A Project Report

on

AN EXPERT SYSTEM FOR DISEASE PREDICTION AND FERTILIZER RECOMMENDATION USING DEEP LEARNING

Submitted in partial fulfillment of the requirements

for the award of degree of

BACHELOR OF TECHNOLOGY

in

Information Technology

by

K. Geethika Reddy (20WH1A1270)

G. Krishna Prathibha (20WH1A12B0)

G. Sneha (20WH1A12B5)

Under the esteemed guidance of

Mr. A. Rajashekar Reddy

Assistant Professor

Department of Information Technology

BVRIT HYDERABAD College of Engineering for Women

Rajiv Gandhi Nagar, Nizampet Road, Bachupally, Hyderabad – 500090

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)

(NAAC 'A' Grade & NBA Accredited- ECE, EEE, CSE & IT)

June, 2024

DECLARATION

We hereby declare that the work presented in this project entitled **"An Expert System for Disease Prediction and Fertilizer Recommendation using Deep Learning"** submitted towards completion of IV year II sem of B.Tech IT at "BVRIT HYDERABAD College of Engineering for Women", Hyderabad is an authentic record of our original work carried out under the esteemed guidance of Mr. A. Rajashekar Reddy, Assistant Professor, Department of Information Technology.

Komatireddy Geethika Reddy(20WH1A1270)

Goda Krishna Prathibha (20WH1A12B0)

Gunjari Sneha (20WH1A12B5)

BVRIT HYDERABAD

College of Engineering for Women

Rajiv Gandhi Nagar, Nizampet Road, Bachupally, Hyderabad – 500090

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)

(NAAC 'A' Grade & NBA Accredited- ECE, EEE, CSE & IT)

CERTIFICATE

This is to certify that the Project report on **" An Expert System for Disease Prediction and Fertilizer Recommendation using Deep Learning "** is a bonafide work carried out by **K. Geethika Reddy (20WH1A1270), G. Krishna Prathibha (20WH1A12B0)** and **G. Sneha (20WH1A12B5)** in the partial fulfillment for the award of B.Tech degree in **Information Technology , BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad** affiliated to Jawaharlal Nehru Technological University, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide	**Head of the Department**
A. Rajashekar Reddy	**Dr. Aruna Rao S L**
Assistant Professor	**Professor & HoD**
Department of IT	**Department of IT**

External Examiner

ACKNOWLEDGEMENT

We would like to express our profound gratitude and thanks to **Dr. K. V. N. Sunitha, Principal, BVRIT HYDERABAD College of Engineering for Women** for providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. Aruna Rao S L, Professor & Head, Department of IT, BVRIT HYDERABAD College of Engineering for Women** for all the timely support, constant guidance and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Mr. A. Rajashekar Reddy, Assistant Professor, Department of IT, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinators **Dr. J Kavitha, Associate Professor and Dr. Mukhtar Ahmad Sofi , Assistant Professor**, all the faculty and staff of Department of IT who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

K Geethika Reddy (20WH1A1270)

G Krishna Prathibha (20WH1A12B0)

G Sneha (20WH1A12B5)

ABSTRACT

The agricultural production of tomatoes presents unique challenges that are pivotal to the preservation of global food supply and economic prosperity. Despite this, an important crop may still be vulnerable to a range of diseases. To make the matters worse, imprecise fertilizers application also aggravates diseases. This eventually results in poor harvests for farmers. It however does not only connote but also exists a tomato plant disease detection system, and this system the efficiency of the different advanced deep learning models. Their only problem is that they always give disease detection but not fertilizer recommendations. This paper will try to fill this gap by coming up with. A hybrid expert system for disease detection and user-friendly efficient fertilizer recommendations for tomato plants. The proposed expert system will employ advanced deep learning techniques such as MobileNetV2 and DenseNet121, DenseNet201 for disease recognition and feature extraction Moreover, machine learning models are based on rules, containing decision trees, random forests and others are used for fertilization recommendation. Dataset of "tomato farm" is used for training and validation. The presented AI based system for disease diagnosis and fertilizer recommendation could convert tomato cultivation towards a new direction that will improve the crop yield, reduce the environment burden and boost the sustainability of the agricultural activities. Its implementation will provide a glimmer of hope for a better and a more robust future for tomato production; it offers a holistic solution to the challenges faced by farmers and eventually to the improvement of the overall agricultural industry.

Keywords: Leaf disease detection, Fertilizer Recommendation, Transfer learning, Prediction

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

Abbrevation	Meaning
CNN	Convolutional Neural Network
ReLU	Reftified Linear Unit
GUI	Graphic User Interface
ML	Machine Learning
IoT	Internet of Things

CONTENTS

1. INTRODUCTION

In the realm of global agriculture, tomato production stands as a cornerstone for ensuring food security and economic stability. However, this vital crop faces substantial threats from various diseases, often exacerbated by inefficient fertilizer practices, resulting in significant losses for farmers. To address these pressing challenges, this project proposes an innovative expert system that seamlessly integrates disease detection with precise fertilizer recommendations tailored specifically for tomato plants.

The methodology employed leverages cutting-edge deep learning models such as MobileNet and DenseNet to achieve accurate disease detection. Concurrently, rule-based machine learning algorithms including decision trees, random forests, and XGBoost are utilized to offer targeted and efficient fertilizer recommendations. Potential future iterations of the system may explore alternative algorithms like Support Vector Machines or Naïve Bayes to further enhance performance and adaptability.

The proposed expert system aims to revolutionize tomato cultivation practices by providing farmers with a comprehensive tool to optimize yield, reduce environmental impact, and enhance agricultural sustainability. Through the integration of disease detection and fertilizer recommendation, this project promises to pave the way for a more resilient and efficient future in tomato agriculture.

This expert system is developed and validated using a comprehensive dataset, ensuring the reliability and effectiveness of the proposed solution. By harnessing the power of advanced technologies in both disease detection and agronomy, this project seeks to address critical pain points faced by tomato farmers worldwide.

The seamless integration of disease detection with fertilizer recommendations holds significant promise for transforming tomato cultivation practices. Farmers stand to benefit from a holistic

approach that not only identifies plant diseases accurately but also provides tailored guidance on optimal fertilizer application. This combination is poised to optimize crop yield, mitigate losses due to diseases and nutrient deficiencies, and contribute to the overall sustainability of the agricultural industry.

The proposed expert system represents a groundbreaking advancement in tomato cultivation techniques. Its implementation has the potential to empower farmers with actionable insights, ultimately fostering a more resilient and sustainable future for tomato agriculture on a global scale.

The project focuses on addressing key challenges in tomato cultivation by developing an integrated expert system that combines advanced disease detection with precise fertilizer recommendations. By leveraging state-of-the-art deep learning models and rule-based machine learning algorithms, the system aims to revolutionize farming practices to optimize crop yield, reduce losses due to diseases and nutrient deficiencies, and promote agricultural sustainability. This innovative approach not only identifies plant diseases accurately but also provides actionable guidance on optimal fertilizer usage, offering farmers a comprehensive tool to enhance productivity and economic stability in tomato agriculture. Through the seamless integration of technology and agronomy, the project seeks to empower farmers with practical solutions that can significantly improve the resilience and efficiency of tomato cultivation worldwide.

1.1. Objective

i) To develop a hybrid expert system integrating deep learning (MobileNetV2, DenseNet) for tomato disease detection.

ii) Implementation of machine learning models to provide precise and user-friendly fertilizer recommendations..

iii) Utilizing the ”Tomato Village”dataset for training and validation of the proposed system.

iv) Enhancing tomato crop yield by optimizing disease diagnosis and tailored fertilizer application.

v) Promote sustainability in agriculture by reducing environmental burden and improving overall productivity.

1.2. Problem Definition

The problem at hand revolves around the agricultural challenges faced in the cultivation of tomatoes, a crop of utmost significance for global food security and economic stability. Two critical issues exacerbate the viability of tomato production: the prevalence of diseases affecting the crop and the inefficiency in fertilizer application, leading to significant losses for farmers. The current reliance on labor-intensive manual inspection for disease detection results in inaccuracies, while conventional fertilizer recommendation systems lack precision, resulting in suboptimal nutrient utilization and potential environmental harm. These challenges underscore the need for an innovative solution that can address both disease detection and fertilizer application inefficiencies. The overarching problem is to develop a comprehensive, integrated system that leverages advanced computer vision and machine learning techniques to provide rapid and accurate disease detection, coupled with data-driven algorithms for precise fertilizer recommendations.

1.3. Modules

1) Disease Detection
2) Fertilizer Recommendation
3) GUI

2. LITERATURE SURVEY

M. Bakr, S. Abdel-Gaber, M. Nasr, and M. Hazman[1], The paper ”Tomato disease detection model based on densenet and trans- fer learning” introduces a transfer learning based model for detecting tomato leaf diseases. This study proposes a model of DenseNet201 as a transfer learning- based model and CNN classifier. A comparison study between four deep learning models (VGG16, Inception V3, ResNet152V2 and DenseNet201) done in order to determine the best accuracy in using transfer learning in plant disease detection. The used images dataset contains 22930 photos of tomato leaves in 10 differ- ent classes, 9 disorders and one healthy class.Plant diseases are a foremost risk to the safety of food. They have the potential to significantly reduce agricultural products quality and quantity. In agriculture sectors, it is the most prominent chal- lenge to recognize plant diseases. In computer vision, the Convolutional Neural Network (CNN) produces good results when solving image classification tasks. For plant disease diagnosis, many deep learning architectures have been applied.with this, they got 99.0 accuracy.

S. Ashok, G. Kishore, V. Rajesh, S. Suchitra, [2] The purpose of this project is to evaluate the Tomato Plant Leaf disease using image processing techniques based on Image segmentation, clustering, and open-source algorithms, thus all contributing to a reliable, safe, and accurate system of leaf disease with the specialization to Tomato Plants.Early Detection of Plant Leaf Detection is a major necessity in a growing agricultural economy like India. Not only as an agricultural economy but also with a large amount of population to feed, it is necessary that leaf diseases in plants are detected at a very early stage and predictive mechanisms to be adopted to make them safe and avoid losses to the agri-based economy.

S. Ahmed, M. B. Hasan, T. Ahmed, M. R. K. Sony, and M. H. Kabir, [3] In this paper author proposes a lightweight transfer learning-based approach for detecting diseases from tomato leaves. It utilizes an effective preprocessing method to enhance the leaf images with illumination correction for improved classification. Our system extracts features using a combined model consisting of a pretrained MobileNetV2 architecture and a classifier network for effective prediction. Traditional augmentation approaches are replaced by runtime augmentation to avoid data leakage and address the class imbalance issue. Evaluation on tomato leaf images from the PlantVillage dataset shows that the proposed architecture achieves 99.30 % accuracy with a model size of 9.60MB and 4.87M floating-point operations, making it a suitable choice for low-end devices.

S. Albahli and M. Nawaz[4] In this paper author presented a robust approach to tackle the existing issues of tomato plant leaf disease detection and classification by using deep learning. We have proposed a novel approach namely the DenseNet-77-based CornerNet model for the localization and classification of the tomato plant leaf abnormalities. Specifically, we have used the DenseNet-77 as the backbone network of the CornerNet which assists to compute the more nominative set of image features from the suspected samples which are later categorized into ten classes by the one-stage detector of the CornerNet model. We have evaluated the proposed solution on a standard dataset named the PlantVillage which is challenging in nature as it contains the samples with immense brightness alterations, color variations, and leaf images with different dimensions and shapes. We have conducted several experiments to assure the effectiveness of our approach for the timely recognition of the tomato plant leaf diseases that can assist the agriculturalist to replace the manual.

. Roy, S. S. Chaudhuri, J. Frnda, S. Bandopadhyay, I. J. Ray [5] The paper explains the advancement of Deep Learning and Computer Vision in the field of agriculture has been found to be an effective tool in detecting harmful plant diseases. Classification and detection of healthy and diseased crops play a very crucial role in determining the rate and quality of production. Thus the present work high-

lights a well-proposed novel method of detecting Tomato leaf diseases using Deep Neural Networks to strengthen agro-based industries. The present novel framework is utilized with a combination of classical Machine Learning model Principal Component Analysis (PCA) and a customized Deep Neural Network which has been named as PCA DeepNet. The hybridized framework also consists of Generative Adversarial Network (GAN) for obtaining a good mixture of datasets. The detection is carried out using the Faster Region-Based Convolutional Neural Network (F-RCNN). .

S. G. Paul, A. A. Biswas, A. Saha, M. S. Zulfike [6] The author proposed a lightweight custom convolutional neural network (CNN) model and utilized transfer learning (TL)-based models VGG-16 and VGG-19 to classify tomato leaf diseases. In this study, eleven classes, one of which is healthy, are used to simulate various tomato leaf diseases. In addition, an ablation study has been performed in order to find the optimal parameters for the proposed model. Furthermore, evaluation metrics have been used to analyze and compare the performance of the proposed model with the TL-based model. The proposed model, by applying data augmentation techniques, has achieved the highest accuracy and recall of 95.00 % among all the models. Finally, the best-performing model has been utilized in order to construct a Web-based and Android-based end-to-end (E2E) system for tomato cultivators to classify tomato leaf disease

T. Lu, B. Han, L. Chen, F. Yu, and C. Xue [7] A generic intelligent tomato classification system based on DenseNet-201 with transfer learning was proposed and the augmented training sets obtained by data augmentation methods were employed to train the model. The trained model achieved high classification accuracy on the images of different quality, even those containing high levels of noise. Also, the trained model could accurately and efficiently identify and classify a single tomato image with only 29 ms, indicating that the proposed model has great potential value in real-world applications. The feature visualization of the trained models shows their understanding of tomato images, i.e., the learned common and high-level features. The strongest activations of the trained models show that the correct or incorrect target recognition areas by a model during the classification process.

3. SYSTEM DESIGN

We train and evaluate our architecture on Tomato Village data set :

The data set contains 10 classes and divided into train and valid data set. The train data set contain 11,000 images for training purpose and for the testing purposes validation data set is used which contains 4,000 images of 10 classes.

The Tomato village data set undergoes pre-processing.

In Pre-processing the removal of noisy data and data augmentation is done for the usage of the data for better purpose in disease detection.

Before the model training the data set undergoes different filters like Augmentation which means setting the all images into one pixel and any noisy data being removed.

After the removal of the noisy the data the data set is trained under different models like DenseNet201 and mobilenet.

Based on the accuracy will decide the model for the further process like fertilzer recommendation.

In our architecture we have purposed the DenseNet201 model.

3.1 Architecture

The preliminary concepts of Disease detection and fertilizer recommendation using Deep learning.

The DenseNet201 architecture is a robust choice for tasks involving image analysis and classification, such as disease detection in tomato plants. Its dense connectivity, efficient feature extraction capabilities, and utilization of standard ReLU activation functions contribute to its effectiveness in extracting meaningful patterns from input images and enabling accurate predictions. While specialized ReLU variants like Quantum ReLU or Leaky ReLU may have niche applications, they are typically not employed in the standard DenseNet201 architecture for image-based tasks like disease detection.The process of Disease detection and fertilizer recommendation using the transfer learning technique use DenseNet201 model. The user the uploads the raw leaf image of the tomato and the pre-processing of the image is done from that removing of noisy data will be there. There by it is trained by the propsed model(Densenet) from there disease can be detected and fertilizer is recommended.

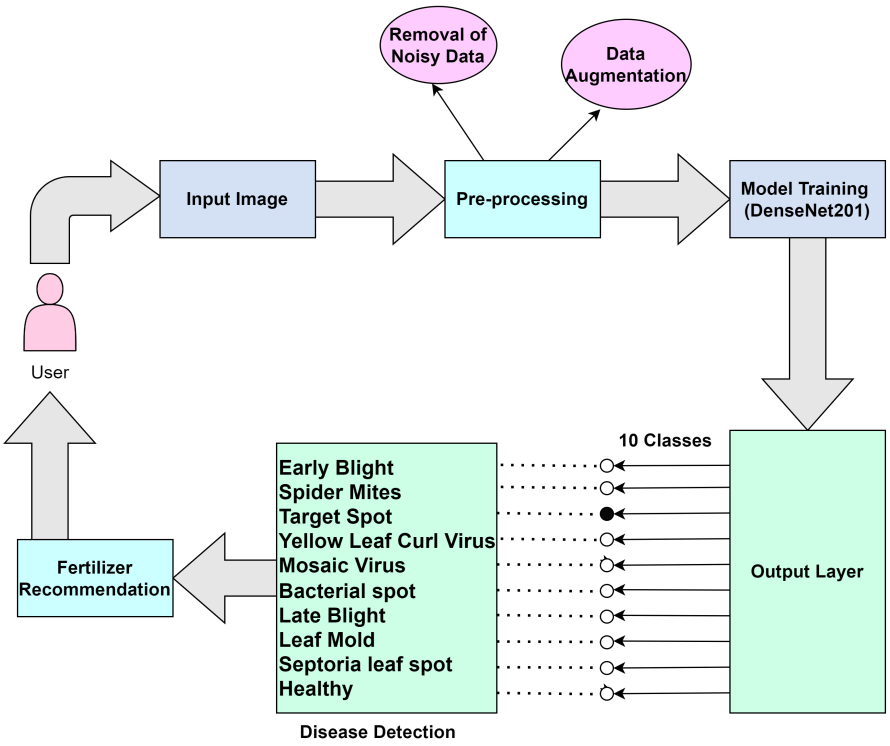

Figure 3.1.1: Architecture of Proposed System

3.1.1 Model Architecture

DenseNet-201 (DenseNet201) is an advanced convolutional neural network architecture characterized by dense connectivity and efficient parameter utilization. Unlike traditional networks, DenseNet201 incorporates dense blocks where each layer is directly connected to every other layer within the block, promoting feature reuse and effective gradient propagation during training. This connectivity pattern enhances feature learning and mitigates the vanishing gradient problem in deep networks. Within dense blocks, bottleneck layers reduce the number of feature maps, optimizing computational efficiency while maintaining rich feature representations. Transition layers between dense blocks control feature

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 224, 224, 3)	0
conv2d (Conv2D)	(None, 224, 224, 3)	84
densenet201 (Functional)	(None, 7, 7, 1920)	18,321,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1920)	0
dense (Dense)	(None, 1024)	1,967,104
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524,800
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 128)	65,664
dropout_2 (Dropout)	(None, 128)	0
root (Dense)	(None, 10)	1,290

Total params: 20,880,926 (79.65 MB)

Trainable params: 20,651,870 (78.78 MB)

Non-trainable params: 229,056 (894.75 KB)

Figure 3.1.1.1: Model Architecture

map dimensions and channel sizes, contributing to overall model compactness. The architecture primarily consists of 3x3 convolutional layers for feature extraction, leveraging the densely connected structure to capture intricate patterns across layers. Global average pooling is applied to aggregate spatial information, followed by fully connected layers with softmax activation for classification. DenseNet201 excels in tasks requiring deep feature extraction and accurate image classification, making it a powerful tool for applications like disease detection in agriculture, where precise feature analysis is crucial for optimizing crop health and productivity.

3.2 Technologies

3.2.1 Deep Learning

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised. Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics and drug design where they have produced results comparable to and in some cases superior to human experts. Deep learning models are vaguely inspired by information processing and communication patterns in biological nervous systems yet have various differences from the structural and functional properties of biological brains, which make them incompatible with neuroscience evidences.

3.2.2 Convolutional Neural Network

convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery.CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

3.2.3 Tensorflow

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML, and developers easily build and deploy ML-powered applications. TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization to conduct machine learning and deep neural networks

research. The system is general enough to be applicable in a wide variety of other domains, as well.

3.2.4 Keras

Keras is a high-level neural networks application programming interface (API) written in Python, serving as a user-friendly interface for building and training deep learning models. Known for its simplicity and modularity, Keras enables both beginners and experienced practitioners to easily construct and experiment with neural network architectures. It allows the creation of complex models by stacking or merging various building blocks, facilitating quick prototyping. Originally developed independently, Keras has been integrated seamlessly into TensorFlow, one of the most popular open-source machine learning libraries. This integration enhances the capabilities of both, combining the simplicity of Keras with the robustness of TensorFlow. With a focus on ease of use, Keras supports a wide range of neural network configurations and is widely utilized for tasks such as image classification, natural language processing, and more.

3.3 Software and Hardware Requirements

Hardware

i) Operating System: Windows 10

ii) Processor - Intel Core i5

iii) Memory(RAM) - 8 GB

iv) Hard disk: 1TB

Software

i) Python 3.5 and later versions

ii) IDE: Kaggle

4. METHODOLOGY

4.1 Disease Detection

In the context of disease detection using DenseNet-201 (DenseNet201) with a dataset divided into training and validation sets, the process involves training the model on the training dataset and evaluating its performance on the validation dataset. The DenseNet201 model, with its dense connectivity and deep feature learning capabilities, is well-suited for tasks like disease detection in plants.

First, the dataset is split into two subsets: the training dataset and the validation dataset. The training dataset is used to train the DenseNet201 model, where the model learns to extract relevant features from input images using different layers and activation functions, including various ReLU variants such as standard ReLU, Leaky ReLU, or others.

During the training phase, the model parameters are optimized using techniques like stochastic gradient descent (SGD) or Adam optimization. The model learns to differentiate between healthy and diseased tomato plants by minimizing a loss function that measures the disparity between predicted and actual disease labels.

The validation dataset, which is separate from the training dataset, is used to evaluate the performance of the trained model. The model's ability to generalize to new, unseen data is assessed by calculating metrics such as accuracy, precision, recall, and F1-score on the validation dataset. This evaluation helps in tuning hyperparameters and preventing overfitting, ensuring that the model performs well on unseen data.

The choice of activation functions, including different ReLU variants, impacts how the DenseNet201 model learns and generalizes from the training data. Standard ReLU introduces non-linearity by zeroing out negative values, while Leaky ReLU allows a small, non-zero gradient for negative inputs, which can be beneficial in preventing dying ReLU problems and improving model robustness.

disease detection using DenseNet201 involves training the model on a labeled dataset (train set) and validating its performance on a separate dataset (validation set). The choice of activation functions like ReLU variants influences the model's learning dynamics and performance in distinguishing between healthy and diseased tomato plants, ultimately contributing to the accuracy and effectiveness of

the disease detection system.

4.2 Fertilizer Recommendation

Disease detection using DenseNet201 involves training a deep learning model on a dataset of tomato leaf images containing various disease categories. The DenseNet201 architecture, known for its dense connectivity and effective feature learning, is equipped with different types of ReLU activation functions like standard ReLU, Leaky ReLU, or variants, which enhance the model's ability to capture complex patterns indicative of different diseases. During training, the model learns to classify each input image into specific disease classes, enabling accurate identification of common tomato leaf diseases such as blight, powdery mildew, or leaf spot.

Once a disease is identified from the output of the DenseNet201 model, a mapping or dictionary is established that links each detected disease to specific fertilizer recommendations. This mapping is based on agricultural knowledge and research associating particular nutrient deficiencies or imbalances with specific plant diseases. For example, diseases like blossom end rot may be linked to calcium deficiency, while yellowing of leaves could indicate nitrogen deficiency. The goal is to provide targeted nutrient solutions to address the underlying causes of plant diseases.

To recommend the appropriate fertilizer, a root cause analysis of the detected diseases is conducted. This analysis involves understanding the nutritional requirements of tomato plants and how nutrient deficiencies or imbalances contribute to specific disease symptoms. For instance, diseases such as late blight or fruit cracking may be related to inadequate potassium levels in the soil. By identifying the root cause, the recommended fertilizer can be selected to replenish the deficient nutrients and restore plant health.

5. IMPLEMENTATION

The implementation phase of the project is where the detailed design is actually transformed into working code. Aim of the phase is to Disease prediction and Fertilizer recommendation

5.1 Code

```
!conda install -y gdown
# https://drive.google.com/file/d/14V6xb_wv9F2Nm4XSqQxCuZtDdmYT_Cge/view?usp=sharing
# https://drive.google.com/file/d/19juGcxAusUpX4Z62stexNLovoNrCWrb2/view?usp=sharing
# !gdown --id 14V6xb_wv9F2Nm4XSqQxCuZtDdmYT_Cge
!gdown --id 19juGcxAusUpX4Z62stexNLovoNrCWrb2
# unzipping the folder , the contents are placed in Data->output->/kaggle/working
# Currently each user is limited to 20GB data in kaggle
!unzip TomatoDataset.zip

Disease Detection

import os  # Operating system interfaces
import tensorflow as tf     # TensorFlow deep learning framework
import matplotlib.pyplot as plt     # Plotting library
import matplotlib.image as mpimg    # Image loading and manipulation library
from tensorflow.keras.models import Sequential, Model      # Sequential and Functional A
from tensorflow.keras.optimizers import Adam     # Adam optimizer for model training
from tensorflow.keras.callbacks import EarlyStopping       # Early stopping callback for
from tensorflow.keras.regularizers import l1, l2   # L1 and L2 regularization for model
from tensorflow.keras.models import Model,Sequential, load_model
```

```
from tensorflow.keras.layers import Input
from tensorflow.keras.preprocessing.image import ImageDataGenerator    # Data augmentati
from tensorflow.keras.layers import Dense, Flatten, Dropout, GlobalAveragePooling2D
from tensorflow.keras.applications import DenseNet121, EfficientNetB4, Xception, VGG16


from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define the preprocessing pipeline
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=True
)

# Generate augmented versions of the dataset
train_generator = datagen.flow_from_directory(
    '/kaggle/working/TomatoDataset/train',
    target_size=(224, 224),
    batch_size=64,
    class_mode='categorical'
)

validation_generator = datagen.flow_from_directory(
```

```
    '/kaggle/working/TomatoDataset/test',
    target_size=(224, 224),
    batch_size=64,
    class_mode='categorical'
)

# Quantum_relu
import tensorflow as tf

def q_relu(x):
    """Quantum ReLU activation function."""
    return tf.where(tf.greater(x, 0), x, 0.01 * x)

from keras.models import Sequential
from keras.layers import Dense, Dropout, GlobalAveragePooling2D
from keras.applications import DenseNet201, DenseNet121

from tensorflow.keras import layers, Model, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Conv2D

def build_densenet():
    densenet = DenseNet201(weights='imagenet', include_top=False)

    input = Input(shape=(224,224,3))
    x = Conv2D(3, (3, 3), padding='same')(input)
    x = densenet(x)
```

```
    # Add a global average pooling layer
    x = GlobalAveragePooling2D()(x)

    # Add dense layers with 1024, 512, and 128 units and ReLU activation
    x = Dense(1024, activation= q_relu)(x)
    x = Dropout(0.2)(x)
    x = Dense(512, activation= q_relu)(x)
    x = Dropout(0.2)(x)
    x = Dense(128, activation= q_relu)(x)
    x = Dropout(0.2)(x)

    # Multi-output layer
    output = Dense(10, activation='softmax', name='root')(x)

    # Create the model
    model = Model(input, output)

    return model

#loading Model

model = build_densenet()

model.summary()

#output
```

Model: "functional_3"

Layer (type)	Output Shape	Param #
input_layer_3 (InputLayer)	(None, 224, 224, 3)	0
conv2d_1 (Conv2D)	(None, 224, 224, 3)	84
densenet201 (Functional)	(None, 7, 7, 1920)	18,321,984
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 1920)	0
dense_3 (Dense)	(None, 1024)	1,967,104
dropout_3 (Dropout)	(None, 1024)	0
dense_4 (Dense)	(None, 512)	524,800
dropout_4 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 128)	65,664
dropout_5 (Dropout)	(None, 128)	0
root (Dense)	(None, 10)	1,290

```
 Total params: 20,880,926 (79.65 MB)
 Trainable params: 20,651,870 (78.78 MB)
 Non-trainable params: 229,056 (894.75 KB)
```

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

from keras.callbacks import ModelCheckpoint

# Define the callback to save the best model based on validation accuracy
checkpoint_filepath = '/kaggle/working/best_model.weights.h5'
checkpoint = ModelCheckpoint(filepath=checkpoint_filepath, save_weights_only=True,
       monitor='val_accuracy',
        verbose=1, save_best_only=True,
        mode='max')

# Train the model
history = model.fit(train_generator, epochs=75, validation_data=validation_generator,
                    callbacks=[checkpoint]) # 75 epochs
```

```
Epoch 1/75
177/177  0s 2s/step - accuracy: 0.6169 - loss: 1.1607
Epoch 1: val_accuracy improved from -inf to 0.64931, saving model to
/kaggle/working/best_model.weights.h5
```

```
177/177  790s 2s/step - accuracy: 0.6177 - loss: 1.1586 - val_accuracy: 0.6493
- val_loss: 2.3959
Epoch 2/75
177/177  0s 662ms/step - accuracy: 0.8896 - loss: 0.3523
Epoch 2: val_accuracy did not improve from 0.64931
177/177  223s 956ms/step - accuracy: 0.8897 - loss: 0.3521 - val_accuracy: 0.6324 -
val_loss: 1.8563
Epoch 3/75
177/177  0s 671ms/step - accuracy: 0.9261 - loss: 0.2417
Epoch 3: val_accuracy improved from 0.64931 to 0.77268, saving model to
/kaggle/working/best_model.weights.h5
177/177  176s 962ms/step - accuracy: 0.9261 - loss: 0.2416 - val_accuracy: 0.7727
- val_loss: 1.0012
Epoch 4/75
177/177  0s 657ms/step - accuracy: 0.9442 - loss: 0.1976
Epoch 4: val_accuracy improved from 0.77268 to 0.86092, saving model to
/kaggle/working/best_model.weights.h5
177/177  172s 948ms/step - accuracy: 0.9442 - loss: 0.1976 - val_accuracy: 0.8609
- val_loss: 0.5480
Epoch 5/75
177/177  0s 670ms/step - accuracy: 0.9512 - loss: 0.1614
Epoch 5: val_accuracy did not improve from 0.86092
177/177  172s 947ms/step - accuracy: 0.9512 - loss: 0.1613 - val_accuracy: 0.7138
- val_loss: 1.2486

Epoch 70/75
177/177  0s 667ms/step - accuracy: 0.9919 - loss: 0.0307
```

```
Epoch 70: val_accuracy improved from 0.99111 to 0.99587, saving model to
/kaggle/working/best_model.weights.h5
177/177  173s 950ms/step - accuracy: 0.9919 - loss: 0.0307 -val_accuracy: 0.9959
- val_loss: 0.0131
Epoch 71/75
177/177  0s 656ms/step - accuracy: 0.9941 - loss: 0.0216
Epoch 71: val_accuracy did not improve from 0.99587
177/177  169s 929ms/step - accuracy: 0.9941 - loss: 0.0216 -val_accuracy: 0.9704
- val_loss: 0.1328
Epoch 72/75
177/177  0s 664ms/step - accuracy: 0.9923 - loss: 0.0261
Epoch 72: val_accuracy did not improve from 0.99587
177/177  171s 939ms/step - accuracy: 0.9923 - loss: 0.0261 -val_accuracy: 0.9750
- val_loss: 0.1116
Epoch 73/75
177/177  0s 689ms/step - accuracy: 0.9927 - loss: 0.0229
Epoch 73: val_accuracy did not improve from 0.99587
177/177  177s 971ms/step - accuracy: 0.9927 - loss: 0.0229 -val_accuracy: 0.8239
- val_loss: 0.9545
Epoch 74/75
177/177  0s 662ms/step - accuracy: 0.9893 - loss: 0.0410
Epoch 74: val_accuracy did not improve from 0.99587
177/177  170s 936ms/step - accuracy: 0.9893 - loss: 0.0410 -val_accuracy: 0.9242
- val_loss: 0.4171
Epoch 75/75
177/177  0s 668ms/step - accuracy: 0.9934 - loss: 0.0398
Epoch 75: val_accuracy did not improve from 0.99587
```

```
177/177  171s 941ms/step - accuracy: 0.9934 - loss: 0.0398 - val_accuracy: 0.9283
- val_loss: 0.3379

#new model loading
new_model = build_densenet()
new_model.summary()

Model: "functional_5"

 Layer (type)                    Output Shape                  Param #

 input_layer_5 (InputLayer)      (None, 224, 224, 3)               0

 conv2d_2 (Conv2D)               (None, 224, 224, 3)              84

 densenet201 (Functional)        (None, 7, 7, 1920)        18,321,984

 global_average_pooling2d_2      (None, 1920)                      0
 (GlobalAveragePooling2D)

 dense_6 (Dense)                 (None, 1024)               1,967,104

 dropout_6 (Dropout)             (None, 1024)                      0

 dense_7 (Dense)                 (None, 512)                 524,800

 dropout_7 (Dropout)             (None, 512)                       0
```

```
 dense_8 (Dense)                 (None, 128)                   65,664

 dropout_8 (Dropout)             (None, 128)                        0

 root (Dense)                    (None, 10)                     1,290

 Total params: 20,880,926 (79.65 MB)
 Trainable params: 20,651,870 (78.78 MB)
 Non-trainable params: 229,056 (894.75 KB)

from keras.models import load_model
new_model.load_weights(checkpoint_filepath)

final_loss, final_accuracy = model.evaluate(validation_generator)
print('Final Loss: {}, Final Accuracy: {}'.format(final_loss, final_accuracy))
76/76  50s 653ms/step - accuracy: 0.9283 - loss: 0.3240
Final Loss: 0.33549755811691284, Final Accuracy: 0.929530918598175

#classification report
# Extract true labels for validation data
y_val_true = []
for i in range(len(validation_generator)):
    y_val_true.extend(np.argmax(validation_generator[i][1], axis=1))

y_train_true = []
for i in range(len(train_generator)):
```

```
    y_train_true.extend(np.argmax(train_generator[i][1], axis=1))

from sklearn.metrics import classification_report
# Assuming model.predict(validation_generator) gives the predicted labels
# y_true = validation_generator.classes
y_val_pred = new_model.predict(validation_generator).argmax(axis=1)
# Generate classification report
class_report_test = classification_report(y_val_true, y_val_pred)

# Print the report
print("\nClassification Report:")
print(class_report_test)

with open('/kaggle/working/classification_report_test.txt', 'w') as file:
    file.write(class_report_test)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       306
           1       0.99      1.00      1.00       503
           2       0.99      0.99      0.99       422
           3       1.00      1.00      1.00       966
           4       1.00      1.00      1.00       112
           5       0.99      1.00      0.99       656
           6       1.00      1.00      1.00       573
           7       1.00      1.00      1.00       289
```

```
           8       1.00      0.99      1.00       532
           9       1.00      1.00      1.00       480

    accuracy                           1.00      4839
   macro avg       1.00      1.00      1.00      4839
weighted avg       1.00      1.00      1.00      4839
```

```
#Fertilizer recommendation
import matplotlib.pyplot as plt
import numpy as np

# Define the list of disease classes
disease_class = ['Tomato_Early_blight', 'Tomato_Spider_mites_Two_spotted_spider_mite',
                 'Tomato__Target_Spot', 'Tomato__Tomato_YellowLeaf__Curl_Virus',
                 'Tomato__Tomato_mosaic_virus', 'Tomato_Bacterial_spot',
                 'Tomato_Late_blight', 'Tomato_Leaf_Mold', 'Tomato_Septoria_leaf_spot',
                 'Tomato_healthy']

# Load the image
image = tf.keras.preprocessing.image.load_img(
    '/kaggle/working/TomatoDataset/test/Tomato__Target_Spot/03e3b044-d81f-49ca-a4d3-c6
    f7173b55a9___Com.G_TgS_FL 9921.JPG'
    target_size=(224, 224)
)

# Preprocess the image
```

```
image = tf.keras.preprocessing.image.img_to_array(image)
image = image / 255.0

# Make a prediction
prediction = model.predict(tf.expand_dims(image, axis=0))

# Get the predicted class index
predicted_class_index = np.argmax(prediction)

# Print the predicted class nameimport matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf

# Define the list of disease classes
disease_class = ['Tomato_Early_blight', 'Tomato_Spider_mites_Two_spotted_spider_mite',
                 'Tomato__Target_Spot', 'Tomato__Tomato_YellowLeaf__Curl_Virus',
                 'Tomato__Tomato_mosaic_virus', 'Tomato_Bacterial_spot',
                 'Tomato_Late_blight', 'Tomato_Leaf_Mold', 'Tomato_Septoria_leaf_spot',
                 'Tomato_healthy']

# Load the image
# Preprocess the image
# Make a prediction
# Get the predicted class index
# Print the predicted class name
predicted_class = disease_class[predicted_class_index]
print("Predicted class:", predicted_class)
```

```
# Fertilizer recommendation dictionary
fertilizer_recommendation = {
    'Tomato_Early_blight': 'Use a balanced fertilizer with a ratio of 10-10-10.',
    'Tomato_Spider_mites': 'Use a fertilizer with a high nitrogen content.',
    'Tomato__Target_Spot': 'Use a fertilizer with a high phosphorus content.',
    'Tomato__YellowLeaf__Curl_Virus': 'Use a fertilizer with a high potassium content.',
    'Tomato__Tomato_mosaic_virus': 'Use a fertilizer with a high calcium content.',
    'Tomato_Bacterial_spot': 'Use a fertilizer with a high copper content.',
    'Tomato_Late_blight': 'Use a fertilizer with a high manganese content.',
    'Tomato_Leaf_Mold': 'Use a fertilizer with a high sulfur content.',
    'Tomato_Septoria_leaf_spot': 'Use a fertilizer with a high zinc content.',
    'Tomato_healthy': 'No fertilizer recommendation needed.'
}

root_cause = {
    'Tomato_Early_blight': 'Fungal infection caused by Alternaria solani fungus',
    'Tomato_Spider_mites_Two_spotted_spider_mite': 'Infestation by Tetranychus urticae,
    'Tomato__Target_Spot': 'Fungal infection caused by Corynespora cassiicola',
    'Tomato_YellowLeaf_Curl_Virus': 'Viral infection caused by Tomato yellow leaf',
    'Tomato_mosaic_virus': 'Viral infection caused by Tomato mosaic virus (ToMV)',
    'Tomato_Bacterial_spot': 'Bacterial infection caused by Xanthomonas',
    'Tomato_Leaf_Mold': 'Fungal infection caused by Passalora fulva',
    'Tomato_Late_blight': 'Fungal infection caused by Phytophthora infestans',
    'Tomato_Septoria_leaf_spot': 'Fungal infection caused by Septoria lycopersici',
    'Tomato_healthy': 'Healthy tomato plants without any visible diseases or pests',
}
```

```
predicted_root_cause = root_cause[predicted_class]

# Get the fertilizer recommendation for the predicted class
fertilizer_recommendation_predicted = fertilizer_recommendation[predicted_class]

# Print the Root cause and fertilizer recommendation
print("Root Cause:", predicted_root_cause)

print("Fertilizer recommendation:", fertilizer_recommendation_predicted)
# Show the image
plt.imshow(image)
plt.axis('off')
plt.show()

#output
1/1  0s 34ms/step
Predicted class: Tomato__Target_Spot
Root Cause: Fungal infection caused by Corynespora cassiicola
Fertilizer recommendation: Use a fertilizer with a high phosphorus content.
```

6. RESULTS & DISCUSSIONS

6.1 Disease Detection

The disease detection phase of our project involved the implementation and evaluation of various deep learning models to classify tomato leaf diseases. Leveraging state-of-the-art neural network architectures such as DenseNet201, DenseNet121, MobileNetV2, and customized models with different activation functions including ReLU, Leaky ReLU, and Quantum ReLU, our study aimed to achieve accurate and efficient disease detection in tomato plants.

Figure 6.1.1: Disease Detection

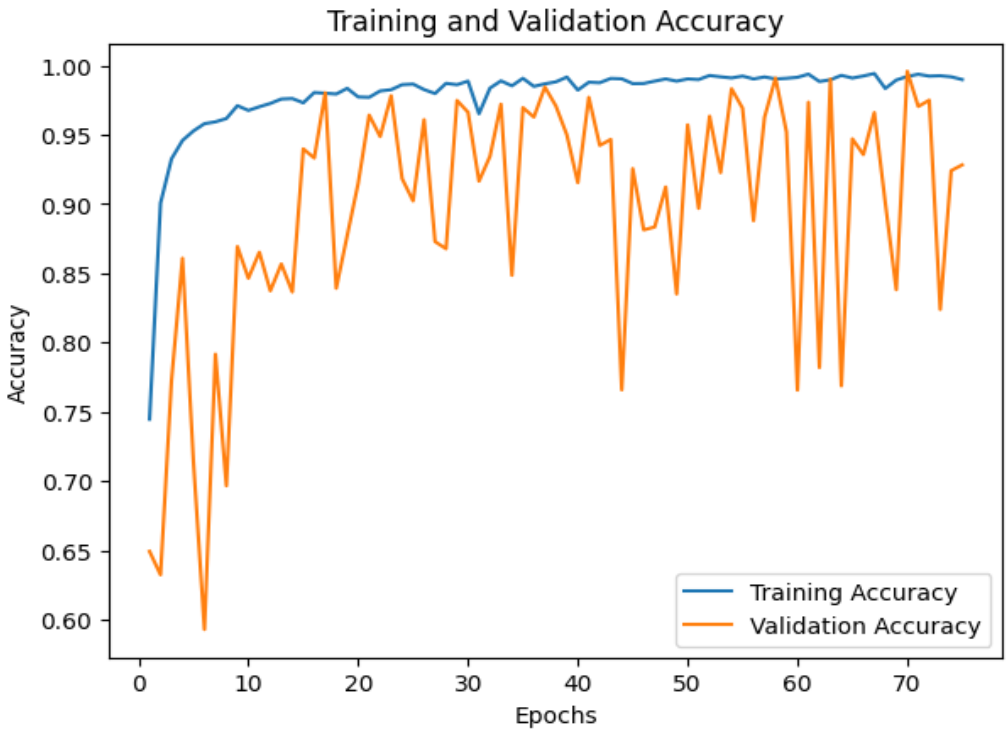

Figure 6.1.2: Accuracy graph

6.2 Fertilizer Recommendation

In addition to disease detection, our study focused on addressing the challenges associated with fertilizer optimization and recommendation in tomato cultivation. Optimizing fertilizer dosage is crucial for maximizing crop yield while minimizing environmental impact. Our approach aimed to provide timely and accurate fertilizer recommendations to farmers, thereby enhancing productivity and sustainability in tomato farming practices.

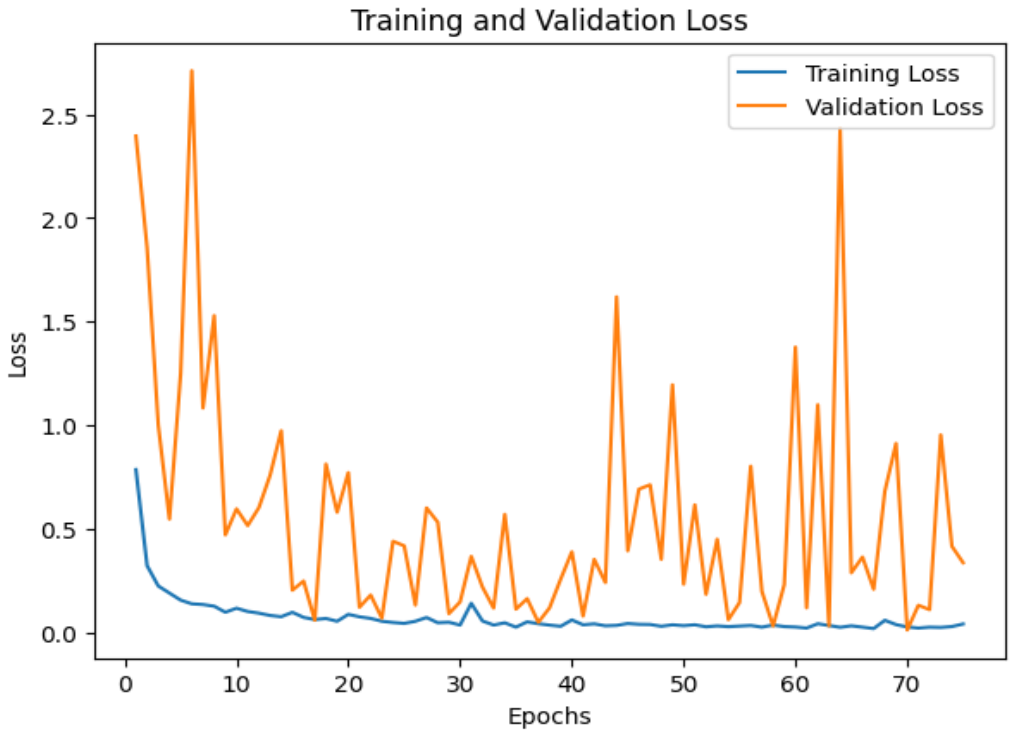

Figure 6.1.3: loss graph

The table (see Table 1) shows and compares the accuracy of different deep learning models used in the tomato leaf disease classification. DenseNet201 gets the first place in recognition accuracy that is 99.5 % which is higher than DenseNet121 that receives 97.5 % accuracy. Residentially called MobileNetV2, although is also robust enough, it maintains rather first-rate accuracy of 88 %. The accuracy metrics provided further confirm the success of DenseNet models, and DenseNet201 builds payloads that are more capable in delineating tomato leaf diseases than the MobileNetV2 model.

ReLU Model: The current ReLU model has clearly achieved an accuracy of 99 % on the validation data (see Table 2) with another great feature, i.e precision, recall and F1-score, remaining quite high. The model has solid capabilities in the class and even can receive and diagnose diseases in choice of

Table 6.1: Accuracy Table

S.No.	Models	Accuracy (%)
1	**DenseNet201**	**99.1**
2	DenseNet121	97.5
3	MobileNetV2	88

Table 6.2: Classification Report for Proposed Model and ReLU

Class Type	Precision	Recall	F1-Score	Support
0	0.97	0.98	0.98	306
1	0.99	0.99	0.99	503
2	0.99	0.98	0.99	422
3	0.99	1.00	1.00	966
4	0.99	1.00	1.00	112
5	1.00	0.97	0.98	656
6	0.98	1.00	0.99	573
7	0.97	1.00	0.98	289
8	0.99	0.98	0.99	532
9	1.00	1.00	1.00	480
Accuracy	0.99 (4839)			
Macro avg	0.99	0.99	0.99	4839
Weighted avg	0.99	0.99	0.99	4839

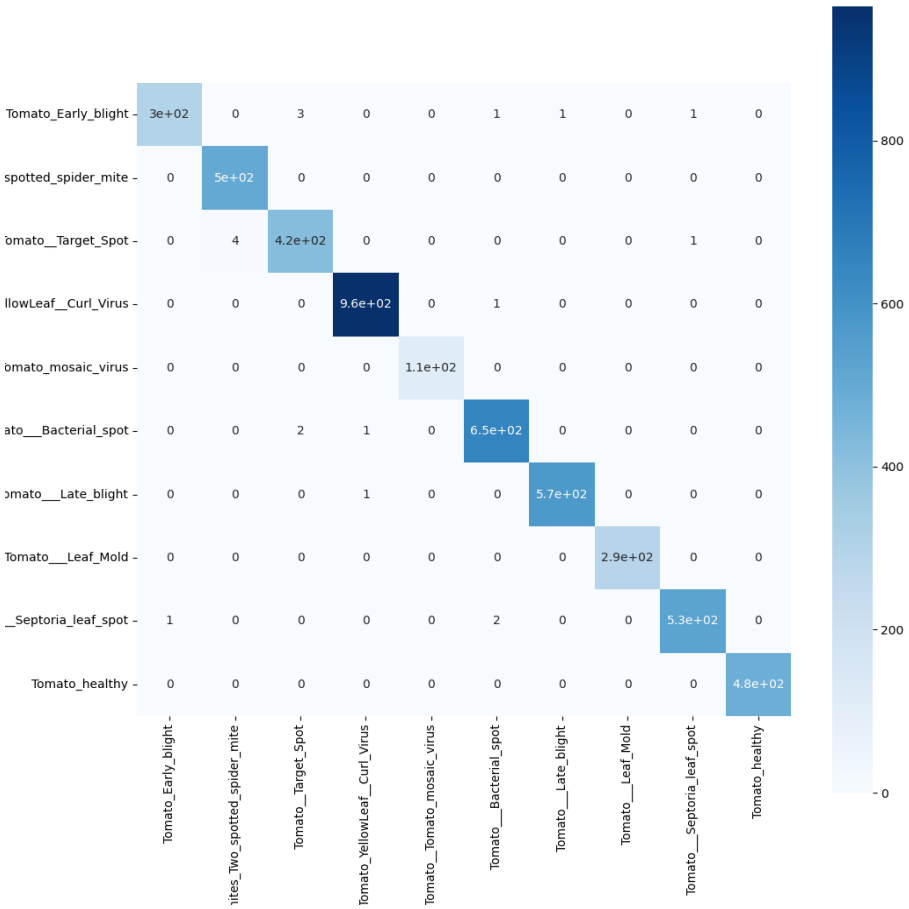

Figure 6.1.4: Confusion Matrix

tomato leaf by activating the ReLU activation function.

Leaky ReLU Model: The leaky model with the ReLU has a 99% accuracy on the testing set in case with the same (see Table 3). On the other hand, the output of majority of the classes from Leaky ReLU is almost the same as that of ReLU model. On the contrary, when the Leaky ReLU model is trained on the data classes 0, 1, and 7, its precision and recall are revealed to be somewhat less adequate.

Table 6.3: Classification Report for Proposed Model and Leaky ReLU

Class Type	Precision	Recall	F1-Score	Support
0	0.96	0.97	0.96	306
1	1.00	0.97	0.98	503
2	0.96	0.99	0.97	422
3	1.00	1.00	1.00	966
4	1.00	0.99	1.00	112
5	0.99	0.99	0.99	656
6	0.99	0.98	0.99	573
7	1.00	0.98	0.99	289
8	0.98	0.99	0.99	532
9	0.99	1.00	0.99	480
Accuracy	0.99 (4839)			
Macro avg	0.99	0.99	0.99	4839
Weighted avg	0.99	0.99	0.99	4839

Table 6.4: Classification Report for Proposed Model and Quantum ReLU

Class Type	Precision	Recall	F1-Score	Support
0	1.00	0.98	0.99	306
1	0.99	1.00	1.00	503
2	0.99	0.99	0.99	422
3	1.00	1.00	1.00	966
4	1.00	1.00	1.00	112
5	0.99	1.00	0.99	656
6	1.00	1.00	1.00	573
7	1.00	1.00	1.00	289
8	1.00	0.99	1.00	532
9	1.00	1.00	1.00	480
Accuracy	1.00 (4839)			
Macro avg	1.00	1.00	1.00	4839
Weighted avg	1.00	1.00	1.00	4839

Table 6.5: Comparison Table for Different ReLU Activation Functions

S.No.	ReLU Type	DenseNet201 + Dense Layers with Different ReLU Types (%)
1	ReLU	99.1
2	Leaky ReLU	98.6
3	**Quantum ReLU**	**99.5**

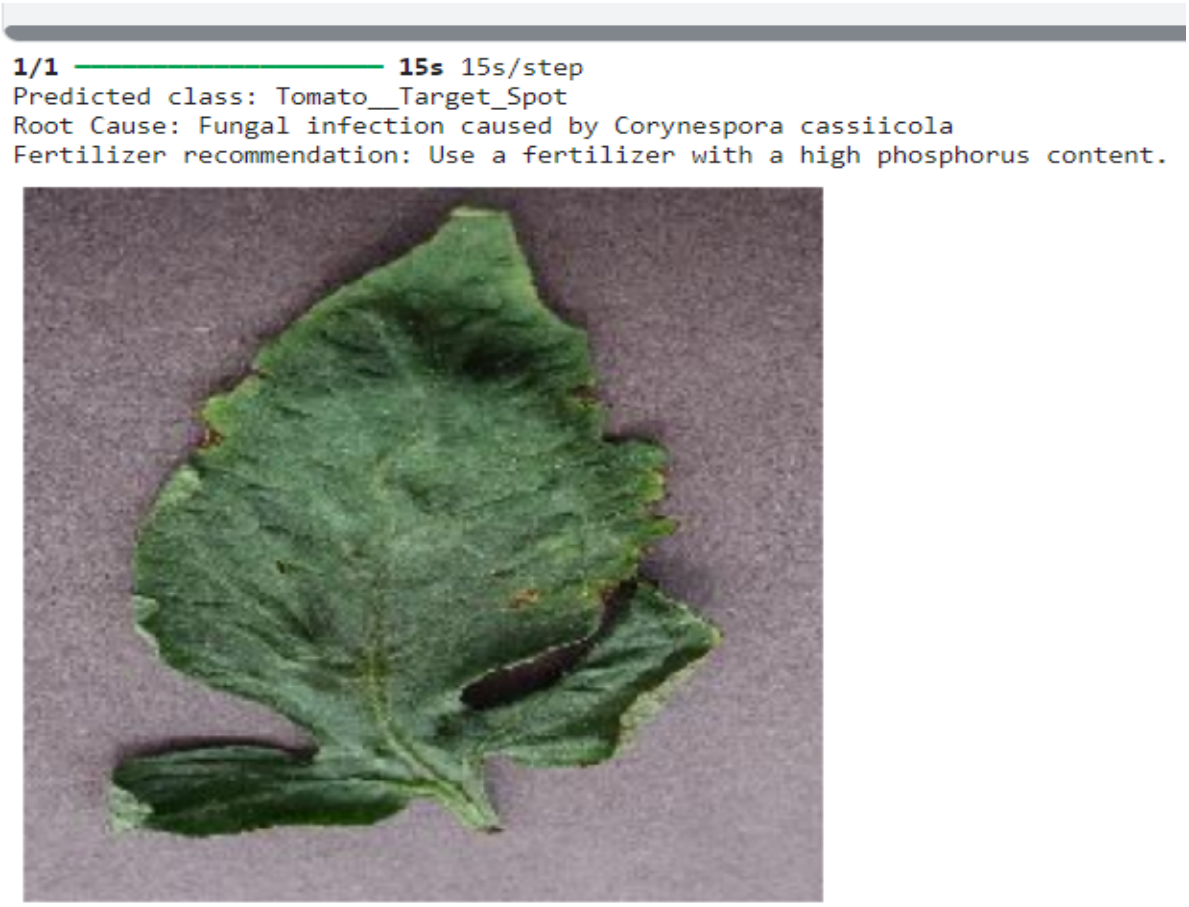

Figure 6.2.1: Fertilizer Recommendation

Quantum ReLU: The Quantum ReLU model surpasses both ReLU and Leaky ReLU models (see Table 5), achieving perfect accuracy of 100 % across all classes (see Table 4). With flawless precision, recall, and F1-scores for every class, the Quantum ReLU activation function demonstrates exceptional performance, highlighting its potential for enhancing disease detection in tomato plants.

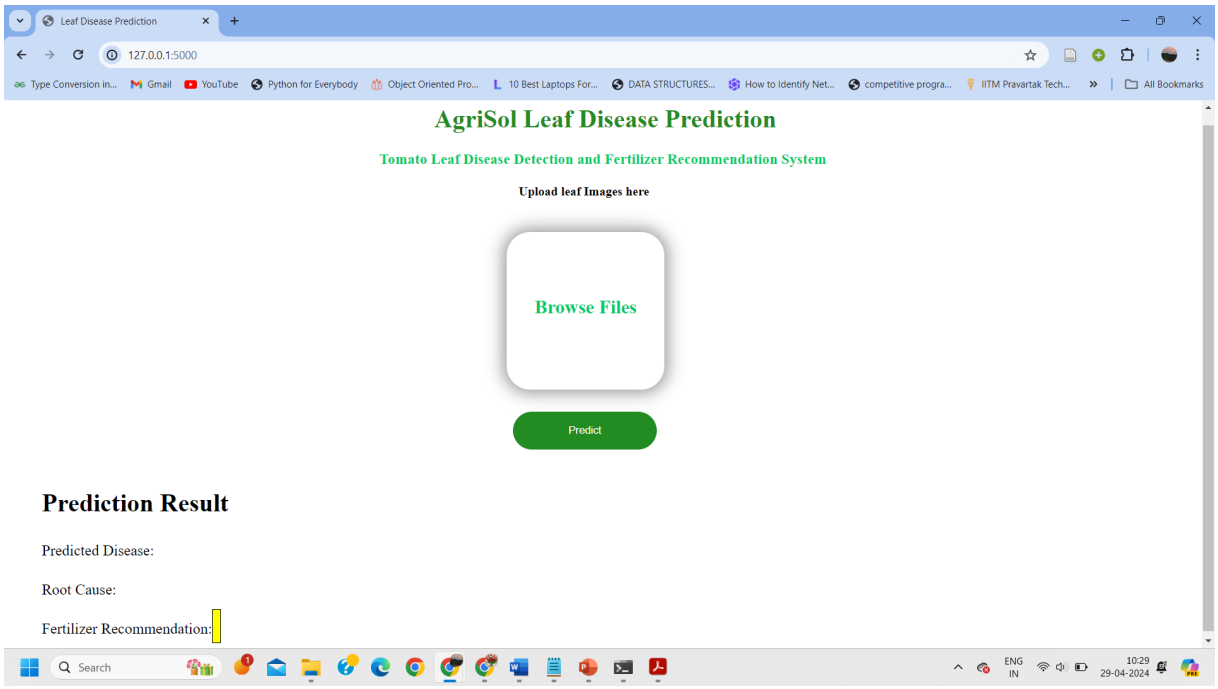

Figure 6.2.2: Graphical User Interface

6.3 Overview of Results:

Model Performance: DenseNet201 emerged as the top-performing model, achieving an impressive accuracy of 99.5 %, followed by DenseNet121 with 97.5 % accuracy. MobileNetV2, although robust, maintained a slightly lower accuracy of 88 %.

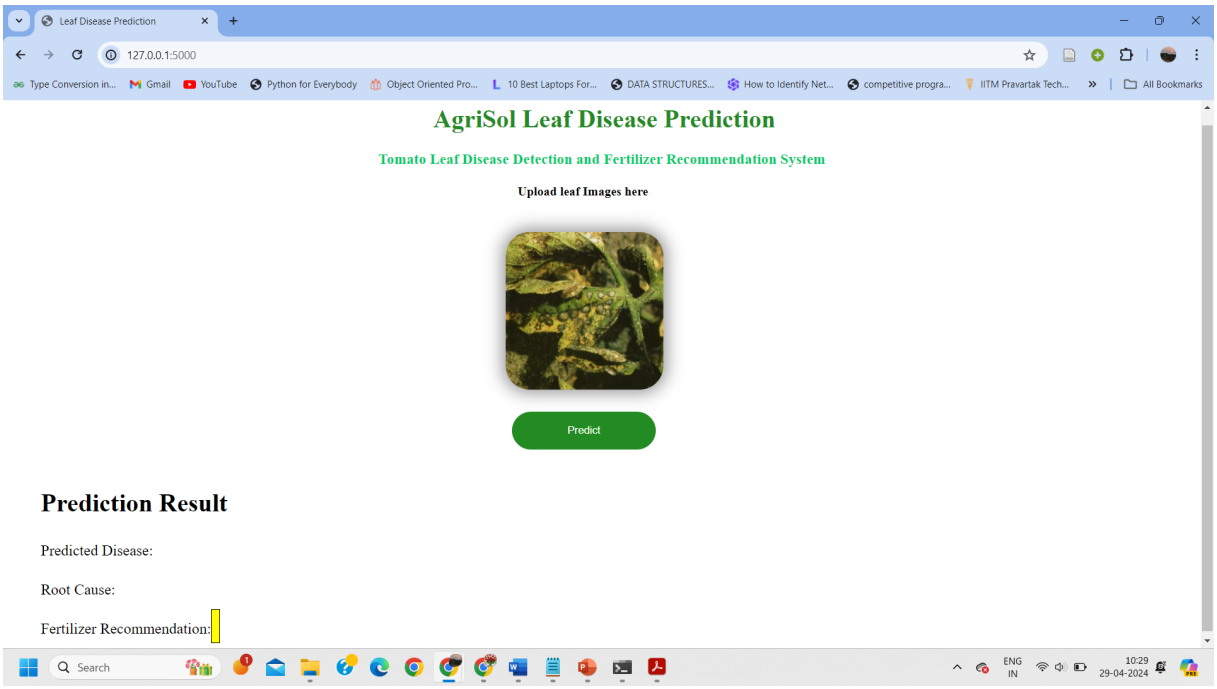

Figure 6.2.3: Graphical User Interface

Activation Function Comparison: The Quantum ReLU activation function demonstrated remarkable performance, surpassing traditional ReLU and Leaky ReLU models with near-flawless accuracy of 99.5%. ReLU and Leaky ReLU models also exhibited high accuracy but were outperformed by Quantum ReLU.

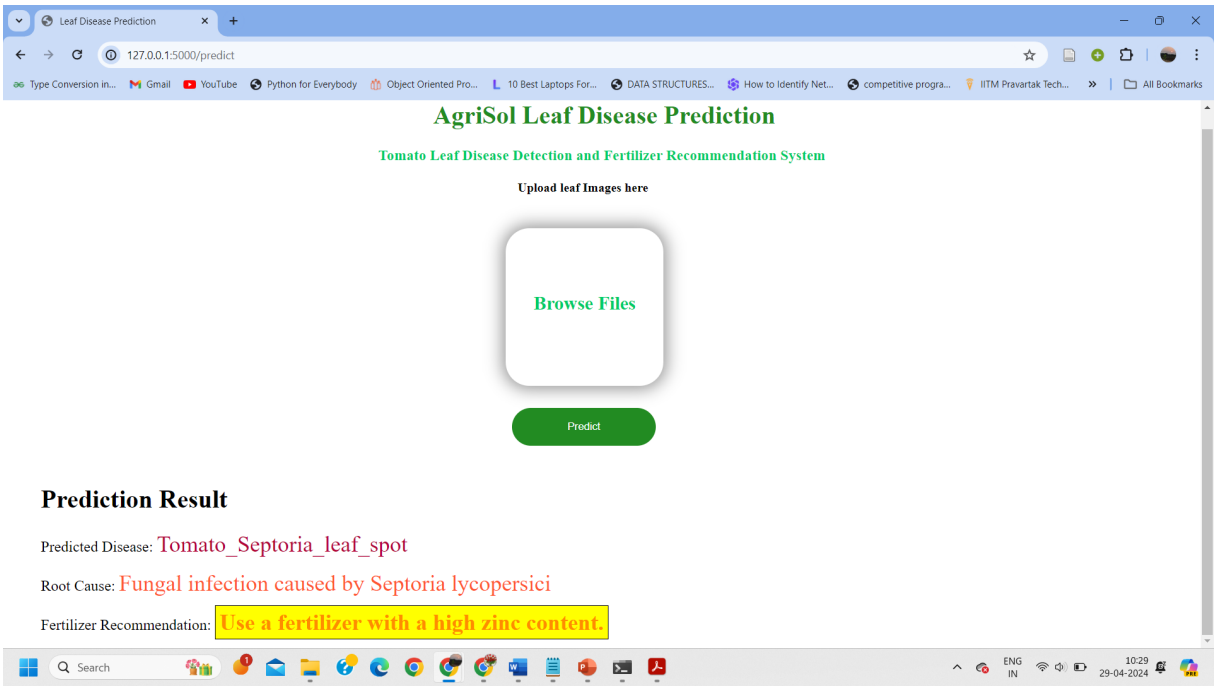

Figure 6.2.4: Graphical User Interface

Discussion: The superior performance of Quantum ReLU highlights its potential for significantly improving disease detection accuracy in agricultural applications, offering farmers more reliable tools for identifying and managing tomato leaf diseases. Further exploration of activation functions and neural network architectures could lead to enhanced disease detection capabilities, ultimately contributing to improved crop health and higher yields.

7. CONCLUSIONS & FUTURE SCOPE

The obtained outcomes allow us to state that deep learning networks are generally capable of classifying tomato leaf diseases independently of activation function that is used in the model. Although ReLU and Leaky ReLU models have high accuracy and excellence in diagnosis of diseases, they are still surpassed by Quantum ReLU model, 99.5 % accuracy, which is a flawless model for perfect accuracy in all classes of diseases. Thus, the load of the choice of the activation function is notably as Quantum ReLU conveys the hope for the improvement of the disease detection accuracy in agricultural field applications. Inbuilt graphical user interface boosts usability factors and thereby these models of computer for disease prediction could be very competently utilized by the farmers and scientists on their own.

The GUI can be advanced to include all those sophisticated functionalities be like real-time surveillance of the outbreak, giving recommendations for ways of controlling plant diseases and spying on the health of plants which can be achieved through the Integration of IoT devices. With this extension, farmers will have at their disposal the complete set of preventive tools for better planning and decision making. This will, on the last analysis, provide better health to crops and increased yields. Developing transfer learning strategies together with compression techniques are likely to allow us to boost model performance and lessen the requirements for time-consuming inferences. Through the process of resource consumption optimizing, the platform may become more scalable and get to users from different conditions especially in resource-short environments. That's how it will facilitate widespread adoption and usability in agricultural areas.

REFERENCES

[1] M. Bakr, S. Abdel-Gaber, M. Nasr, and M. Hazman, "Tomato disease de- tection model based on densenet and transfer learning," Applied Computer Science, vol. 18, no. 2, 2022

[2] S. Ashok, G. Kishore, V. Rajesh, S. Suchitra, S. G. Sophia, and B. Pavithra, "Tomato leaf disease detection using deep learning techniques," in 2020 5th International Conference on Communication and Electronics Systems (IC- CES), pp. 979–983, IEEE, 2020

[3] S. Ahmed, M. B. Hasan, T. Ahmed, M. R. K. Sony, and M. H. Kabir, "Less is more: Lighter and faster deep neural architecture for tomato leaf disease classification," IEEE Access, vol. 10, pp. 68868–68884, 2022

[4] S. Albahli and M. Nawaz, "Dcnet: Densenet-77-based cornernet model for the tomato plant leaf disease detection and classification," Frontiers in Plant Science, vol. 13, p. 957961, 2022

[5] K.Roy, S. S. Chaudhuri, J. Frnda, S. Bandopadhyay, I. J. Ray, S. Banerjee, and J. Nedoma, "Detection of tomato leaf diseases for agro-based industries using novel pca deepnet," IEEE Access, vol. 11, pp. 14983–15001, 2023

[6] S. G. Paul, A. A. Biswas, A. Saha, M. S. Zulfiker, N. A. Ritu, I. Za- han, M. Rahman, and M. A. Islam, "A real-time application-based convolu- tional neural network approach for tomato leaf disease classification," Array, vol. 19, p. 100313, 2023

[7] T. Lu, B. Han, L. Chen, F. Yu, and C. Xue, "A generic intelligent tomato clas- sification system for practical applications using densenet-201 with transfer learning," Scientific Reports, vol. 11, no. 1, p. 15824, 2021