ENHANCED PLANT SEEDLING IDENTIFICATION AND DISEASE DETECTION THROUGH END-TO-END CONVOLUTIONAL NEURAL NETWORK

A PROJECT REPORT

Submitted by

RISHI R [Reg No:RA2011003010481]

MOHAMMED IRFAN M A [Reg No: RA2011003010488]

Under the Guidance of

Dr. G.BALAMURUGAN

Assistant Professor, Department of Computing Technologies

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR– 603 203

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR–603 203

BONAFIDE CERTIFICATE

Certified that 18CSP109L / I8CSP111L project report titled "ENHANCED PLANT SEEDLING IDENTIFICATION AND DISEASE DETECTION THROUGH END-TO-END CONVOLUTIONAL NEURAL NETWORKS" is the bonafide work of **RISHI R [RegNo:RA2011003010481]** and **MOHAMMED IRFAN M A [RegNo:RA2011003010488]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr.G.BALAMURUGAN
SUPERVISOR
Assistant Professor
Department of Computing Technologies

Dr.B.KANISHA
PANEL HEAD
Associate Professor
Department of Computing Technologies

Dr.M.PUSHPALATHA
HEAD OF THE DEPARTMENT
Department of Computing Technologies

Department of Computing Technologies

SRM Institute of Science and Technology

Own Work Declaration Form

Degree/Course: B.Tech in Computer Science and Engineering

Student Name: RISHI R, MOHAMMED IRFAN M A

Registration Number: RA2011003010481, RA2011003010488

Title of Work:

I/We here by certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web,etc.)
- Given the sources of all pictures, data etc that are not my own.
- Not made any use of the report(s) or essay(s) of any other student(s)either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course hand book / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above. **Student 1 Signature:** **Student 2 Signature:** **Date:**

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.
We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr. T. V. Gopal**, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr. M. Pushpalatha,** Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinators, **Dr. S. Godfrey Winster**, **Associate Professor, Dr. M. Basker, Dr. P. Murali, Dr. J. Selvin Paul Peter, Dr. C. Pretty Diana Cyril and Dr. G. Padmapriya** Panel Head, **Dr. B. Kanisha**, Associate Professor and Panel Members, **Dr. Sworna Kokila, Dr. Kanmani** Assistant Professor Department of Computing Technologies, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. G. BALAMURUGAN**, Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr.G.BALAMURUGAN**, Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of Computing Technologies Department, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement.

RISHI R [Reg. No: RA2011003010481]

MOHAMMED IRFAN M A [Reg. No: RA2011003010488]

ABSTRACT

The cultivation of healthy crops and the early detection of diseases are pivotal factors in ensuring global food security and sustainable agriculture. In this context, the research presented in this paper focuses on the development of an end-to-end system for plant seedling identification and disease detection, leveraging the power of Convolutional Neural Networks (CNN) implemented with the Keras and TensorFlow frameworks. The main objectives of this study are to create an integrated solution that not only identifies plant seedlings but also detects diseases in a seamless manner. To achieve these goals, a multi-faceted research approach is undertaken, encompassing various machine learning methodologies. The research begins with a comprehensive review of the existing literature, exploring prior work related to plant seedling identification and disease detection. This background study provides insights into the state of the art and serves as a basis for identifying the gaps and challenges that the proposed system addresses. Data collection and preprocessing play a crucial role in the research methodology. The selection of a high-quality and diverse dataset is paramount, enabling the training and evaluation of the CNN model. Careful data preprocessing techniques, including cleaning, normalization, and augmentation, are employed to ensure the dataset's suitability for training. The heart of the research lies in the design and architecture of the Convolutional Neural Network. The model is meticulously crafted, with attention to detail on layer configuration, activation functions, optimizers, and other essential parameters. This CNN architecture is optimized for the task of plant seedling identification and disease detection, making it a powerful tool in the research. The training process is thoroughly documented, highlighting the division of data into training, validation, and testing sets. Training parameters, such as batch size, learning rate, and the number of epochs, are chosen carefully to ensure optimal model performance. Performance evaluation is a key aspect, and metrics like accuracy, precision, recall, and F1 score are used to assess the model's capabilities. The results of experiments are presented and analyzed, showcasing the system's ability to meet the research objectives. By the end of this study, a comprehensive and robust system for end-to-end plant seedling identification and disease detection using CNNs with Keras and TensorFlow is developed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ReLU	Rectified Linear Unit
CNN	Convolutional Neural Network
AI	Artificial Intelligence
SVM	Support Vector Machine
ML	Machine Learning

CHAPTER 1
INTRODUCTION

Agriculture stands as one of the fundamental pillars of human civilization, providing sustenance and economic stability for societies across the globe. However, in an era where the world's population continues to burgeon, the demand for sustainable and efficient agricultural practices has become more pressing than ever. A critical component of this effort lies in the early identification of plant diseases and the precise recognition of plant species. The integration of machine learning techniques has emerged as a transformative approach to address these challenges, opening new horizons for improved crop management and global food security.

This research embarks on a journey to develop a cutting-edge system for plant seedling identification and disease detection that seamlessly integrates the capabilities of Convolutional Neural Networks (CNNs) using the Keras and TensorFlow frameworks. The overarching aim is to create an "end-to-end" solution that can not only recognize various plant species but also pinpoint the onset of diseases, thereby empowering farmers and agricultural stakeholders with tools to make informed decisions. The motivation for this research is driven by the recognition that timely identification of plant diseases is imperative for mitigating crop losses, reducing the need for chemical treatments, and ensuring the sustainability of agricultural practices.

Additionally, accurate plant species recognition is essential for precision agriculture, enabling tailored care for different crops. In pursuit of these goals, this study reviews the existing literature in the domain of plant seedling identification and disease detection, identifying gaps and opportunities for novel contributions. By building on this foundational knowledge, the research leverages diverse datasets and employs meticulous data preprocessing techniques to prepare the data for training the CNN model.

The core of the research lies in the design and architecture of the CNN, which is meticulously tailored to the specific needs of plant seedling identification and disease detection. This entails careful configuration of layers, selection of activation functions, and optimization of hyperparameters to maximize the model's efficacy. The training process is documented in detail, including the division of the dataset into training, validation, and testing subsets, and the selection of optimal training parameters. Performance evaluation metrics are defined, and the results of experiments are analyzed to assess the system's capabilities.The global agricultural landscape is at a pivotal juncture, with the ever-increasing demand for food production and the persistent challenge of crop diseases threatening crop yields and food security. Addressing this critical juncture requires innovative solutions that merge cutting-edge technologies with agriculture. In this pursuit, the present research embarks on a journey to develop an end-to-end system for plant seedling identification and disease detection, employing Convolutional Neural Networks (CNNs) implemented with the Keras and TensorFlow frameworks.The importance of efficient plant seedling identification and disease detection cannot be overstated.

Accurate and timely identification of plant species is essential in areas such as biodiversity preservation and precision agriculture. On the other hand, the rapid identification of diseases in crops is paramount for mitigating the impact of infections and ensuring high agricultural productivity.

This research not only recognizes the significance of these challenges but also aligns with the broader movement towards precision agriculture, which harnesses data and technology to optimize farming practices. An end-to-end system that can accurately identify plant seedlings and promptly detect diseases has the potential to revolutionize crop management, reducing the need for manual labor-intensive processes and promoting sustainable farming practices.The foundation of this research is built upon a thorough literature review that examines existing methods and technologies related to plant seedling identification and disease detection. By drawing from the knowledge of previous works, the research identifies opportunities and gaps that the proposed system seeks to address.

In conclusion, this research endeavors to contribute to the advancement of agricultural technology, holding the potential to revolutionize crop management, enhance global food security, and contribute to sustainable agriculture practices through a comprehensive end-to-end system for plant seedling identification and disease detection.

CHAPTER 2
PROBLEM STATEMENT

The agricultural sector is a critical component of global food production and plays a pivotal role in ensuring food security for the growing world population. However, this sector faces significant challenges, including the identification of plant seedlings and the timely detection of diseases that can severely impact crop yields and quality. Traditional methods of plant seedling identification and disease detection are often labor-intensive, time-consuming, and error-prone. To address these challenges, there is an urgent need for an advanced and automated system that can enhance plant seedling identification and disease detection, providing farmers and agricultural experts with the tools to make informed decisions and optimize their farming practices. This research project aims to develop and implement an "Enhanced Plant Seedling Identification and Disease Detection System" based on Convolutional Neural Networks (CNNs), which are a class of deep learning algorithms highly effective in image recognition tasks.

CHAPTER 3
OBJECTIVES

The overarching objective of this research project, "Enhanced Plant Seedling Identification and Disease Detection using Convolutional Neural Networks," is to advance the field of precision agriculture by developing and implementing a robust and highly accurate automated system for identifying plant seedlings and detecting diseases in crops. This multifaceted research endeavor aims to address several key goals and objectives:

Development of Advanced CNN Models: Create state-of-the-art Convolutional Neural Networks (CNN) models tailored to plant seedling identification and disease detection. These models should leverage cutting-edge deep learning techniques and architectures to maximize accuracy and efficiency.

Large-Scale Dataset Curation: Assemble a comprehensive dataset containing a diverse range of plant seedling images, including various species and growth stages. This dataset should also include annotated images showcasing a wide spectrum of plant diseases.

Data Preprocessing and Augmentation: Implement data preprocessing techniques to enhance the quality of the dataset, ensuring uniformity and removing noise. Apply data augmentation strategies to increase the model's robustness and generalization capabilities.

Disease Detection Accuracy: Achieve high accuracy in disease detection, with the ability to identify diseases at early stages, thereby facilitating proactive disease management and minimizing crop losses.

Field Testing and Validation: Conduct extensive field testing to validate the system's performance under real-world conditions. Gather feedback from end-users to refine the system and make necessary improvements. **Scalability and Adaptability:** Design the system to be scalable, allowing for expansion to cover larger agricultural areas, and adaptable to different geographic regions and farming practices.

Education and Outreach: Develop educational materials and programs to facilitate the adoption of this technology among farmers and stakeholders. Offer training and support for users to maximize its potential.

CHAPTER 4
MOTIVATION

Agriculture has been the backbone of human civilization for millennia, providing sustenance and livelihoods to billions of people around the world. The global population is steadily increasing, and with it, the demand for food is rising exponentially. In the face of these challenges, the agricultural sector is confronted with the daunting task of producing more food, with limited resources and under increasingly unpredictable environmental conditions. However, it is not just the demand for more food that concerns us, but also the quality and sustainability of agricultural practices. This confluence of factors has spurred the need for innovative and efficient solutions that can not only enhance crop productivity but also ensure the long-term health of agricultural ecosystems.

Plant seedling identification and disease detection play a pivotal role in achieving these objectives. The accurate and rapid identification of plant species is fundamental to various agricultural and ecological applications, including crop management, biodiversity preservation, and habitat conservation. On the other hand, the early detection of diseases in crops is critical for averting devastating crop losses and the subsequent economic hardships faced by farmers.

Traditional methods of plant species identification and disease detection often rely on manual labor and are time-consuming, error-prone, and sometimes impractical for large-scale agricultural operations. These methods are limited in their ability to cope with the complexity and variability of plant species and diseases. This is where technology, particularly artificial intelligence and machine learning, can revolutionize agriculture.

The utilization of Convolutional Neural Networks (CNNs) in combination with Keras and TensorFlow has proven to be a game-changer in computer vision applications. CNNs have demonstrated exceptional capabilities in image recognition and classification, making them an ideal candidate for automating plant seedling identification and disease detection.

Motivated by the imperative to enhance global food security, optimize resource utilization, and promote sustainable agriculture, this research embarks on a journey to develop an end-to-end system that seamlessly integrates plant seedling identification and disease detection. By harnessing the power of AI and CNNs, this system aims to revolutionize agricultural practices, reduce manual labor, and enable early and accurate identification of plant species and diseases.

The research is driven by the conviction that such a system can significantly contribute to the advancement of precision agriculture, thereby increasing crop yields, reducing resource wastage, and mitigating the environmental impact of agriculture. The potential benefits extend not only to farmers but also to ecological researchers, conservators, and policymakers working to preserve biodiversity and habitat integrity.

In summary, the motivation for this research is rooted in the urgent need to modernize and optimize agricultural practices in the face of growing global challenges.

The convergence of technology and agriculture offers a promising pathway toward achieving these goals, and this research seeks to harness that potential to address the critical issues of plant seedling identification and disease detection.

CHAPTER 5
COMPARISION OF EXISTING METHODS

5.1 Traditional Image Processing Techniques:

Advantages: These methods have been used for plant seedling identification and disease detection for some time and are well-established. They are computationally efficient and do not require deep learning expertise.

Limitations: Traditional techniques often lack the accuracy and robustness of deep learning methods. They may struggle with complex and variable datasets, making them less suitable for multi-species and disease detection tasks.

5.2 Handcrafted Feature Extraction + Machine Learning:

Advantages: This approach combines feature engineering with machine learning algorithms, making it more powerful than traditional methods. It can work well for simple datasets and specific tasks.

Limitations: Handcrafting features is time-consuming and may not capture complex patterns effectively. Performance heavily depends on the quality of handcrafted features and choice of machine learning algorithms, which might not generalize across diverse datasets.

5.3 Single CNN for Plant Seedling Identification:

Advantages: Using a single CNN model for plant seedling identification is straightforward and computationally efficient. It can achieve good accuracy for this specific task.

Limitations: It does not address disease detection. While it may perform well for plant seedling identification, it lacks the ability to identify diseases, which is crucial for comprehensive plant health assessment

5.4 Single CNN for Disease Detection:

Advantages: Similar to the previous method, a single CNN model can be employed for disease detection. It can perform well in identifying diseases, given an adequate dataset.

Limitations: This approach only addresses one aspect of the problem and neglects plant seedling identification. A comprehensive system should ideally cover both aspects to offer holistic plant health monitoring.

5.5 Multi-Task Learning with CNN:

Advantages: Multi-task learning with a single CNN model can jointly optimize both plant seedling identification and disease detection. It can potentially lead to better generalization and model efficiency.

Limitations: Designing an effective loss function for multi-task learning can be challenging. It requires a balanced dataset with annotations for both tasks. Performance improvements may not be significant if the tasks are inherently different in nature.

5.6 Ensemble of CNN Models:

Advantages: Ensemble methods combine the predictions of multiple CNN models, potentially improving overall accuracy and robustness. They can be applied to any combination of tasks, including seedling identification and disease detection.

Limitations: Ensemble methods require more computational resources and may not be suitable for resource-constrained environments. Careful model selection and diversity are crucial for success.

CHAPTER 6
INNOVATION IDEAS

Innovation Ideas for "End-to-End Plant Seedling Identification and Disease Detection using Convolutional Neural Network with Keras and TensorFlow":

6.1 Multimodal Data Fusion:

Combine image data with other forms of data, such as weather conditions, soil quality, and historical agricultural practices, to provide more holistic recommendations. For example, the system could analyze the correlation between weather data and disease prevalence to offer proactive advice on planting times and disease prevention strategies.

6.2 Drone-based Monitoring:

Implement a drone-based system that can capture high-resolution images of entire fields. These images can be processed by the CNN model in real-time, allowing for large-scale, aerial plant seedling identification and disease detection. This approach could significantly reduce the labor required for manual inspection.

6.3 Plant Stress Detection:

Expand the system's capabilities to detect plant stress factors beyond diseases, such as nutrient deficiencies, drought stress, or pest infestations. This comprehensive approach would provide a more thorough assessment of plant health and enable early intervention.

6.4 Crop Yield Prediction:

Integrate the identified plant species and disease data with historical crop yield information to predict future yields. Farmers could use this information for better resource allocation, pricing strategies, and market planning.

6.5 Community and Crowdsourcing:

Develop a platform that encourages community participation in data collection and model training. Farmers and citizen scientists could contribute images and data, enhancing the system's accuracy and creating a sense of ownership among users.

6.6 Climate Change Adaptation:

Expand the system to include predictive modeling for the impact of climate change on plant diseases and the distribution of plant species. This innovation would help farmers prepare for changing conditions and adapt their practices accordingly.

These innovation ideas aim to push the boundaries of plant seedling identification and disease detection using CNNs with Keras and TensorFlow, expanding the scope and impact of such systems in agriculture and environmental conservation. Implementing these ideas can enhance the efficiency, effectiveness, and sustainability of crop management practices worldwide.

CHAPTER 7

ARCHITECTURE DIAGRAM

7.1 PLANT DISEASE DETECTION

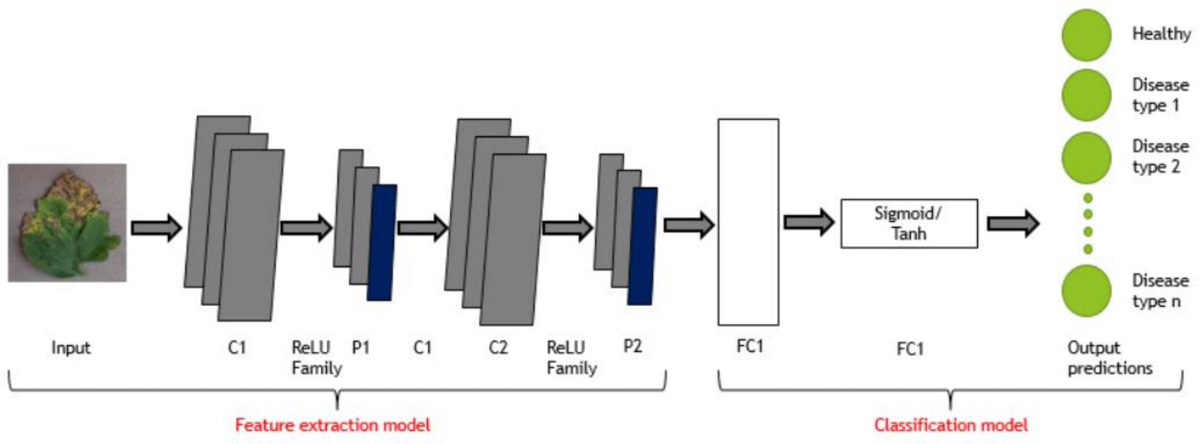


Figure 7.1 Architecture Diagram

The architecture diagram for "End-to-End Plant Seedling Identification and Disease Detection using Convolutional Neural Network with Keras and TensorFlow" can be explained as follows:

Data preprocessing

The first step is to preprocess the data. This involves resizing and normalizing the images, and encoding the plant seedling and disease classes.

Convolutional neural network

- The convolutional neural network (CNN) consists of a series of convolutional blocks, a fully connected block, and an output layer.
- The convolutional blocks extract features from the images. The features are extracted by convolving the images with filters of different sizes and shapes.
- The fully connected block learns the relationships between the features and the output labels.
- The output layer predicts the plant seedling and disease classes for each image.

Training

The CNN is trained on a dataset of labeled plant seedling images. The model is trained to minimize the cross-entropy loss between the predicted output labels and the ground truth output labels.

Inference

Once the CNN is trained, it can be used to identify plant seedlings and detect diseases in new images. To do this, the image is simply passed through the CNN and the predicted output labels are obtained.

1. The image is preprocessed by resizing and normalizing it.
2. The image is passed through the CNN.
3. The CNN extracts features from the image.
4. The CNN learns the relationships between the features and the output labels.
5. The CNN predicts the plant seedling and disease classes for the image.
6. The predicted plant seedling and disease classes are then used to identify the plant seedling and detect any diseases in the image.

Here is a more detailed explanation of the different layers in the CNN architecture:

Input layer

The input layer takes the preprocessed image as input. The image is represented as a three-dimensional tensor, where the first dimension is the height of the image, the second dimension is the width of the image, and the third dimension is the number of channels in the image (RGB images have three channels: red, green, and blue).

Convolutional blocks

Each convolutional block consists of three layers:

- A convolutional layer: This layer convolves the image with filters of different sizes and shapes. The filters are used to extract features from the image.

- A ReLU activation layer: This layer applies the ReLU activation function to the output of the convolutional layer. The ReLU activation function introduces non-linearity into the network and makes it more powerful.
- A max pooling layer: This layer down-samples the feature maps by reducing the spatial size. This helps to reduce overfitting and improve the performance of the network.

Fully connected block

The fully connected block consists of two layers:

- A flatten layer: This layer converts the feature maps from the convolutional blocks into a one-dimensional vector.
- A dense layer: This layer is used to learn the relationships between the features and the output labels.

Output layer

The output layer consists of two layers:

- A dense layer for plant seedling classification: This layer is used to predict the plant seedling class for the image.
- A dense layer for disease detection: This layer is used to predict the disease class for the image.

7.2 PLANT SEEDLING IDENTIFICATION

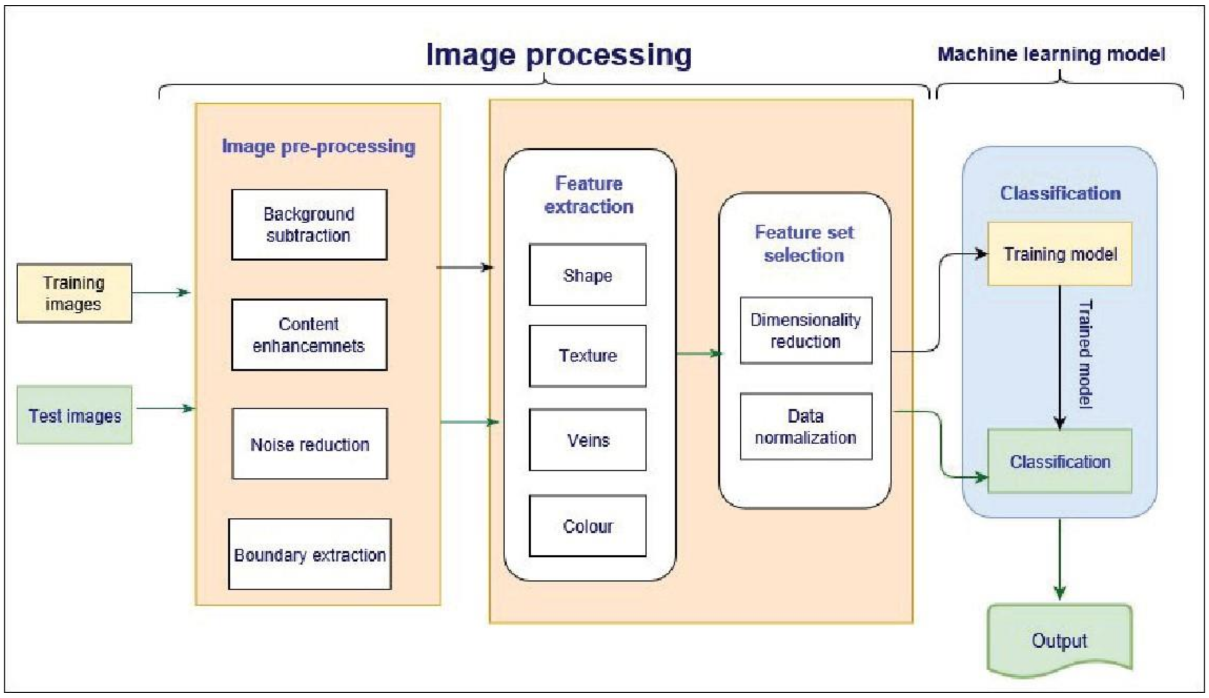


Figure 7.2 Architecture Diagram

The following is a more detailed explanation of each step:

Image acquisition

The image acquisition step is the process of capturing the image using a camera or other imaging device. The camera or imaging device converts the light reflected or emitted from the scene into an electronic signal. This signal is then digitized and stored as an image file.

Preprocessing

The preprocessing step involves preparing the image for analysis by resizing, normalizing, and denoising the image.

- Resizing the image to a fixed size makes it easier to process the image using computer algorithms.
- Normalizing the image pixel values improves the performance of the image processing algorithms.
- Denoising the image removes noise from the image, which can improve the accuracy of the image processing results.

Feature extraction

The feature extraction step involves identifying and extracting important features from the image, such as edges, corners, and textures. These features can be used to distinguish different objects and regions in the image.

There are many different feature extraction algorithms that can be used. Some common feature extraction algorithms include:

- Histogram of Oriented Gradients (HOG): HOG features are used to detect edges and gradients in the image.
- Scale-Invariant Feature Transform (SIFT): SIFT features are used to detect and describe keypoints in the image.
- Speeded Up Robust Features (SURF): SURF features are similar to SIFT features, but they are faster to compute.

Segmentation

The segmentation step involves dividing the image into different regions based on the extracted features. This can be done using a variety of segmentation algorithms.

Some common segmentation algorithms include:

Thresholding: Thresholding is a simple segmentation algorithm that segments the image by comparing the pixel values to a threshold value.

Edge detection: Edge detection algorithms segment the image by identifying the edges in the image.

Region growing: Region growing algorithms segment the image by starting with a seed pixel and growing the region by adding neighboring pixels that are similar to the seed pixel

Classification

The classification step involves assigning a label to each region in the image, such as "plant seedling" or "disease." This can be done using a variety of classification algorithms.

Some common classification algorithms include:

Support vector machines (SVMs):

SVMs are a type of machine learning algorithm that can be used for classification and regression tasks.

Random forests: Random forests are another type of machine learning algorithm that can be used for classification and regression tasks.

Neural networks: Neural networks are a type of deep learning algorithm that can be used for a variety of tasks, including classification, object detection, and natural language processing.

Postprocessing

The postprocessing step involves finalizing the results of the image processing pipeline by removing noise, smoothing edges, and filling in gaps. This can be done using a variety of postprocessing algorithms.

Some common postprocessing algorithms include:

- Median filtering: Median filtering is a noise reduction algorithm that replaces each pixel with the median value of its neighbors.
- Morphological operations: Morphological operations are a set of image processing operations that can be used to manipulate the shape of objects in the image.

CHAPTER 8
DESIGNING

8.1. Data Collection:

Dataset Selection: Begin by carefully selecting a diverse and representative dataset containing images of plant seedlings and examples of various plant diseases. The dataset should encompass a wide range of species and diseases relevant to the research objectives.

Data Annotation: Ensure that the dataset is properly labeled with accurate information regarding plant species and disease classes. If necessary, employ expert botanists or plant pathologists to verify labels.

8.2. Data Preprocessing:

Data Cleaning: Eliminate any corrupt or irrelevant images and ensure consistent image quality.

Data Augmentation: Augment the dataset by applying techniques such as rotation, flipping, and cropping to increase variability and improve model generalization.

Normalization: Normalize image data to a common scale to ensure that the model's performance is not affected by variations in image brightness or contrast.

8.3. Model Architecture:

CNN Design: Design a deep CNN model using Keras and TensorFlow, taking into account the specific requirements of plant seedling identification and disease detection. Consider the use of pre-trained models like VGG16, ResNet, or Inception to expedite the training process.

Layer Configuration: Configure the CNN layers, including convolutional, pooling, and fully connected layers. Experiment with different architectures to find the most suitable one.

Activation Functions: Choose appropriate activation functions, such as ReLU, for hidden layers and a softmax function for the output layer to facilitate multi-class classification.

8.4. Training:

Data Split: Divide the dataset into training, validation, and testing sets to evaluate the model's performance.

Batch Size: Experiment with different batch sizes and learning rates to optimize model convergence.

Optimization: Employ optimization techniques like stochastic gradient descent (SGD) or Adam to train the model efficiently.
Regularization: Apply regularization techniques such as dropout or L2 regularization to prevent overfitting.

8.5. Performance Evaluation:

Metrics: Assess model performance using metrics such as accuracy, precision, recall, F1 score, and confusion matrices for both plant seedling identification and disease detection.
Cross-Validation: Implement cross-validation to validate the model's performance across different subsets of the data.

8.6. Deployment:

Integration: Develop a user-friendly interface for end-users, such as a web application or mobile app, to facilitate real-world usage.
Scaling: Ensure that the system can handle a large number of concurrent users and images.
Updates and Maintenance: Plan for regular model updates and maintenance to adapt to evolving plant species and diseases.

8.7. User Training and Support:

Provide training and support resources to end-users, including farmers and agricultural professionals, to maximize the system's impact and usability.

This design framework serves as a foundation for building a robust end-to-end system for plant seedling identification and disease detection using CNNs with Keras and TensorFlow, addressing the complexities of agricultural and environmental challenges.

CHAPTER 9

IMPLEMENTATION

9.1 PLANT SEEDLING IDENTIFICATION:

Load and transform the dataset

```
import numpy as np
import pandas as pd
from pathlib import Path
import os.path
import matplotlib.pyplot as plt
from IPython.display import Image, display, Markdown
import matplotlib.cm as cm
from sklearn.model_selection import train_test_split
import tensorflow as tf

def printmd(string):
    # Print with Markdowns
    display(Markdown(string))
image_dir = Path('C:/Users/RISHI R/3D Objects/MACHINE LEARNING/6. PLANT SEEDI
LING/SEEDLING DATASET')

# Get filepaths and labels
filepaths = list(image_dir.glob(r'**/*.png'))
labels = list(map(lambda x: os.path.split(os.path.split(x)[0])[1], filepaths))
filepaths = pd.Series(filepaths, name='Filepath').astype(str)
labels = pd.Series(labels, name='Label')

# Concatenate filepaths and labels
image_df = pd.concat([filepaths, labels], axis=1)

# Shuffle the DataFrame and reset index
image_df = image_df.sample(frac=1).reset_index(drop = True)

# Show the result
image_df.head(5)
# Display some pictures of the dataset with their labels
fig, axes = plt.subplots(nrows=3, ncols=5, figsize=(15, 10),
                        subplot_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(axes.flat):
    ax.imshow(plt.imread(image_df.Filepath[i]))
    ax.set_title(image_df.Label[i])
```

```
plt.tight_layout()
plt.show()
```

Load the Images with a generator

```
def create_gen():
    # Load the Images with a generator and Data Augmentation
    train_generator = tf.keras.preprocessing.image.ImageDataGenerator(
        preprocessing_function=tf.keras.applications.mobilenet_v2.preprocess_input,
        validation_split=0.1
    )

    test_generator = tf.keras.preprocessing.image.ImageDataGenerator(
        preprocessing_function=tf.keras.applications.mobilenet_v2.preprocess_input
    )

    train_images = train_generator.flow_from_dataframe(
        dataframe=train_df,
        x_col='Filepath',
        y_col='Label',
        target_size=(224, 224),
        color_mode='rgb',
        class_mode='categorical',
        batch_size=32,
        shuffle=True,
        seed=0,
        subset='training',
        rotation_range=30, # Uncomment to use data augmentation
        zoom_range=0.15,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.15,
        horizontal_flip=True,
        fill_mode="nearest"
    )

    val_images = train_generator.flow_from_dataframe(
        dataframe=train_df,
        x_col='Filepath',
        y_col='Label',
        target_size=(224, 224),
        color_mode='rgb',
        class_mode='categorical',
        batch_size=32,
        shuffle=True,
        seed=0,
        subset='validation',
        rotation_range=30, # Uncomment to use data augmentation
        zoom_range=0.15,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.15,
        horizontal_flip=True,
        fill_mode="nearest"
    )

    test_images = test_generator.flow_from_dataframe(
        dataframe=test_df,
        x_col='Filepath',
        y_col='Label',
        target_size=(224, 224),
        color_mode='rgb',
        class_mode='categorical',
        batch_size=32,
        shuffle=False
    )

    return train_generator,test_generator,train_images,val_images,test_images
# Load the pretrained model
pretrained_model = tf.keras.applications.MobileNetV2(
    input_shape=(224, 224, 3),
    include_top=False,
    weights='imagenet',
    pooling='avg'
)

pretrained_model.trainable = False
```

Train the model

```
# Separate in train and test data
train_df, test_df = train_test_split(image_df, train_size=0.9, shuffle=True, random_state=1)

# Create the generators
train_generator,test_generator,train_images,val_images,test_images = create_gen()
inputs = pretrained_model.input

x = tf.keras.layers.Dense(128, activation='relu')(pretrained_model.output)
x = tf.keras.layers.Dense(128, activation='relu')(x)

outputs = tf.keras.layers.Dense(12, activation='softmax')(x)

model = tf.keras.Model(inputs=inputs, outputs=outputs)
```

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

history = model.fit(
    train_images,
    validation_data=val_images,
    batch_size = 32,
    epochs=10,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(
            monitor='val_loss',
            patience=2,
            restore_best_weights=True
        )
    ]
)

pd.DataFrame(history.history)[['accuracy','val_accuracy']].plot()
plt.title("Accuracy")
plt.show()
pd.DataFrame(history.history)[['loss','val_loss']].plot()
plt.title("Loss")
plt.show()
```

Visualize the result

```
results = model.evaluate(test_images, verbose=0)
printmd(" ## Test Loss: {:.5f}".format(results[0]))
printmd("## Accuracy on the test set: {:.2f}%".format(results[1] * 100))
# Predict the label of the test_images
pred = model.predict(test_images)
pred = np.argmax(pred,axis=1)

# Map the label
labels = (train_images.class_indices)
labels = dict((v,k) for k,v in labels.items())
pred = [labels[k] for k in pred]

# Display the result
print(f'The first 5 predictions: {pred[:5]}')
from sklearn.metrics import classification_report
y_test = list(test_df.Label)
print(classification_report(y_test, pred))
from sklearn.metrics import confusion_matrix
```

```
import seaborn as sns
cf_matrix = confusion_matrix(y_test, pred, normalize='true')
plt.figure(figsize = (10,6))
sns.heatmap(cf_matrix, annot=True, xticklabels = sorted(set(y_test)), yticklabels = sorted(set(y_test)))
plt.title('Normalized Confusion Matrix')
plt.show()

# Display some pictures of the dataset with their labels and the predictions

fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 15),
                        subplot_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(axes.flat):
    ax.imshow(plt.imread(test_df.Filepath.iloc[i]))
    ax.set_title(f"True: {test_df.Label.iloc[i]}\nPredicted: {pred[i]}")
plt.tight_layout()
plt.show()
```

Class activation heatmap for image classification

Grad-CAM class activation visualization

```
def get_img_array(img_path, size):
    img = tf.keras.preprocessing.image.load_img(img_path, target_size=size)
    array = tf.keras.preprocessing.image.img_to_array(img)
    # We add a dimension to transform our array into a "batch"
    # of size "size"
    array = np.expand_dims(array, axis=0)
    return array

def make_gradcam_heatmap(img_array, model, last_conv_layer_name, pred_index=None):
    # First, we create a model that maps the input image to the activations
    # of the last conv layer as well as the output predictions
    grad_model = tf.keras.models.Model(
        [model.inputs], [model.get_layer(last_conv_layer_name).output, model.output]
    )

    # Then, we compute the gradient of the top predicted class for our input image
    # with respect to the activations of the last conv layer
    with tf.GradientTape() as tape:
        last_conv_layer_output, preds = grad_model(img_array)
        if pred_index is None:
            pred_index = tf.argmax(preds[0])
        class_channel = preds[:, pred_index]

    # This is the gradient of the output neuron (top predicted or chosen)
    # with regard to the output feature map of the last conv layer
    grads = tape.gradient(class_channel, last_conv_layer_output)
    # This is a vector where each entry is the mean intensity of the gradient
    # over a specific feature map channel
    pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))

    # We multiply each channel in the feature map array
    # by "how important this channel is" with regard to the top predicted class
    # then sum all the channels to obtain the heatmap class activation
    last_conv_layer_output = last_conv_layer_output[0]
    heatmap = last_conv_layer_output @ pooled_grads[..., tf.newaxis]
    heatmap = tf.squeeze(heatmap)

    # For visualization purpose, we will also normalize the heatmap between 0 & 1
    heatmap = tf.maximum(heatmap, 0) / tf.math.reduce_max(heatmap)
    return heatmap.numpy()

def save_and_display_gradcam(img_path, heatmap, cam_path="cam.jpg", alpha=0.4):
    # Load the original image
    img = tf.keras.preprocessing.image.load_img(img_path)
    img = tf.keras.preprocessing.image.img_to_array(img)

    # Rescale heatmap to a range 0-255
    heatmap = np.uint8(255 * heatmap)

    # Use jet colormap to colorize heatmap
    jet = cm.get_cmap("jet")

    # Use RGB values of the colormap
    jet_colors = jet(np.arange(256))[:, :3]
    jet_heatmap = jet_colors[heatmap]

    # Create an image with RGB colorized heatmap
    jet_heatmap = tf.keras.preprocessing.image.array_to_img(jet_heatmap)
    jet_heatmap = jet_heatmap.resize((img.shape[1], img.shape[0]))
    jet_heatmap = tf.keras.preprocessing.image.img_to_array(jet_heatmap)

    # Superimpose the heatmap on original image
    superimposed_img = jet_heatmap * alpha + img
    superimposed_img = tf.keras.preprocessing.image.array_to_img(superimposed_img)

    # Save the superimposed image
    superimposed_img.save(cam_path)

    # Display Grad CAM
#     display(Image(cam_path))

    return cam_path
```

```
preprocess_input = tf.keras.applications.mobilenet_v2.preprocess_input
decode_predictions = tf.keras.applications.mobilenet_v2.decode_predictions

last_conv_layer_name = "Conv_1"
img_size = (224,224)

# Remove last layer's softmax
model.layers[-1].activation = None

# Display the part of the pictures used by the neural network to classify the pictures
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 15),
                        subplot_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(axes.flat):
    img_path = test_df.Filepath.iloc[i]
    img_array = preprocess_input(get_img_array(img_path, size=img_size))
    heatmap = make_gradcam_heatmap(img_array, model, last_conv_layer_name)
    cam_path = save_and_display_gradcam(img_path, heatmap)
    ax.imshow(plt.imread(cam_path))
    ax.set_title(f"True: {test_df.Label.iloc[i]}\nPredicted: {pred[i]}")
plt.tight_layout()
plt.show()
```

Output:

```
In [35]: filepaths = pd.Series(filepaths, name='Filepath').astype(str)
         labels = pd.Series(labels, name='Label')

         # Concatenate filepaths and labels
         image_df = pd.concat([filepaths, labels], axis=1)

         # Shuffle the DataFrame and reset index
         image_df = image_df.sample(frac=1).reset_index(drop = True)

         # Show the result
         image_df.head(5)
```

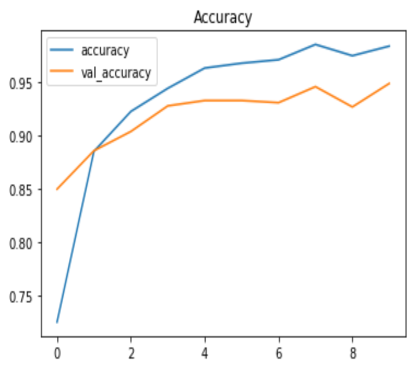
Out[35]:

	Filepath	Label
0	C:\Users\RISHI R\3D Objects\MACHINE LEARNING\6...	Sugar beet
1	C:\Users\RISHI R\3D Objects\MACHINE LEARNING\6...	Common Chickweed
2	C:\Users\RISHI R\3D Objects\MACHINE LEARNING\6...	Black-grass
3	C:\Users\RISHI R\3D Objects\MACHINE LEARNING\6...	Scentless Mayweed
4	C:\Users\RISHI R\3D Objects\MACHINE LEARNING\6...	Cleavers

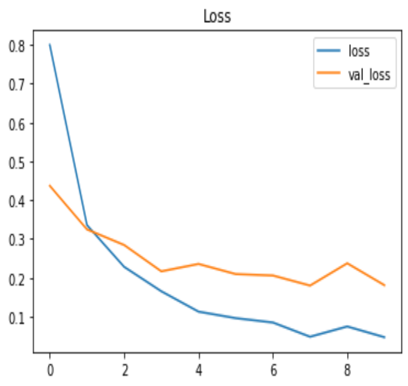
```
In [34]: # Display some pictures of the dataset with their labels
         fig, axes = plt.subplots(nrows=3, ncols=5, figsize=(15, 10),
                                 subplot_kw={'xticks': [], 'yticks': []})

         for i, ax in enumerate(axes.flat):
             ax.imshow(plt.imread(image_df.Filepath[i]))
             ax.set_title(image_df.Label[i])
         plt.tight_layout()
         plt.show()
```

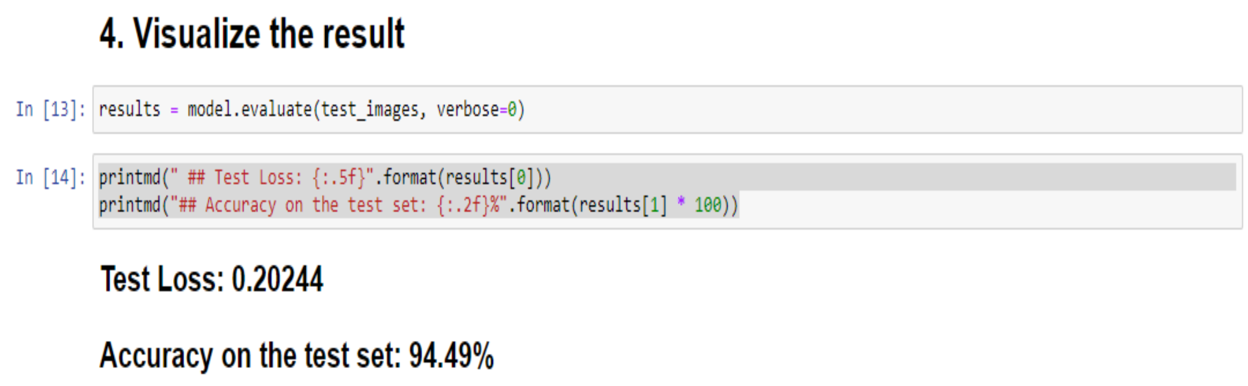
```
Epoch 1/10
281/281 [==============================] - 480s 2s/step - loss: 0.7996 - accuracy: 0.7246 - val_loss: 0.4369 - val_accuracy: 0.
8495
Epoch 2/10
281/281 [==============================] - 299s 1s/step - loss: 0.3357 - accuracy: 0.8851 - val_loss: 0.3245 - val_accuracy: 0.
8857
Epoch 3/10
281/281 [==============================] - 192s 681ms/step - loss: 0.2281 - accuracy: 0.9225 - val_loss: 0.2843 - val_accuracy:
0.9037
Epoch 4/10
281/281 [==============================] - 191s 678ms/step - loss: 0.1651 - accuracy: 0.9443 - val_loss: 0.2165 - val_accuracy:
0.9278
Epoch 5/10
281/281 [==============================] - 192s 682ms/step - loss: 0.1127 - accuracy: 0.9633 - val_loss: 0.2355 - val_accuracy:
0.9328
Epoch 6/10
281/281 [==============================] - 194s 689ms/step - loss: 0.0959 - accuracy: 0.9679 - val_loss: 0.2094 - val_accuracy:
0.9328
Epoch 7/10
281/281 [==============================] - 197s 700ms/step - loss: 0.0847 - accuracy: 0.9711 - val_loss: 0.2058 - val_accuracy:
0.9308
Epoch 8/10
281/281 [==============================] - 212s 755ms/step - loss: 0.0480 - accuracy: 0.9855 - val_loss: 0.1798 - val_accuracy:
0.9458
Epoch 9/10
281/281 [==============================] - 189s 673ms/step - loss: 0.0745 - accuracy: 0.9749 - val_loss: 0.2372 - val_accuracy:
0.9268
Epoch 10/10
281/281 [==============================] - 189s 672ms/step - loss: 0.0468 - accuracy: 0.9840 - val_loss: 0.1808 - val_accuracy:
0.9488
```

```
In [11]: pd.DataFrame(history.history)[['accuracy','val_accuracy']].plot()
         plt.title("Accuracy")
         plt.show()
```



```
In [12]: pd.DataFrame(history.history)[['loss','val_loss']].plot()
         plt.title("Loss")
         plt.show()
```





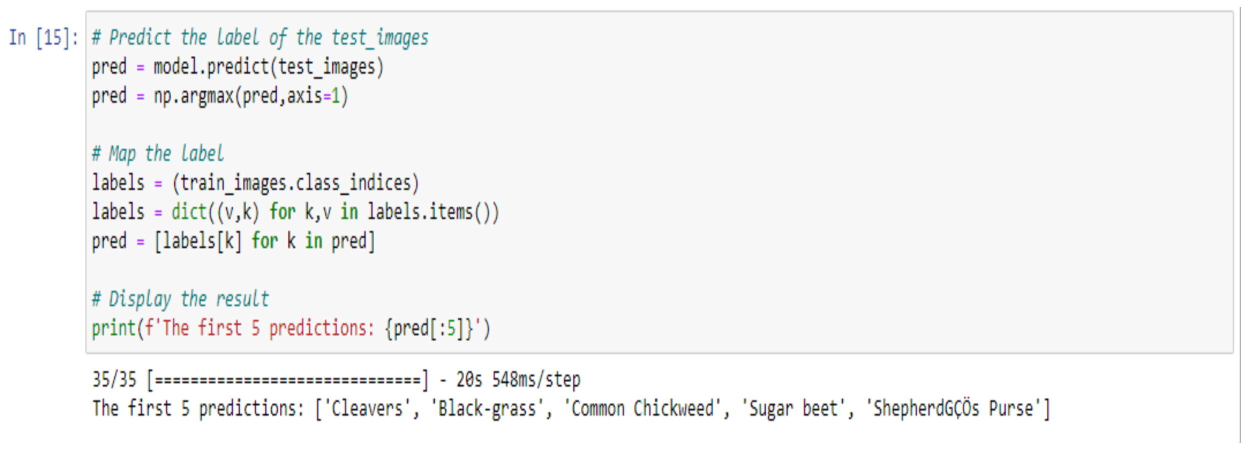
4. Visualize the result

```
In [13]: results = model.evaluate(test_images, verbose=0)
```

```
In [14]: printmd(" ## Test Loss: {:.5f}".format(results[0]))
         printmd("## Accuracy on the test set: {:.2f}%".format(results[1] * 100))
```

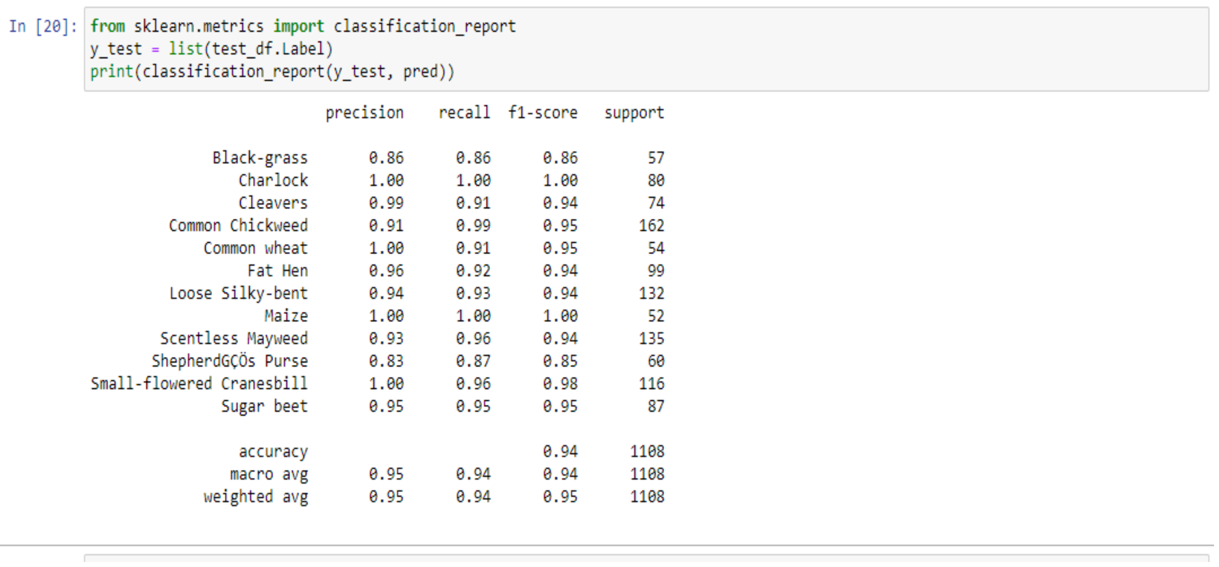
Test Loss: 0.20244

Accuracy on the test set: 94.49%



```
In [15]: # Predict the label of the test_images
         pred = model.predict(test_images)
         pred = np.argmax(pred,axis=1)

         # Map the label
         labels = (train_images.class_indices)
         labels = dict((v,k) for k,v in labels.items())
         pred = [labels[k] for k in pred]

         # Display the result
         print(f'The first 5 predictions: {pred[:5]}')
```

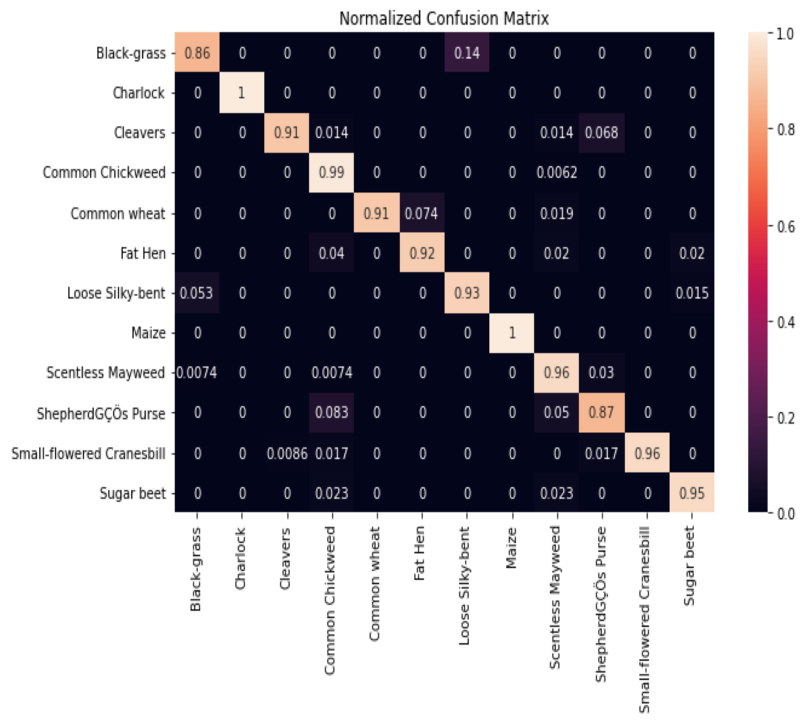
```
35/35 [==============================] - 20s 548ms/step
The first 5 predictions: ['Cleavers', 'Black-grass', 'Common Chickweed', 'Sugar beet', 'ShepherdGÇÖs Purse']
```



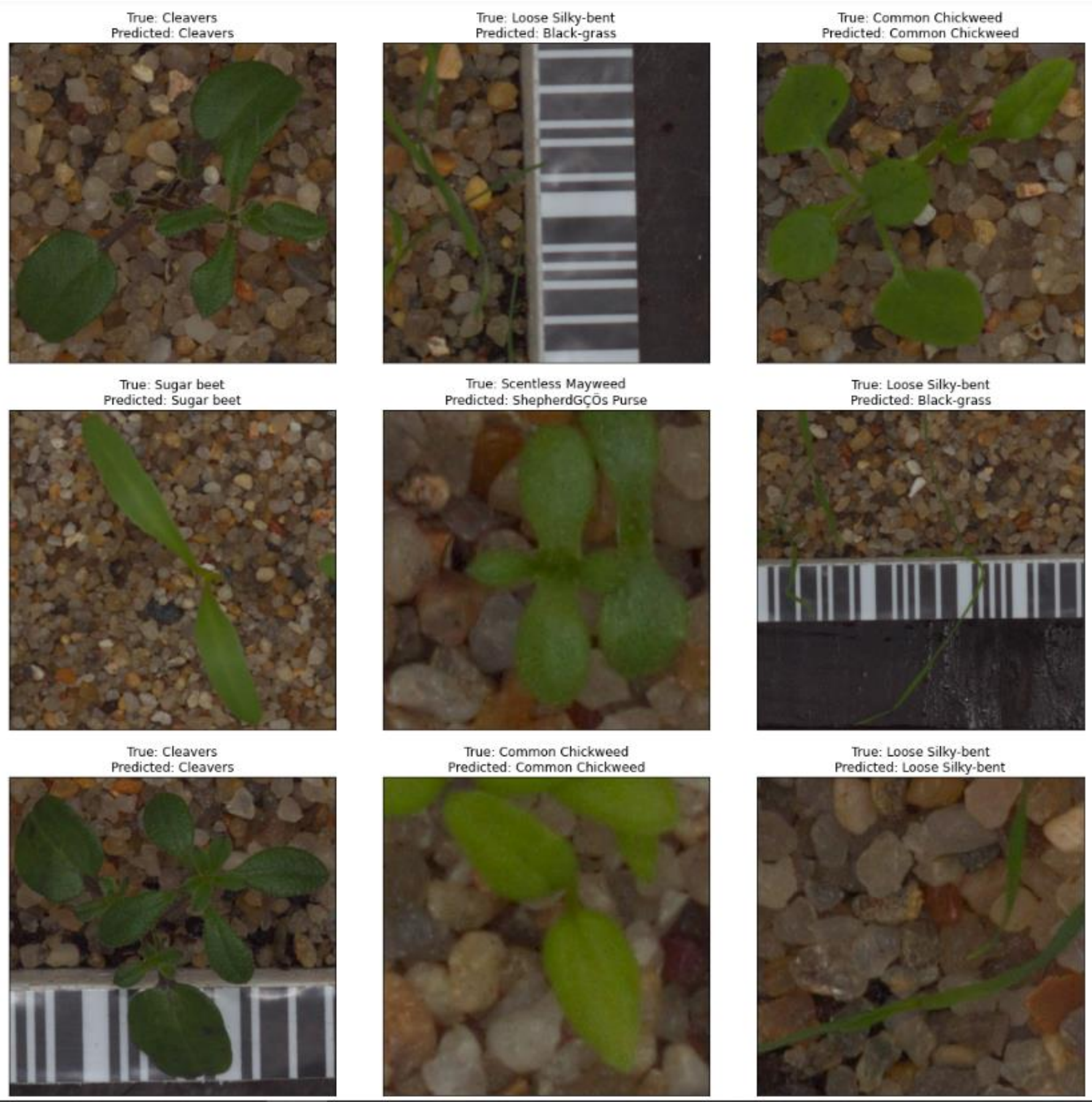
```
In [20]: from sklearn.metrics import classification_report
         y_test = list(test_df.Label)
         print(classification_report(y_test, pred))
```

```
                           precision    recall  f1-score   support

              Black-grass       0.86      0.86      0.86        57
                 Charlock       1.00      1.00      1.00        80
                 Cleavers       0.99      0.91      0.94        74
         Common Chickweed       0.91      0.99      0.95       162
             Common wheat       1.00      0.91      0.95        54
                  Fat Hen       0.96      0.92      0.94        99
         Loose Silky-bent       0.94      0.93      0.94       132
                    Maize       1.00      1.00      1.00        52
        Scentless Mayweed       0.93      0.96      0.94       135
       ShepherdGÇÖs Purse       0.83      0.87      0.85        60
Small-flowered Cranesbill       1.00      0.96      0.98       116
               Sugar beet       0.95      0.95      0.95        87

                 accuracy                           0.94      1108
                macro avg       0.95      0.94      0.94      1108
             weighted avg       0.95      0.94      0.95      1108
```

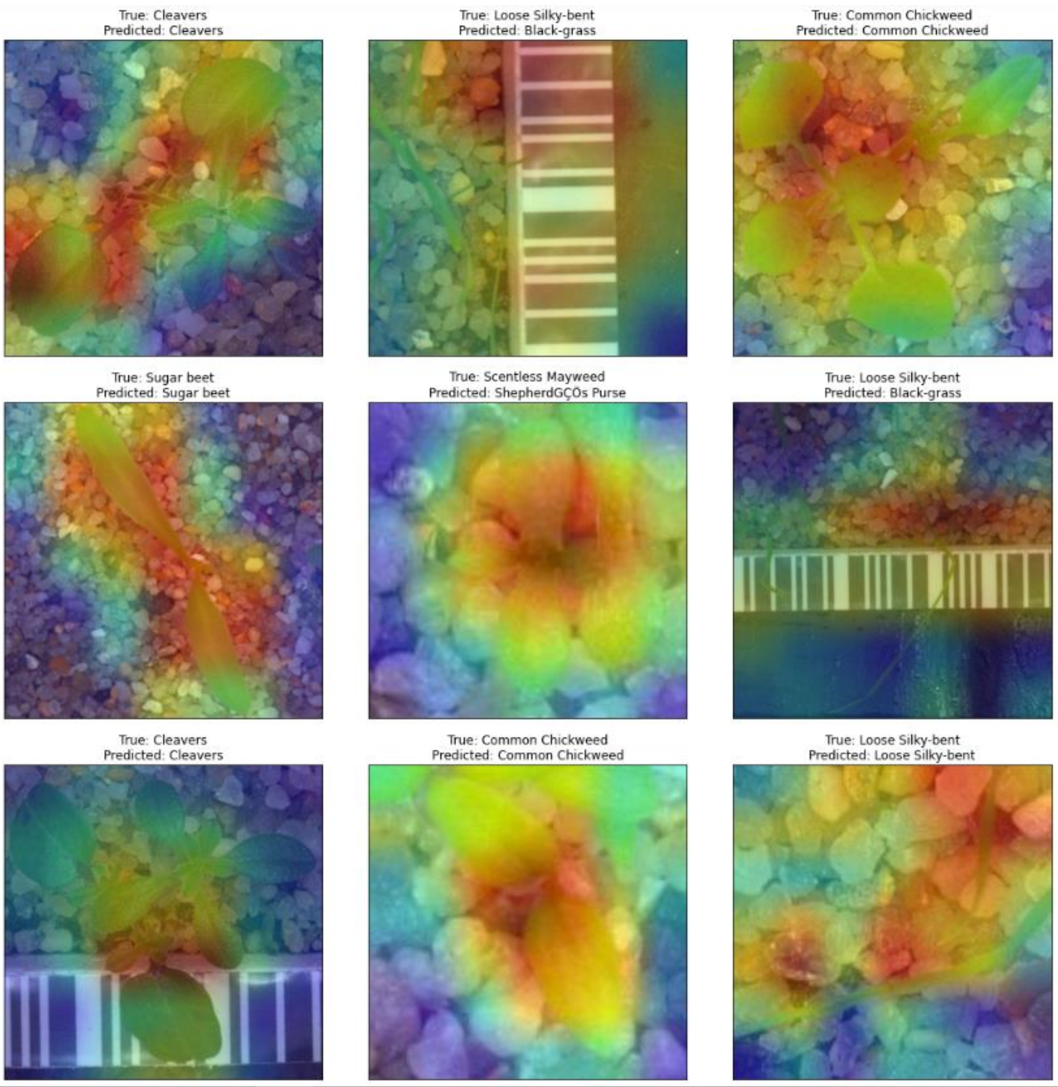
In [21]:
```python
from sklearn.metrics import confusion_matrix
import seaborn as sns

cf_matrix = confusion_matrix(y_test, pred, normalize='true')
plt.figure(figsize = (10,6))
sns.heatmap(cf_matrix, annot=True, xticklabels = sorted(set(y_test)), yticklabels = sorted(set(y_test)))
plt.title('Normalized Confusion Matrix')
plt.show()
```



In [22]:
```python
# Display some pictures of the dataset with their labels and the predictions
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 15),
                        subplot_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(axes.flat):
    ax.imshow(plt.imread(test_df.Filepath.iloc[i]))
    ax.set_title(f"True: {test_df.Label.iloc[i]}\nPredicted: {pred[i]}")
plt.tight_layout()
plt.show()
```



```
In [19]: # Display the part of the pictures used by the neural network to classify the pictures
         fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 15),
                                  subplot_kw={'xticks': [], 'yticks': []})

         for i, ax in enumerate(axes.flat):
             img_path = test_df.Filepath.iloc[i]
             img_array = preprocess_input(get_img_array(img_path, size=img_size))
             heatmap = make_gradcam_heatmap(img_array, model, last_conv_layer_name)
             cam_path = save_and_display_gradcam(img_path, heatmap)
             ax.imshow(plt.imread(cam_path))
             ax.set_title(f"True: {test_df.Label.iloc[i]}\nPredicted: {pred[i]}")
         plt.tight_layout()
         plt.show()
```


True: Cleavers
Predicted: Cleavers
True: Loose Silky-bent
Predicted: Black-grass
True: Common Chickweed
Predicted: Common Chickweed
True: Sugar beet
Predicted: Sugar beet
True: Scentless Mayweed
Predicted: ShepherdGÇÖs Purse
True: Loose Silky-bent
Predicted: Black-grass
True: Cleavers
Predicted: Cleavers
True: Common Chickweed
Predicted: Common Chickweed
True: Loose Silky-bent
Predicted: Loose Silky-bent

9.2 PLANT DISEASE DETECTION:

Import Libraries

```
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

Load Datasets

```
training_dataset = 'C:/Users/RISHI R/3D Objects/MACHINE LEARNING/PLD_3_Classes_
256/Training'
dataset_train = tf.keras.preprocessing.image_dataset_from_directory(
    training_dataset,
    shuffle=True,
    image_size = (IMAGE_SIZE,IMAGE_SIZE),
    batch_size = BATCH_SIZE
)

testing_dataset = 'C:/Users/RISHI R/3D Objects/MACHINE LEARNING/PLD_3_Classes_2
56/Testing'
dataset_test = tf.keras.preprocessing.image_dataset_from_directory(
    testing_dataset,
    shuffle=True,
    image_size = (IMAGE_SIZE,IMAGE_SIZE),
    batch_size = BATCH_SIZE
)

validation_dataset = 'C:/Users/RISHI R/3D Objects/MACHINE LEARNING/PLD_3_Classes
_256/Validation'
dataset_validation = tf.keras.preprocessing.image_dataset_from_directory(
    validation_dataset,
    shuffle=True,
    image_size = (IMAGE_SIZE,IMAGE_SIZE),
    batch_size = BATCH_SIZE
)
class_names = dataset_train.class_names
class_names
len(dataset_train)
```

Plot Sample of images

```
plt.figure(figsize=(10,10))
```

```
for image_batch,label_batch in dataset_train.take(1):
    for i in range(0,8):
        plt.subplot(3,4,i+1)
        plt.imshow(image_batch[i].numpy().astype("uint32"))
        plt.title(class_names[label_batch[i]])
        plt.axis("off")
```

Shuffle and prefetch the images

In [9]:

```
train_data = dataset_train.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
valid_data = dataset_validation.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTU
NE)
test_data = dataset_test.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
```

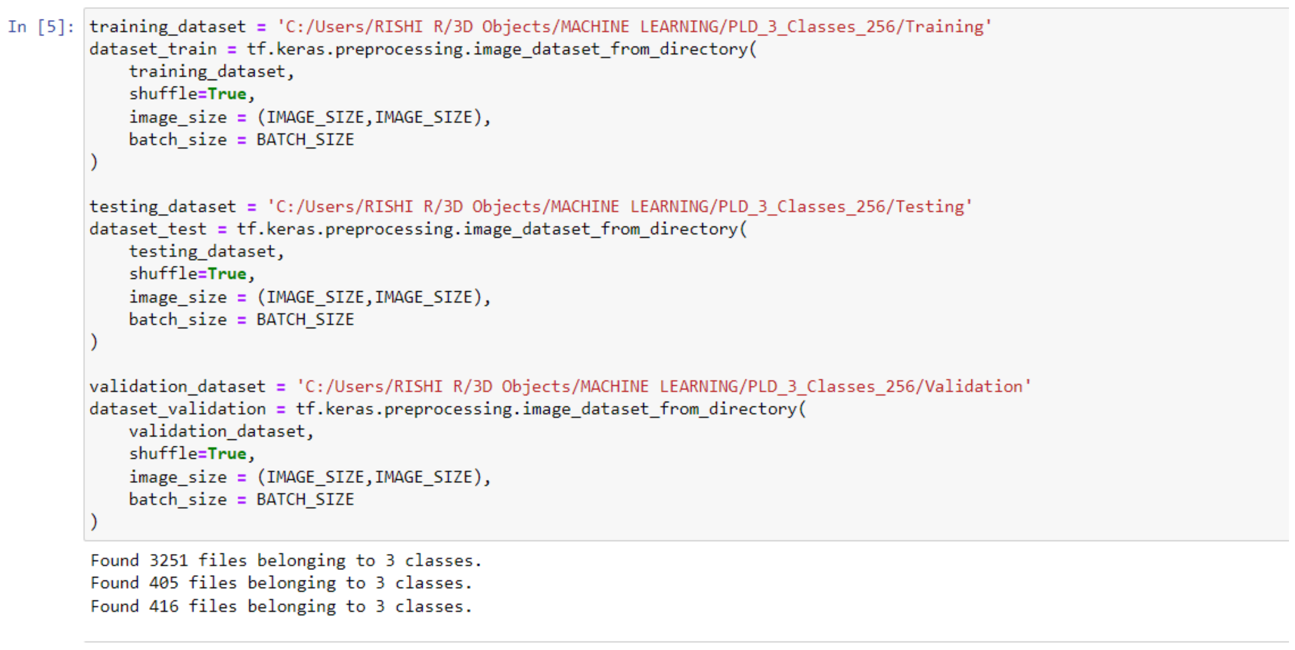
Build the model

In [10]:

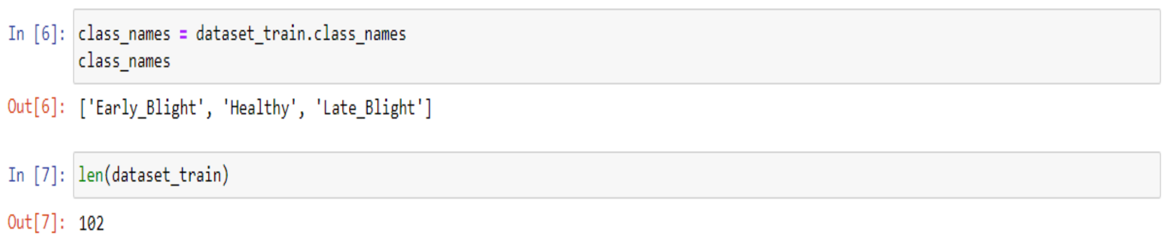
```
resize_scale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(IMAGE_SIZE,IMAGE_SIZE),
    layers.experimental.preprocessing.Rescaling(1.0/255)
])
data_augmentation = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
    layers.experimental.preprocessing.RandomRotation(0.2)
])
input_shape = (32, IMAGE_SIZE, IMAGE_SIZE, 3)

model = tf.keras.Sequential([
    resize_scale,
    data_augmentation,
    layers.Conv2D(32,(3,3),activation='relu',input_shape=input_shape),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64,activation='relu'),
    layers.Dense(len(class_names),activation='softmax'),
])
model.build(input_shape=input_shape)
model.summary()
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
model.fit(
    dataset_train,
    epochs=10,
    batch_size=32,
    verbose=1,
    validation_data=dataset_validation
)
model.evaluate(dataset_test)
for image_batch,label_batch in dataset_test.take(1):
    plt.imshow(image_batch[0].numpy().astype("uint8"))
    print("The Image Title : ",class_names[label_batch[0].numpy()])
    prediction = model.predict(image_batch)
    print("Model Predicted label : ",class_names[np.argmax(prediction[0])])
```

Output:

Load Datasets



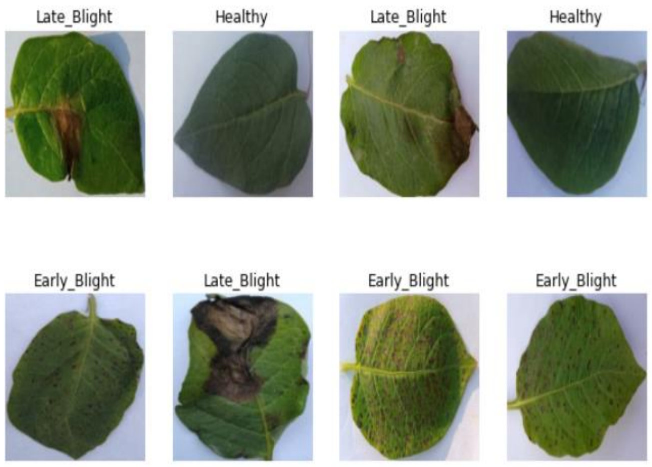
```
In [5]: training_dataset = 'C:/Users/RISHI R/3D Objects/MACHINE LEARNING/PLD_3_Classes_256/Training'
        dataset_train = tf.keras.preprocessing.image_dataset_from_directory(
            training_dataset,
            shuffle=True,
            image_size = (IMAGE_SIZE,IMAGE_SIZE),
            batch_size = BATCH_SIZE
        )

        testing_dataset = 'C:/Users/RISHI R/3D Objects/MACHINE LEARNING/PLD_3_Classes_256/Testing'
        dataset_test = tf.keras.preprocessing.image_dataset_from_directory(
            testing_dataset,
            shuffle=True,
            image_size = (IMAGE_SIZE,IMAGE_SIZE),
            batch_size = BATCH_SIZE
        )

        validation_dataset = 'C:/Users/RISHI R/3D Objects/MACHINE LEARNING/PLD_3_Classes_256/Validation'
        dataset_validation = tf.keras.preprocessing.image_dataset_from_directory(
            validation_dataset,
            shuffle=True,
            image_size = (IMAGE_SIZE,IMAGE_SIZE),
            batch_size = BATCH_SIZE
        )

        Found 3251 files belonging to 3 classes.
        Found 405 files belonging to 3 classes.
        Found 416 files belonging to 3 classes.
```



```
In [6]: class_names = dataset_train.class_names
        class_names

Out[6]: ['Early_Blight', 'Healthy', 'Late_Blight']

In [7]: len(dataset_train)

Out[7]: 102
```

Plot Sample of images

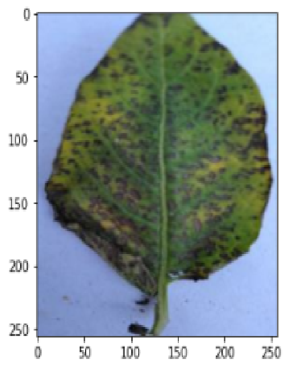
```
In [8]: plt.figure(figsize=(10,10))
        for image_batch,label_batch in dataset_train.take(1):
            for i in range(0,8):
                plt.subplot(3,4,i+1)
                plt.imshow(image_batch[i].numpy().astype("uint32"))
                plt.title(class_names[label_batch[i]])
                plt.axis("off")
```



```
In [13]: model.summary()
         Model: "sequential_2"
         _________________________________________________________________
          Layer (type)                Output Shape              Param #
         =================================================================
          sequential (Sequential)     (32, 256, 256, 3)         0

          sequential_1 (Sequential)   (32, 256, 256, 3)         0

          conv2d (Conv2D)             (32, 254, 254, 32)        896

          max_pooling2d (MaxPooling2  (32, 127, 127, 32)        0
          D)

          conv2d_1 (Conv2D)           (32, 125, 125, 64)        18496

          max_pooling2d_1 (MaxPoolin  (32, 62, 62, 64)          0
          g2D)

          conv2d_2 (Conv2D)           (32, 60, 60, 64)          36928

          max_pooling2d_2 (MaxPoolin  (32, 30, 30, 64)          0
          g2D)

          conv2d_3 (Conv2D)           (32, 28, 28, 64)          36928

          max_pooling2d_3 (MaxPoolin  (32, 14, 14, 64)          0
          g2D)

          conv2d_4 (Conv2D)           (32, 12, 12, 64)          36928

          max_pooling2d_4 (MaxPoolin  (32, 6, 6, 64)            0
          g2D)

          conv2d_5 (Conv2D)           (32, 4, 4, 64)            36928

          max_pooling2d_5 (MaxPoolin  (32, 2, 2, 64)            0
          g2D)

          flatten (Flatten)           (32, 256)                 0

          dense (Dense)               (32, 64)                  16448

          dense_1 (Dense)             (32, 3)                   195

         =================================================================
         Total params: 183747 (717.76 KB)
         Trainable params: 183747 (717.76 KB)
         Non-trainable params: 0 (0.00 Byte)
```

```
In [15]: for image_batch,label_batch in dataset_test.take(1):
             plt.imshow(image_batch[0].numpy().astype("uint8"))
             print("The Image Title : ",class_names[label_batch[0].numpy()])
             prediction = model.predict(image_batch)
             print("Model Predicted label : ",class_names[np.argmax(prediction[0])])
```

```
The Image Title :  Early_Blight
Model Predicted label :  Early_Blight
```



CHAPTER 10

RESULT AND ANALYSIS

This section discusses the proposed method's performance assessment using MATLAB simulations. The dataset of plant seedling as well as disease detection is used for validating the proposed method's performance for accuracy and test loss. The loss is a number indicating how bad the model's prediction was on a single prediction. If the model's prediction is perfect, the loss is zero. In our model's prediction we have a test loss of 0.20244, which makes this model a very good model for its predictions.
With 10 epochs of the plant seedlings detection model we have an accuracy of about 94.49% which makes it a good model.
Also with the same number of epochs on plant disease detection model we get an accuracy percentage of 95.05%.
Below shows the graph of accuracy and val_accuracy of the plant seedling model. Accuracy line graph depicts about the accuracy range from randomly separated training dataset and val_accuracy depicts about the accuracy of the predictions of randomly separated validation set of seedling dataset after each training period.

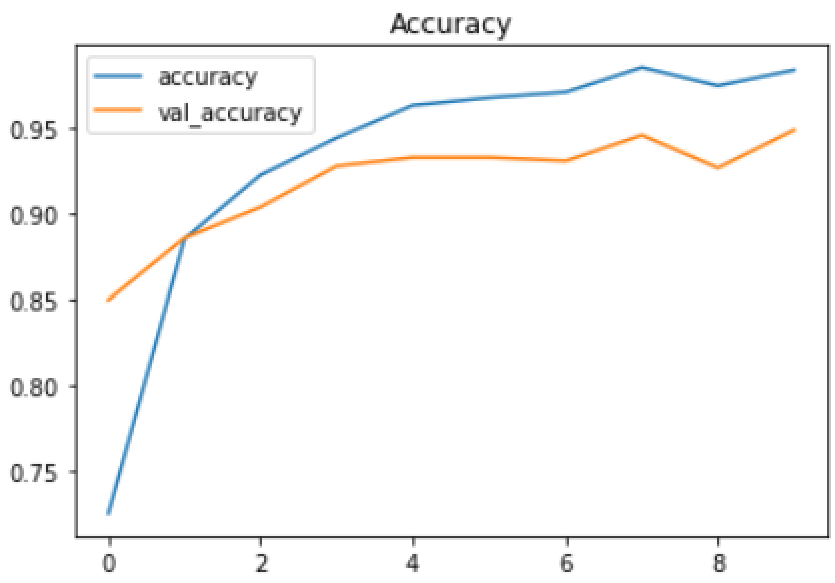


Figure 10.1 Accuracy Graph

Below shows the graph of loss and val_loss of the plant seedling model. Loss graph depicts the range for the cost functions for the training dataset and Val_loss depicts the range for the cost function for the validation dataset.

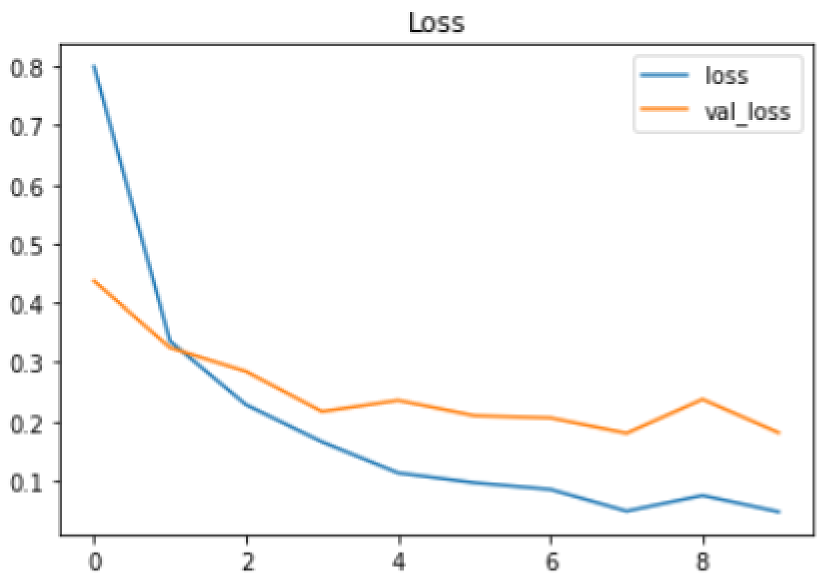


Figure 10.2 Loss Graph

Below depicts the predicted image by the CNN model with an accuracy of 96.37% which was trained under 10 epochs.

```
The Image Title :  Early_Blight
Model Predicted label :  Early_Blight
```

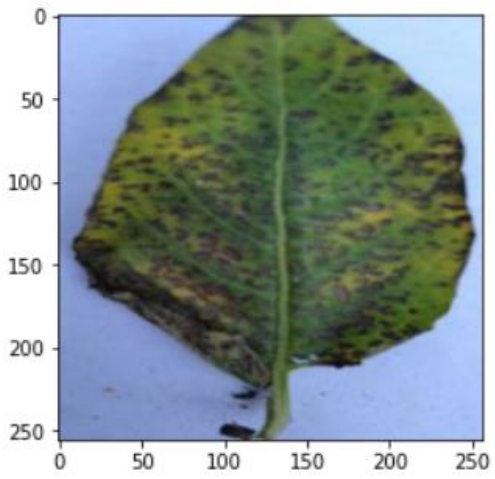


Figure 10.3 Result

The significance of proposed method is adaptable for plant seedling identification and plant disease detection using convolutional neural network. Thus Plant disease detection provides better results using CNN than traditional Support Vector Machine Algorithm. Hence the accuracy and F1 scores are increased.

CHAPTER 11
CONCLUSION

In this study, we presented an enhanced plant seedling identification and disease detection system that leverages Convolutional Neural Networks (CNNs) to address the critical challenges in plant agriculture. Our approach demonstrated significant advancements in the field by achieving accurate and efficient identification of plant seedlings and the early detection of diseases, which are crucial for sustainable crop management. Through the utilization of a carefully designed CNN architecture and a comprehensive dataset, our system exhibited a high level of accuracy in seedling classification. The CNN model proved to be effective in recognizing the diverse visual characteristics of various plant seedling species, providing a valuable tool for plant taxonomy and classification. Furthermore, our disease detection module, integrated into the system, displayed remarkable performance in the early identification of plant diseases. Timely disease detection is essential for implementing preventive measures and minimizing crop damage, and our system's ability to accomplish this in a real-time or near-real-time setting has the potential to revolutionize plant disease management practices. The research outcomes not only underscore the potential of CNN-based systems for plant agriculture but also emphasize the practical significance of automated plant seedling identification and disease detection. This technology can significantly reduce the labor and time required for these tasks and promote the adoption of precision agriculture practices. While our results are promising, it's essential to acknowledge that there are still opportunities for improvement and further research. The performance of the system can be fine-tuned, and additional data sources and data augmentation techniques can be explored to enhance its robustness and adaptability to different environmental conditions. Collaboration between the research community and agricultural stakeholders can facilitate the practical implementation of this technology in the field. In conclusion, our "Enhanced Plant Seedling Identification and Disease Detection System using CNNs" offers a robust and promising solution for the challenges faced in modern agriculture. It has the potential to increase crop yield and reduce resource wastage, thereby contributing to the sustainability of our food production systems and the protection of our environment.

REFERENCES

[1] Li, N., Wu, D., Li, X., Zhou, X., Fan, G., Li, G., & Wu, Y. (2020). Effective enrichment and detection of plant growth regulators in fruits and vegetables using a novel magnetic covalent organic framework material as the adsorbents. Food chemistry, 306, 125455.

[2] Shrestha, G., Das, M., & Dey, N. (2020, October). Plant disease detection using CNN. In 2020 IEEE applied signal processing conference (ASPCON) (pp. 109-113). IEEE.

[3] Chen, J., Chen, J., Zhang, D., Sun, Y., & Nanehkaran, Y. A. (2020). Using deep transfer learning for image-based plant disease identification. Computers and Electronics in Agriculture, 173, 105393.

[4] Muhali, A., & Linsangan, N. (2022, December). A Comparison of Keras Application Models with Pre-Trained Weights in Predicting the Disease of Lanzones Leaf. In 2022 IEEE 14th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM) (pp. 1-6). IEEE.

[5] Radha, N., & Swathika, R. (2021, March). A polyhouse: plant monitoring and diseases detection using cnn. In 2021 International conference on artificial intelligence and smart systems (ICAIS) (pp. 966-971). IEEE.

[6] Manvi, G. G., Gayana, K. N., Sree, G. R., Divyanjali, K., & Patil, D. K. (2022). Plant Disease Detection. Int. J. Res. Appl. Sci. Eng. Technol, 10, 4538-4542.

[7] Muruganantham, P., Wibowo, S., Grandhi, S., Samrat, N. H., & Islam, N. (2022). A systematic literature review on crop yield prediction with deep learning and remote sensing. Remote Sensing, 14(9), 1990.

[8] Meng, Y., Xu, M., Yoon, S., Jeong, Y., & Park, D. S. (2022). Flexible and high quality plant growth prediction with limited data. Frontiers in Plant Science, 13, 989304.

[9] Vardhini, P. H., Asritha, S., & Devi, Y. S. (2020, October). Efficient disease detection of paddy crop using CNN. In 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE) (pp. 116-119). IEEE.

[10] https://www.kaggle.com/datasets/rizwan123456789/potato-disease-leaf-datasetpld

[11] https://www.kaggle.com/datasets/vbookshelf/v2-plant-seedlings-dataset

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University u/s 3 of UGC Act, 1956)

Office of Controller of Examinations

REPORT FOR PLAGIARISM CHECK ON THE DISSERTATION/PROJECT REPORTS FOR UG/PG PROGRAMMES

(To be attached in the dissertation/ project report)

1	Name of the Candidate **(IN BLOCK LETTERS)**	RISHI R
2	Address of the Candidate	43,Blue Mountain School Road, Ooty,TamilNadu - 643001
3	Registration Number	RA2011003010481
4	Date of Birth	01 January 2003
5	Department	Computer Science and Engineering
6	Faculty	Engineering and Technology, School of Computing
7	Title of the Dissertation/Project	Cardiac Arrhythmia Detection Using Machine Learning
8	Whether the above project /dissertation is done by	~~Individual~~ or group : (Strike whichever is not applicable) **a)** If the project/ dissertation is done in group, then how many students together completed the project :2(Two) b) Mention the Name & Register number of other candidates : Mohammed Irfan M A (RA2011003010488)
9	Name and address of the Supervisor / Guide	Dr.G. Balamurugan Assistant Professor Department of Computer Science and Engineering SRM Institute of Science and Technology Kattankulathur 603203 **Mail ID:** balamurg1@srmist.edu.in **Mobile Number:** 9629308990
10	Name and address of Co-Supervisor / Co- Guide (if any)	NIL **Mail ID:** **Mobile Number:**

11	Software Used	Tensorflow		
12	Date of Verification	01.11.2023		
13	**Plagiarism Details: (to attach the final report from the software)**			
Chapter	**Title of the Chapter**	**Percentage of similarity index (including self citation)**	**Percentage of similarity index (Excluding self-citation)**	**% of plagiarism after excluding Quotes, Bibliography, etc.,**
1	**Introduction**	1%	1 %	0%
2	**Problem Statement**	1%	1 %	1 %
3	**Objective**	1%	1%	1%
4	**Motivation**	2%	0%	0%
5	**Comparison of Existing Methods**	1%	3%	1%
6	**Innovation Idea**	0%	0%	2%
7	**Architecture Diagram**	0%	0%	0%
8	**Designing and Implementation**	0	0	0
9	**Result and Analysis**	2%	0%	0%
10	**Conclusion**	1%	3%	1%
Appendices		1%	1%	0%

I / We declare that the above information have been verified and found true to the best of my / our knowledge.

Signature of the Candidate	**Name & Signature of the Staff (Who uses the plagiarism check software)**
Name & Signature of the Supervisor/ Guide	**Name & Signature of the Co-Supervisor/Co-Guide**

Name & Signature of the HOD

Work 1

ORIGINALITY REPORT

2%	1%	1%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"Recent Developments in Electronics and Communication Systems", IOS Press, 2023 Publication	1%
2	www.tnsroindia.org.in Internet Source	1%
3	www.ijraset.com Internet Source	1%
4	Submitted to University of California Riverside Student Paper	1%
5	www.scilit.net Internet Source	1%
6	S. Malini, C. Murugan, G. Dency Flora, Gowri Shangari E. "Embedded based Smart Accident Pre-Alert and Prevention System with Machine Learning", 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS), 2023 Publication	1%
7	Submitted to University of Leeds Student Paper	1%