PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

Capstone Project Report (Phase-2)

on

Disease Prediction in Citrus Fruit

Submitted by

GANIKA KODNANI- (PES1PG22CA072)

Feb 2024 – June 2024

under the guidance of

Guide Details

Ms. Deepika D

Assistant Professor

Department of Computer Applications,

PESU Bengaluru– 560085

FACULTY OF ENGINEERING DEPARTMENT OF COMPUTER APPLICATIONS PROGRAM – MASTER OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the project entitled

Disease Prediction in Citrus Fruit

is a bonafide work carried out by

GANIKA KODNANI- (PES1PG22CA072)

In partial fulfilment for the completion of Capstone Project, Phase-1 work in the Program of Study MCA under rules and regulations of PES University, Bengaluru during the period Feb 2024 – Jun 2024. The project report has been approved as it satisfies the academic requirements of 4th semester MCA.

Internal Guide	Chairperson
Ms. Deepika D	Dr. Veena s
Department of Computer Applications,	Department of Computer Application,
PES University, Bengaluru - 560085	PES University, Bengaluru - 560085

DECLARATION

I, Ganika Kodnani, bearing PES1PG22CA072 hereby declare that the Capstone project phase-2 entitled, Disease Prediction in Citrus Fruit, is an original work done by me under the guidance of

Ms. Deepika D, Assistant Professor, PES University, and is being submitted in partial fulfilment of the requirements for completion of 4th Semester course in the Program of Study MCA. All corrections/suggestions indicated for internal assessment have been incorporated inthe report.

The plagiarism check has been done for the report and is below the given threshold.

I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other course.

PLACE:

DATE:

GANIKA KODNANI

ACKNOWLEDGMENT

I would like to express my profound gratitude for the opportunity to work on the project titled "Disease Prediction in Citrus Fruit ." This project has been an incredible journey, providing me with invaluable knowledge, experiences, and insights. It is a blessing to have had the space and time to dedicate myself fully to this endeavor.

First, my heartfelt gratitude and respect to **Dr. J Suryaprasad,** Vice Chancellor of PES University and to **Dr. Veena S,** Chairperson, Department of Computer Applications. With utmost thanks and dedication, I would like to thank my guide **Ms Deepika D,** where she was aside in every step of work that I have done and with some important advices and corrective measures.

I would also like to extend my thanks and gratitude to every faculty of Department of Computer Applications and to my family inmates and friends who were concerned for the project.

Thank you everyone.

GANIKA KODNANI

ABSTRACT

The project named “Prediction of Diseases in Citrus Fruits” aims at establishing an automated system for detecting a disease from an image of citrus fruit. Early symptoms diagnosis is crucial for effective control and prevention of such diseases since they lead to huge economic losses. These conventional methods are also time-consuming and often unreliable.

To overcome these problems, this study employs deep learning specifically convolutional neural networks (CNNs) for automating disease detection. This involves dataset collection and preprocessing which includes steps like resizing, normalization and augmentation’s optimization model performance. The CNN model is trained using this data set and its performance evaluated using accuracy, precision, recall, F1-scores among other metrics.

Deep learning models were able to accurately identify diseases in citrus fruits according to the results of the project. It works faster than traditional methods while being more accurate as well as scaling up easily. In fact, it can be used for early recognition of diseases or precision agriculture in general.

However, the future recommendations include increasing data size; combining multiple sources; implementing the model on edge devices for efficient use.

INDEX

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

It is a project that aims to improve the management of citrus crops by developing a deep learning based disease detection system for citrus fruits. The significance of detecting diseases in citrus fruits has been detailed, as well as the various limitations of traditional manual inspection methods and the need for automated and efficient ones.

To recognize diseases properly, Convolutional Neural Networks (CNNs) which are types of deep learning are discussed. Basically, deep learning models can understand what is necessary in an image and use this knowledge to identify a disease with precision. In addition, Python, TensorFlow or PyTorch, OpenCV and Pillow are beneficial tools referred to in the presentation when modeling software development, image preprocessing and data augmentation techniques respectively.

Thus; frontend technologies such as HTML CSS JavaScript while backend technologies include Python (Flask or Django) or Node.js (Express.js). Consequently, farmers through this interface can always access the disease detection systems because they receive immediate information about any outbreak.

1.1 Project description

Problem Definition

The laborious and manual procedure of identifying illnesses in citrus fruits is the issue this initiative attempts to solve. The visual inspection and manual diagnosis used in traditional illness detection techniques are labor-intensive, subjective, and frequently prone to error. The citrus industry may suffer financial losses as a result of this inefficiency in disease identification, which impedes efficient disease management and prevention.

Proposed Solution

The proposed solution for Disease Prediction in Citrus Fruit using machine learning can be:

- Develop a dataset of labeled citrus fruit images, including healthy and diseased samples.
- Train a machine learning model, such as a convolutional neural network (CNN), using the dataset to classify citrus fruit diseases.
- Implement an image processing pipeline to preprocess acquired images for noise reduction and enhancement.
- Deploy the trained model into a user-friendly interface to enable users to upload or capture images and receive disease detection results with corresponding disease labels or probabilities.

Purpose

"Disease Prediction in Citrus Fruit" is a project aimed at creating a machine learning-based system for precise illness detection and classification in citrus fruits. Preventing crop losses and maintaining healthy plant growth are dependent on the project's primary goal, which is to develop an automated and effective method for early disease identification in fruits.

The project's goal is to analyse and categorise visual signs and patterns linked to different citrus illnesses that damage fruits by utilising machine learning algorithms, such as deep learning or conventional machine learning techniques. Farmers and agricultural specialists can use this to quickly detect and diagnose problems affecting citrus fruits, enabling them to implement the necessary disease control strategies, such as focused treatments or preventive actions.

The goal of the project is to provide a reliable and user-friendly tool that can assist farmers, researchers, and agricultural professionals in monitoring and managing plant health, improving crop productivity, and reducing economic losses caused by diseases affecting citrus fruits.

Scope

"Disease Prediction in Citrus Fruit" is a project whose goal is to create an intelligent system that can precisely identify and categorise diseases that impact citrus fruits. The approach include preprocessing and improving citrus sample photos, collecting pertinent features, and using a variety of annotated datasets to train machine learning models like random forests, support vector machines, and convolutional neural networks (CNNs). The aim is to differentiate between healthy and diseased citrus samples based on learned disease patterns. The project may also include the development of a real-time disease detection system, allowing users to upload images for immediate analysis. Performance evaluation using appropriate metrics will be conducted to assess the system's effectiveness. The scope of the project is subject to specific requirements, available resources, and time constraints, necessitating clear objectives and deliverables for successful completion.

2. LITERATURE SURVEY

2.1 Domain Survey

The fields of agriculture and the citrus industry encompass this initiative. Citrus fruit disease detection and control will be made easier for farmers, agricultural researchers, and crop management experts with the help of the established method. By using the method, crop health can be enhanced, yield can be increased, and financial losses from disease outbreaks can be decreased.

Convolutional neural networks (CNNs), a type of deep learning, and image analysis methods are both included in this research. To enable automatic illness identification, a model will be trained with a collection of photographs of citrus fruits using deep learning methods. To improve the deep learning model's accuracy and performance, the project may also make use of methods including data augmentation, image preprocessing, and assessment metrics. The project may also require the use of libraries and software frameworks.

2.2 Related Work

Research Paper 1

Title: Citrus Leaf Disease Detection Using Hybrid CNN-RF Model

Author: HEENA KALIM, ANURADHA CHUG, AMIT PRAKASH SINGH

Publisher: IEEE

Key findings:
The proposed hybrid model of Convolutional Neural Network (CNN) and Random Forest (RF) achieved an accuracy of 87% in detecting citrus leaf diseases such as Black Spot, Canker, and Greening. The VGG16-Random Forest algorithm outperformed the ResNet50-Random Forest and InceptionV3-Random Forest models.

Approach:
The researchers used a CNN model as a feature extractor and a Random Forest classifier to identify citrus leaf diseases. They trained and tested the model using a dataset of 58 healthy and 536 unhealthy citrus leaf images.

Conclusion:
The study demonstrates the effectiveness of the hybrid CNN-RF model in accurately detecting citrus leaf diseases, which is crucial for improving citrus fruit yields and reducing economic losses. The findings suggest that the proposed approach can be a valuable tool for automated disease detection in precision agriculture.

Research Paper 2

Title: Classification of Diseases in Citrus Fruits using SqueezeNet

Author: EJAZ KHAN, MUHAMMAD ZIA UR REHMAN, FAWAD AHMED, MUHAMMAD ATTIQUE

Publisher: IEEE

Key Findings: The paper proposes a deep learning-based approach for the classification of six different citrus diseases using the SqueezeNet model. The SqueezeNet model outperformed the MobileNetV2 model, achieving an accuracy of 96%.

Approach: The authors performed data augmentation to expand the privately acquired dataset. They then retrained two lightweight pre-trained deep learning models, SqueezeNet and MobileNetV2, on the augmented dataset. Features were extracted from the retrained models and optimized using the Whale Optimization Algorithm.

Conclusion: The proposed citrus fruit disease classification technique using the SqueezeNet model outperformed a current technique in the literature, demonstrating the effectiveness of the deep learning-based approach for accurate and efficient detection and classification of citrus diseases.

Research Paper 3

Title: Fruit Diseases Identification and Classification Using Deep Learning Model

Author: MOHAMMED AHMED MATBOLI, AYMAN ATIA

Publisher: IEEE

Key Findings: The paper presents a solution to identify and classify common fruit diseases using deep learning models. Five different transfer learning models were evaluated, with the customized CNN model achieving the highest accuracy of 99.16%.

Approach: The researchers used transfer learning techniques to build a deep learning model for fruit disease identification and classification. They compared the performance of various transfer learning models and proposed a customized CNN model that outperformed the existing approaches.

Conclusion: The proposed solution can help reduce the errors and challenges in accurately detecting fruit diseases by the human eye. The deep learning-based approach provides an automated and reliable system for early detection and classification of common fruit diseases.

Research Paper 4

Title: Disease Detection of Citrus Plants Using Image Processing Techniques

Author: GENDLAL VAIDYA, SHUBHALAKSHMEE WARUTKAR, NEHA CHAUDHARY, SHREESH KAWATHEKAR, SNEHAL NASARE, SANKET DHENGRE
Publisher: IEEE

Key Findings: The paper presents an overview of various image processing and machine learning techniques for detecting and classifying diseases in citrus plants. The proposed model uses a machine learning-based approach, specifically the CNN algorithm, to detect diseases on citrus plant leaves, stems, and fruits.
Approach: The authors created a dataset of citrus plant diseases and employed techniques like image pre-processing, segmentation, feature extraction, and machine learning algorithms such as K-means, GLCM, and SVM for disease detection and classification.
Conclusion: The proposed model was able to successfully identify and categorize various citrus plant diseases, demonstrating the potential of automated disease detection using image processing and machine learning techniques in the agriculture industry.

Research Paper 5

Title: Deep Metric Learning Based Citrus Disease Classification With Sparse Data

Author: SIVASUBRAMANIAM JANARTHAN, SELVARAJAH THUSEETHAN, SUTHARSHAN RAJASEGARAR, QIANG LYU, YONGQIANG ZHENG, AND JOHN YEARWOODDATE
Publisher: IEEE

Key Findings: The paper proposes a lightweight, deep metric learning-based approach for citrus disease classification, which can effectively handle sparse data. The proposed method achieved high classification accuracy of 95.04% while requiring fewer parameters (less than 2.3 M) and being time-efficient (less than 10 ms) in detecting the citrus diseases.
Approach: The authors developed a deep learning-based architecture that includes an embedding module, a cluster prototype module, and a simple neural network classifier. They also employed a patch-based classification mechanism to improve the detection performance.

Conclusion: The proposed approach demonstrates the ability to learn with fewer resources and without compromising accuracy, making it suitable for deployment on resource-constrained devices like mobile phones for practical citrus disease monitoring by farmers.

2.3 Existing Systems

Citrus Doctor: Citrus Doctor is a mobile application developed for citrus farmers. It utilizes image recognition technology to identify diseases in citrus plants.

Plantix: Plantix is a mobile app that supports the identification of plant diseases in various crops, including citrus.

2.4 Technology Survey

Machine Learning: Machine learning is a technique in artificial intelligence concerned with the development of algorithms and statistical methods through which computers can learn from experience or data. The process allows them to make better decisions without being explicitly programmed on specific tasks.

The main idea behind machine learning is that computers can learn by themselves. Instead of humans telling the computer exactly what to do for every situation, we give the computer lots of information (data). The computer then looks at this data and finds patterns on its own. This is really helpful when there are too many situations to write rules for, or when the rules are too complicated for humans to figure out.

There are three primary sorts of machine learning:

Supervised learning: In these models are trained using labelled data. In these models need to find the mapping function to map the input variables. Supervised learning needs a mentor to train the model. This method can be used for classification and regression.

Unsupervised learning: In this method, you find patterns in data that is not labelled. Structures and patterns can be found in data by using this method without requiring an expert's advice because it develops its insights. This way could be applied in such activities like grouping similar objects (clustering) or determining connections between objects (association).

Reinforcement Learning: The calculation learns by connection with an environment, accepting input in the shape of rewards or punishments for its activities. The objective is to learn a approach that maximizes the aggregate compensate.

NumPy: NumPy is a Python library that specializes in working with arrays and matrices. It offers a wide range of functions and tools for performing numerical computations and data processing tasks efficiently. NumPy is a powerful and versatile library that enables efficient array-oriented computing and numerical processing in Python, making it a fundamental tool for scientific computing and data analysis.

Pandas: It is a python library used for working with data sets.it has multiple functionalities such as analysing, cleaning, exploring and manipulating data. Pandas can delete rows that are not relevant or might contain wrong values or null values.

Scikit-Learn: It is a machine learning library for the python programming language. It has a large number of algorithms that can be readily deployed by programmers and data scientists in machine learning models.

Keras: Keras is an open-source library that gives a high-level Python interface for working with fake neural systems. It serves as a user-friendly wrapper around lower-level profound learning systems, permitting engineers to effortlessly construct and prepare complex neural organize models. keras is a capable and adaptable library that disentangles the prepare of building and preparing complex neural arrange models in Python, making it a prevalent choice for both apprentice and experienced profound learning professionals.

Image Processing: Image processing techniques are commonly employed to enhance and analyze images of citrus fruits. This includes operations such as image segmentation, feature extraction, and image enhancement to improve disease detection accuracy.

CNN: Convolutional neural networks are abie to analyse cata much in the same way a human brain does, through supervised learning. A CNN comprises - an input layer, an output layer and many intermediate layers for data processing. Convolu-tional layers operate mathematically to examine and pull-out aspects or details in data transported to them covertly. For the stackable nature of CNN, all the features are transferred from one layer to another. Indeed, this is one of the ways that CNN manages to prevent overfitting in which it does so by analysing hierarchical structures that are within the underlying dataset causing them to decompose as they move from being simple to more complicated across different stages of processing. The input could be an image, for instance, given as tensor which is a multi-dimensional array where channels correspond to different parts of data. These feature maps are then produced by convolutions that convert the input mentioned above ilso feature maps capable of capturing the essential patterns and traits. The process is repeated in multiple levels, modelled after how neurons work in the human brain; because all the layers are interconnected in CNNs, whatever the different layers output is merged in the last layer to form a single output; and the more intricate characteristics are coped with the increased number of channels in the feature maps.

To sum it up. CNNs are robust artificial neural networks, which will easily recognize the data features. This by splitting information into distributed fragments. Just like how human brains understand messages.

Flask: A web application framework or a simply Web Framework represents a collection of libraries and modules that enables web application developers to write applications without worrying about low-level details such as protocol, thread management. Flask is often referred to as a micro framework.it is designed to keep the core of the application simple and scalable.

Matplotlib: Matplotlib is a library for making charts and intuitively visualizations in python. Matplotlib makes things simple and difficult things conceivable.

TensorFlow: TensorFlow is a free and open-source computer program library for machine learning and manufactured insights. It can be utilized over a run of assignments but has a specific centre on preparing and induction of profound neural networks.

3. HARDWARE AND SOFTWARE REQUIREMENTS

3.1 Hardware Requirements

Hardware Component	Minimum Requirements
Processor (CPU)	Dual core processor
Memory (RAM)	8GB RAM
Networking	Stable internet connection with adequate bandwidth.
Hard Disk	454 GB

3.2 Software Requirements

Software Requirements

Browsers	Google Chrome
OS	Windows 8 and above

Tools And Technologies

Language	Python 3.10.0
IDE	Visual Studio Code
ML Libraries	TensorFlow v2.11.0
HTML version	HTML5
CSS version	CSS3
Web Framework	Flask 2.2.2

4. Software Requirements Specification

4.1 Users

General users: User interacts with the user interface give the image input and get the predicted result.

4.2 Functional Requirements

Image Dataset Collection: The system ought to incorporate a mechanism for gathering and organising photographs of citrus fruits impacted by various illnesses.

Image Pre-processing: To improve the acquired photos' quality and get rid of any noise or artefacts, the system should preprocess them.

Feature Extraction: The preprocessed images should have pertinent features that the system can extract.

Model Training: The system should train a machine learning model using the prepared dataset.

Disease Classification: The trained model should be capable of classifying the input images into different citrus disease categories accurately. It should provide the disease label or a probability distribution over the possible disease classes for each input image.

User Interface: The system should provide a user-friendly interface for users to interact with the application.

4.3 Non-Functional Requirements

Performance: Real-time or almost real-time performance requires the system to be able to process photos and identify diseases in an acceptable amount of time.

Precision: To reduce the number of false positives and false negatives, the system's disease detection should be highly accurate.

Usability: People with differing degrees of technical proficiency should be able to easily navigate and utilise the user interface.

Maintainability: Easy upgrades, bug patches, and enhancements should be possible because to the system's modular and manageable design.

5. SYSTEM DESIGN

5.1 Architecture Diagram

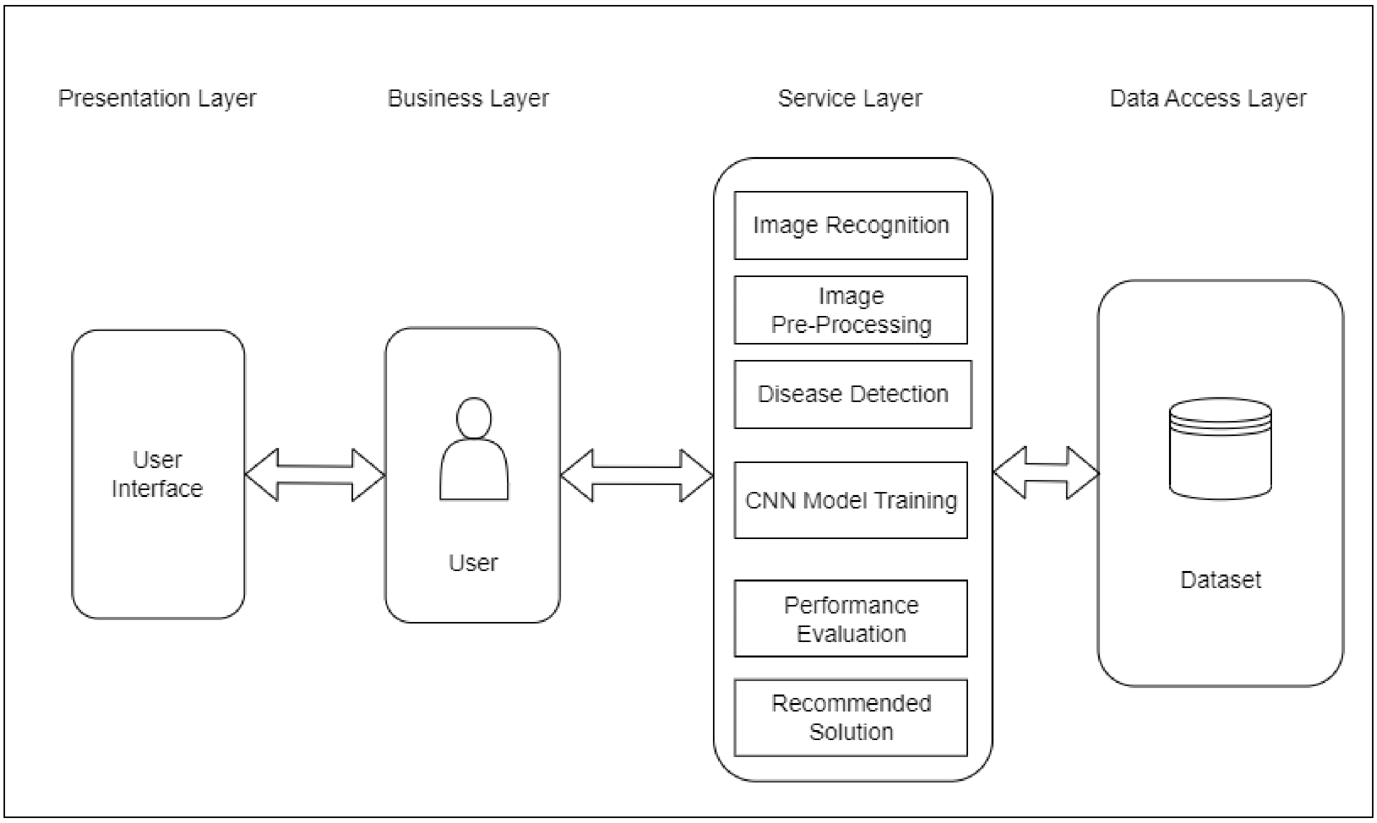


Figure 5.1

Presentation Layer: This is the topmost layer which is responsible for the user interface. It's where the user interacts with the system.

Business Layer: This layer indicates the user's interaction with the system.

Service Layer: This layer lists various services that the user can interact with. These services include image recognition, image pre-processing, disease detection, CNN model training. The system assists in identifying and managing citrus fruit diseases using image recognition techniques.

Data Service Layer: This is the bottom layer, represented by Parse.

5.2 Block Diagram / Context Flow Diagram

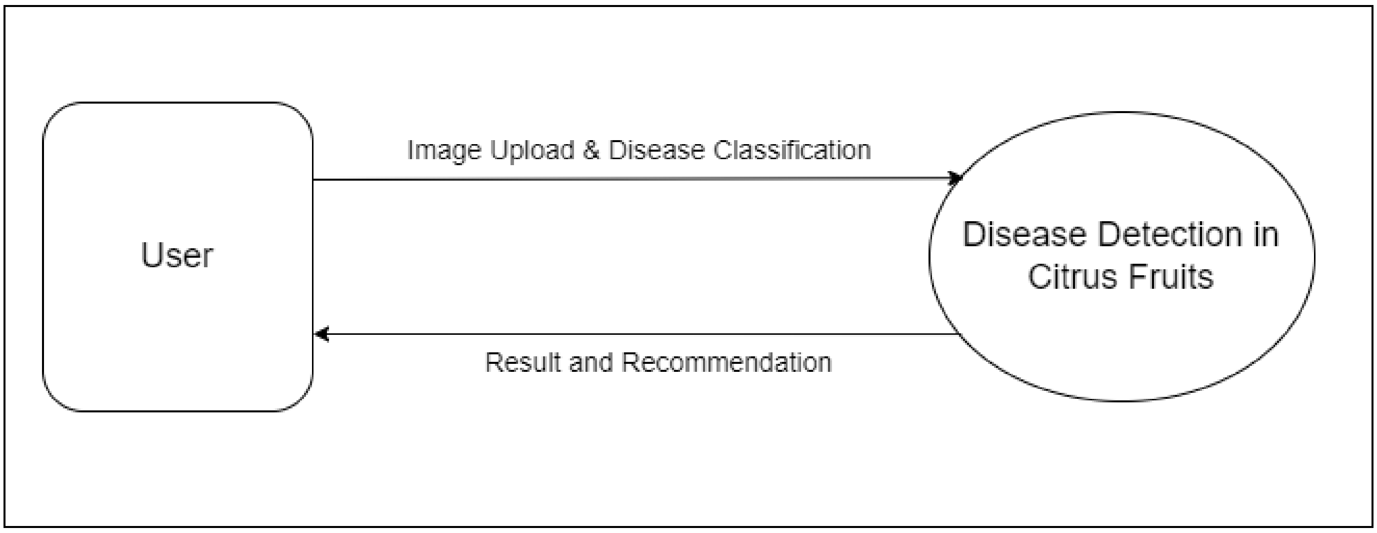

User
Image Upload & Disease Classification
Disease Detection in Citrus Fruits
Result and Recommendation

Figure 5.2

User: The process begins with a user who wants to identify diseases in citrus fruits.

Image Upload & Disease Classification: The user uploads an image of a citrus fruit (presumably showing signs of disease). This image is then analyzed for disease classification.

Disease Detection in Citrus Fruits: The analysis phase involves identifying the specific disease affecting the citrus fruit. Various algorithms or techniques may be used to determine the disease type.

Result and Recommendation: Based on the analysis, the system provides results and recommendations to the user. These recommendations could include treatment options, preventive measures, or further steps to take.

6. DETAILED DESIGN

6.1 Process Flow Diagram

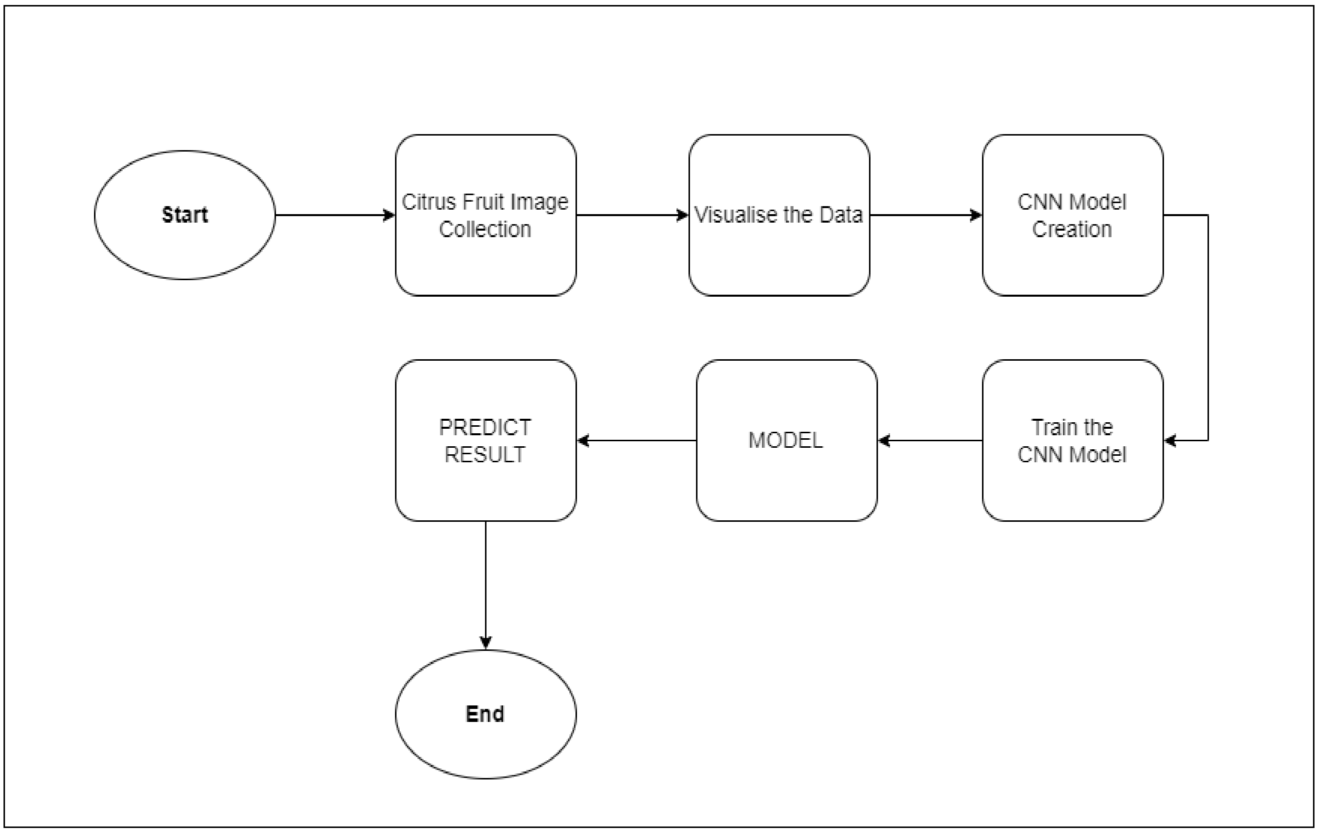


Figure 6.1

Start: Initiation of the project.
Citrus Fruit Image Collection: Gather a dataset of citrus fruit images, which will be used for analysis and model training.
Visualise the Data: Analyze and visualize the dataset to understand its characteristics and distribution, aiding in effective preprocessing and model training.
CNN Model Creation: Define and create the Convolutional Neural Network (CNN) model architecture, specifying the layers and hyperparameters.
Train the CNN Model: Train the CNN model using the preprocessed dataset, allowing the model to learn patterns and features that distinguish healthy fruits from diseased ones. Evaluate performance with metrics like accuracy, precision, recall, and F1-score.
MODEL: This step represents the trained model, ready for use in making predictions.
Predict Result: Use the trained model to predict the health status of citrus fruits in new images, determining if they are healthy or diseased.
End: Conclusion of the process. Use the predictions for applications such as early disease detection and precision agriculture, enabling timely interventions and better crop management.

6.2 Use Case Diagram

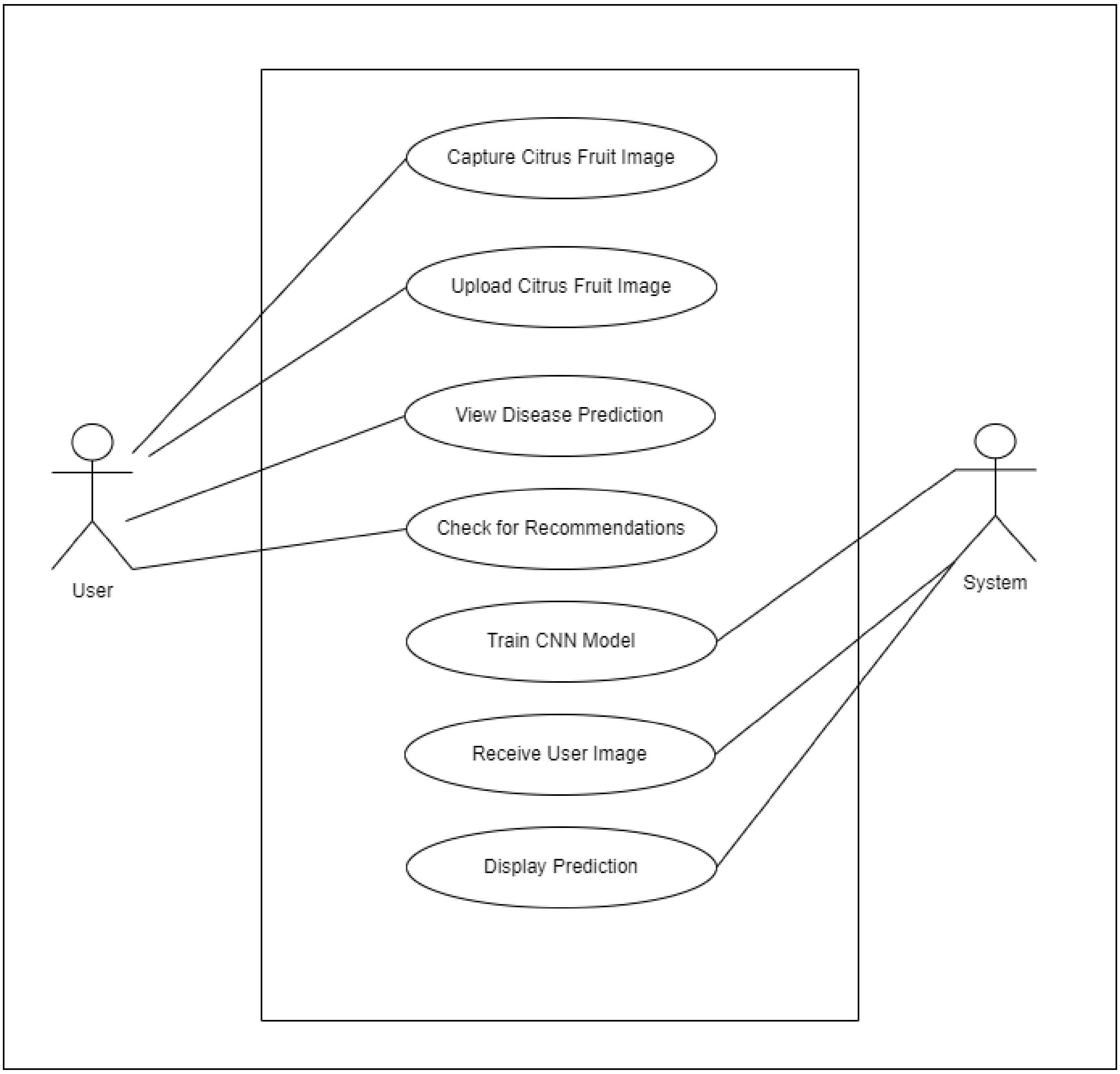


Figure 6.2

6.3 Activity Diagram

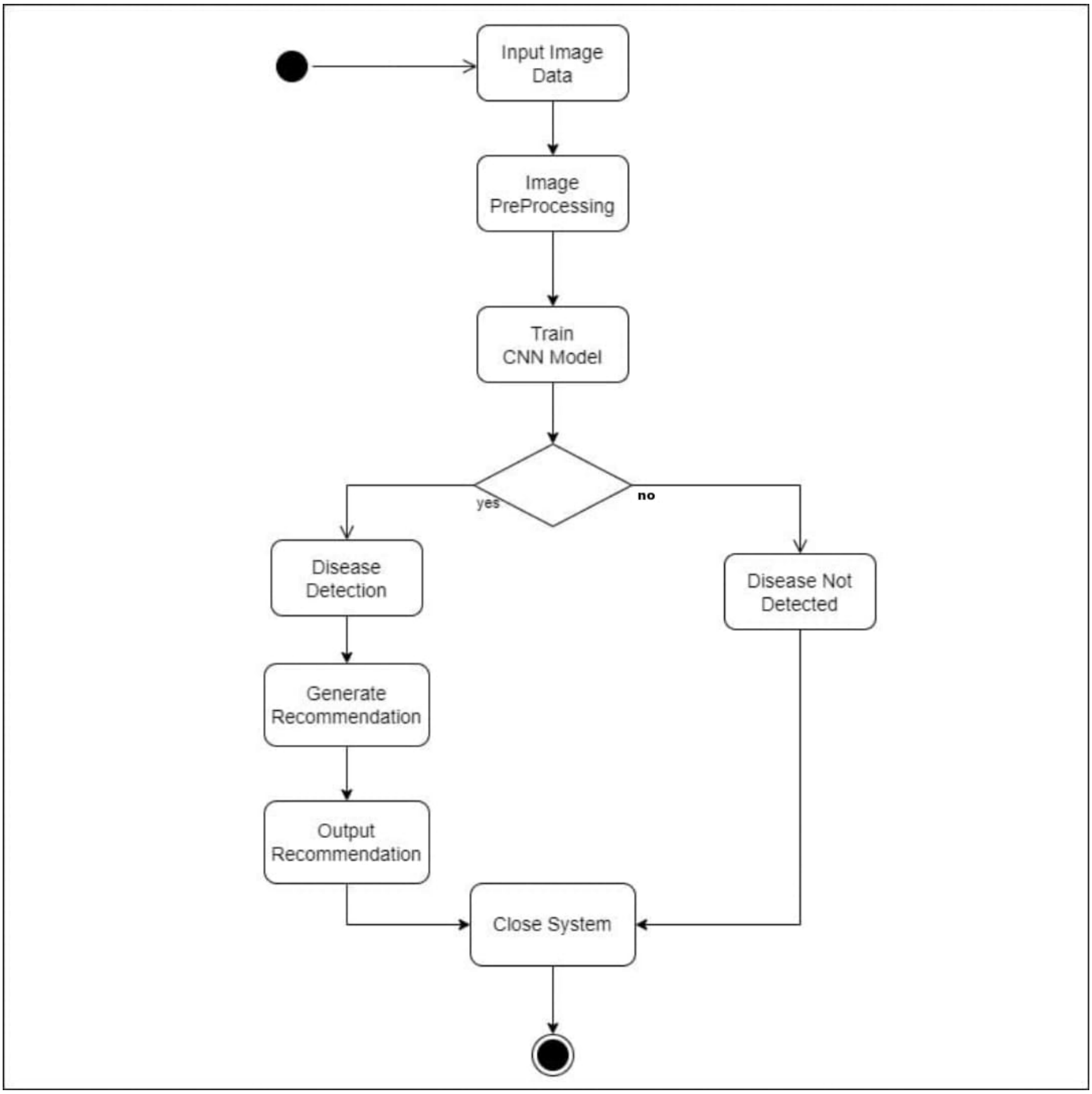


Figure 6.3

Input Image Data: The process begins with collecting images of diseased and healthy fruits.

Image Preprocessing: The collected image data is prepared for analysis. This step ensures that the images are in a suitable format for further processing.

Train CNN Model: A Convolutional Neural Network (CNN) model is created and trained using the preprocessed image data. The CNN architecture plays a crucial role in identifying patterns and features related to plant diseases.

Disease Detection: The trained CNN model analyzes input images and predicts whether the plant is healthy or diseased. If a disease is detected, the model proceeds to the next step.

Generate Recommendation: Based on the disease detection results, specific recommendations or actions are suggested. These could include treatment options, preventive measures, or further steps for managing the disease.

Output Recommendation: The system provides the recommendations as output, which can guide farmers or users in taking appropriate actions for crop health.

Close System: The process concludes, and the system awaits new input for disease prediction.

7. IMPLEMENTATION

7.1 Methodology

The proposed methodology consists of steps like pre-processing, augmentation, Model Deployment and testing etc.

Data Collection
A dataset of citrus fruit images was compiled, including both healthy and diseased samples. Images were sourced from agricultural databases and field surveys. Each image was labeled according to the type of disease or health status of the fruit.

Image Preprocessing
Image preprocessing involved several steps:

Resizing: Ensuring uniform image dimensions.
Normalization: Standardizing pixel values to improve model performance.
Data Augmentation: Techniques such as rotation, flipping, and scaling were applied to increase the diversity of the dataset.

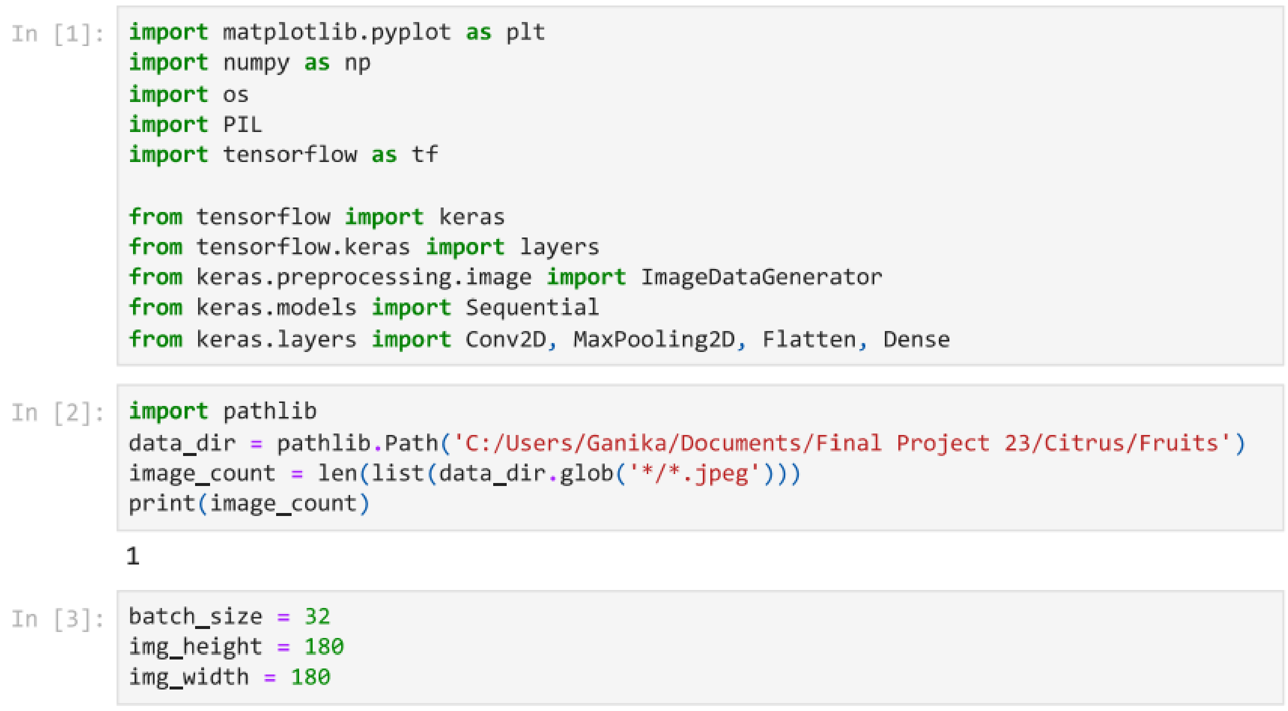
CNN Model Development
A Convolutional Neural Network (CNN) was designed and trained using TensorFlow. The architecture included multiple convolutional layers, pooling layers, and fully connected layers. The model was trained using the preprocessed dataset, with hyperparameters optimized for best performance.

Model Evaluation
The model's performance was evaluated using metrics such as accuracy, precision, recall, and F1-score. A validation set was used to assess the model's ability to generalize to new data.

User Interface Development
A web-based interface was developed to provide a user-friendly platform for disease detection. The frontend was built using HTML, CSS, and JavaScript, while the backend was implemented using Flask. Users can upload images of citrus fruits, and the system provides immediate disease detection results.

7.2 Screenshots

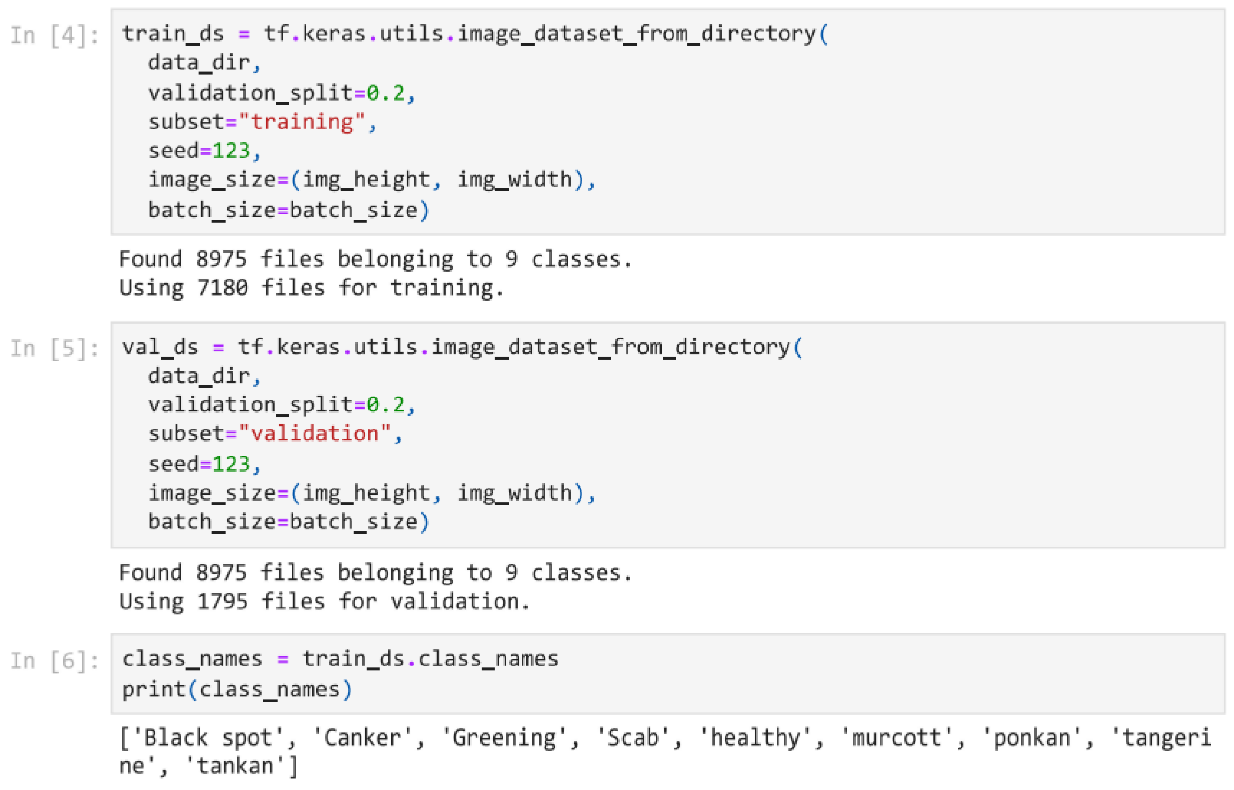
1. Import Dataset



```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
        import os
        import PIL
        import tensorflow as tf

        from tensorflow import keras
        from tensorflow.keras import layers
        from keras.preprocessing.image import ImageDataGenerator
        from keras.models import Sequential
        from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

In [2]: import pathlib
        data_dir = pathlib.Path('C:/Users/Ganika/Documents/Final Project 23/Citrus/Fruits')
        image_count = len(list(data_dir.glob('*/*.jpeg')))
        print(image_count)

        1

In [3]: batch_size = 32
        img_height = 180
        img_width = 180
```

Figure 7.1

2. Load Data using Keras Utility



```
In [4]: train_ds = tf.keras.utils.image_dataset_from_directory(
          data_dir,
          validation_split=0.2,
          subset="training",
          seed=123,
          image_size=(img_height, img_width),
          batch_size=batch_size)
```

```
Found 8975 files belonging to 9 classes.
Using 7180 files for training.
```

```
In [5]: val_ds = tf.keras.utils.image_dataset_from_directory(
          data_dir,
          validation_split=0.2,
          subset="validation",
          seed=123,
          image_size=(img_height, img_width),
          batch_size=batch_size)
```

```
Found 8975 files belonging to 9 classes.
Using 1795 files for validation.
```

```
In [6]: class_names = train_ds.class_names
        print(class_names)
```

```
['Black spot', 'Canker', 'Greening', 'Scab', 'healthy', 'murcott', 'ponkan', 'tangeri
ne', 'tankan']
```

Figure 7.2

3. Visualize the Data

```
In [7]: import matplotlib.pyplot as plt

        plt.figure(figsize=(10, 10))
        for images, labels in train_ds.take(1):
            for i in range(9):
                ax = plt.subplot(3, 3, i + 1)
                plt.imshow(images[i].numpy().astype("uint8"))
                plt.title(class_names[labels[i]])
                plt.axis("off")
```

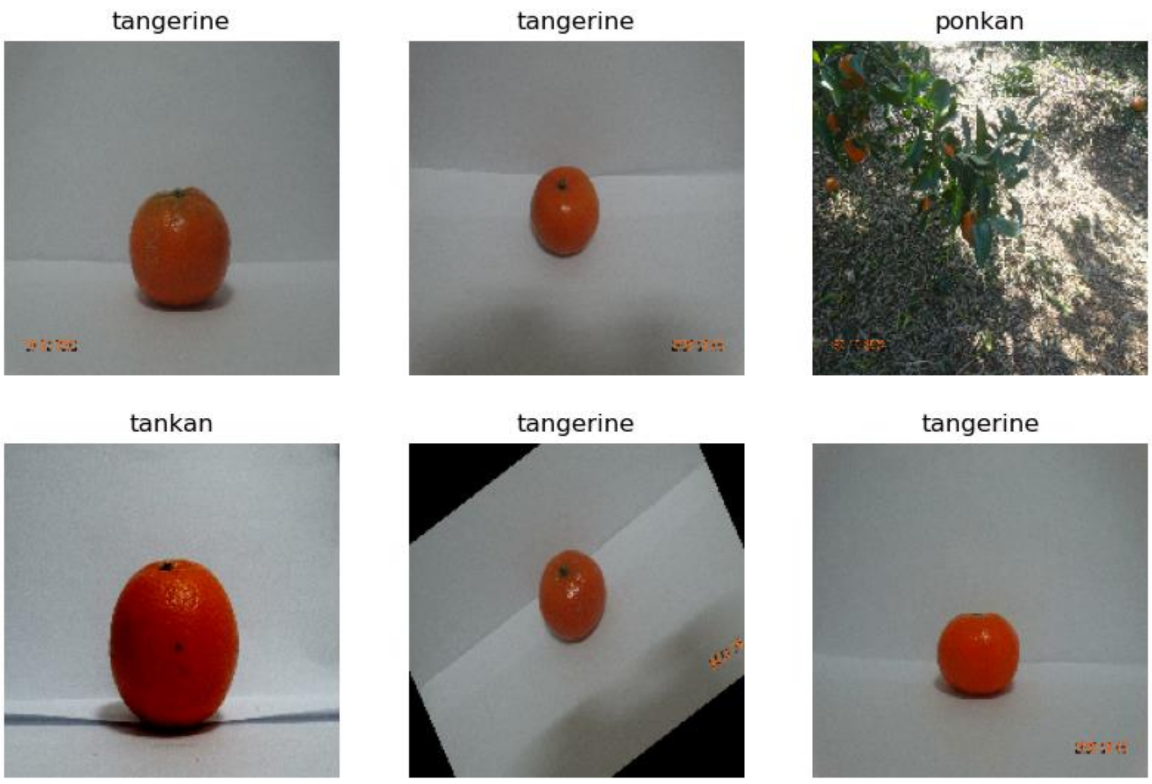


Figure 7.3

4. Configure and Standardize the Data

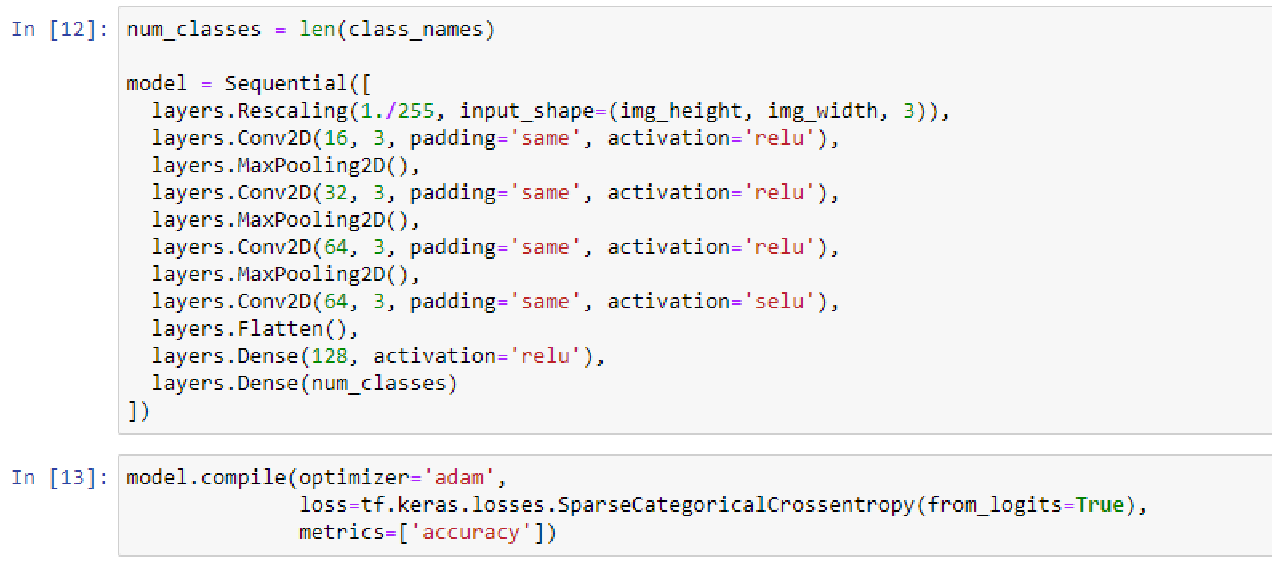
```
In [9]: AUTOTUNE = tf.data.AUTOTUNE

        train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
        val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

```
In [10]: normalization_layer = layers.Rescaling(1./255)
```

```
In [11]: normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
         image_batch, labels_batch = next(iter(normalized_ds))
         first_image = image_batch[0]
         # Notice the pixel values are now in `[0,1]`.
         print(np.min(first_image), np.max(first_image))

         0.0 1.0
```

Figure 7.4

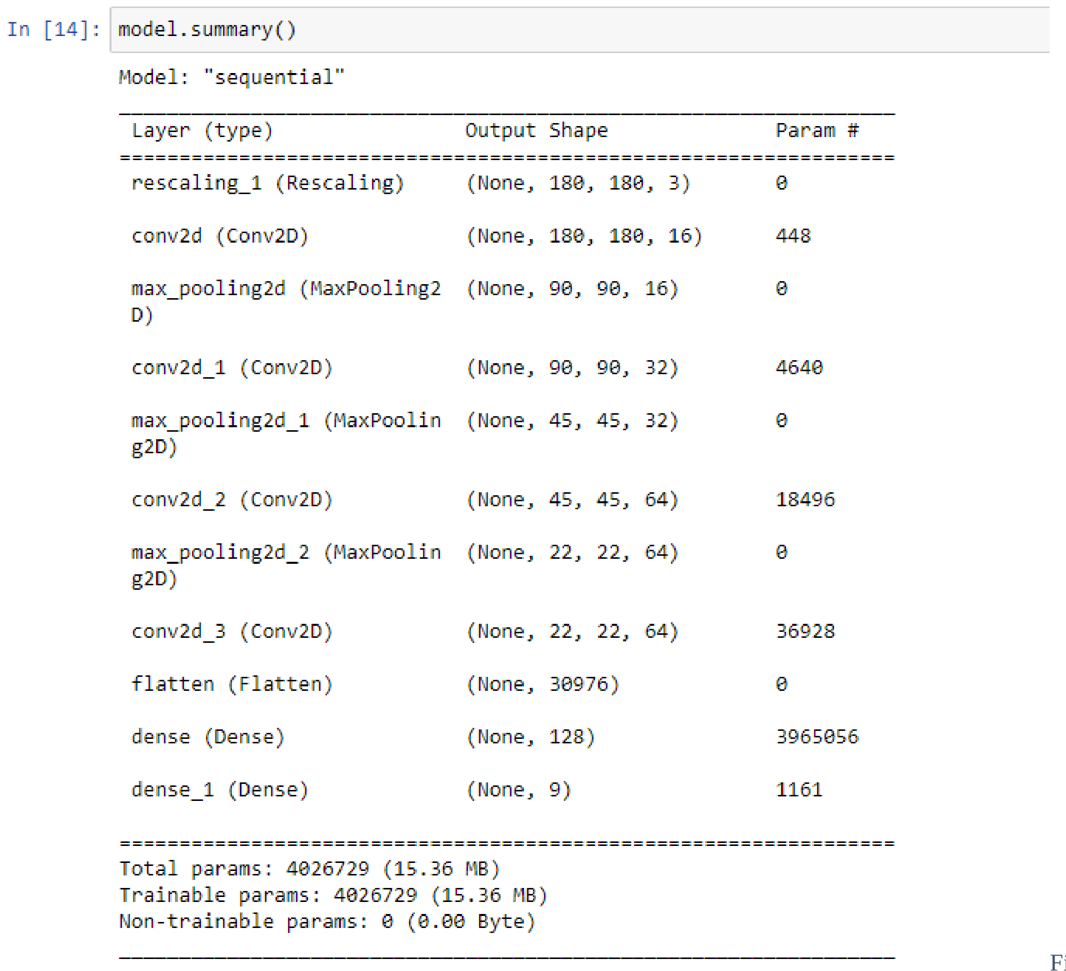
5. Model Creation



```
In [12]: num_classes = len(class_names)

         model = Sequential([
           layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
           layers.Conv2D(16, 3, padding='same', activation='relu'),
           layers.MaxPooling2D(),
           layers.Conv2D(32, 3, padding='same', activation='relu'),
           layers.MaxPooling2D(),
           layers.Conv2D(64, 3, padding='same', activation='relu'),
           layers.MaxPooling2D(),
           layers.Conv2D(64, 3, padding='same', activation='selu'),
           layers.Flatten(),
           layers.Dense(128, activation='relu'),
           layers.Dense(num_classes)
         ])

In [13]: model.compile(optimizer='adam',
                       loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                       metrics=['accuracy'])
```

Figure 7.5

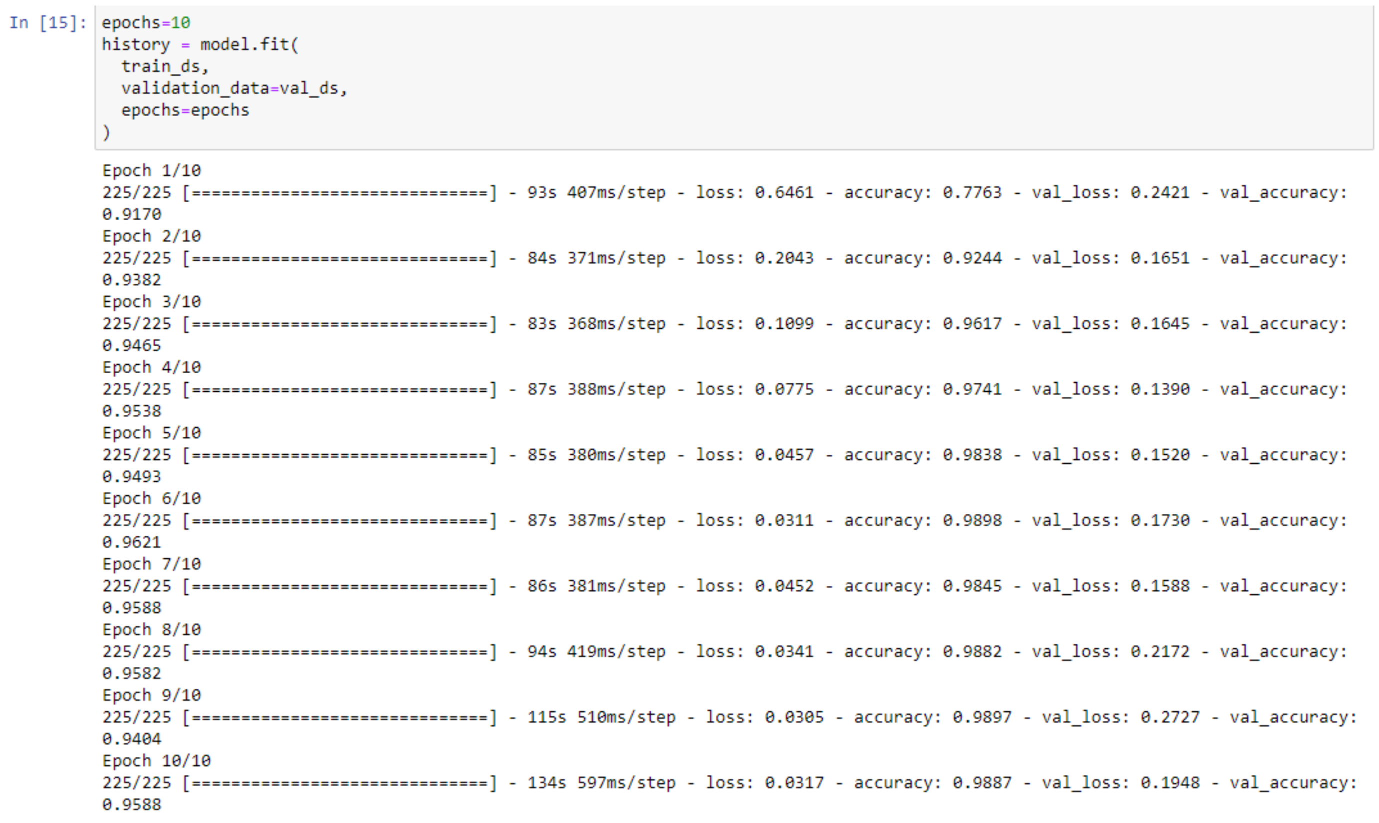
6. Model Summary



```
In [14]: model.summary()

         Model: "sequential"
         _________________________________________________________________
          Layer (type)                Output Shape              Param #
         =================================================================
          rescaling_1 (Rescaling)     (None, 180, 180, 3)       0

          conv2d (Conv2D)             (None, 180, 180, 16)      448

          max_pooling2d (MaxPooling2  (None, 90, 90, 16)        0
          D)

          conv2d_1 (Conv2D)           (None, 90, 90, 32)        4640

          max_pooling2d_1 (MaxPoolin  (None, 45, 45, 32)        0
          g2D)

          conv2d_2 (Conv2D)           (None, 45, 45, 64)        18496

          max_pooling2d_2 (MaxPoolin  (None, 22, 22, 64)        0
          g2D)

          conv2d_3 (Conv2D)           (None, 22, 22, 64)        36928

          flatten (Flatten)           (None, 30976)             0

          dense (Dense)               (None, 128)               3965056

          dense_1 (Dense)             (None, 9)                 1161

         =================================================================
         Total params: 4026729 (15.36 MB)
         Trainable params: 4026729 (15.36 MB)
         Non-trainable params: 0 (0.00 Byte)
         _________________________________________________________________
```

Figure 7.6

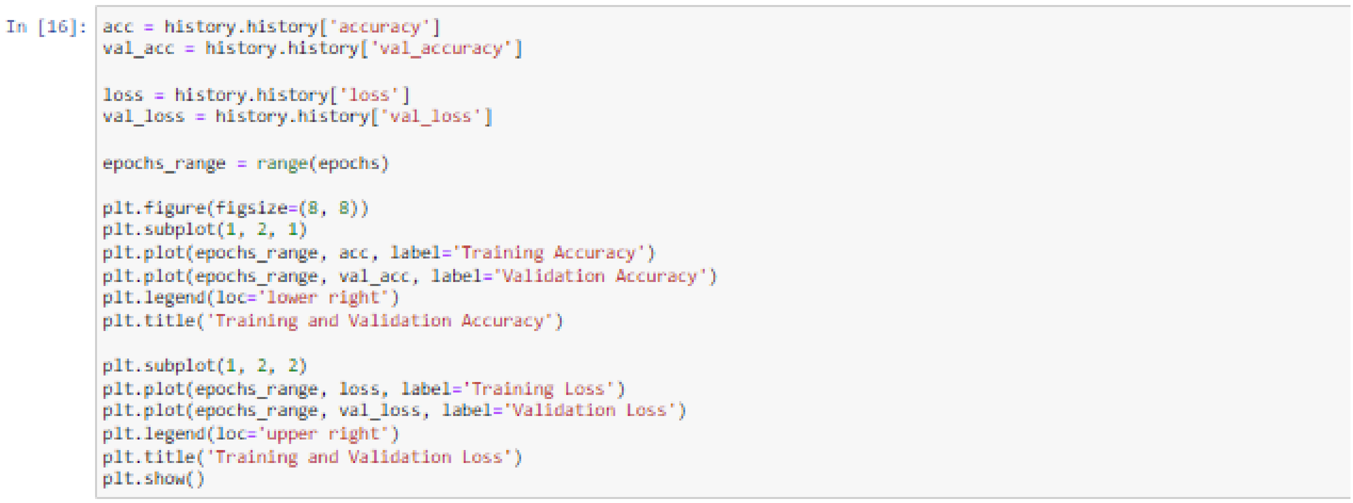
7. Train the Model



```
In [15]: epochs=10
         history = model.fit(
           train_ds,
           validation_data=val_ds,
           epochs=epochs
         )

Epoch 1/10
225/225 [==============================] - 93s 407ms/step - loss: 0.6461 - accuracy: 0.7763 - val_loss: 0.2421 - val_accuracy:
0.9170
Epoch 2/10
225/225 [==============================] - 84s 371ms/step - loss: 0.2043 - accuracy: 0.9244 - val_loss: 0.1651 - val_accuracy:
0.9382
Epoch 3/10
225/225 [==============================] - 83s 368ms/step - loss: 0.1099 - accuracy: 0.9617 - val_loss: 0.1645 - val_accuracy:
0.9465
Epoch 4/10
225/225 [==============================] - 87s 388ms/step - loss: 0.0775 - accuracy: 0.9741 - val_loss: 0.1390 - val_accuracy:
0.9538
Epoch 5/10
225/225 [==============================] - 85s 380ms/step - loss: 0.0457 - accuracy: 0.9838 - val_loss: 0.1520 - val_accuracy:
0.9493
Epoch 6/10
225/225 [==============================] - 87s 387ms/step - loss: 0.0311 - accuracy: 0.9898 - val_loss: 0.1730 - val_accuracy:
0.9621
Epoch 7/10
225/225 [==============================] - 86s 381ms/step - loss: 0.0452 - accuracy: 0.9845 - val_loss: 0.1588 - val_accuracy:
0.9588
Epoch 8/10
225/225 [==============================] - 94s 419ms/step - loss: 0.0341 - accuracy: 0.9882 - val_loss: 0.2172 - val_accuracy:
0.9582
Epoch 9/10
225/225 [==============================] - 115s 510ms/step - loss: 0.0305 - accuracy: 0.9897 - val_loss: 0.2727 - val_accuracy:
0.9404
Epoch 10/10
225/225 [==============================] - 134s 597ms/step - loss: 0.0317 - accuracy: 0.9887 - val_loss: 0.1948 - val_accuracy:
0.9588
```

Figure 7.7

8. Visualise Training Results



```
In [16]: acc = history.history['accuracy']
         val_acc = history.history['val_accuracy']

         loss = history.history['loss']
         val_loss = history.history['val_loss']

         epochs_range = range(epochs)

         plt.figure(figsize=(8, 8))
         plt.subplot(1, 2, 1)
         plt.plot(epochs_range, acc, label='Training Accuracy')
         plt.plot(epochs_range, val_acc, label='Validation Accuracy')
         plt.legend(loc='lower right')
         plt.title('Training and Validation Accuracy')

         plt.subplot(1, 2, 2)
         plt.plot(epochs_range, loss, label='Training Loss')
         plt.plot(epochs_range, val_loss, label='Validation Loss')
         plt.legend(loc='upper right')
         plt.title('Training and Validation Loss')
         plt.show()
```

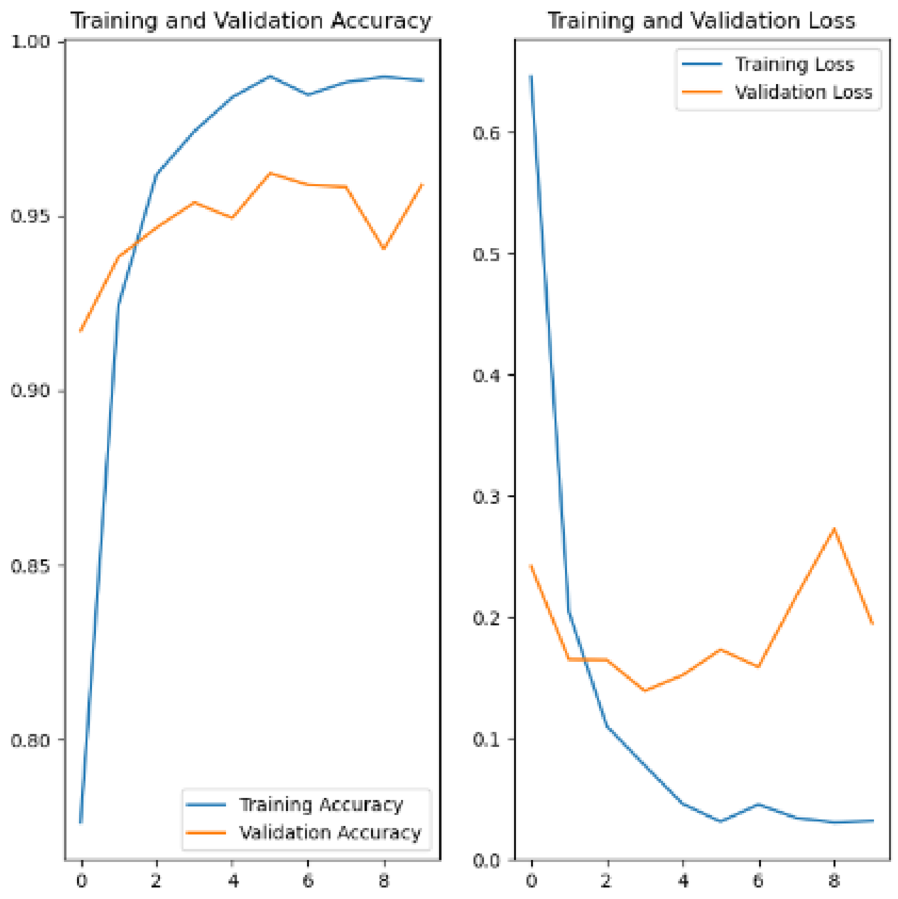
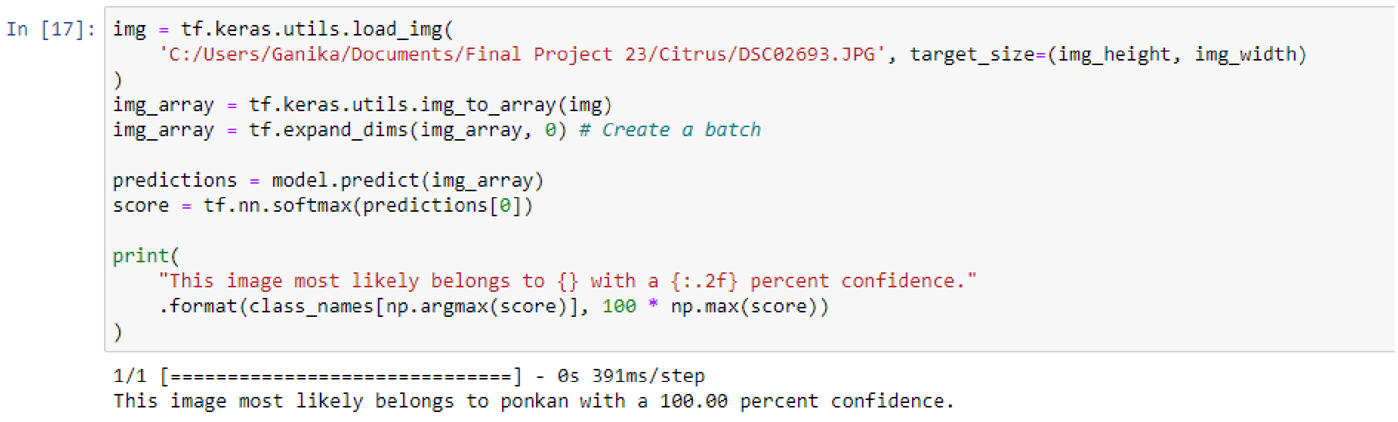


Figure 7.8

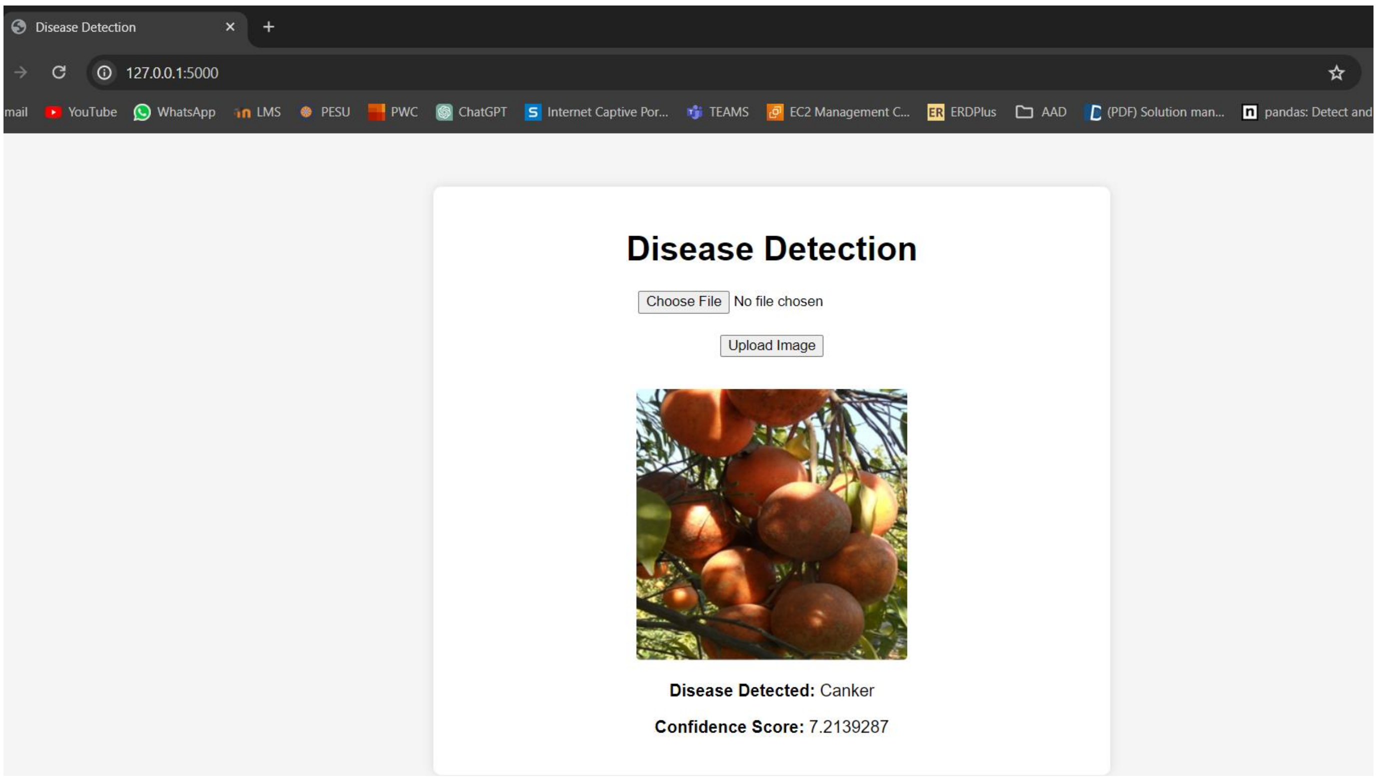
9. Predict the Output



```
In [17]: img = tf.keras.utils.load_img(
             'C:/Users/Ganika/Documents/Final Project 23/Citrus/DSC02693.JPG', target_size=(img_height, img_width)
         )
         img_array = tf.keras.utils.img_to_array(img)
         img_array = tf.expand_dims(img_array, 0) # Create a batch

         predictions = model.predict(img_array)
         score = tf.nn.softmax(predictions[0])

         print(
             "This image most likely belongs to {} with a {:.2f} percent confidence."
             .format(class_names[np.argmax(score)], 100 * np.max(score))
         )

         1/1 [==============================] - 0s 391ms/step
         This image most likely belongs to ponkan with a 100.00 percent confidence.
```

Figure 7.9

10. User Interface


Disease Detection
127.0.0.1:5000
mail YouTube WhatsApp LMS PESU PWC ChatGPT Internet Captive Por... TEAMS EC2 Management C... ERDPlus AAD (PDF) Solution man... pandas: Detect and
Choose File No file chosen
Upload Image

Disease Detection

Disease Detected: Canker

Confidence Score: 7.2139287

Figure 7.10

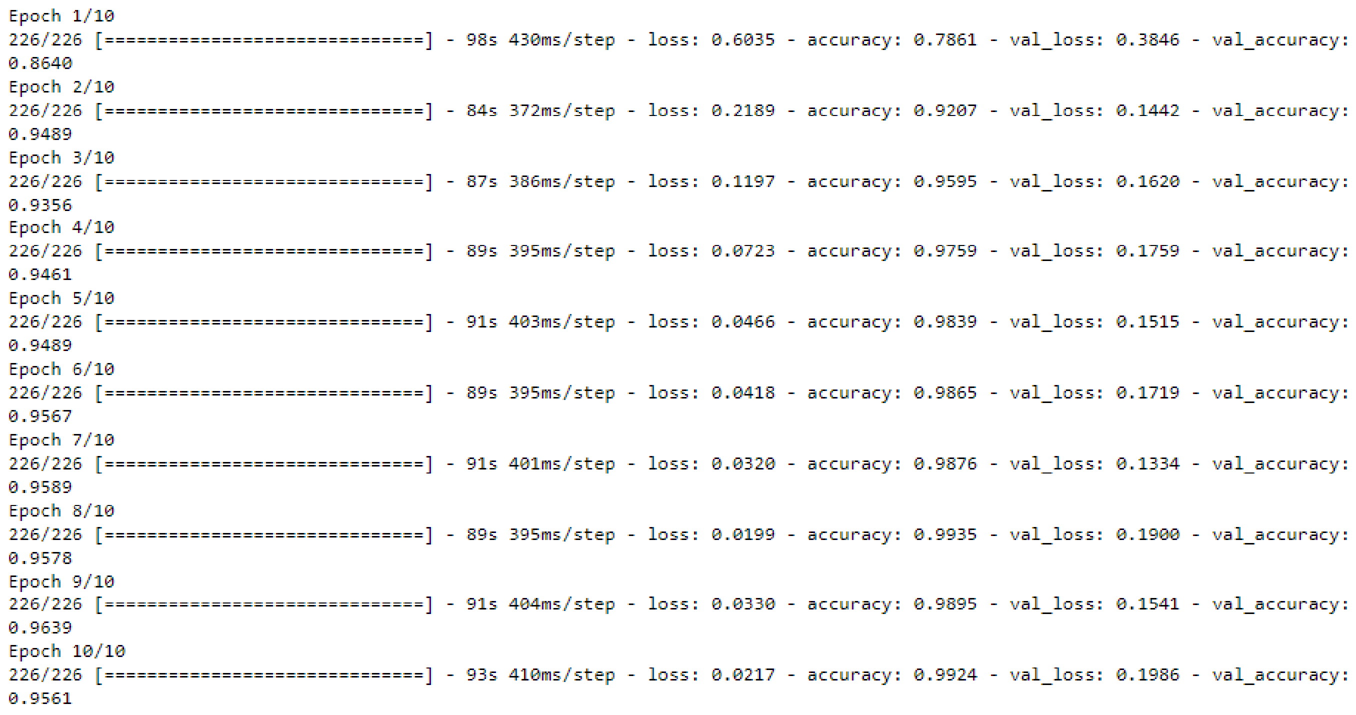
8. SOFTWARE TESTING

8.1 Manual Testing

Test Scenario	Step Detail	Expected Result	Actual Result	Pass/Fail/Non-executed/suspended
TC01	Navigate to http://127.0.0.1:5000/	Site Should Open	Site Opened as expected	Pass
TCO2	User uploads an image	User uploaded the image	Image uploaded successfully	Pass
TC03	User sees the result	Result is successfully shown	Output generated successfully	Pass
TC04	User upload normal image	Misleading result	Output generated	Failed

TC04	Load Pre-trained model.	Model loads successfully without any errors.	Model loaded successfully without any error.	Pass
TC05	Verify the existence of directories for each category	Directories for each category exist or created successfully.	Directories for each category exist or were created successfully.	Pass
TC06	Predict the category for images	Images are correctly classified.	Images were correctly classified.	Pass
TC07	Calculate the F1-score for model prediction and determine accuracy.	The model returns a f1-score with the highest probability and accuracy is correctly computed.	The model returned a f1-score score with the highest probability, and the accuracy was correctly computed.	Pass
TC08	Load and preprocess images to be classified.	Images are loaded and resized to (250,250) successfully without error.	Images were loaded and resized to (250,250) successfully without error.	Pass
TC09	Attempt to load and classify invalid or corrupted images.	The model skips or logs errors for invalid images without crashing.	The model skipped or logged errors for invalid images without crashing.	Pass

9. RESULT AND CONCLUSION



```
Epoch 1/10
226/226 [==============================] - 98s 430ms/step - loss: 0.6035 - accuracy: 0.7861 - val_loss: 0.3846 - val_accuracy:
0.8640
Epoch 2/10
226/226 [==============================] - 84s 372ms/step - loss: 0.2189 - accuracy: 0.9207 - val_loss: 0.1442 - val_accuracy:
0.9489
Epoch 3/10
226/226 [==============================] - 87s 386ms/step - loss: 0.1197 - accuracy: 0.9595 - val_loss: 0.1620 - val_accuracy:
0.9356
Epoch 4/10
226/226 [==============================] - 89s 395ms/step - loss: 0.0723 - accuracy: 0.9759 - val_loss: 0.1759 - val_accuracy:
0.9461
Epoch 5/10
226/226 [==============================] - 91s 403ms/step - loss: 0.0466 - accuracy: 0.9839 - val_loss: 0.1515 - val_accuracy:
0.9489
Epoch 6/10
226/226 [==============================] - 89s 395ms/step - loss: 0.0418 - accuracy: 0.9865 - val_loss: 0.1719 - val_accuracy:
0.9567
Epoch 7/10
226/226 [==============================] - 91s 401ms/step - loss: 0.0320 - accuracy: 0.9876 - val_loss: 0.1334 - val_accuracy:
0.9589
Epoch 8/10
226/226 [==============================] - 89s 395ms/step - loss: 0.0199 - accuracy: 0.9935 - val_loss: 0.1900 - val_accuracy:
0.9578
Epoch 9/10
226/226 [==============================] - 91s 404ms/step - loss: 0.0330 - accuracy: 0.9895 - val_loss: 0.1541 - val_accuracy:
0.9639
Epoch 10/10
226/226 [==============================] - 93s 410ms/step - loss: 0.0217 - accuracy: 0.9924 - val_loss: 0.1986 - val_accuracy:
0.9561
```

Figure 9.1

The training progress of a deep learning model across ten epochs is captured in the training log that is delivered. The main elements and their importance are broken down as follows:

Epochs and Steps:

- **Epochs:** Each epoch denotes a full pass through the entire training dataset. The model was trained across 10 epochs
- **Steps:** The notation 226/226 signifies that the dataset was split into 226 batches, with the model being updated 226 times per epoch.

Training Time:

- The duration for each epoch is specified. As an illustration, the first epoch took 98 seconds, while the next epochs took 84–93 seconds approximately.

Metrics:

- **Loss:** This indicates how closely the model's predictions match the real labels. It is desirable to have a lower loss value. As you can see, the model is getting better as the training loss drops from 0.6035 in the first epoch to 0.0217 in the tenth.

- **Accuracy:** This is the percentage of correct predictions. The training accuracy increases, as seen by the fact that it goes from 78.61% (0.7861) in the first epoch to 99.24% (0.9924) in the tenth.

- **Validation Loss (Val_loss):** This measures the loss on the validation set, used to evaluate the model's performance on unseen data. Ideally, it should also decrease along with the training loss. However, in this case, the validation loss fluctuates, indicating variability in performance.

- **Validation Accuracy (Val_accuracy):** This represents the accuracy on the validation set. It rises from 86.40% (0.8640) in the first epoch to 96.39% (0.9639) in the ninth epoch, suggesting good generalization to new data.

Epoch-wise Analysis:

- **Epoch 1:** The model begins with a training accuracy of 78.61% and a validation accuracy of 86.40%. The initial high validation accuracy suggests that the model quickly grasps some useful features.
- **Epochs 2-4:** There is a significant improvement in training accuracy, with a concurrent increase in validation accuracy. However, the validation loss during epochs 3 and 4 (0.1620 and 0.1759) indicates some overfitting, as the validation loss doesn't consistently decrease.
- **Epochs 5-7:** Training accuracy continues to rise, surpassing 98%. Validation accuracy also improves, peaking at 95.89% in epoch 7. The validation loss decreases in epoch 7 (0.1334), indicating improved generalization.
- **Epochs 8-10:** Training accuracy exceeds 99%. Although validation accuracy remains high, there are fluctuations in validation loss, suggesting slight overfitting, particularly with an increased validation loss in epoch 10 (0.1986).

The model shows substantial improvement in both training and validation accuracy, reflecting effective learning. Consistently high validation accuracy indicates good generalization to unseen data. Fluctuations in validation loss suggest variability in model performance, potentially due to factors like dataset size, complexity, and noise. Overall, the training log indicates a successful training process with high accuracy and reasonable validation performance. Further tuning and evaluation may be needed to reduce validation loss fluctuations and ensure consistent generalization.

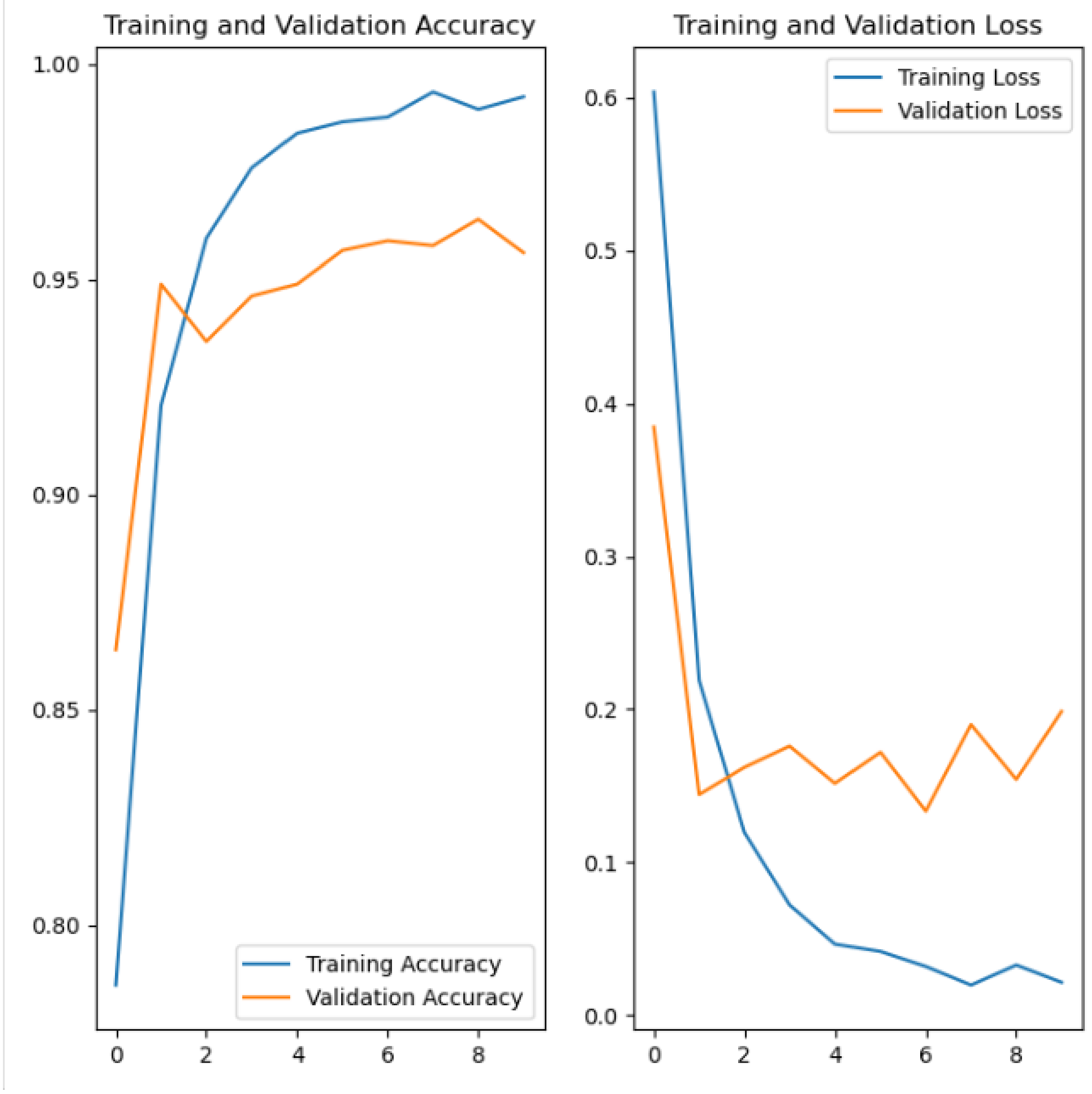


Figure 9.2

The provided graph illustrate the training and validation accuracy, as well as the training and validation loss, of a deep learning model over 10 epochs.

Training and Validation Accuracy

- **X-Axis:** Epochs (from 0 to 9)
- **Y-Axis:** Accuracy (from 0.80 to 1.00)

Blue Line (Training Accuracy):

- The training accuracy starts at approximately 0.78 (78%) in the first epoch.
- There is a steady increase in accuracy with each epoch, reaching about 0.99 (99%) by the ninth epoch.
- This consistent upward trend indicates that the model is effectively learning from the training data.

Orange Line (Validation Accuracy):

- The validation accuracy starts at around 0.86 (86%) in the first epoch.
- It increases along with the training accuracy, peaking at around 0.96 (96%) by the seventh epoch.
- After the seventh epoch, the validation accuracy slightly fluctuates but remains relatively high, indicating good generalization to unseen data.

Training and Validation Loss

- **X-Axis:** Epochs (from 0 to 9)
- **Y-Axis:** Loss (from 0.0 to 0.6)

Blue Line (Training Loss):

- The training loss starts at about 0.60 in the first epoch.
- There is a rapid decrease in loss, which continues to decline steadily, reaching close to 0.02 by the ninth epoch.
- This decrease shows that the model is minimizing errors in its predictions on the training data.

Orange Line (Validation Loss):

- The validation loss starts at around 0.40 in the first epoch.
- Initially, it drops significantly, but then it begins to fluctuate, indicating variability in the model's performance on the validation set.
- Although the validation loss decreases in some epochs, it exhibits ups and downs, suggesting that the model might be overfitting slightly towards the later epochs.

Training Accuracy: Shows a consistent improvement, suggesting effective learning.

Validation Accuracy: Remains high, indicating good generalization, although slight fluctuations suggest some variability.

Training Loss: Decreases steadily, indicating that the model is becoming better at minimizing errors.

Validation Loss: Fluctuates, which might suggest overfitting or variability in performance on unseen data.

10. FUTURE ENHANCEMENTS

This research demonstrates the transformative potential of deep learning in revolutionizing disease detection in citrus fruits. By developing an automated, efficient, and accurate system, the project paves the way for improved agricultural practices and crop management. The findings highlight the significant benefits of using CNNs for disease detection, offering a scalable and reliable solution for the citrus industry.

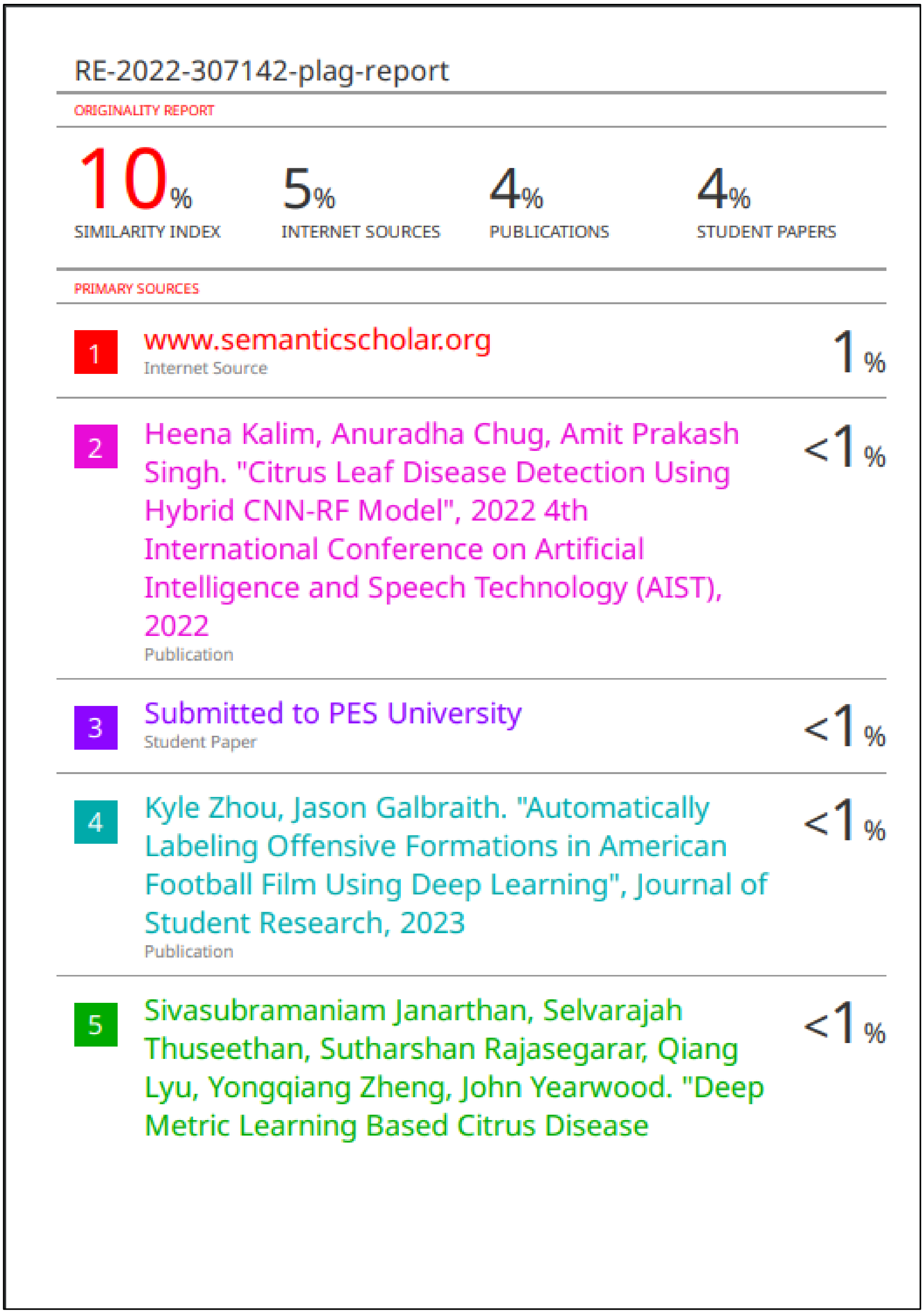
Future Enhancements include:

- Testing in low-end devices
- Deploy our model in low-end devices like embedded systems and mobile phones.
- It may include expanding the dataset, incorporating multi-modal data
- Integration of additional machine learning techniques to improve detection accuracy

BIBLIOGRAPHY

1. https://data.mendeley.com/datasets/3f83gxmv57/1
2. https://data.mendeley.com/datasets/bm4jmg53kw/1
3. https://data.mendeley.com/datasets/wchp3bryrm/2
4. https://www.kaggle.com/datasets/myprojectdictionary/citrus-leaf-disease-image
5. https://data.mendeley.com/datasets/5tz79rtd39/1
6. M. A. Matboli and A. Atia, "Fruit Disease's Identification and Classification Using Deep Learning Model," 2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), Cairo, Egypt, 2022, pp. 432-437, doi: 10.1109/MIUCC55081.2022.9781688.
7. G. Shireesha and B. E. Reddy, "Citrus Fruit and Leaf Disease Detection Using DenseNet," 2022 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON), Bangalore, India, 2022, pp. 1-5, doi: 10.1109/SMARTGENCON56628.2022.10083852.
8. H. Kalim, A. Chug and A. P. Singh, "Citrus Leaf Disease Detection Using Hybrid CNN-RF Model," 2022 4th International Conference on Artificial Intelligence and Speech Technology (AIST), Delhi, India, 2022, pp. 1-4, doi: 10.1109/AIST55798.2022.10065093.
9. E. Khan, M. Z. U. Rehman, F. Ahmed and M. A. Khan, "Classification of Diseases in Citrus Fruits using SqueezeNet," 2021 International Conference on Applied and Engineering Mathematics (ICAEM), Taxila, Pakistan, 2021, pp. 67-72, doi: 10.1109/ICAEM53552.2021.9547133.
10. G. Vaidya, N. Chaudhary, S. Nasare, S. Warutkar, S. Kawathekar and S. Dhengre, "Disease Detection of Citrus Plants Using Image Processing Techniques," 2023 11th International Conference on Emerging Trends in Engineering & Technology - Signal and Information Processing (ICETET - SIP), Nagpur, India, 2023, pp. 1-6, doi: 10.1109/ICETET-SIP58143.2023.10151494.

PLAGIARISM REPORT



RE-2022-307142-plag-report

ORIGINALITY REPORT

10%	5%	4%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.semanticscholar.org Internet Source	1%
2	Heena Kalim, Anuradha Chug, Amit Prakash Singh. "Citrus Leaf Disease Detection Using Hybrid CNN-RF Model", 2022 4th International Conference on Artificial Intelligence and Speech Technology (AIST), 2022 Publication	<1%
3	Submitted to PES University Student Paper	<1%
4	Kyle Zhou, Jason Galbraith. "Automatically Labeling Offensive Formations in American Football Film Using Deep Learning", Journal of Student Research, 2023 Publication	<1%
5	Sivasubramaniam Janarthan, Selvarajah Thuseethan, Sutharshan Rajasegarar, Qiang Lyu, Yongqiang Zheng, John Yearwood. "Deep Metric Learning Based Citrus Disease	<1%

POSTER

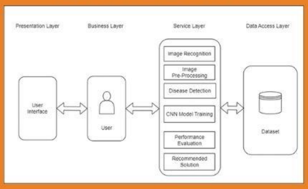
Disease Prediction in Citrus Fruits

By: Ganika Kodnani
Guide: Ms. Deepika D (Assistant Professor, Department of Computer Application

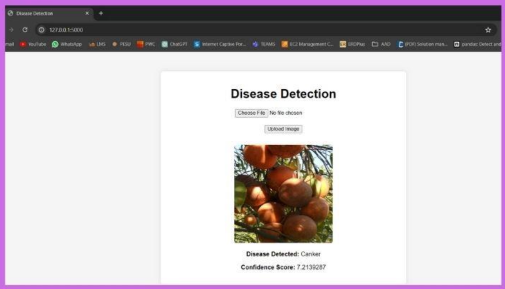
ABSTRACT:

This project uses deep learning to automatically detect diseases in citrus fruits from images. It's faster and more accurate than traditional methods, allowing for early intervention and improved crop management. The system has the potential to be even more powerful with a larger dataset and wider applications.

ARCHITECTURAL DIAGRAM:



RESULT:



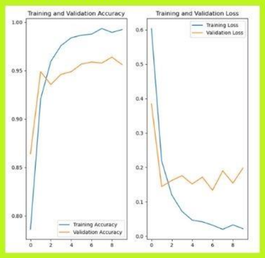
Disease Detection

Disease Detected: Canker

Confidence Score: 7.2139287

TOOLS:

PERFORMANCE:



CONCLUSION:

This project offers an automated disease detection system for citrus fruits using deep learning. By leveraging a user-friendly interface, it empowers to identify diseases early, leading to improved crop health and reduced losses. Future advancements could include real-time detection and even expand to predict disease outbreaks based on environmental factors.