimal amount of code, and with a clear separation of the documentation and the implementation.

FastAPI is also fully asynchronous, meaning that it can take advantage of the async/await syntax introduced in Python 3.5 to perform non-blocking I/O operations. This can make your API much more efficient, especially when handling a large number of concurrent requests. In addition to its automatic documentation and validation, FastAPI also includes support for OpenAPI and JSON Schema. This means that you can use FastAPI to build an API that can be easily consumed by a wide range of clients, including those built using Swagger or Postman.

FastAPI is easy to get started with and includes several useful features out of the box. For example, it includes support for dependency injection, which allows you to easily share common functionality between different parts of your API. It also includes support for automatic routing, which means that you can define your API endpoints using simple function decorators. Overall, FastAPI is a powerful and easy-to-use tool for building APIs with Python. Its combination of automatic documentation and validation, async support, and OpenAPI/JSON Schema integration make it a great choice for developers looking to build high-performance, well-documented APIs quickly and easily.

There are several advantages to using FastAPI for building APIs:

1. Fast performance: FastAPI is built on top of Starlette, which is a high-performance ASGI (Asynchronous Server Gateway Interface) framework. This means that FastAPI can handle a large number of concurrent requests efficiently.
2. Automatic documentation: FastAPI uses Python-type hints to automatically generate web-based API documentation. This means that you don't have to write separate documentation for your API, and can focus on writing code.
3. Automatic data validation: FastAPI uses Pydantic to automatically validate incoming and outgoing data. This can help to prevent errors and improve the reliability of your API.
4. Async support: FastAPI is fully asynchronous, which means that it can take advantage of the async/await syntax introduced in Python 3.5 to perform non-blocking I/O operations. This can make your API much more efficient, especially when handling a large number of concurrent requests.
5. OpenAPI/JSON Schema integration: FastAPI includes support for OpenAPI and JSON Schema, which means that your API can be easily consumed by a wide range of clients, including those built using Swagger or Postman.
6. Easy to use: FastAPI is easy to get started with and includes several useful features out of the box, such as dependency injection and automatic routing. This makes it a great choice for developers looking to build APIs quickly and easily.

1.1.2 OpenCV

OpenCV (Open Source Computer Vision) is a powerful and widely used open-source library for computer vision and image processing tasks. With its comprehensive collection of functions and algorithms, OpenCV provides a robust framework for developing applications in various domains, including robotics, augmented reality, object detection, facial recognition, and medical imaging, among others. In this article, we will explore the key features, capabilities, and applications of OpenCV, as well as delve into some of its core functionalities and use cases.

Introduction to OpenCV: OpenCV was initially developed by Intel in 1999 and later became an open-source project. Since then, it has gained immense popularity and has been widely adopted by researchers, developers, and enthusiasts worldwide. The library is written in C++ and provides interfaces for several programming languages, including Python, Java, and MATLAB, making it accessible to a diverse range of developers.

Key Features and Capabilities: OpenCV offers a comprehensive set of functions and algorithms that enable developers to perform a wide range of computer vision tasks. Some of the key features and capabilities of OpenCV include:

1. Image and Video Processing: OpenCV provides a rich set of functions for reading, manipulating, and processing images and videos. It offers support for various image formats and allows operations such as resizing, cropping, rotation, filtering, and blending. Additionally, OpenCV provides capabilities for video capture, playback, and analysis.
2. Feature Detection and Extraction: OpenCV includes algorithms for detecting and extracting features from images, such as corners, edges, and keypoints. These features can be used for tasks like object recognition, image stitching, and motion tracking.
3. Object Detection and Tracking: OpenCV provides pre-trained models and algorithms for object detection and tracking. These algorithms can be used to identify and track objects in real-time video streams or static images. Popular techniques such as Haar cascades and deep learning-based methods like YOLO (You Only Look Once) are supported.
4. Machine Learning and Deep Learning Integration: OpenCV seamlessly integrates with popular machine learning and deep learning frameworks like

TensorFlow, PyTorch, and Caffe. This allows developers to leverage the power of machine learning and deep learning algorithms for tasks like image classification, object recognition, and semantic segmentation.

5. Camera Calibration and 3D Reconstruction: OpenCV includes functions for camera calibration, which is essential for obtaining accurate measurements from images. It also provides tools for 3D reconstruction from multiple images, enabling the creation of 3D models from 2D images.
6. Image Stitching and Panorama Generation: OpenCV offers functionalities for stitching multiple images together to create panoramic images. It utilizes techniques like feature detection, image registration, and blending to seamlessly merge overlapping images.
7. Facial Recognition: OpenCV provides tools for facial detection and recognition. It includes pre-trained models for face detection and offers methods for facial landmark detection, emotion recognition, and face identification.

Applications of OpenCV: The versatility of OpenCV makes it applicable to a wide range of domains and industries. Some notable applications of OpenCV include:

1. Robotics: OpenCV is extensively used in robotics for tasks like object recognition, obstacle avoidance, visual navigation, and robotic perception.
2. Augmented Reality (AR): OpenCV is a key component in AR applications that involve real-time object tracking, marker detection, and virtual object placement in the real world.
3. Medical Imaging: OpenCV is utilized in medical imaging applications for tasks like tumor detection, organ segmentation, and image analysis for diagnostic purposes.
4. Autonomous Vehicles: OpenCV plays a vital role in autonomous driving systems by enabling tasks such as lane detection, traffic sign recognition, pedestrian detection, and object tracking.
5. Security and Surveillance: OpenCV is used in security and surveillance systems for tasks like face recognition, motion.

1.1.3 PyTorch

PyTorch is a popular open-source deep learning framework that has gained significant traction among researchers and developers in recent years. Built on top of the Torch library, PyTorch provides a dynamic and intuitive interface for building and training neural networks. With its ease of use, flexibility, and powerful capabilities, PyTorch has become a preferred choice for deep learning practitioners. In this article, we will explore the key features, advantages, and applications of PyTorch, as well as delve into its core components and ecosystem.

Introduction to PyTorch: PyTorch was developed by Facebook's AI Research lab and was released as an open-source project in 2016. Since then, it has gained widespread adoption and has become one of the most popular frameworks for deep learning. PyTorch combines the flexibility of Python programming with the computational efficiency of low-level frameworks, making it an ideal choice for both research and production environments.

Key Features and Advantages of PyTorch:

1. Dynamic Computational Graph: PyTorch's defining feature is its dynamic computational graph, which allows for dynamic creation and modification of the computational graph during runtime. This flexibility enables developers to write and debug code more easily, as they can use standard Python control flow statements and debug interactively.
2. Automatic Differentiation: PyTorch provides automatic differentiation, a powerful technique that enables efficient computation of gradients for backpropagation. With PyTorch, developers can define complex neural network architectures and compute gradients with minimal effort, making it easier to train deep learning models.
3. GPU Acceleration: PyTorch supports seamless integration with GPUs, enabling high-performance training and inference on parallel hardware. By leveraging GPUs, PyTorch can significantly speed up the computation of neural networks, making it suitable for training large-scale models and handling complex tasks.
4. Rich Ecosystem and Community: PyTorch has a vibrant and active community of developers, researchers, and enthusiasts. This active community contributes to the development of new models, algorithms, and tools, and provides support

and resources for newcomers. Additionally, PyTorch has a rich ecosystem of libraries and extensions that further enhance its functionality and ease of use.

5. Extensibility and Customization: PyTorch offers a highly extensible framework that allows developers to customize and extend its functionality. Developers can create custom neural network layers, loss functions, and optimization algorithms, enabling them to address specific research or application requirements.
6. Seamless Integration with Python and Numpy: PyTorch seamlessly integrates with the Python programming language, which is widely used in the machine learning community. This integration allows developers to leverage Python's vast ecosystem of libraries and tools, such as NumPy, for data preprocessing, visualization, and evaluation.

Core Components of PyTorch:

1. Tensor: The fundamental data structure in PyTorch is the tensor, which is similar to multi-dimensional arrays. Tensors can store and manipulate data efficiently and are the building blocks of neural networks in PyTorch.
2. Module: In PyTorch, a module represents a neural network layer or a complete neural network. Modules encapsulate trainable parameters, define the forward pass, and provide methods for model training and inference.
3. Optimizer: PyTorch provides various optimization algorithms, such as SGD (Stochastic Gradient Descent), Adam, and RMSprop, through its optimizer module. Optimizers are used to update the parameters of the neural network during the training process.
4. DataLoader: PyTorch's DataLoader simplifies the process of loading and batching data for training. It provides features like shuffling, parallel data loading, and customizable data transformations, making it easy to handle large datasets efficiently.

Applications of PyTorch: PyTorch finds applications in various domains and industries, including:

1. Computer Vision: PyTorch is widely used for tasks such as image classification, object detection, semantic segmentation, and image generation.

Its flexibility and extensive library support, including torchvision, make it a popular choice among computer vision researchers and practitioners.

2. Natural Language Processing (NLP): PyTorch is extensively used in NLP tasks, including sentiment analysis, machine translation, text generation, and language modeling. Libraries such as Transformers provide pre-trained models and tools for efficient NLP model development.
3. Reinforcement Learning: PyTorch's dynamic computational graph and ease of integration with other libraries, such as Gym, make it well-suited for reinforcement learning tasks. Researchers and developers leverage PyTorch to implement and train complex reinforcement learning algorithms.
4. Research and Education: PyTorch's simplicity and flexibility make it an ideal framework for researchers and educators in the field of deep learning. Its dynamic nature allows for rapid prototyping and experimentation, facilitating innovative research and efficient teaching.
5. Industry Applications: PyTorch is adopted by numerous companies and organizations for a wide range of applications, including recommendation systems, fraud detection, financial analysis, and healthcare. Its versatility, extensive community support, and performance make it a valuable tool for industry applications.

1.2 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision and are widely used for various image-related tasks, including image classification, object detection, and image segmentation. With their ability to automatically learn hierarchical features from raw pixel data, CNNs have achieved remarkable success and outperformed traditional machine learning approaches. In this article, we will delve into the fundamentals of CNNs, explore their architecture, discuss their advantages, and examine their applications in different domains.

Introduction to Convolutional Neural Networks: Convolutional Neural Networks, inspired by the structure of the visual cortex in animals, are specifically designed to process and analyze visual data. They excel at capturing spatial patterns and extracting meaningful features from images. Unlike traditional neural networks, CNNs utilize

convolutional layers, pooling layers, and fully connected layers to achieve translation invariance, parameter sharing, and hierarchical feature learning.

Architecture of Convolutional Neural Networks: The architecture of CNNs consists of several key components:

1. Convolutional Layers: Convolutional layers apply a set of learnable filters (also known as kernels) to the input image. Each filter performs convolution with the input, producing feature maps that capture local spatial information. These feature maps highlight relevant patterns and edges present in the image.
2. Activation Functions: Activation functions introduce non-linearity into the network, enabling CNNs to learn complex relationships. Common activation functions used in CNNs include Rectified Linear Unit (ReLU), sigmoid, and hyperbolic tangent.
3. Pooling Layers: Pooling layers reduce the spatial dimensionality of feature maps while preserving the most important information. Max pooling, average pooling, and sum pooling are commonly used pooling operations.
4. Fully Connected Layers: Fully connected layers connect every neuron in one layer to every neuron in the next layer, similar to traditional neural networks. These layers help in combining high-level features learned from previous layers and making final predictions.
5. Dropout: Dropout is a regularization technique commonly used in CNNs to prevent overfitting. It randomly sets a fraction of the neurons to zero during training, forcing the network to learn more robust representations.

Advantages of Convolutional Neural Networks:

1. Translation Invariance: CNNs exhibit translation invariance, meaning they can recognize objects regardless of their position in the image. This property is achieved through the use of shared weights in convolutional layers, enabling the network to learn and detect patterns irrespective of their spatial location.
2. Parameter Sharing: CNNs leverage parameter sharing, which reduces the number of learnable parameters in the network. By sharing weights across different regions of the input, CNNs can efficiently learn and generalize from limited training data.

3. Hierarchical Feature Learning: CNNs learn features hierarchically, starting from low-level features (e.g., edges, corners) to high-level features (e.g., complex shapes, objects). This hierarchical learning allows CNNs to capture both local and global spatial information, enabling effective representation of complex visual patterns.

Applications of Convolutional Neural Networks: CNNs have found numerous applications across various domains, including:

1. Image Classification: CNNs have achieved significant success in image classification tasks, surpassing human-level performance in some cases. They can accurately classify images into predefined categories, enabling applications such as autonomous driving, medical diagnosis, and facial recognition.
2. Object Detection: CNN-based object detection algorithms can localize and classify objects within an image. This technology is crucial in fields like autonomous robotics, surveillance, and augmented reality.
3. Image Segmentation: CNNs can perform pixel-level segmentation, assigning semantic labels to each pixel in an image. This technique is essential in medical imaging, autonomous vehicles, and video analysis.

1.3 HTML

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> directly introduce content to the page. Other tags such as <p> surround and provide

information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. The inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), the former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997. A form of HTML, known as HTML5, is used to display video and audio, primarily using the <canvas> element, in collaboration with JavaScript.

<html>: This is called the HTML root element. All other elements are contained within it.

<head>: The head tag contains the "behind the scenes" elements for a webpage. Elements within the head aren't visible on the front end of a webpage. HTML elements used inside the <head> element include:

- <style>-This HTML tag allows us to insert styling into our web pages and make them appealing to look at with the help of CSS.
- <title>-The title is what is displayed on the top of your browser when you visit a website and contains the title of the webpage that you are viewing.
- <base>-It specifies the base URL for all relative URLs in a document.
- <noscript>– Defines a section of HTML that is inserted when the scripting has been turned off in the user's browser.
- <script>-This tag is used to add functionality to the website with the help of JavaScript.
- <meta>-This tag encloses the metadata of the website that must be loaded every time the website is visited. For eg:- the metadata charset allows you to use the standard UTF-8 encoding on your website. This in turn allows the users to view your webpage in the language of their choice. It is a self-closing tag.
- <link>– The 'link' tag is used to tie together HTML, CSS, and JavaScript. It is self-closing.

<body>: The body tag is used to enclose all the visible content of a webpage. In other words, the body content is what the browser will show on the front end.

1.4 CSS

Cascading Style Sheets (**CSS**) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML, or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

CSS is among the core languages of the **open web** and is standardized across Web browsers according to W3C specifications. Previously, the development of various parts of CSS specification was done synchronously, which allowed the versioning of the latest recommendations. You might have heard about CSS1, CSS2.1, or even CSS3. There will never be a CSS3 or a CSS4; rather, everything is now CSS without a version number.

After CSS 2.1, the scope of the specification increased significantly and the progress on different CSS modules started to differ so much, that it became more effective to develop and release recommendations separately per module. Instead of versioning the CSS specification, W3C now periodically takes a snapshot of the latest stable state of the CSS specification and individual modules' progress. CSS modules now have version numbers, or levels, such as CSS Color Module Level 5.

1.5 JavaScript

JavaScript is a programming language that is commonly used to create interactive elements on websites. It is an object-oriented language that is supported by all modern web browsers, and it is the primary language used for client-side web development. JavaScript is executed on the client side, which means that it runs in the user's web browser rather than on the server. This makes it an ideal language for creating interactive elements on websites, such as form validation, real-time updates, and asynchronous requests. One of the key features of JavaScript is its support for objects and object-oriented programming. An object in JavaScript is a collection of properties and methods that can be used to represent real-world entities. For example, a JavaScript object might represent a user, a product, or a piece of data. JavaScript also has a number of built-in objects, such as arrays, dates, and math, that can be used to perform common tasks. Additionally, it has a number of features that make it well-

suited for web development, such as its support for events, its ability to manipulate the DOM (Document Object Model), and its ability to communicate with servers using AJAX (Asynchronous JavaScript and XML). In summary, JavaScript is a programming language that is commonly used to create interactive elements on websites. It is an object-oriented language that is supported by all modern web browsers, and it has a number of features that make it well-suited for web development.

1.6 React.js

Let's say one of your friends posted a photograph on Facebook. Now you go and like the image and then you started checking out the comments too. Now while you are browsing over comments you see that the likes count has increased by 100, since you liked the picture, even without reloading the page. This magical count change is because of ReactJS.

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. 'V' denotes the view in MVC. ReactJS is an open-source, component-based front-end library responsible only for the view layer of the application. It is maintained by Facebook.

React uses a declarative paradigm that makes it easier to reason about your application and aims to be both efficient and flexible. It designs simple views for each state in your application, and React will efficiently update and render just the right component when your data changes. The declarative view makes your code more predictable and easier to debug.

A React application is made of multiple components, each responsible for rendering a small, reusable piece of HTML. Components can be nested within other components to allow complex applications to be built out of simple building blocks. A component may also maintain an internal state – for example, a TabList component may store a variable corresponding to the currently open tab.

Prerequisites: Download Node packages with their latest version.

How does it work: While building client-side apps, a team of Facebook developers realized that the DOM is slow (The Document Object Model (DOM) is an application programming interface (API) for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.). So,

to make it faster, React implements a virtual DOM that is a DOM tree representation in JavaScript. So when it needs to read or write to the DOM, it will use the virtual representation of it. Then the virtual DOM will try to find the most efficient way to update the browser's DOM. Unlike browser DOM elements, React elements are plain objects and are cheap to create. React DOM takes care of updating the DOM to match the React elements. The reason for this is that JavaScript is very fast and it's worth keeping a DOM tree in it to speed up its manipulation. Although React was conceived to be used in the browser, because of its design it can also be used in the server with Node.js.

So why should you use React?

- The ability to work with a friendlier and optimized VIrtual Browser (*Virtual-DOM*) as compared to Real DOM.
- You just need **JavaScript knowledge** to work with React.
- React Native allows you to **create Mobile Applications** (iOS and Android).
- React team at Facebook tests all features and versions of React on Facebook.com as a result the React Library is **battle-ready**.
- React allows developers to declare **stateful user interfaces**.

1.7 React Native

React Native is a popular framework for building cross-platform mobile applications. It allows developers to write mobile apps using JavaScript and share a single codebase across multiple platforms, including iOS and Android. With its efficient and intuitive development approach, React Native has gained significant traction in the mobile app development community. In this article, we will explore the key features and advantages of React Native, discuss its architecture and development workflow, and examine its impact on the mobile app development landscape.

Introduction to React Native: React Native, developed by Facebook, is an open-source framework that combines the power of React, a JavaScript library for building user interfaces, with the ability to render native components on mobile devices. It allows developers to create native-like mobile apps using familiar web technologies, such as JavaScript, HTML, and CSS. By leveraging a single codebase, developers can save time and effort by avoiding the need to write separate code for each platform.

Key Features of React Native:

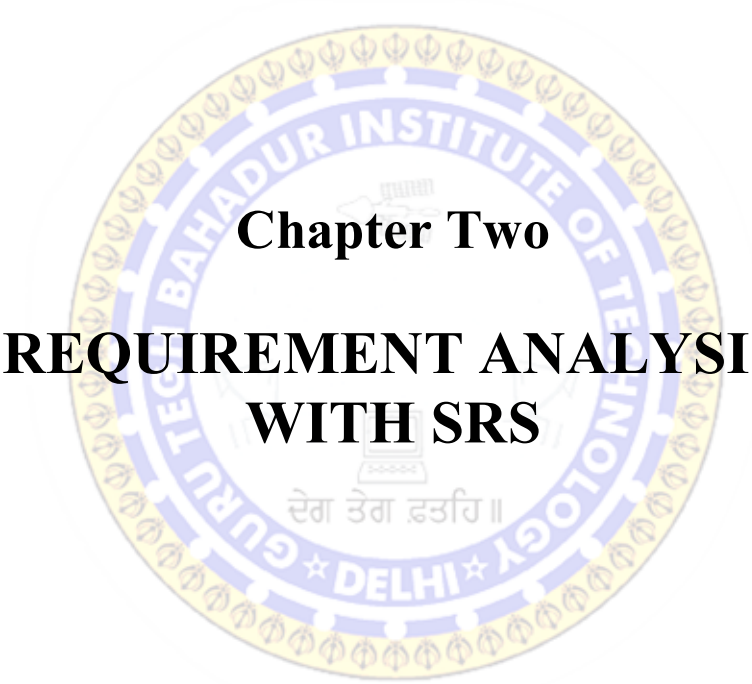
1. Cross-platform Compatibility: React Native enables developers to build mobile apps that run on both iOS and Android platforms with a single codebase. This dramatically reduces development time and allows for efficient sharing of code and resources.
2. Native Performance: React Native bridges the gap between JavaScript and native components by using a native rendering engine. This allows apps built with React Native to deliver high-performance and smooth user experiences comparable to those of apps built using native frameworks.
3. Hot Reloading: React Native's hot reloading feature allows developers to see the changes they make in the code immediately reflected in the app, without the need to rebuild or restart the entire application. This significantly speeds up the development process and enables quick iteration and debugging.
4. Modular Architecture: React Native follows a modular architecture, allowing developers to reuse and combine pre-built UI components to create complex and feature-rich mobile applications. This modular approach promotes code reusability, maintainability, and scalability.
5. Community and Ecosystem: React Native benefits from a large and active community of developers, which contributes to the growth of its ecosystem. The ecosystem includes a vast collection of third-party libraries, tools, and plugins that extend the functionality of React Native and provide solutions for common mobile app development challenges.

Architecture and Development Workflow: React Native adopts a hybrid approach, combining JavaScript and native components to build mobile apps. The architecture consists of three main layers:

1. JavaScript Layer: The JavaScript layer is responsible for managing the application logic and rendering the UI components using the React framework. Developers write JavaScript code that defines the app's behavior, data flow, and user interface.
2. Bridge Layer: The bridge layer acts as a communication channel between the JavaScript layer and the native layer. It allows JavaScript code to invoke native

modules and components and vice versa, enabling seamless integration of native functionality into the app.

3. Native Layer: The native layer consists of platform-specific code written in Java or Kotlin for Android and Objective-C or Swift for iOS. This layer handles tasks such as rendering native UI components, accessing device capabilities, and interacting with platform-specific APIs.



Chapter Two

REQUIREMENT ANALYSIS WITH SRS

Requirement Analysis with SRS

2.1 External Interface Requirement:

We classify External Interfaces into 4 types, those are:

1. User Interface:

Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.

2. Hardware interface:

Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and the communication protocols to be used.

3. Software Interface:

Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data-sharing mechanism must be implemented in a specific way (for example, the use of a global data area in a multitasking operating system), specify this as an implementation constraint.

4. Communication Interface:

Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

2.2 Non-Functional Requirement:

1. Performance Requirements:

If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real-time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.

2. Safety Requirements:

Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.

3. Security Requirements:

Specify any requirements regarding security or privacy issues surrounding the use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.

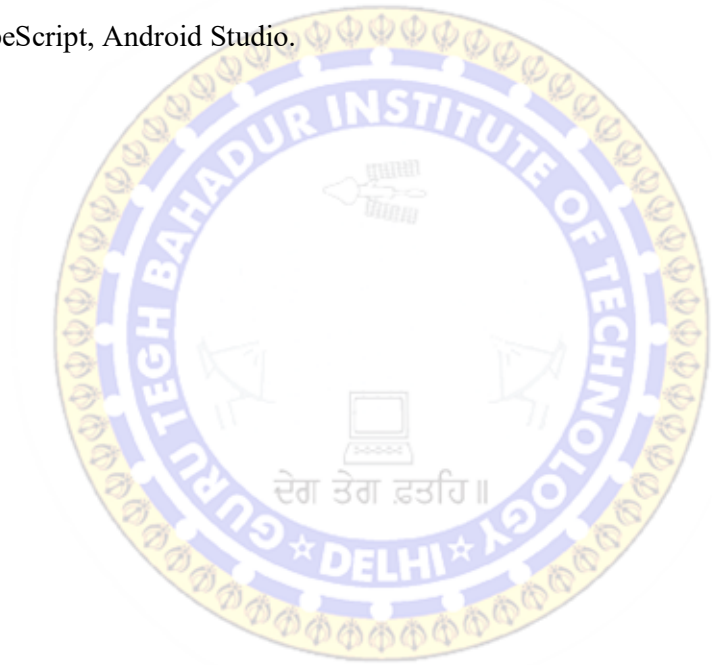
4. Software Quality Attributes:

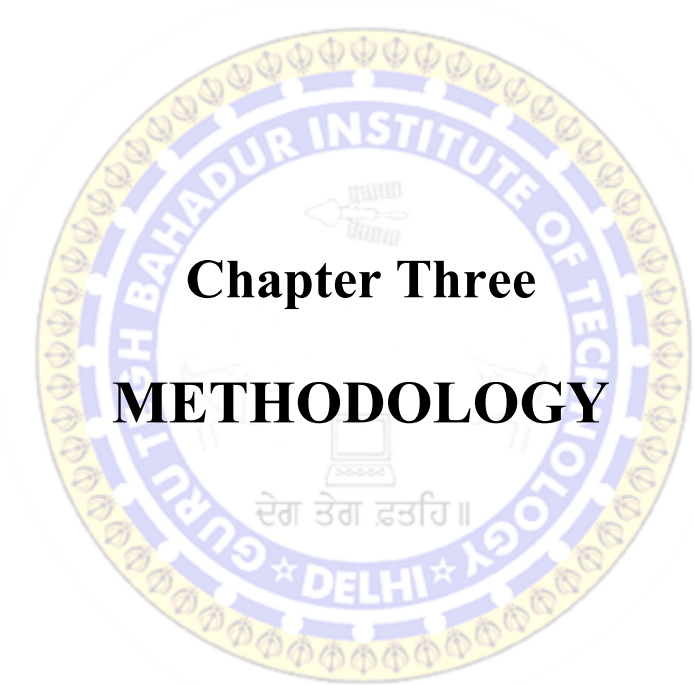
Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are adaptability, availability, correctness, flexibility, interoperability, maintainability, portability,

reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.

2.3 Other Requirements:

- Python 3.7(VSCode, pip)
- Back-End technologies -PyTorch, FastAPI, uvicorn, OpenCV, Jupyter Notebook.
- Front-End Technologies – React.js, React Native, HTML, CSS, JavaScript, TypeScript, Android Studio.





Chapter Three

METHODOLOGY

Methodology

1. Importing the necessary libraries:

- numpy for numerical computations
- pandas for data manipulation and analysis
- matplotlib.pyplot for data visualization
- torch for deep learning with PyTorch
- torchvision for computer vision tasks
- datetime for tracking time
- torchsummary for model summary

2. Setting up data transformations:

- transform defines a series of transformations to be applied to the images, including resizing, center cropping, and converting to tensors.

3. Loading the dataset:

- datasets.ImageFolder loads the image dataset from the "Dataset" folder, applying the defined transformations.

4. Splitting the dataset:

- The dataset is split into training, validation, and test sets.
- The indices list is shuffled randomly.
- The dataset is divided based on the specified ratios (0.85, 0.70), and the indices are used to create subsets for each set.

5. Defining the CNN model:

- The CNN class is defined as a subclass of nn.Module.
- The model consists of convolutional layers followed by dense (fully connected) layers.
- ReLU activation functions and batch normalization are used after each convolutional layer.
- The model takes an input image of size (3, 224, 224) and outputs a tensor of size K, where K represents the number of target classes.

6. Setting the device for training:

- The code checks if a GPU is available and assigns the device accordingly.
- The device variable is later reassigned to "cpu" for compatibility reasons.

7. Model summary:

- The summary function from torchsummary is used to display a summary of the model architecture.

8. Defining the loss function and optimizer:

- Cross-entropy loss (nn.CrossEntropyLoss) and Adam optimizer (torch.optim.Adam) are used.

9. Defining the batch gradient descent training function:

- The function batch_gd performs the training loop for the specified number of epochs.

- The train and validation losses are computed and stored for each epoch.

10. Creating data loaders:

- Data loaders are created for the training, validation, and test sets using the torch.utils.data.DataLoader class.

11. Training the model:

- The batch_gd function is called to train the model using the training and validation loaders.
- The model parameters are saved to a file(plant_disease_model_1_latest.pt).

12. Evaluating the model:

- The accuracy function computes the accuracy of the model on the training, validation, and test sets.

13. Processing predictions:

- The function single_prediction is defined to make predictions on individual images.
- The image is loaded, transformed, and passed through the model.
- The predicted disease class is obtained by finding the index with the highest output probability.

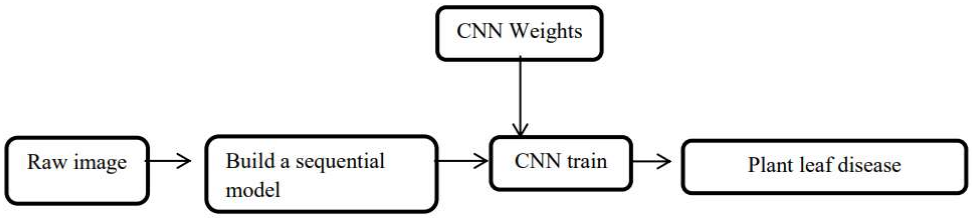


Fig. 1: Methodology diagram of the Plant Disease Detection System

Chapter Four

SYSTEM DESIGN

System Design

4.1 System architecture and Work flow diagram

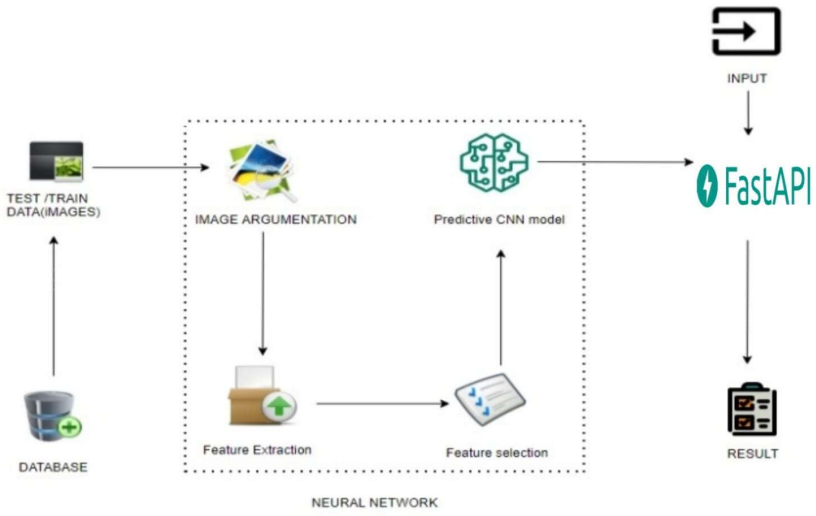


Fig. 2: System Architecture

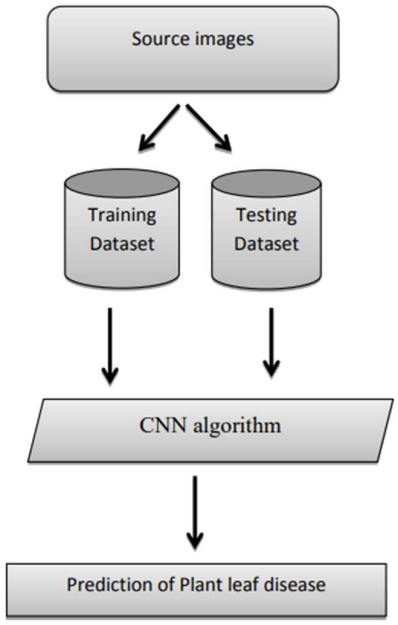


Fig. 3: Work Flow Diagram

4.2 Data flow diagram (DFD)

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model. Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top-down approach to Systems Design. Symbols and Notations Used in DFDs Using any 10 convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

External entity: an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

Process: any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.

Data store: files or repositories that hold information for later use, such as a database table or a membership form.

Data flow: the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like "Billing details."

DFD levels and layers A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish. DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

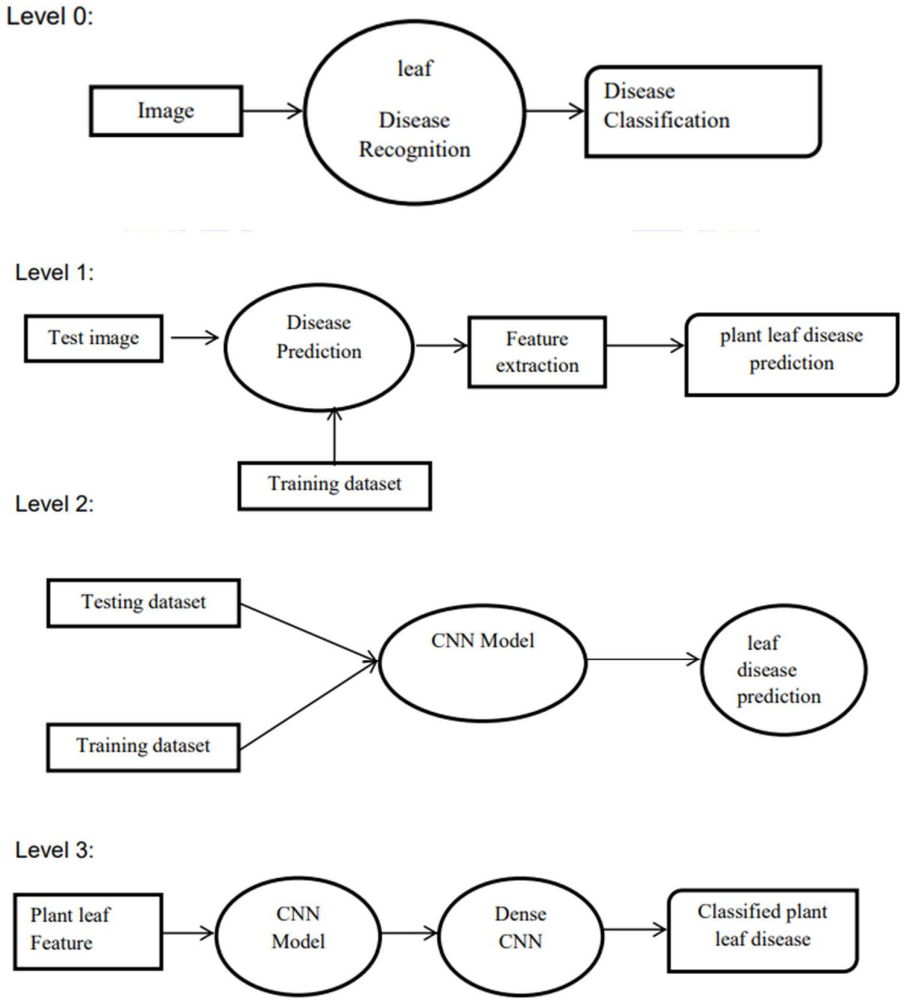


Fig. 4: Data Flow Diagram

4.3 ER Diagram

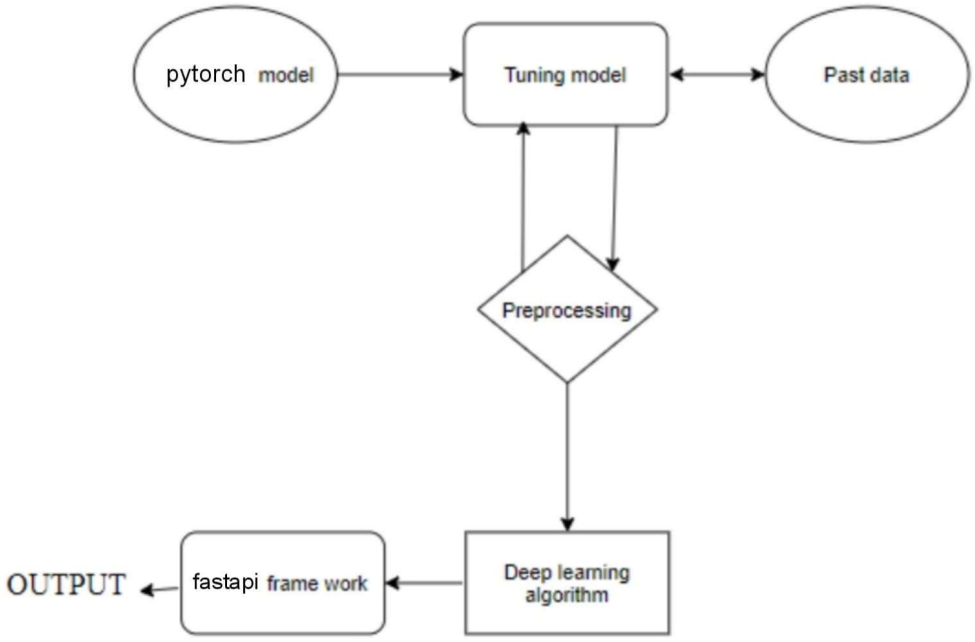


Fig. 5: Entity Relationship Diagram

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

4.4 Class Diagram

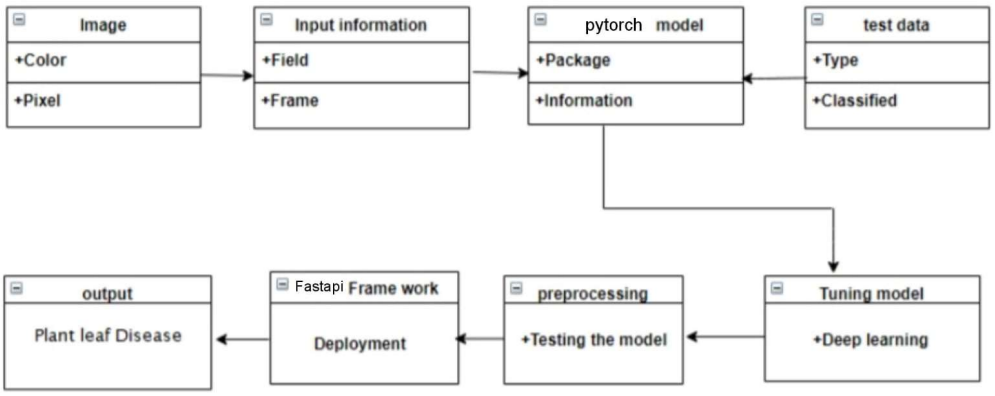


Fig. 6 Class diagram

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So, a collection of class diagrams represents the whole system. 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 and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and 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 rework as many times as possible to make it correct.

4.5 Use Case diagram

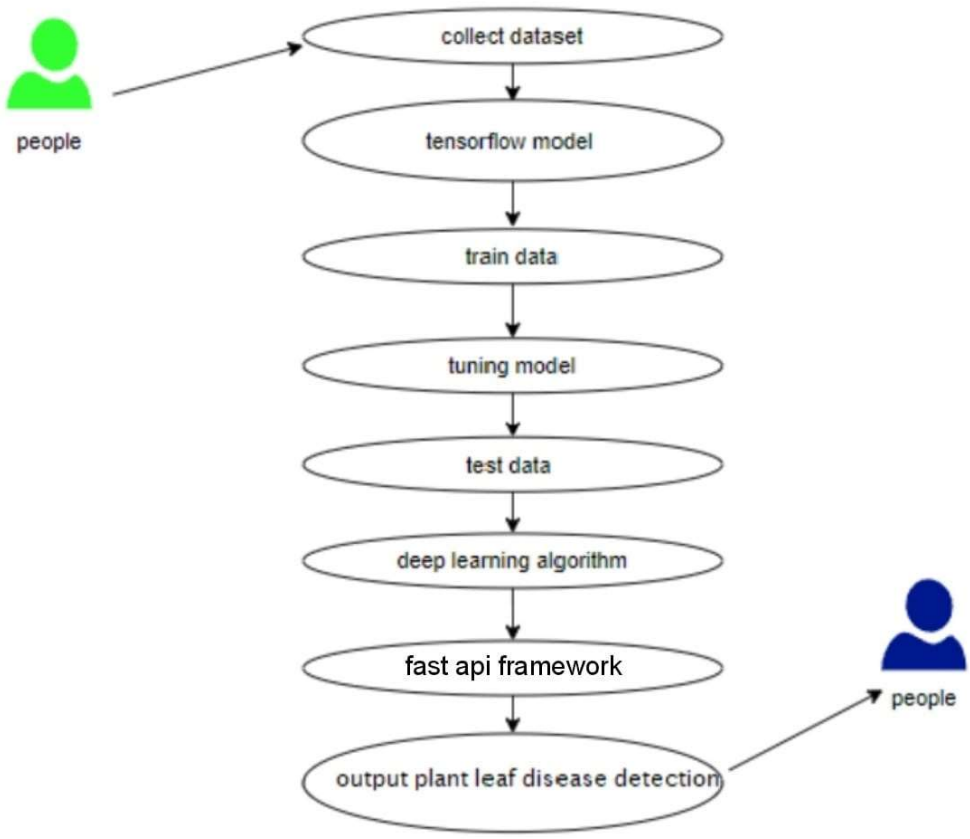
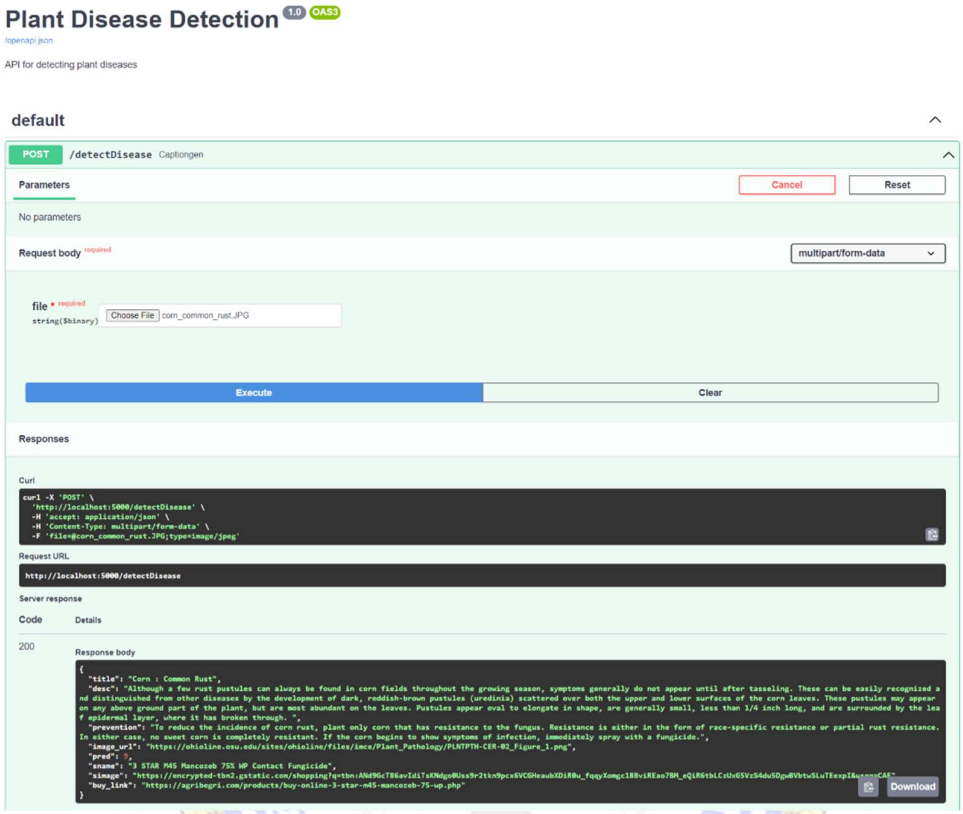


Fig. 7: Use Case Diagram

Use case diagrams are considered for high level requirement analysis of a system. So, when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

Results



Our project successfully uses a leaf image as an input to identify the ailment using a convolutional neural network and then outputs the disease's description and supplements.

Observations

- The obtained accuracies for both the test and validation datasets (98.9% and 98.7% respectively) indicate that the trained model performs exceptionally well in correctly identifying and classifying plant diseases.
- This suggests that the model has learned meaningful patterns and features from the input images, allowing it to make accurate predictions.

Conclusion

- Food is a basic necessity for humans, and farmers play a crucial role in producing food.
- Farmers often face yield losses due to pests and diseases, resulting in agricultural losses.
- Accurate detection of plant leaf diseases is essential to minimize yield losses caused by diseases.
- Convolutional Neural Networks (CNNs) can be a significant approach for detecting plant leaf diseases.
- The Kaggle Plant Village Dataset, consisting of over 32,000 images of healthy and diseased leaves, can be used for training the CNN model.
- Deep learning techniques, including image recognition and preprocessing, can be applied to efficiently detect plant leaf diseases.
- This system not only provides accurate disease detection but also offers remedies for the identified diseases.
- By leveraging the concept of Deep Learning and image analysis, plant leaf diseases can be detected quickly and accurately.
- The system can aid farmers in managing and preventing crop losses, thus contributing to sustainable agriculture.

Future Scope

- New technologies and methods enable faster and more efficient applications for users.
- Pretrained CNN models achieve 98% accuracy in plant disease recognition and classification.
- Expanding the scope of CNN for classification allows quick and accurate results.
- CNN models significantly reduce leaf image processing and analysis time.
- Faster and efficient leaf image recognition enables the development of applications with quick and accurate results for users.

REFERENCES

[1] Acharya, D., & Pant, A. (2017). Deep learning-based classification for plant diseases. Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 1575-1580.

[2] A.K. Mishra, V. D. Bharadi, and M. B. Nagori, "Plant Disease Detection Techniques: A Review," Indian Journal of Science and Technology, vol. 9, no. 45, 2016.

[3] "Automated Identification of Plant Diseases using Machine Learning Techniques" by Anupama Vijaya Kumar.

[4] "Computer Vision: Models, Learning, and Inference" by Simon J.D. Prince.

[5] "Deep Learning Models for Plant Disease Detection and Diagnosis" by Ferentinos, K. P. (2018).

[6] "Deep learning models for plant disease detection and diagnosis" by Ferentinos, K. P. (2020). In Advances in Experimental Medicine and Biology, 1138, 21-35.

[7] "Deep learning in agriculture: A survey" by Kamilaris, A., & Prenafeta-Boldú, F. X. (2018).

[8] "Deep neural networks-based recognition of plant diseases by leaf image classification" by Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016). Computational Intelligence and Neuroscience, 2016, 1-11.

[9] "Deep neural networks based recognition of plant diseases under limited labeled training data" by Sladojevic, S., Gavrilovic, M., Zikic, V., & Anderla, A. (2018). Neurocomputing, 283, 279-288.

[10] "Going Deeper with Convolutions" by C. Szegedy et al., in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[11] "Image Analysis, Classification, and Change Detection in Remote Sensing: With Algorithms for ENVI/IDL and Python" by Morton J. Canty.

[12] "Learning React Native: Building Native Mobile Apps with JavaScript" by Bonnie Eisenman.

[13] "Pattern Recognition and Machine Learning" by Christopher M. Bishop.

[14] "Plant Disease Detection and Diagnosis" by Jennifer Olson and Margery Daughtrey.

[15] "Plant disease detection: Review on detection techniques and recent advances" by Das, S., Raza, S., & Pradhan, M. (2018). Expert Systems with Applications, 120, 162-184.

[16] "Plant disease detection and classification using image processing techniques: A review" by Sharma, G., & Prakash, S. (2017). Journal of Computing and Electronics in Agriculture, 143, 152-167.

[17] "Plant disease detection using deep learning techniques: A systematic literature review" by J. De Lima Morais, D. C. Pereira, M. C. A. Figueiredo, and A. De M. A. Zanetti. Computers and Electronics in Agriculture, vol. 174, 2020.

[18] "Plant disease recognition from hyperspectral reflectance image based on deep learning framework" by Yan, Y., Zhang, W., & Ma, Y. (2016). Computers and Electronics in Agriculture, 124, 366-373.

[19] "Review on recent advances in detection of plant diseases from remote sensing images" by Ghosal, S., Das, S., & Saha, S. (2018). Computers and Electronics in Agriculture, 145, 201-215.

[20] "Review on Detection and Classification Techniques of Plant Diseases" by P. Gautam and P. S. Patil, International Journal of Advanced Research in Computer Science and Software Engineering, vol. 5, no. 12, pp. 732-736, 2015.

[21] "Review of Image Processing Techniques for Plant Disease Detection" by M. Fuentes, C. Ornelas-Paz, and J. Y. Castro, Computers and Electronics in Agriculture, vol. 144, pp. 1-14, 2017.

[22] "Using deep learning for image-based plant disease detection" by Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Frontiers in Plant Science, 7, 1419.

[23] "Very Deep Convolutional Networks for Large-Scale Image Recognition" by K. Simonyan and A. Zisserman, in Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2015.

[24] "Deep Learning" by Y. LeCun, Y. Bengio, and G. Hinton, Nature, vol. 521, no. 7553, pp. 436-444, 2015.

[25] "Plant Disease Detection Techniques: A Review" by A. K. Mishra, V. D. Bharadi, and M. B. Nagori, Indian Journal of Science and Technology, vol. 9, no. 45, 2016.

[26] "Plant Disease Detection Using Deep Learning Techniques: A Systematic Literature Review" by J. De Lima Morais, D. C. Pereira, M. C. A. Figueiredo, and A. De M. A. Zanetti, Computers and Electronics in Agriculture, vol. 174, 2020.

[27] "Plant leaf disease detection and classification using image processing techniques: A review" by Sharma, G., & Prakash, S. (2017). Journal of Computing and Electronics in Agriculture, 143, 152-167.

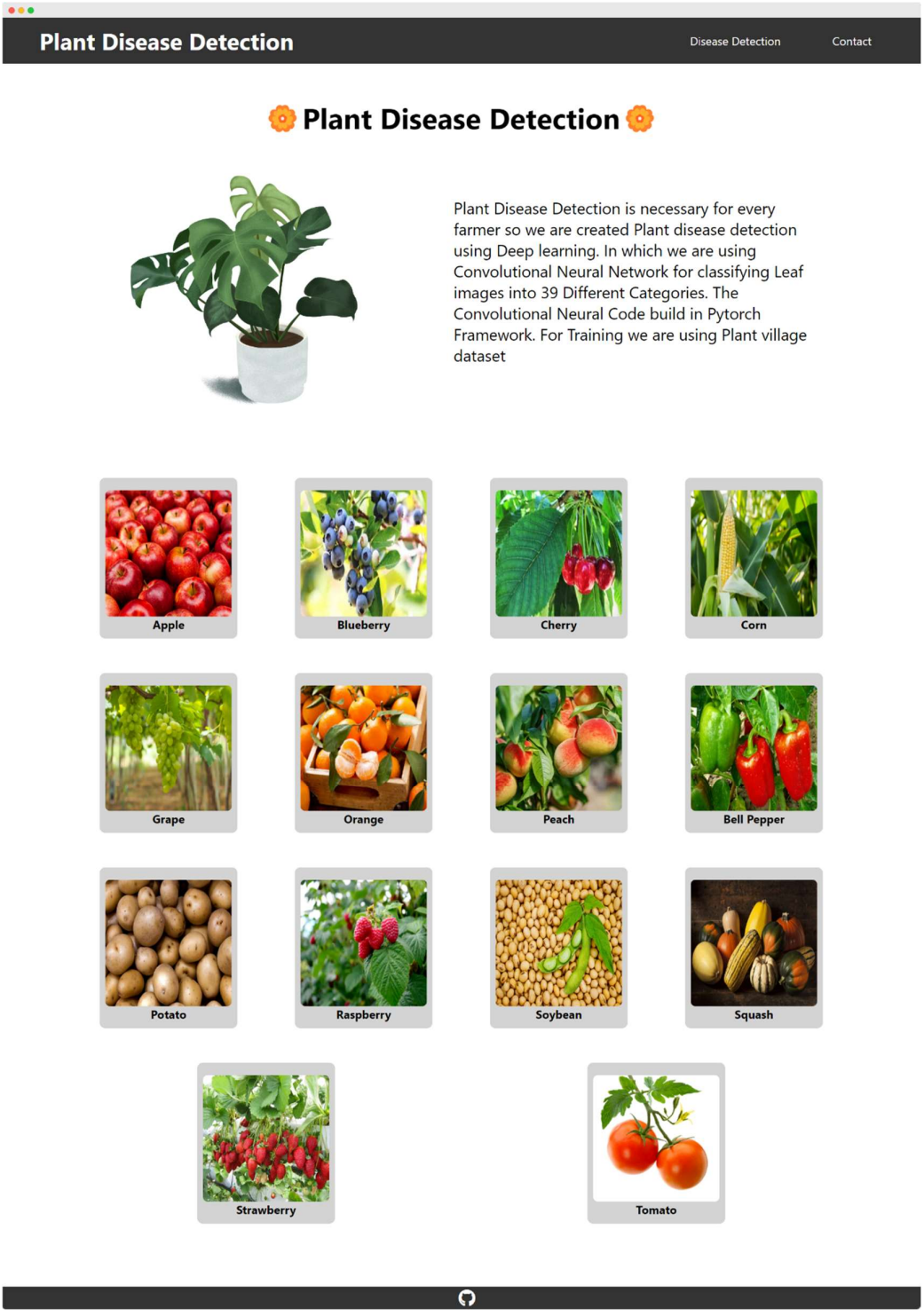
[28] "Deep Learning Models for Plant Disease Detection and Diagnosis" by Ferentinos, K. P. (2018). Computers in Biology and Medicine, 103, 137-145.

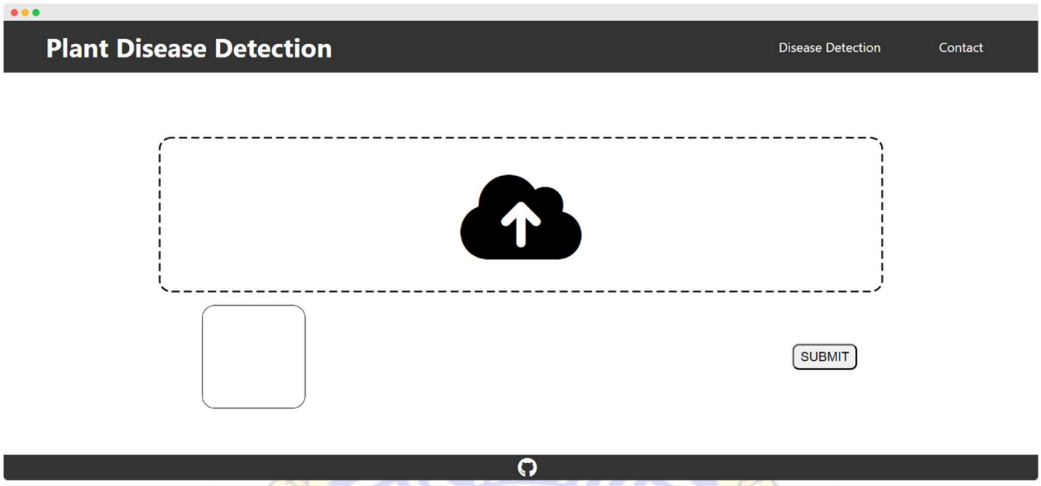
[29] "Deep learning-based classification for plant diseases" by Acharya, D., & Pant, A. (2017). Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 1575-1580.

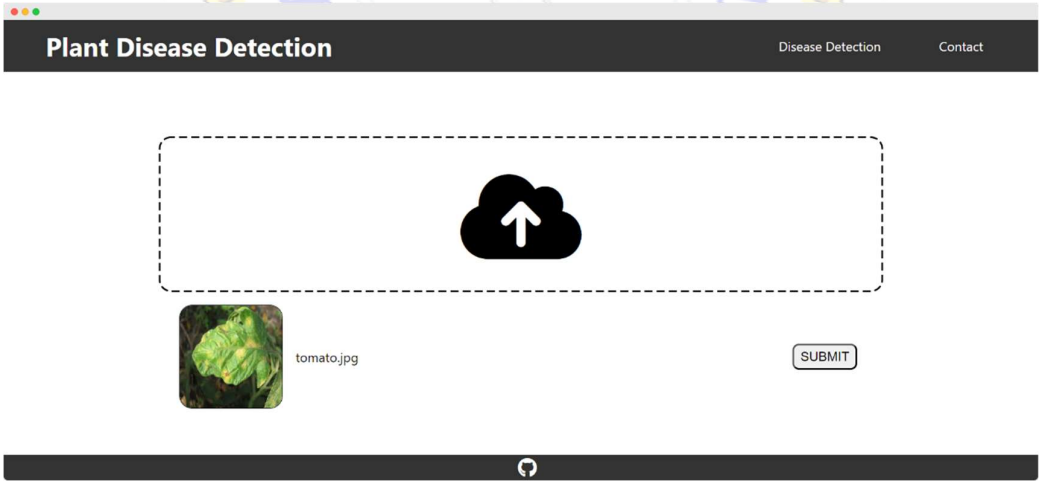
[30] "Deep Learning in Agriculture: A Survey" by Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Computers and Electronics in Agriculture, 147, 70-90.

APPENDIX A

SCREENSHOTS


Plant Disease Detection
Disease Detection
Contact
Plant Disease Detection
Plant Disease Detection is necessary for every farmer so we are created Plant disease detection using Deep learning. In which we are using Convolutional Neural Network for classifying Leaf images into 39 Different Categories. The Convolutional Neural Code build in Pytorch Framework. For Training we are using Plant village dataset
Apple
Blueberry
Cherry
Corn
Grape
Orange
Peach
Bell Pepper
Potato
Raspberry
Soybean
Squash
Strawberry
Tomato


Plant Disease Detection
Disease Detection
Contact
SUBMIT


Plant Disease Detection
Disease Detection
Contact
tomato.jpg
SUBMIT

Plant Disease Detection

Disease Detection Contact

Tomato : Bacterial Spot

Brief Descritpion :

Bacterial spot of tomato is a potentially devastating disease that, in severe cases, can lead to unmarketable fruit and even plant death. Bacterial spot can occur wherever tomatoes are grown, but is found most frequently in warm, wet climates, as well as in greenhouses. The disease is often an issue in Wisconsin.Bacterial spot can affect all above ground parts of a tomato plant, including the leaves, stems, and fruit. Bacterial spot appears on leaves as small (less than ? inch), sometimes water-soaked (i.e., wet-looking) circular areas. Spots may initially be yellow-green, but darken to brownish-red as they age. When the disease is severe, extensive leaf yellowing and leaf loss can also occur. On green fruit, spots are typically small, raised and blister-like, and may have a yellowish halo. As fruit mature, the spots enlarge (reaching a maximum size of ¼ inch) and turn brown, scabby and rough. Mature spots may be raised, or sunken with raised edges.

Prevent This Plant Disease By follow below steps :

Plant pathogen-free seed or transplants to prevent the introduction of bacterial spot pathogens on contaminated seed or seedlings. If a clean seed source is not available or you suspect that your seed is contaminated, soak seeds in water at 122°F for 25 min. to kill the pathogens. To keep leaves dry and to prevent the spread of the pathogens, avoid overhead watering (e.g., with a wand or sprinkler) of established plants and instead use a drip-tape or soaker-hose. Also to prevent spread, DO NOT handle plants when they are wet (e.g., from dew) and routinely sterilize tools with either 10% bleach solution or (better) 70% alcohol (e.g., rubbing alcohol). Where bacterial spot has been a recurring problem, consider using preventative applications of copper-based products registered for use on tomato, especially during warm, wet periods. Keep in mind however, that if used excessively or for prolonged periods, copper may no longer control the disease. Be sure to read and follow all label instructions of the product that you select to ensure that you use it in the safest and most effective manner possible.

Supplements :

CUREAL Best Fungicide & Bactericide

Plant Disease Detection

Disease Detection Contact

Contact the Developers

Gurman Singh

gurman29singh@gmail.com

29th-pixel

gurman-singh-8b94a21ba

Harneet Singh Channa

harneetsinghch110@gmail.com

hsc-code

harneet-singh-channa-8473831ba

Kuldeep Singh

kuldeepsingh2670@gmail.com

kuldeep2670

kuldeep2670

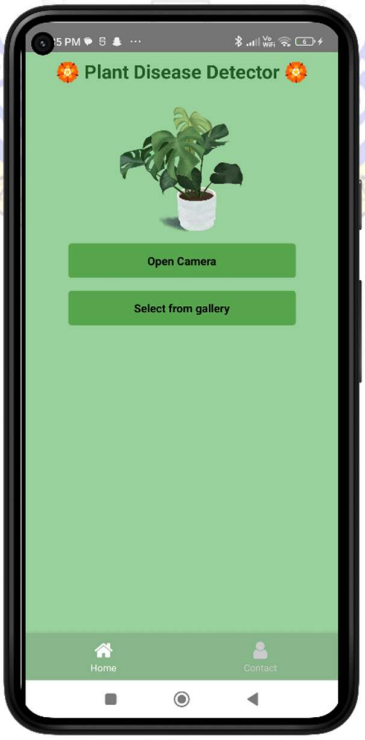
Sahibjot Singh

sahibjot7180@gmail.com

sahibjot366

sahibjot-singh


5 PM
Plant Disease Detection
Plant Disease Detection is necessary for every farmer so we are created Plant disease detection using Deep learning. In which we are using Convolutional Neural Network for classifying Leaf images into 39 Different Categories. The Convolutional Neural Code build in Pytorch Framework. For Training we are using Plant village dataset
Apple
Blueberry
Cherry
Corn
Grape
Orange
Try it !!
Home
Contact


5 PM
Plant Disease Detector
Open Camera
Select from gallery
Home
Contact


Plant Disease Detector
Open Camera
Select from gallery
FIND DISEASE
Home
Contact
Plant Disease Detector
X
Grape : Esca | Black Measles
Brief Descritpion :
Grapevine measles, also called esca, black measles or Spanish measles, has long plagued grape growers with its cryptic expression of symptoms and, for a long time, a lack of identifiable causal organism(s). The name 'measles' refers to the superficial spots found on the fruit (Fig. 1). During the season, the spots may coalesce over the skin surface, making berries black in appearance. Spotting can develop anytime between fruit set and a few days prior to harvest. Berries affected at fruit set tend not to mature and will shrivel and dry up. In addition to spotting, fruit affected later in the season will also have an acrid taste.Leaf symptoms
Home
Contact
Our Team of Developers
Gurman Singh
Harneet Singh
Kuldeep Singh
Sahibjot Singh
Home
Contact

APPENDIX B

SOURCE CODE

Back-End

app.py

```
import uvicorn
import os
from fastapi import FastAPI, responses, File, UploadFile
from fastapi.middleware.cors import CORSMiddleware
import cv2
import io
import predictor
import numpy as np
import pandas as pd
from PIL import Image
import torchvision.transforms.functional as TF
import torch
import cv2

app = FastAPI(title='Plant Disease Detection',
          description="API for detecting plant diseases", version="1.0", debug=True)

origins = [
    "http://localhost",
    "http://localhost:3000",
    "http://localhost:8080",
    "http://localhost:5000",
    "http://127.0.0.1:5000",
    "http://192.168.1.41:3000"
]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

diseaseInfoPath = os.path.dirname(os.getcwd()).replace(
    '\\', '/') + '/Plant-Disease-Detection-App/BackEnd/assets/disease_info.csv'
supplementInfoPath = os.path.dirname(os.getcwd()).replace(
    '\\', '/') + '/Plant-Disease-Detection-App/BackEnd/assets/supplement_info.csv'
modelPath = os.path.dirname(os.getcwd()).replace(
    '\\', '/') + '/Plant-Disease-Detection-App/BackEnd/assets/plant_disease_model_1_latest.pt'
diseaseInfo = pd.read_csv(diseaseInfoPath, encoding='cp1252')
supplementInfo = pd.read_csv(supplementInfoPath, encoding='cp1252')

model = predictor.CNN(39)
model.load_state_dict(torch.load(modelPath))
model.eval()


def prediction(image):
    # image = Image.open(image_path)
    image = image.resize((224, 224))
    input_data = TF.to_tensor(image)
    input_data = input_data.view((-1, 3, 224, 224))
    output = model(input_data)
    output = output.detach().numpy()
    index = np.argmax(output)
```

```
    return index

def load_image_into_numpy_array(data):
    npimg = np.frombuffer(data, np.uint8)
    frame = cv2.imdecode(npimg, cv2.IMREAD_COLOR)
    return cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)


@app.post('/detectDisease')
async def captiongen(file: UploadFile = File(...)):
    request_object_content = await file.read()
    image = Image.open(io.BytesIO(request_object_content))
    pred = prediction(image)
    title = diseaseInfo['disease_name'][pred]
    description = diseaseInfo['description'][pred]
    prevent = diseaseInfo['Possible Steps'][pred]
    image_url = diseaseInfo['image_url'][pred]
    supplement_name = supplementInfo['supplement name'][pred]
    supplement_image_url = supplementInfo['supplement image'][pred]
    supplement_buy_link = supplementInfo['buy link'][pred]
    return responses.JSONResponse(content={"title": title, "desc" : description, "prevention" : prevent,
"image_url": image_url, "pred": int(pred), "sname": supplement_name, "simage":
supplement_image_url, "buy_link": supplement_buy_link})


if __name__ == "__main__":
    uvicorn.run(app, port=5000)
```

predictor.py

```
import torch.nn as nn
from PIL import Image

class CNN(nn.Module):
    def __init__(self, K):
        super(CNN, self).__init__()
        self.conv_layers = nn.Sequential(
            # conv1
            nn.Conv2d(in_channels=3, out_channels=32,
                    kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.Conv2d(in_channels=32, out_channels=32,
                    kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.MaxPool2d(2),
            # conv2
            nn.Conv2d(in_channels=32, out_channels=64,
                    kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(64),
            nn.Conv2d(in_channels=64, out_channels=64,
                    kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(64),
            nn.MaxPool2d(2),
            # conv3
            nn.Conv2d(in_channels=64, out_channels=128,
```

```
                kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(128),
            nn.Conv2d(in_channels=128, out_channels=128,
                kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(128),
            nn.MaxPool2d(2),
            # conv4
            nn.Conv2d(in_channels=128, out_channels=256,
                kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(256),
            nn.Conv2d(in_channels=256, out_channels=256,
                kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(256),
            nn.MaxPool2d(2),
        )

        self.dense_layers = nn.Sequential(
            nn.Dropout(0.4),
            nn.Linear(50176, 1024),
            nn.ReLU(),
            nn.Dropout(0.4),
            nn.Linear(1024, K),
        )

    def forward(self, X):
        out = self.conv_layers(X)

        # Flatten
        out = out.view(-1, 50176)

        # Fully connected
        out = self.dense_layers(out)

        return out


idx_to_classes = {0: 'Apple___Apple_scab',
                  1: 'Apple___Black_rot',
                  2: 'Apple___Cedar_apple_rust',
                  3: 'Apple___healthy',
                  4: 'Background_without_leaves',
                  5: 'Blueberry___healthy',
                  6: 'Cherry___Powdery_mildew',
                  7: 'Cherry___healthy',
                  8: 'Corn___Cercospora_leaf_spot Gray_leaf_spot',
                  9: 'Corn___Common_rust',
                  10: 'Corn___Northern_Leaf_Blight',
                  11: 'Corn___healthy',
                  12: 'Grape___Black_rot',
                  13: 'Grape___Esca_(Black_Measles)',
                  14: 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)',
                  15: 'Grape___healthy',
                  16: 'Orange___Haunglongbing_(Citrus_greening)',
                  17: 'Peach___Bacterial_spot',
                  18: 'Peach___healthy',
                  19: 'Pepper,_bell___Bacterial_spot',
```

```
20: 'Pepper,_bell___healthy',
21: 'Potato___Early_blight',
22: 'Potato___Late_blight',
23: 'Potato___healthy',
24: 'Raspberry___healthy',
25: 'Soybean___healthy',
26: 'Squash___Powdery_mildew',
27: 'Strawberry___Leaf_scorch',
28: 'Strawberry___healthy',
29: 'Tomato___Bacterial_spot',
30: 'Tomato___Early_blight',
31: 'Tomato___Late_blight',
32: 'Tomato___Leaf_Mold',
33: 'Tomato___Septoria_leaf_spot',
34: 'Tomato___Spider_mites Two-spotted_spider_mite',
35: 'Tomato___Target_Spot',
36: 'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
37: 'Tomato___Tomato_mosaic_virus',
38: 'Tomato___healthy'}
```

Model Training Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import torch
from torchvision import datasets, transforms, models  # datsets  , transforms
from torch.utils.data.sampler import SubsetRandomSampler
import torch.nn as nn
import torch.nn.functional as F
from datetime import datetime

%load_ext nb_black

transform = transforms.Compose(
    [transforms.Resize(255), transforms.CenterCrop(224), transforms.ToTensor()]
)
dataset = datasets.ImageFolder("Dataset", transform=transform)
indices = list(range(len(dataset)))
split = int(np.floor(0.85 * len(dataset)))  # train_size
validation = int(np.floor(0.70 * split))  # validation
print(0, validation, split, len(dataset))
print(f"length of train size :{validation}")
print(f"length of validation size :{split - validation}")
print(f"length of test size :{len(dataset)-validation}")

np.random.shuffle(indices)
train_indices, validation_indices, test_indices = (
    indices[:validation],
    indices[validation:split],
    indices[split:],
)

train_sampler = SubsetRandomSampler(train_indices)
validation_sampler = SubsetRandomSampler(validation_indices)
test_sampler = SubsetRandomSampler(test_indices)

targets_size = len(dataset.class_to_idx)

class CNN(nn.Module):
```

```
    def __init__(self, K):
        super(CNN, self).__init__()
        self.conv_layers = nn.Sequential(
            # conv1
            nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.Conv2d(in_channels=32, out_channels=32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.MaxPool2d(2),
            # conv2
            nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(64),
            nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(64),
            nn.MaxPool2d(2),
            # conv3
            nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(128),
            nn.Conv2d(in_channels=128, out_channels=128, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(128),
            nn.MaxPool2d(2),
            # conv4
            nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(256),
            nn.Conv2d(in_channels=256, out_channels=256, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(256),
            nn.MaxPool2d(2),
        )
        self.dense_layers = nn.Sequential(
            nn.Dropout(0.4),
            nn.Linear(50176, 1024),
            nn.ReLU(),
            nn.Dropout(0.4),
            nn.Linear(1024, K),
        )
    def forward(self, X):
        out = self.conv_layers(X)

        # Flatten
        out = out.view(-1, 50176)

        # Fully connected
        out = self.dense_layers(out)

        return out

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)

device = "cpu"
model = CNN(targets_size)
```

```
model.to(device)
from torchsummary import summary
summary(model, (3, 224, 224))
criterion = nn.CrossEntropyLoss()  # this include softmax + cross entropy loss
optimizer = torch.optim.Adam(model.parameters())

def batch_gd(model, criterion, train_loader, test_laoder, epochs):
    train_losses = np.zeros(epochs)
    test_losses = np.zeros(epochs)

    for e in range(epochs):
        t0 = datetime.now()
        train_loss = []
        for inputs, targets in train_loader:
            inputs, targets = inputs.to(device), targets.to(device)
            optimizer.zero_grad()
            output = model(inputs)
            loss = criterion(output, targets)
            train_loss.append(loss.item())  # torch to numpy world
            loss.backward()
            optimizer.step()

        train_loss = np.mean(train_loss)

        validation_loss = []

        for inputs, targets in validation_loader:
            inputs, targets = inputs.to(device), targets.to(device)
            output = model(inputs)
            loss = criterion(output, targets)
            validation_loss.append(loss.item())  # torch to numpy world

        validation_loss = np.mean(validation_loss)

        train_losses[e] = train_loss
        validation_losses[e] = validation_loss

        dt = datetime.now() - t0

        print(
            f"Epoch : {e+1}/{epochs} Train_loss:{train_loss:.3f} Test_loss:{validation_loss:.3f}
Duration:{dt}"
        )

    return train_losses, validation_losses

device = "cpu"

batch_size = 64
train_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=train_sampler
)
test_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=test_sampler
)
validation_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=validation_sampler
)

train_losses, validation_losses = batch_gd(
```

```
    model, criterion, train_loader, validation_loader, 5
)

# torch.save(model.state_dict() , 'plant_disease_model_1.pt')

targets_size = 39
model = CNN(targets_size)
model.load_state_dict(torch.load("plant_disease_model_1_latest.pt"))
model.eval()

def accuracy(loader):
    n_correct = 0
    n_total = 0

    for inputs, targets in loader:
        inputs, targets = inputs.to(device), targets.to(device)

        outputs = model(inputs)

        _, predictions = torch.max(outputs, 1)

        n_correct += (predictions == targets).sum().item()
        n_total += targets.shape[0]

    acc = n_correct / n_total
    return acc


train_acc = accuracy(train_loader)
test_acc = accuracy(test_loader)
validation_acc = accuracy(validation_loader)

print(
    f"Train Accuracy : {train_acc}\nTest Accuracy : {test_acc}\nValidation Accuracy :
{validation_acc}"
)

transform_index_to_disease = dataset.class_to_idx

transform_index_to_disease = dict(
    [(value, key) for key, value in transform_index_to_disease.items()]
)  # reverse the index

data = pd.read_csv("disease_info.csv", encoding="cp1252")

from PIL import Image
import torchvision.transforms.functional as TF
def single_prediction(image_path):
    image = Image.open(image_path)
    image = image.resize((224, 224))
    input_data = TF.to_tensor(image)
    input_data = input_data.view((-1, 3, 224, 224))
    output = model(input_data)
    output = output.detach().numpy()
    index = np.argmax(output)
    print("Original : ", image_path[12:-4])
    pred_csv = data["disease_name"][index]
    print(pred_csv)

single_prediction("test_images/Apple_ceder_apple_rust.JPG")
```

Front-End – Web App

App.js

```
import "./App.css";
import {BrowserRouter, Routes, Route, Navigate} from 'react-router-dom';
import Navbar from "./components/Navbar";
import Home from "./components/Home";
import DiseaseDetection from "./components/Disease-Detection";
import Contact from "./components/Contact";
import Footer from "./components/Footer";
function App() {
 return (
   <BrowserRouter>
     <Navbar />
     <Routes>
       <Route path ='/' element={<Home/>} />
       <Route path ='/disease-detection' element={<DiseaseDetection/>} />
       <Route path ='/contact' element={<Contact/>} />
       <Route path ="*" element = {<Navigate to = "/"/>} />
     </Routes>
     <Footer/>
   </BrowserRouter>
 );
}
export default App;
```

Navbar.js

```
import './Navbar.css';
import {Link} from "react-router-dom";
export default function Navbar(){
   return (
   <nav className="nav">
      <h1 className="title"><Link to="/">Plant Disease Detection</Link> </h1>
      <ul>
         <li>
            <Link to="/disease-detection"> Disease Detection</Link>
         </li>
         <li>
            <Link to="/contact">Contact</Link>
         </li>
      </ul>
   </nav>
   )
}
```

Home.js

```
import "./Home.css";
import plant from './images/plant.png';
function Home() {
   return (
      <>
            <div className="container1">
            <h1> 🏵 Plant Disease Detection 🏵 </h1>

            <div className="body">
               <div className="plant">
```

```
                <img src={plant} alt="plant"/>
              </div>
              <div className="content">
                <p>Plant Disease Detection is necessary for every farmer so we are created Plant
disease detection using Deep learning. In which we are using Convolutional Neural Network for
classifying Leaf images into 39 Different Categories. The Convolutional Neural Code build in Pytorch
Framework. For Training we are using Plant village dataset</p>
              </div>
            </div>

            <div className="categories">
              <div className="c1">
                <img src="https://post.healthline.com/wp-
content/uploads/2020/09/Do_Apples_Affect_Diabetes_and_Blood_Sugar_Levels-732x549-
thumbnail-1-732x549.jpg"></img>
                <h6>Apple</h6>
              </div>
              <div className="c1">
                <img src="https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQNkX9PwImXVpsUnEu69_BZRVNYgUhpQJ6nLw&usqp=
CAU"></img>
                <h6>Blueberry</h6>
              </div>
              <div className="c1">
                <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTMS9w-
BMX5br8CHuVMqjO5Rr9WbMqPCAExbA&usqp=CAU"></img>
                <h6>Cherry</h6>
              </div>
              <div className="c1">
                <img src="https://i0.wp.com/post.healthline.com/wp-content/uploads/2021/03/corn-
cob-field-1296x728-header.jpg?w=1155&h=1528"></img>
                <h6>Corn</h6>
              </div>
              <div className="c1">
                <img src="https://minnetonkaorchards.com/wp-content/uploads/2022/04/White-Green-
Grapes-SS-1102331153.jpg"></img>
                <h6>Grape</h6>
              </div>
              <div className="c1">
                <img src="https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcRCZIcc3BogbCvOiUuiyZWQptOYzqtZUvIrbg&usqp=CAU
"></img>
                <h6>Orange</h6>
              </div>
              <div className="c1">
                <img src="https://cdn.shopify.com/s/files/1/2045/8185/products/redhaven-
peach2_300x300.jpg?v=1642175876"></img>
                <h6>Peach</h6>
              </div>
              <div className="c1">
                <img
src="https://www.almanac.com/sites/default/files/styles/large/public/image_nodes/bell_peppers_dleon
is_gettyimages.jpg?itok=Z6m3cDpS"></img>
                <h6>Bell Pepper</h6>
              </div>
              <div className="c1">
                <img src="https://www.thespruce.com/thmb/6k5-ACTx6OXqmO--
k23ddujl9T4=/5750x0/filters:no_upscale():max_bytes(150000):strip_icc()/how-to-harvest-potatoes-
2540062-03-6ef71ddbaa974a2583d7ddf8d74f7c1e.jpg"></img>
                <h6>Potato</h6>
```

```
            </div>
            <div className="c1">
              <img src="https://www.gardeningknowhow.com/wp-
content/uploads/2021/07/ripening-raspberries.jpg"></img>
              <h6>Raspberry</h6>
            </div>
            <div className="c1">
              <img
src="https://d2jx2rerrg6sh3.cloudfront.net/images/news/ImageForNews_745986_16823072833517897.jpg"></img>
              <h6>Soybean</h6>
            </div>
            <div className="c1">
              <img src="https://hips.hearstapps.com/hmg-prod/images/is-squash-a-fruit-squash-
1561591075.jpg"></img>
              <h6>Squash</h6>
            </div>
            <div className="c1">
              <img src="https://m.media-amazon.com/images/I/71VEOI6MYGL.jpg"></img>
              <h6>Strawberry</h6>
            </div>
            <div className="c1">
              <img src="https://media.istockphoto.com/id/187110938/photo/garden-
tomatoes.jpg?s=612x612&w=0&k=20&c=b1K8Tifv9HqZ3JDnkF3H--
yT5wv7NhV2LfTkQGk5RcI="></img>
              <h6>Tomato</h6>
            </div>
          </div>

        </div>
      </>
    );
}

export default Home;
```

Disease-Detecction.js

```
import React, { useState } from "react";
import "./DiseaseDetection.css";

export default function DiseaseDetection() {
  const [selectedFile, setSelectedFile] = useState(null);
  const [file, setFile] = useState();
  const [fname, setFname] = useState("");
  const [isLoading, setLoading] = useState(false);

  const [title, setTitle] = useState("");
  const [desc, setDesc] = useState("");
  const [prevention, setPrevention] = useState("");
  const [image_url, setImage_url] = useState("");
  const [pred, setPred] = useState(0);
  const [sname, setSname] = useState("");

  const changeHandler = (event) => {
    setSelectedFile(event.target.files[0]);
    setFname(event.target.files[0].name);
    setFile(URL.createObjectURL(event.target.files[0]));
  };
```

```
const handleSubmit = (event) => {
  event.preventDefault();
  const formData2 = new FormData();
  formData2.append("file", selectedFile);

  const requestOptions = {
    method: "POST",
    body: formData2,
  };

  fetch("http://127.0.0.1:5000/detectDisease", requestOptions)
    .then((response) => response.json())
    .then(function (response) {
      let a = response;
      console.log(a);
      setLoading(true);
      setTitle(response.title);
      setDesc(response.desc);
      setPrevention(response.prevention);
      setImage_url(response.image_url);
      setPred(response.pred);
      setSname(response.sname);
    })
    .catch((err) => {
      setTitle(
        "Error occured!! Please check your connection or upload a valid image."
      );
    });
};
return (
  <div>
    {!isLoading ? (
      <>
        <center>
          <form onSubmit={handleSubmit}>
            <div className="upload-box">
              <input
                type="file"
                id="image"
                onChange={changeHandler}
                style={{ display: "none" }}
              />

              <label for="image">
                <i class="fa-solid fa-cloud-arrow-up fa-xl"></i>
              </label>
            </div>
            <div className="image-box">
              <div className="image-preview">
                <img src={file} alt=""/>
                <p>{fname}</p>
              </div>

              <button type="submit" className="btn">
                SUBMIT
              </button>
            </div>
          </form>
        </center>
```

```
  </>
) : (
  <div>
    <div className="container">
      <div className="disease-title">
        <h1>
          <b>{title} </b>
        </h1>
      </div>
      <br />
      <div className="disease-image">
        <img src={image_url} className="disease-img" alt=""/>
      </div>

      <br />
      <div className="disease-box">
        <div className="disease-desc">
          <h5>
            {pred === 3 ||
            pred === 5 ||
            pred === 7 ||
            pred === 11 ||
            pred === 15 ||
            pred === 18 ||
            pred === 20 ||
            pred === 23 ||
            pred === 24 ||
            pred === 25 ||
            pred === 28 ||
            pred === 38 ? (
              <b> Tips to Grow Healthy Plants :</b>
            ) : (
              <b>Brief Descritpion :</b>
            )}
          </h5>
          <p className="paragraph">{desc}</p>
        </div>

        <div className="disease-prevention">
          <h5>
            {pred === 3 ||
            pred === 5 ||
            pred === 7 ||
            pred === 11 ||
            pred === 15 ||
            pred === 18 ||
            pred === 20 ||
            pred === 23 ||
            pred === 24 ||
            pred === 25 ||
            pred === 28 ||
            pred === 38 ? (
              <b> Benefits :</b>
            ) : (
              <b>Prevent This Plant Disease By follow below steps :</b>
            )}
          </h5>
          <p className="paragraph">{prevention}</p>
        </div>
      </div>
```

```
          <div className="supplement">
            {pred !== 4 ? (
              <>
                <h5>
                  {pred === 3 ||
                  pred === 5 ||
                  pred === 7 ||
                  pred === 11 ||
                  pred === 15 ||
                  pred === 18 ||
                  pred === 20 ||
                  pred === 23 ||
                  pred === 24 ||
                  pred === 25 ||
                  pred === 28 ||
                  pred === 38 ? (
                    <b> Fertilizer :</b>
                  ) : (
                    <b>Supplements :</b>
                  )}
                </h5>

                <p className="paragraph">{sname}</p>
              </>
            ) : (
              <center></center>
            )}
          </div>
        </div>
      </div>
    )}
  </div>
 );
}
```

Contact.js

```
import "./Contact.css";
function Contact() {
 return (
  <>
    <div className="header">
      <h1>Contact the Developers 👨🏻‍💻 </h1>
    </div>
    <div className="developers">
      <div className="dev">
        <h2>Gurman Singh</h2>
        <div className="links">
          <i className="fa-solid fa-envelope"></i>
          <a href="mailto:gurman29singh@gmail.com" target="_blank" rel="noopener noreferrer">
            gurman29singh@gmail.com
          </a>
        </div>
        <div className="links">
          <i className="fa-brands fa-square-github"></i>
          <a href="https://github.com/29th-pixel" target="_blank" rel="noopener noreferrer">
          29th-pixel
          </a>
```

```
        </div>
        <div className="links">
          <i className="fa-brands fa-linkedin"></i>
          <a href="https://www.linkedin.com/in/gurman-singh-8b94a21ba/" target="_blank"
rel="noopener noreferrer">
          gurman-singh-8b94a21ba
          </a>
        </div>
      </div>
      <div className="dev">
      <h2>Harneet Singh Channa</h2>
        <div className="links">
          <i className="fa-solid fa-envelope"></i>
          <a href="mailto:harneetsinghch110@gmail.com" target="_blank" rel="noopener noreferrer">
          harneetsinghch110@gmail.com
          </a>
        </div>
        <div className="links">
          <i className="fa-brands fa-square-github"></i>
          <a href="https://github.com/hsc-code" target="_blank" rel="noopener noreferrer">
          hsc-code
          </a>
        </div>
        <div className="links">
          <i className="fa-brands fa-linkedin"></i>
          <a href="https://www.linkedin.com/in/harneet-singh-channa-8473831ba/" target="_blank"
rel="noopener noreferrer">
          harneet-singh-channa-8473831ba
          </a>
        </div>
      </div>

      <div className="dev">
      <h2>Kuldeep Singh</h2>
        <div className="links">
          <i className="fa-solid fa-envelope"></i>
          <a href="mailto:kuldeepsingh2670@gmail.com" target="_blank" rel="noopener noreferrer">
          kuldeepsingh2670@gmail.com
          </a>
        </div>
        <div className="links">
          <i className="fa-brands fa-square-github"></i>
          <a href="https://github.com/kuldeep2670" target="_blank" rel="noopener noreferrer">
          kuldeep2670
          </a>
        </div>
        <div className="links">
          <i className="fa-brands fa-linkedin"></i>
          <a href="https://www.linkedin.com/in/kuldeep2670/" target="_blank" rel="noopener
noreferrer">
          kuldeep2670
          </a>
        </div>
      </div>

      <div className="dev">
      <h2>Sahibjot Singh</h2>
        <div className="links">
          <i className="fa-solid fa-envelope"></i>
          <a href="mailto:sahibjot7180@gmail.com" target="_blank" rel="noopener noreferrer">
```

```
          sahibjot7180@gmail.com
          </a>
        </div>
        <div className="links">
          <i className="fa-brands fa-square-github"></i>
          <a href="https://github.com/sahibjot366" target="_blank" rel="noopener noreferrer">
          sahibjot366
          </a>
        </div>
        <div className="links">
          <i className="fa-brands fa-linkedin"></i>
          <a href="https://www.linkedin.com/in/sahibjot-singh/" target="_blank" rel="noopener
noreferrer">
          sahibjot-singh
          </a>
        </div>
      </div>
    </div>
  </>
 );
}
export default Contact;
```

Footer.js

```
import './Footer.css';
function Footer() {
 return (
  <footer className='footer'>
    <a
      href='https://github.com/29th-pixel/Plant-Disease-Detection-App'
      target='_blank'
      rel='noopener noreferrer'
    >
      <i class='fa-brands fa-github fa-fade' style={{color: '#FFFFFF'}}></i>
    </a>
  </footer>
 );
}
export default Footer;
```

Front-End – Android App

index.tsx

```
export {default as HomeScreen} from './home-screen'
export {default as DiseaseDetectorScreen} from './disease-detector'
export {default as ContactScreen} from './contact-screen'
```

home-screen.tsx

```
import React from 'react'
import {Text,StyleSheet,ScrollView,View } from 'react-native'
import {SafeAreaProvider} from 'react-native-safe-area-context';
import { commonStyles } from '../../utils/styles/common-styles';
import { data as imageData } from '../../utils/image-data';
import ImageItem from '../../components/image-item';
import Button from '../../components/button';

import type {NavigationProp} from '@react-navigation/native'
import type { HomeStackParamList } from '../../navigator/home-navigator';
const HomeScreen = ({navigation}:{navigation:NavigationProp<HomeStackParamList>}) => {

 const buttonStyle={
   backgroundColor:'#00FF00',
   width:60,
   height:60,
   borderRadius:100,
   shadowColor: '#000',
   shadowOffset: { width: 0, height: 2 },
   shadowOpacity: 1,
   shadowRadius: 8,
   elevation: 16,
   position:'absolute',
   bottom:16,
   right:16}

 return (
   <SafeAreaProvider style={[commonStyles.parentContainer]}>
     <ScrollView contentContainerStyle={styles.scrollview}>
       <Text style={[commonStyles.titleText]}>🏵️ Plant Disease Detection 🏵️</Text>
       <Text style={[commonStyles.text,styles.description]}>Plant Disease Detection is necessary for
every farmer so we are created Plant disease detection using Deep learning. In which we are using
Convolutional Neural Network for classifying Leaf images into 39 Different Categories. The
Convolutional Neural Code build in Pytorch Framework. For Training we are using Plant village
dataset</Text>

       <View style={{ width:'90%' }}>
         {imageData.map((image, index) => {
          if (index % 2 === 0) {
            return (
              <View key={image.title} style={{ flexDirection: 'row' }}>
                <ImageItem imageData={image} />
                {index + 1 < imageData.length && (
                  <ImageItem imageData={imageData[index + 1]} />
                )}
              </View>
            );
          }
```

```
          return null;
        })}
      </View>
    </ScrollView>
    <Button title='Try It !!' style={buttonStyle}
onPress={()=>{navigation.navigate('DiseaseDetectorScreen')}} />
   </SafeAreaProvider>
  )
}
const styles=StyleSheet.create({
  scrollview:{
    alignItems:'center'
  },
  description:{
    color:'#000000',
    marginHorizontal:8,
    marginVertical:8
  }
})
export default HomeScreen
```

disease-detector.tsx

```
import React,{useState} from 'react'
import { SafeAreaProvider } from 'react-native-safe-area-context'
import {StyleSheet,Image,Text,ScrollView,Platform} from 'react-native'
import { commonStyles } from '../../utils/styles/common-styles'
import Button from '../../components/button'
import { BUTTON_COLOR } from '../../utils/constants/colors'
import ImagePicker, { ImageOrVideo } from 'react-native-image-crop-picker';
import OutputModal from '../../components/output-modal'

export type outputData={
  buy_link:string;
  desc:string;
  image_url:string;
  pred:number;
  prevention:string;
  simage:string;
  sname:string;
  title:string;
}
const DiseaseDetectorScreen = () => {
  const buttonStyle={
    backgroundColor:BUTTON_COLOR,
    marginVertical:8
  }
  const [selectedImage,setSelectedImage]=useState<ImageOrVideo>()
  const [output,setOutput]=useState<outputData>();
  const createFormData = (image:ImageOrVideo | undefined) => {

    const formData = new FormData();
    formData.append('file', {
      uri: image?.path,
      type: image?.mime,
      name: 'image.jpg',
    });
    return formData;
  };
```

```
  const openCamera=()=>{
    setOutput(undefined)
    ImagePicker.openCamera({
      width: 250,
      height: 250,
      cropping: false,
    }).then(image => {
      setSelectedImage(image)
    });
  }
  const openGallery=()=>{
    setOutput(undefined)
    ImagePicker.openPicker({
      multiple: false
    }).then(image => {
      setSelectedImage(image)
    });
  }
  const computeOutput=()=>{

    const formData = createFormData(selectedImage);
    fetch('http://127.0.0.1:5000/detectDisease', {
      method: 'POST',
      body: formData,
      headers: {
        'Content-Type': 'multipart/form-data',
      },
    })
      .then((response) => response.json())
      .then((result) => {
        setOutput(result)
      })
      .catch((error) => {
        console.log('Error occured!')
      });
  }
  return (
    <SafeAreaProvider style={[commonStyles.parentContainer,{paddingHorizontal:0}]}>
      <ScrollView contentContainerStyle={style.scrollview} showsVerticalScrollIndicator={false}>
        <Text style={[commonStyles.titleText]}>🏵️ Plant Disease Detector 🏵️</Text>
        <Image source={require('../../assets/images/plant.png')}
style={[commonStyles.image,style.image]} />
        <Button title='Open Camera' onPress={openCamera} style={buttonStyle} />
        <Button title='Select from gallery' onPress={openGallery} style={buttonStyle} />
        {selectedImage?.path?
        <>
        <Image source={{uri:selectedImage.path}} style={[commonStyles.image,style.plantImage]} />
        <Text style={style.name}>{selectedImage.filename}</Text>
        <Button title='FIND DISEASE' onPress={computeOutput} style={buttonStyle} />
        </>
        :
        null}
      </ScrollView>
      <OutputModal visible={output!=undefined} onClose={()=>{setOutput(undefined)}}
output={output} />
    </SafeAreaProvider>
  )
}
const style=StyleSheet.create({
  image:{
```

```
    width:180,
    height:180
  },
  plantImage:{
    width:250,
    height:250,
    borderRadius:8
  },
  scrollview:{
    alignItems:'center',
  },
  name:{
    marginBottom:8
  }
})
export default DiseaseDetectorScreen
```

contact-screen.tsx

```
import React,{useCallback} from 'react'
import { Text,FlatList } from 'react-native';
import {SafeAreaProvider} from 'react-native-safe-area-context';
import { commonStyles } from '../../utils/styles/common-styles';
import { data as contactData } from '../../utils/contact-data';
import ContactItem from '../../components/contact-item';
import { BACKGROUND_COLOR } from '../../utils/constants/colors';
export type contactDataType={
  name:string;
  email:string;
  github:string;
  linkedin:string;
}

const ContactScreen = () => {
  const renderItem=useCallback(({item}:{item:contactDataType})=>{
    return (
      <ContactItem data={item} />
    )
  },[])
  return (
    <SafeAreaProvider
style={[commonStyles.parentContainer,{backgroundColor:BACKGROUND_COLOR,}]}>
      <Text style={commonStyles.titleText}>Our Team of Developers 👨‍💻 </Text>
      <FlatList
        data={contactData}
        keyExtractor={item=>item.email}
        renderItem={renderItem}
      />
    </SafeAreaProvider>
  )
}
export default ContactScreen
```