

POTATO BLIGHT CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK

A major project report submitted in partial fulfillment of the requirement for the
award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Aditya Sharma (201551)

Anshit Goel (201159)

Under the guidance & supervision of

Dr. Amit Kumar

Assistant Professor (SG)



**Department of Computer Science & Engineering and
Information Technology**

TABLE OF CONTENT

TITLE	PAGE NO.
Declaration	ii
Certificate	iii
Acknowledgement	iv
Abstract	v
Chapter-1 (Introduction)	1-5
Chapter-2 (Literature Survey)	6-9
Chapter-3 (System Development)	10-39
Chapter-4 (Testing)	39-39
Chapter-5 (Results & Evaluation)	40-60
Chapter-6 (Conclusion & Future Scope)	61-64
Reference	65

**Jaypee University of Information Technology,
Waknaghat, Solan - 173234 (India)
Candidate's Declaration**

I hereby declare that the work presented in this report entitled '**Potato Blight Classification using CNN**' in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Amit Kumar** (Assistant Professor (SG), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)

Student Name: Aditya Sharma

Roll No.: 201551

(Student Signature with Date)

Student Name: Anshit Goel

Roll No.: 201159

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature with Date)

Supervisor Name: Dr. Amit Kumar

Designation: Assistant Professor(SG)

Department: CSE & IT

Dated: 15/5/2024

CERTIFICATE

This is to certify that the work which is being presented in the project report titled “**Potato Blight Classification using CNN**” in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by Aditya Sharma,201551 & Anshit Goel,201159 during the period from August 2023 to May 2024 under the supervision of Dr. Amit Kumar,Assistant Professor(SG),Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Aditya Sharma – 201551

Anshit Goel – 201159

The above statement made is correct to the best of my knowledge.

(Dr. Amit Kumar)

Associate Professor(SG)

Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat,

AKCNOLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully.

I really grateful and wish my profound my indebtedness to Supervisor **Dr. Amit Kumar, Associate Professor(SG)**, Department of CSE Jaypee University of Information Technology, Wagnaghat. Deep Knowledge & keen interest of my supervisor in the field of “**Deep Learning and Machine Learning**” to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Deepak Gupta, Major Project Coordinator** Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straight forwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Aditya Sharma – 201551

Anshit Goel – 201159

ABSTRACT

Potatoes are one of the most commonly consumed staple foods, ranking fourth on the world food pyramid. Additionally, the global coronavirus pandemic has dramatically increased global potato consumption. On the other hand, potato diseases are the main cause of decline in crop quality and quantity. Disease misclassification and delayed identification can dramatically worsen plant condition due to which Potato farmers are experiencing significant financial losses due to numerous diseases. If farmers can identify these illnesses initially & operate them appropriately, they cannot waste a good percentage of money. Leaf state helps classify various potato diseases. In this project, with the help of convolution neural network we tried to classify a potato plant by its leaf whether it is healthy or not. Mainly there are three classes Late Blight, Early Blight & healthy. According to confidence each potato leaf has been classified.

CNN architecture consists of number of Convolutional layers succeeded by pooling layer to capture the spatial hierarchy of the input image. To improve the model's generalization ability, a transfer learning technique is used that leverages a model pre-trained on a large image dataset. The trained CNN is then estimated on a different test dataset to evaluate its performance in precisely classifying healthy potato plants and potato plants affected by late blight.

A simple CNN architecture is used to train and test the model. Main objective was to create a web API which will take leaf image as input and will classify according to the trained model. This model has high sensitivity & specificity to shows potential as a secure tool for beforehand detection & categorization of late blight in potato crops.

1.1 INTRODUCTION

Potato is one of the most broadly used up crop. If we want that the crop production continues well, then it is very important to identify the reasons which may cause problems in crop production. *Phytophthora infestans* & *Alternaria solani* are two pathogens that cause significant production damage in potato crop, causing diseases named late blight & early blight respectively. Somehow, we can detect these diseases early, then it will allow us for the implementation of preventive measures as well as the reduction of financial and yield losses. Late blight of potato, caused by the oomycete pathogen *Phytophthora infestans*, is a calamitous disease that affects potato cultivation worldwide. Early & accurate detection of late blight is very important to implement timely control measures to deprecate crop losses. In this study, we propose a new approach to classify potato late blight using convolutional neural networks (CNN). In recent years, advances in computer vision and machine learning have opened new opportunities to develop automated and effective disease detection systems. It has achieved remarkable success in various fields such as medical image processing and agriculture. This study concentrates on leveraging the power of his CNN for potato late blight classification. By influencing CNN's unique ability to instinctual pick up hierarchical attribute from image data, we head to develop a fit and dependable model that can distinguish between healthy & late infected potato plants. The use of deep learning techniques, especially CNNs, in agricultural disease classification characterize a paradigm shift towards precision agriculture and early intervention, eventually contributing to sustainable agricultural practices. The very common method for detecting & indeed identifying the plant diseases has been expert naked eye monitoring. However, in many circumstances, this strategy is impractical due to scarcity of experts on farms in remote places and long processing times. As a result, the use of image processing technologies has proven to be an excellent way for continuously checking plant health and early disease diagnosis. Hence, the objective of this project is to create classification methodology using simple convolutional neural network to detect disease by giving input as potato leaf.

1.2 PROBLEM STATEMENT

Develop a deep learning model based on CNNs to classify potato plants as either healthy or affected by potato blight using images of the plants.

Problem statement is to classify whether a plant is healthy or not by using an image of a potato leaf.

It is a multi-class classification problem.

Following are the three classes:

- Late Blight- Potato leaves are more deteriorated.
- Early Blight- Potato leaves are in the early stage of disease.
- Healthy- Leaves are healthy.



1.3 OBJECTIVES OF THE PROJECT

Agriculture sector contributes 18% to India's GDP and about 60% of Indian population work in this sector. So, there is need of new technologies to help in growth of crop production. If we can detect diseases early, we can apply various methods to prevent disease and ultimately production of crops will increase as well as revenue will also increase. If we try to monitor with our naked eyes, then it will be very time consuming and it also requires expertise in the field. So, we can apply image processing techniques to correctly classify plant whether it is healthy or not which will save time as well as resources. So, the objective of this project is to use CNN for image classification and create a web application which will take an image as an input and will classify it according to the saved model.

- **Automatic detection:** Design a CNN model that can instinctual detect & categorize late blight symptoms in potato plants.
- **Early intervention:** Rapid identification of plants infected with late blight allows for early intervention, thereby deprecate crop losses.
- **Precision Agriculture:** Contributes to precision agriculture by integrating advanced technologies for targeted disease management.
- **Non-Invasive Monitoring:** Provides a non-invasive solution for monitoring potato crops, reducing the need for labour-intensive visual inspection.
- **Efficient instruments:** Create effective and user-friendly instruments that can be utilized by farmers, advisors, in the agricultural industry.
- **Sustainable Agriculture:** Minimize the environmental impact of epidemics through targeted interventions and contribute to sustainable agriculture.

1.4 SIGNIFICANCE AND MOTIVATON OF THE PROJECT

The project “Classification of Potato Late Blight Using Convolutional Neural Networks (CNN)” is of great importance due to its possible impact on agriculture & related fields. This project head to automate the prior detection of potato late blight crops by making use of advanced technologies, specifically CNNs. This also allows for timely and accurate interventions, reducing economic losses for farmers, improving labor effectiveness, and promoting environmentally sustainable practices.

The project aligns oneself with the basics of precision agriculture and will facilitate the adoption of cutting-edge technology and contribute to global food security. Additionally, we will advance scientific knowledge by demonstrating practical applications of deep learning & computer vision in the fight against plant diseases, ultimately increasing strength to disease outbreaks and encourage a culture of concoction in agriculture.

The motivation behind the project “Potato Blight Classification Using CNN” comes from the urgent need to address challenges in the agricultural sector, particularly in potato cultivation.

There are several important factors that motivated this project.

- 1. Economic Impact:** Potato late blight poses a serious threat to global food security as it causes significant economic losses through reduced crop yields. This classification system can reduce losses by permitting early detection & intervention.
- 2. Timely disease management:** Old-style methods of detecting diseases in potato crops are often time-consuming and require visual inspection by experts. The motivation is to create a system that can quickly and accurately detect the symptoms of an epidemic, thereby allowing timely treatment of the disease.
- 3. Precision Agriculture:** Precision agriculture aims to optimize resource use and increase productivity.
- 4.** The project aims to subsidize to precision agriculture by leveraging advanced technologies such as CNN to provide targeted solutions for disease management.
- 5. Labour Efficiency:** Manual inspection of large agricultural fields is labour intensive and can delay detection of disease outbreaks. Automating the classification process using CNN increases operational efficiency and reduces the workload of farmers.
- 6. Technological Advancements:** The motivation is to leverage advances in deep learning and computer vision to address real-world agricultural challenges. The application of CNN to classify plant diseases shows that this technology has the potential to revolutionize traditional agricultural practices.

1.5 METHODOLOGY

This problem is an image classification problem. We are using convolutional neural network to train our model as it provides better resource utilization and better accuracy as compare to normal neural networks. It normally contains sequentially number of convolution layers, max pooling layers, activation function followed by normal dense layers. This procedure assimilates data-driven processes, deep learning techniques, & model evaluation to create an effectual and dependable potato late blight classification tool. The iterative character of model training and evaluation allows for refinement & optimization, ensuring the effectual of the system in real agricultural environments. These extra layers provide better accuracy as compare to normal dense layers. ReLU activation function has been used

and also adam optimizer has been used as it provides both dynamically changing learning rate and exponential weighted average concept to reduce noise.

1.6 ORGANIZATION OF THE PROJECT

S. No.	Chapter	Sections
1.	Chapter 1: Introduction	1.1 Introduction 1.2 Problem Statement 1.3 Objectives 1.4 Significance and Motivation of the Project Work 1.5 Methodology 1.6 Organization of Project Report
2.	Chapter 2: Literature Survey	The literature survey should include the existing information available in standard books, journals, popular websites, top organizations' technical/white papers, etc., with a clear emphasis on the recent five years' work. 2.1 Overview of Relevant Literature 2.2 Key Gaps in the Literature
3.	Chapter 3: System Development	3.1 Requirements and Analysis 3.2 Project Design and Architecture 3.3 Data Preparation 3.4 Implementation (include code snippets, algorithms, tools and techniques, etc.) 3.5 Key Challenges (discuss the challenges faced during the development process and how these are addressed)
4.	Chapter 4: Testing	4.1 Testing Strategy (discuss the testing strategy/tools used in the project) 4.2 Test Cases and Outcomes
5.	Chapter 5: Results and Evaluation	5.1 Results (presentation of findings, interpretation of the results, etc.) 5.2 Comparison with Existing Solutions (if applicable)

6.	Chapter 6: Conclusions and Future Scope	6.1 Conclusion (summarize key findings, limitations and contributions to the field). 6.2 Future Scope
----	---	--

LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

“End-to-end hybrid deep learning framework for potato leaf disease prediction.” [1]

The paper we suggest PLDPNet, a novel hybrid deep learning model. The PLDPNet improves the accuracy of disease prediction by utilizing deep feature ensemble fusion and auto-segmentation modules. The suggested PLDPNet performs end-to-end with a recorded accuracy of 98.66%.

“A Convolutional Neural Network Based Potato Leaf Diseases Detection Using Sequential Model” [2]

The Paper focuses on identifying potato leaf diseases in Bangladesh, where the practice of growing potatoes has gained popularity but is hampered by a number of ailments that drive up production costs and cause financial losses for growers. In order to precisely identify and diagnose diseases in potato leaf pictures, a new model based on Convolutional Neural Networks (CNN) is presented in this research.

The model was evaluated on both healthy and diseased potato leaves, and it was able to predict potato leaf illnesses with 94.2% accuracy.

“In-field classification of the asymptomatic biotrophic phase of potato late blight based on deep learning and proximal hyperspectral imaging” [3]

In this paper, a deep learning organization architecture for different pictures is proposed, which combines deep cooperative attention networks with 2D and 3D convolutional neural networks (CNNs).

In test dataset of 2000 photos, the accurateness was 0.739 in the whole band and 0.790 in certain bands.

Using proximal hyperspectral imaging and deep learning, the study shows inspiring findings for the classification of the asymptomatic biotrophic phase of PLB illness.

“Potato Blight Detection Using Fine-Tuned CNN Architecture” [4]

The aim of the study is to personalized CNN with less computation time and information loss to detect potato blight more accurately. With an overall accuracy of 99%, the suggested model surpassed

competing machine and deep learning techniques. Sections on the literature study, techniques and materials, suggested architecture, results and discussion, comparative analysis, conclusion, and future scope make up the remaining portion of the work.

“Automated recognition of optical image based potato leaf blight diseases using deep learning” [5]

This paper discusses the serious problem of fungal blight diseases that impact potato crops all over the world, highlighting the danger they pose to the world’s food security. The quantity and quality of potato yields are greatly decreased by early blight from *Alternaria solani* and late blight from *Phytophthora infestans*. Farmers have traditionally used a subjective and time-consuming visual detection method. In order to address this, the research uses optical images of potato leaves to investigate deep learning models. With an accuracy rate of 92.69%, VGG16 is the most accurate. After more fine-tuning, the methodology achieves an amazing 97.89% accuracy in identifying between healthy potato leaves and late and early blight syndromes.

“Supervised Learning-Based Image Classification for the Detection of Late Blight in Potato Crops” [6]

This paper uses supervised learning and image classification techniques, such as SVM & CNN, to detect disease in potato crops. The scientists assembled a collection of photos of potato crops and built a classification algorithm to distinguish between healthy and diseased plants. The efficacy of the CNN and SVM models was assessed using performance metrics, such as the area under the curve (AUC). The outcomes demonstrated that the SVM had an AUC of 0.87 and the CNN had an AUC of 0.97.

“Leaf Blights Detection and Classification in Large Scale Applications” [7]

The study suggests using entropy reduction, SVM classification, & deep CNN models derived from Squeeze Net and Shuffle Net to enhance the precision and resilience of leaf blight detection. To capture the entire color spectrum and improve accuracy, the CIELAB color space is used. The study’s results indicate promising results, with 98% accuracy in early leaf blight detection.

“Enhanced Field-Based Detection of Potato Blight in Complex Backgrounds Using Deep Learning” [8]

This paper presents an automated system for field-based potato leaf blight disease patch detection using Mask R-CNN architecture. The system was trained on a dataset of photos of potato leaves taken at various times of the day and in various geographical locations. It makes use of transfer learning. Using pictures of plant leaves, recent developments in image processing have shown promise for quick and automatic disease detection. With varying degrees of success, earlier studies have tried to use segmentation &

multiclass SVMs to detect potato disease.

“CNN built Disease Detection Method on Potato Leaves.” [9]

The expertise that is proposed make use of image handling tools/techniques to surely identify & find diseases of potato leaves. It is a ML algorithm that is famous for its intense performance in image organization, & it is used in the model defined in the paper. The algorithm attains a high precision of 97% in categorizing potato leaf as either normal or diseased. The expertise could help farmers notice diseases in potato harvests more effectively & decrease their monetary losses.

“Identification of Disease in Potato Leaves Using Convolutional Neural Network Algorithm.” [10]

The model presented in the paper makes use of CNN, a machine learning algorithm that is well-known for its remarkable and good performance in image classification. This model indeed distinguishes between normal and abnormal leaf characteristics by likening normal & disease-affected potato leaves. This also classifies potato plant leaf as either normal or it is diseased with high degree of precision-97%.

“Image-Based Plant Disease Identification by Deep Learning Meta-Architectures. [11]

This research focuses on the localization and classification of plant diseases in leaves using deep learning meta-architectures. TensorFlow object detection framework was used to apply three meta-architectures: SSD, RCNN, and RFCN. Using an Adam optimizer, the SSD model yielded a mean average precision (MAP) of 73.07% after training and testing the models on a dataset in a controlled environment. The study’s originality was established by the successful documentation of 26 distinct types of faulty leaves & 12 types of healthy leaves within a single outline. The generated weights are utilized for real-time disease detection in both controlled and uncontrolled environments, and the suggested methodology may find application in other agricultural aspects.

2.2 KEY GAPS IN THE LITERATURE

The bounded availability of tagged plant diseases limited the capability to comprehensively train the proposed model, and training each module was found to be time-consuming. [1]

The conclusion of photos in this collection was rated as moderate. [2]

Although it concentrates on potato leaf diseases, it does not consider precluding of diseases that may influence potato crops. [3]

This paper does not cope with the following potential limitations & challenges faced by the CNN architecture fine-tuning process: like Overfitting or hyperparameter tuning. [4]

They used the existing architecture without any innovation in the design. Therefore, the contribution is not important for technical reasons. [5]

The paper does not provide information on the size of the dataset used for training and validation, which could affect the reliability and robustness of model. [6]

No additional features were selected by optimizing different depth models to improve efficiency & accurateness. [7]

This document does not discuss the drawbacks of using different color spaces to improve recognition performance. [8]

It focuses on potato leaf diseases, but does not study the discovery of other different types of diseases and pests that can affect potato plants. [9]

The results performed on potato leaves give better results when the 70: 30 data set is split assimilate to the 80: 20 data set. [10]

This paper does not argue the performance of deep learning approaches in real-world situations and does not compare them with other existing techniques. [11]

SYSTEM DEVELOPMENT

3.1 REQUIREMENT AND ANALYSIS

➤ **HARDWARE REQUIREMENTS**

- PC with 64-bit operating system, x64-based processor
- 8GB RAM or above
- 300GB hard disk or above
- 2GB graphic card or above

➤ **SIMULATION REQUIREMENTS**

- Windows 10
- Python 3.7 (as language)
- Anaconda (Jupyter Notebooks)
- GPU Support
- Python Packages
- Fast API, React JS , GCP
- Necessary Libraries: TensorFlow, Keras (DL Frameworks) , Scikit-Learn(ML Lib) , Numpy/Pandas(For data manipulation) , Matplotlib(For data visualization), OpenCV (Image processing Lib).

3.2 PROJECT DESIGN AND ARCHITECTURE

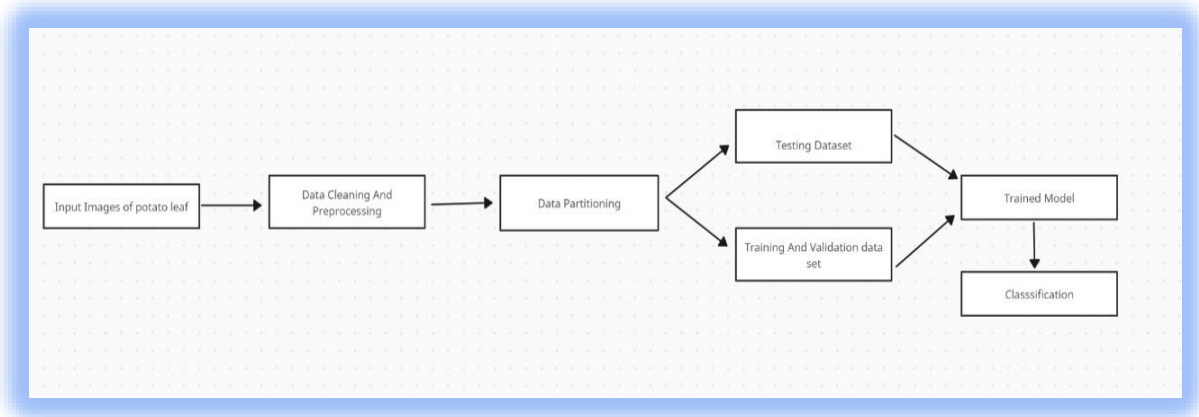


Fig- 3.1 Data Flow Diagram(Level-0)

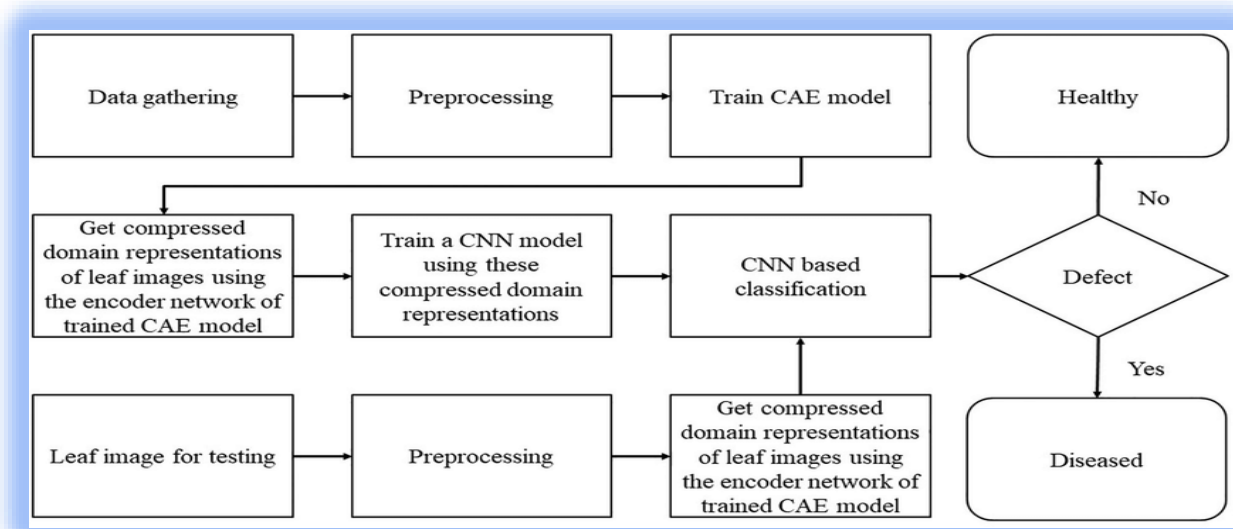


Fig -3.2 Data Flow Diagram (Level-1)

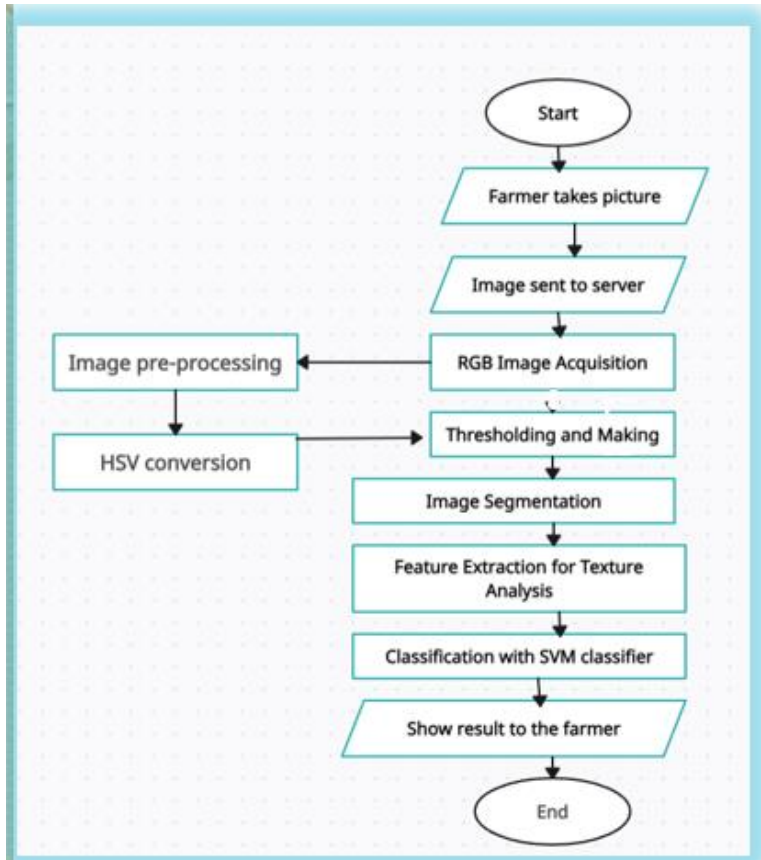


Fig-3.3 Flow Chart

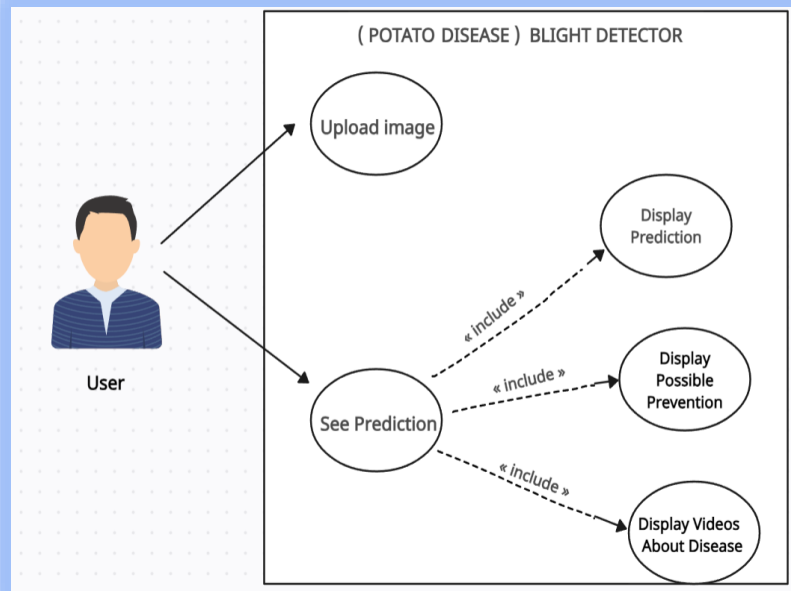


Fig-3.4 Use case Diagram

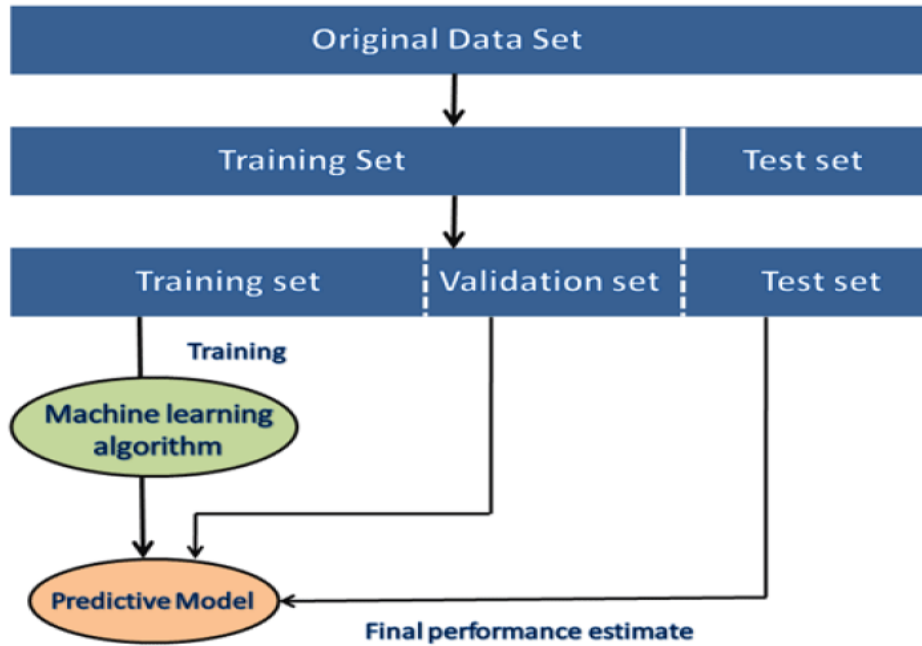


Fig -3.5 State Transition Diagram

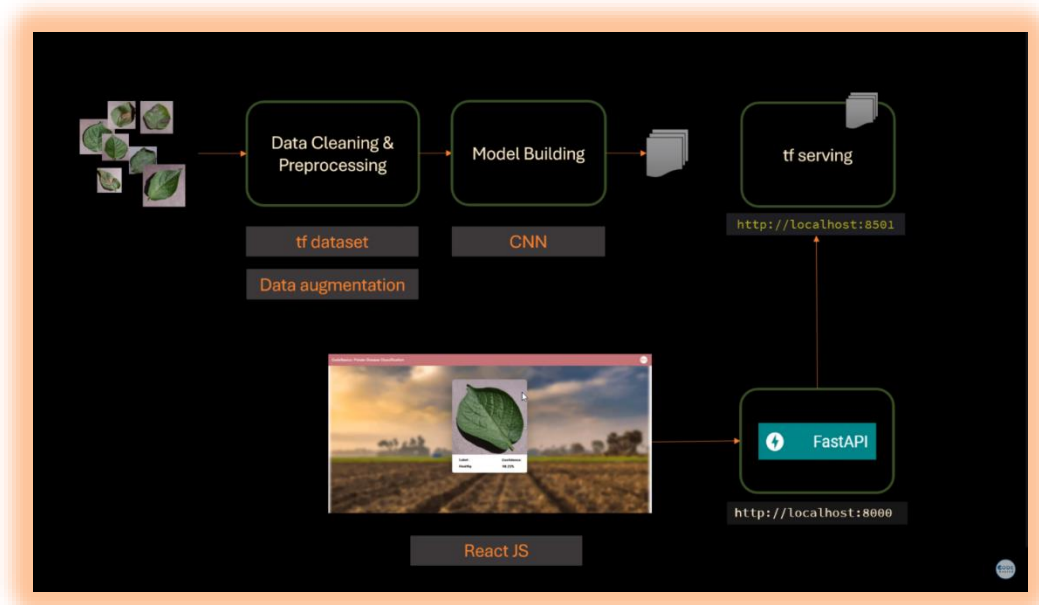


Fig-3.6 Web Application

3.3 DATA PREPARATION

Data Set Used

Every supervised machine learning project commences with a data gathering process.

There are basically three steps to collect dataset:

- 1) Collect and annotate data on your own.
- 2) Write web scrapping scripts to collect images from internet.
- 3) Buy data from third party vendors or use public repositories such as Kaggle.

Dataset occupied from kaggle repository. Dataset consists images belonging to three different classes.

Plant Village (Only extracted Healthy, Early & Late Blight files) from open source: **Kaggle**

Below is the link for the dataset :

<https://www.kaggle.com/arjuntejaswi/plant-village>

Dataset contains three types of images:



Potato_healthy



Potato_Early_blight



Potato_Late_blight

Fig-3.7 Represents different classes of the leaf

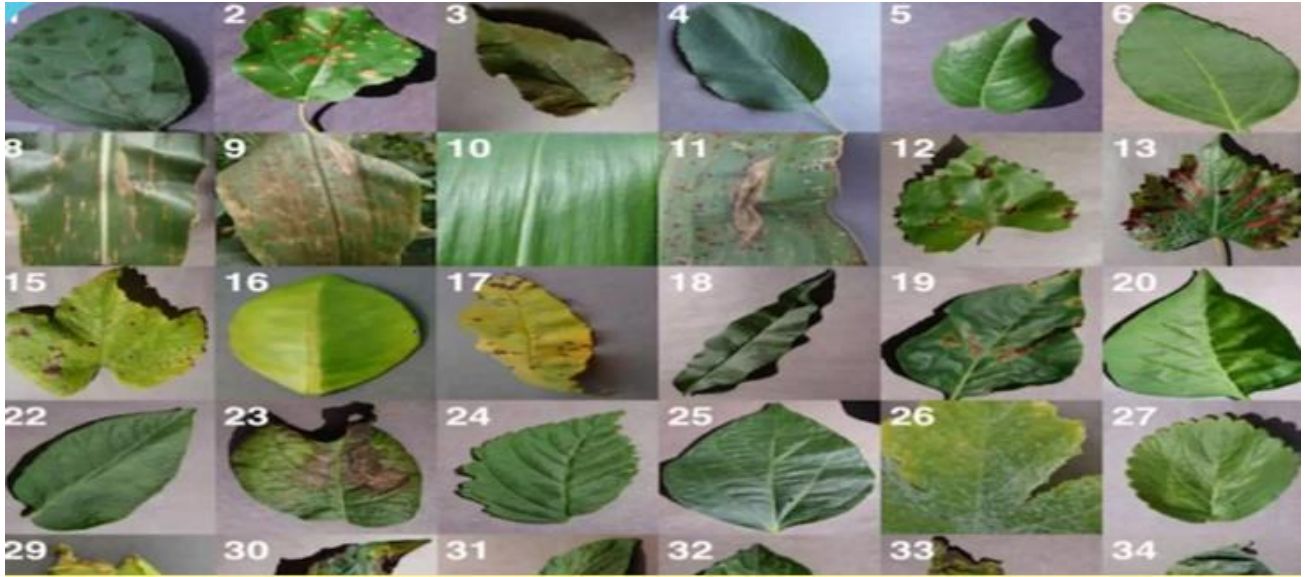


Fig- 3.8 Represents images in Dataset.

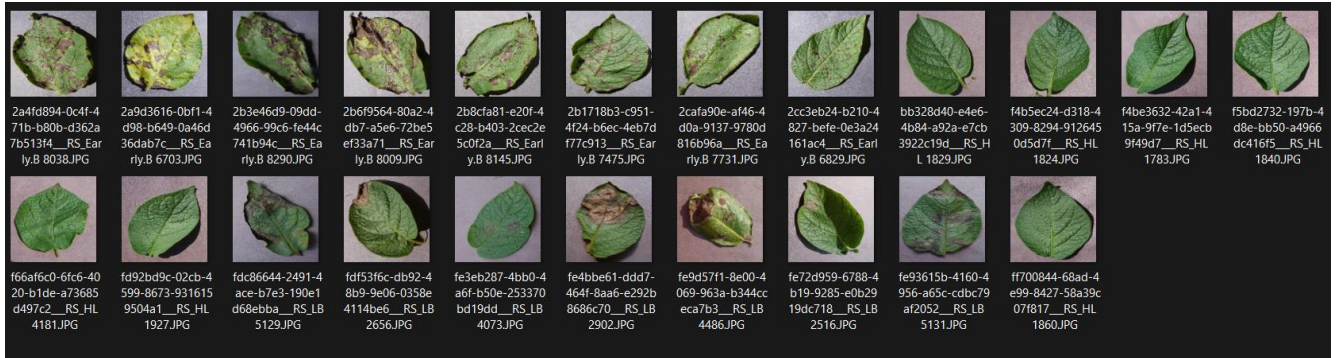


Fig- 3.9 Represents images in Test Dataset.

3.4 IMPLEMENTATION

(Code Snippets)

```
pip install --upgrade pip
```

```
pip install tensorflow
```

```
pip install matplotlib
```

Fig- 3.10 Installing necessary libraries

```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
```

```
IMAGE_SIZE = 256
BATCH_SIZE = 32
CHANNELS=3
EPOCHS =50
```

```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "PlantVillage",
    shuffle=True,
    image_size =(IMAGE_SIZE,IMAGE_SIZE),
    batch_size =BATCH_SIZE
)
```

Fig-3.11 Importing libraries and dataset

```
class_names = dataset.class_names
class_names
```

Fig-3.12 Class names

```
len(dataset)
```

Fig-3.13 length of dataset

```
train_size =0.8  
len(dataset)*train_size
```

```
plt.figure(figsize=(10,10))  
for image_batch, label_batch in dataset.take(1):  
    for i in range(12):  
        ax=plt.subplot(3,4,i+1)  
        plt.imshow(image_batch[i].numpy().astype("uint8"))  
        plt.title(class_names[label_batch[i]])  
        plt.axis("off")
```

Fig- 3.14 Plotting the graph

```
80% ==> training  
20% ==> 10% validation, 10% test
```

```
train_ds=dataset.take(54)  
len(train_ds)
```

```
val_size=0.1  
len(dataset)*val_size
```

```
val_ds=test_ds.take(6)  
len(val_ds)
```

```
test_ds=test_ds.skip(6)  
len(test_ds)
```

Fig-3.15 training and testing dataset length


```
def get_dataset_partitions_tf(ds,train_split=0.8,val_split=0.1,test_split=0.1, shuffle=True, shuffle_size=10000):
    ds_size=len(ds)
    if shuffle:
        ds=ds.shuffle(shuffle_size, seed=12)
    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)
    train_ds= ds.take(train_size)
    val_ds= ds.skip(train_size).take(val_size)
    test_ds=ds.skip(train_size).skip(val_size)
```

```
train_ds,val_ds,test_ds=get_dataset_partitions_tf(dataset)
```

```
len(train_ds)
```

Fig- 3.16 dataset partitioning

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

```
resize_and_rescale= 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)
])
```

Fig-3.17 Data Augmentation

```

input_shape=(BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes=3
model =models.Sequential([
    resize_and_rescale,
    data_augmentation,
    layers.Conv2D(32,(3,3),activation='relu',input_shape=input_shape),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, kernel_size=(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(n_classes,activation='softmax'),
])
model.build(input_shape=input_shape)

```

Fig -3.18 Model building

```

model.summary()

```

Fig -3.19 Model summary

```

model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)

```

```

history=model.fit(
    train_ds,
    epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    verbose=1,
    validation_data=val_ds
)

```

Fig -3.18 Compiling model and model history

```

scores= model.evaluate(test_ds)

```

Fig -3.19 Evaluating model

```
scores
```

```
history.history.keys()
```

Fig -3.20 Model history

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

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

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

Fig -3.21 Plotting and subplotting model

```
import numpy as np
for images_batch, labels_batch in test_ds.take(1):
    first_image = image_batch[0].numpy().astype('uint8')
    first_label = labels_batch[0].numpy()

    print("first image to predict")
    plt.imshow(first_image)
    print("actual label:" ,class_names[first_label])

    batch_prediction = model.predict(images_batch)
    print("predicted label:" ,class_names[np.argmax(batch_prediction[0])])
```

Fig-3.22 Batch Prediction

```

def predict(model,img):
    img_array=tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    img_array=tf.expand_dims(img_array,0)
    predictions = model.predict(img_array)
    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100*(np.max(predictions[0])),2)
    return predicted_class,confidence

plt.figure(figsize=(15,15))
for images,labels in test_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3,3,i+1)
        plt.imshow(images[i].numpy().astype("uint8"))
        predicted_class,confidence = predict(model,images[i].numpy())
        actual_class = class_names[labels[i]]
        plt.title(f"Actual:{actual_class},\n Predicted: {predicted_class}.\n Confidence:{confidence}%")
        plt.axis("off")

```

Fig -3.23 Prediction outcome with confidence

```

model_version=1
model.save(f"../models/{model_version}")

```

Fig- 3.24 Saving model

IMPLEMENTATION – APPLICATION

(Code Snippets)

```

C: > Users > anshi > Desktop > App Development(RN) > JS Appjs > ...
1  import React, {useState} from 'react';
2  import {
3    SafeAreaView,
4    Image,
5    StatusBar,
6    StyleSheet,
7    Text,
8    Platform,
9    Dimensions,
10   useColorScheme,
11   View,
12   TouchableOpacity,
13   ImageBackground,
14 } from 'react-native';
15 import axios from 'axios';
16 import Config from 'react-native-config';
17 import {launchCamera, launchImageLibrary} from 'react-native-image-picker';
18 import {Colors} from 'react-native/Libraries/NewAppScreen';
19 import PermissionsService, {isIOS} from './Permissions';
20
21 axios.interceptors.request.use(
22   async config => {
23     let request = config;
24     request.headers = {
25       'Content-Type': 'application/json',
26       Accept: 'application/json',
27     };
28     request.url = configureUrl(config.url);
29     return request;
30   },

```

Fig- 3.25 Importing libraries and using interceptors request.

```

C: > Users > anshi > Desktop > App Development(RN) > JS Appjs > ...
31   error => error,
32   );
33
34   export const {height, width} = Dimensions.get('window');
35
36   export const configureUrl = url => {
37     let authUrl = url;
38     if (url && url[url.length - 1] === '/') {
39       authUrl = url.substring(0, url.length - 1);
40     }
41     return authUrl;
42   };
43
44   export const fonts = {
45     Bold: {fontFamily: 'Roboto-Bold'},
46   };
47
48   const options = {
49     mediaType: 'photo',
50     quality: 1,
51     width: 256,
52     height: 256,
53     includeBase64: true,
54   };
55
56   const App = () => {
57     const [result, setResult] = useState('');
58     const [label, setLabel] = useState('');
59     const isDarkMode = useColorScheme() === 'dark';
60     const [image, setImage] = useState('');

```

Fig- 3.26 App building.

```

60  const [image, setImage] = useState('');
61  const backgroundStyle = {
62    backgroundColor: isDarkMode ? Colors.darker : Colors.lighter,
63  };
64
65  const getPredication = async params => {
66    return new Promise((resolve, reject) => {
67      var bodyFormData = new FormData();
68      bodyFormData.append('file', params);
69      const url = Config.URL;
70      return axios
71        .post(url, bodyFormData)
72        .then(response => {
73          resolve(response);
74        })
75        .catch(error => {
76          setLabel('Failed to predicting. ');
77          reject('err', error);
78        });
79    });
80  };
81
82  const manageCamera = async type => {
83    try {
84      if (!(await PermissionsService.hasCameraPermission())) {
85        return [];
86      } else {
87        if (type === 'Camera') {
88          openCamera();

```

Fig- 3.27 Framing and Managing camera.

```

89    } else {
90      openLibrary();
91    }
92  }
93  } catch (err) {
94    console.log(err);
95  }
96  };
97
98  const openCamera = async () => {
99    launchCamera(options, async response => {
100      if (response.didCancel) {
101        console.log('User cancelled image picker');
102      } else if (response.error) {
103        console.log('ImagePicker Error: ', response.error);
104      } else if (response.customButton) {
105        console.log('User tapped custom button: ', response.customButton);
106      } else {
107        const uri = response?.assets[0]?.uri;
108        const path = Platform.OS !== 'ios' ? uri : 'file://' + uri;
109        getResult(path, response);
110      }
111    });
112  };
113
114  const clearOutput = () => {
115    setResult('');
116    setImage('');

```

Fig- 3.28 Scanner setting.

```

117     });
118
119     const getResult = async (path, response) => {
120       setImage(path);
121       setLabel('Predicting...');
122       setResult('');
123       const params = {
124         uri: path,
125         name: response.assets[0].fileName,
126         type: response.assets[0].type,
127       };
128       const res = await getPredication(params);
129       if (res?.data?.class) {
130         setLabel(res.data.class);
131         setResult(res.data.confidence);
132       } else {
133         setLabel('Failed to predict');
134       }
135     };
136
137     const openLibrary = async () => {
138       launchImageLibrary(options, async response => {
139         if (response.didCancel) {
140           console.log('User cancelled image picker');
141         } else if (response.error) {
142           console.log('ImagePicker Error: ', response.error);
143         } else if (response.customButton) {
144           console.log('User tapped custom button: ', response.customButton);
145         }

```

Fig- 3.29 Launching image library.

```

145     } else {
146       const uri = response.assets[0].uri;
147       const path = Platform.OS !== 'ios' ? uri : 'file://' + uri;
148       getResult(path, response);
149     }
150   });
151 };
152
153 return (
154   <View style={[backgroundStyle, styles.outer]}>
155     <StatusBar barStyle={isDarkMode ? 'light-content' : 'dark-content'} />
156     <ImageBackground
157       blurRadius={10}
158       source={{uri: 'background'}}
159       style={{height: height, width: width}}
160     />
161     <Text style={styles.title}><'Potato Disease \nPrediction App'</Text>
162     <TouchableOpacity onPress={clearOutput} style={styles.clearStyle}>
163       <Image source={{uri: 'clean'}} style={styles.clearImage} />
164     </TouchableOpacity>
165     {(image?.length && (
166       <Image source={{uri: image}} style={styles.imageStyle} />
167     )) ||
168     null}
169     {(result && label && (
170       <View style={styles.mainOuter}>
171         <Text style={styles.space} style={styles.labelText}>

```

Fig- 3.30 Application settings.

```

171     <Text style={[styles.space, styles.labelText]}>
172       {'Label: \n'}
173     <Text style={styles.resultText}>{label}</Text>
174   </Text>
175   <Text style={[styles.space, styles.labelText]}>
176     {'Confidence: \n'}
177     <Text style={styles.resultText}>
178       | {parseFloat(result).toFixed(2) + '%'}
179     </Text>
180   </Text>
181 </View>
182 ) ||
183 (image && <Text style={styles.emptyText}>{label}</Text>) || (
184   <Text style={styles.emptyText}>
185     Use below buttons to select a picture of a potato plant leaf.
186   </Text>
187 )}
188 <View style={styles.btn}>
189   <TouchableOpacity
190     activeOpacity={0.9}
191     onPress={() => manageCamera('Camera')}
192     style={styles.btnStyle}>
193     <Image source={{uri: 'camera'}} style={styles.imageIcon} />
194   </TouchableOpacity>
195   <TouchableOpacity
196     activeOpacity={0.9}
197     onPress={() => manageCamera('Photo')}
198     style={styles.btnStyle}>
199     <Image source={{uri: 'gallery'}} style={styles.imageIcon} />

```

Fig- 3.31 Frontend process.

```

199     <Image source={{uri: 'gallery'}} style={styles.imageIcon} />
200   </TouchableOpacity>
201 </View>
202 </View>
203 );
204 };
205
206 const styles = StyleSheet.create({
207   title: {
208     alignSelf: 'center',
209     position: 'absolute',
210     top: (isIOS && 35) || 10,
211     fontSize: 30,
212     ...fonts.Bold,
213     color: '#FFF',
214   },
215   clearImage: {height: 40, width: 40, tintColor: '#FFF'},
216   mainOuter: {
217     flexDirection: 'row',
218     justifyContent: 'space-between',
219     position: 'absolute',
220     top: height / 1.6,
221     alignSelf: 'center',
222   },
223   outer: {
224     flex: 1,
225     alignItems: 'center',
226     justifyContent: 'center',
227   },

```

Fig- 3.32 Frontend process.


```

228   btn: {
229     position: 'absolute',
230     bottom: 40,
231     justifyContent: 'space-between',
232     flexDirection: 'row',
233   },
234   btnStyle: {
235     backgroundColor: '#FFF',
236     opacity: 0.8,
237     marginHorizontal: 30,
238     padding: 20,
239     borderRadius: 20,
240   },
241   imageStyle: {
242     marginBottom: 50,
243     width: width / 1.5,
244     height: width / 1.5,
245     borderRadius: 20,
246     position: 'absolute',
247     borderWidth: 0.3,
248     borderColor: '#FFF',
249     top: height / 4.5,
250   },
251   clearStyle: {
252     position: 'absolute',
253     top: 100,
254     right: 30,
255     tint-color: '#FFF',
256     zIndex: 10

```

Fig- 3.33 Frontend process.

```

256     zIndex: 10,
257   },
258   space: {marginVertical: 10, marginHorizontal: 10},
259   labelText: {color: '#FFF', fontSize: 20, ...fonts.Bold},
260   resultText: {fontSize: 32, ...fonts.Bold},
261   imageIcon: {height: 40, width: 40, tint-color: '#000'},
262   emptyText: {
263     position: 'absolute',
264     top: height / 1.6,
265     alignSelf: 'center',
266     color: '#FFF',
267     fontSize: 20,
268     maxWidth: '70%',
269     ...fonts.Bold,
270   },
271 });
272
273 export default App;

```

Fig- 3.34 Saving model

ALGORITHM'S AND TOOLS/TECHNIQUES

What are Convolutional Neural Networks?

Convolutional neural networks are a special type of neural network that has proven to be very effectual in classifying images. Because these networks provide additional layers such as convolutions, pools of these networks are used in computer vision operation & are mostly useful for face acknowledgement & object discovery. Generally, this particular type of network consists of multiple convolutional layers. The last layer is followed by thick layers that are fully connected to each other.

CNNs are designed to take advantage of the two-dimensional structure of images. Combined weights and affiliate connections were used. This is succeeding by a max or average pooling layer that extracts the most salient attribute. The main merits of CNNs over regular neural networks is that they require fewer parameters to be trained compared to neural networks, rather than the same number of hidden layers. Easy to train due to fewer parameters.

Overview of different operations in CNN:

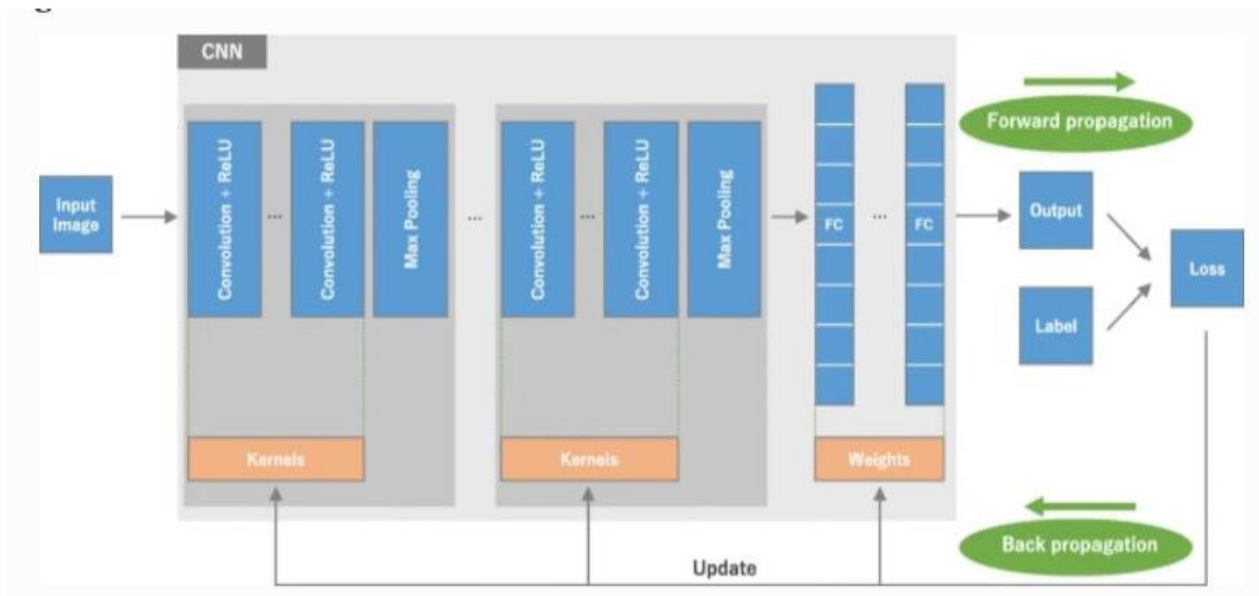


Fig -3.35 Overview of different operations in CNN

As shown in the diagram above, this is a CNN configuration that includes several consecutive operations. CNN accord of a convolutional layer with a kernel or filter applied that is used to determine vertical or horizontal edges, succeed by a ReLU activation function.

The next layer is the max-pooling layer, where a kernel is applicable to extract the most salient attribute from the image. This is trailed by a completely simply associated dense layer.

The accurateness of the model is resolute by forward propagation Loss function, and the backpropagation filters and biases are updated according to the weight of the loss value.

This can be done by gradient descent, AdaDelta, or Adam.

Convolution

In this process, a small tensor is applied across an input image which is called as a kernel or a filter. Its main objective is to abstract important features from the figure. After operation, generation of feature map takes place which is computed through element wise multiplication of each value in image and kernel. This feature map will contain the vertical and horizontal edges which are enhanced in next step.

Figure showing convolution operation

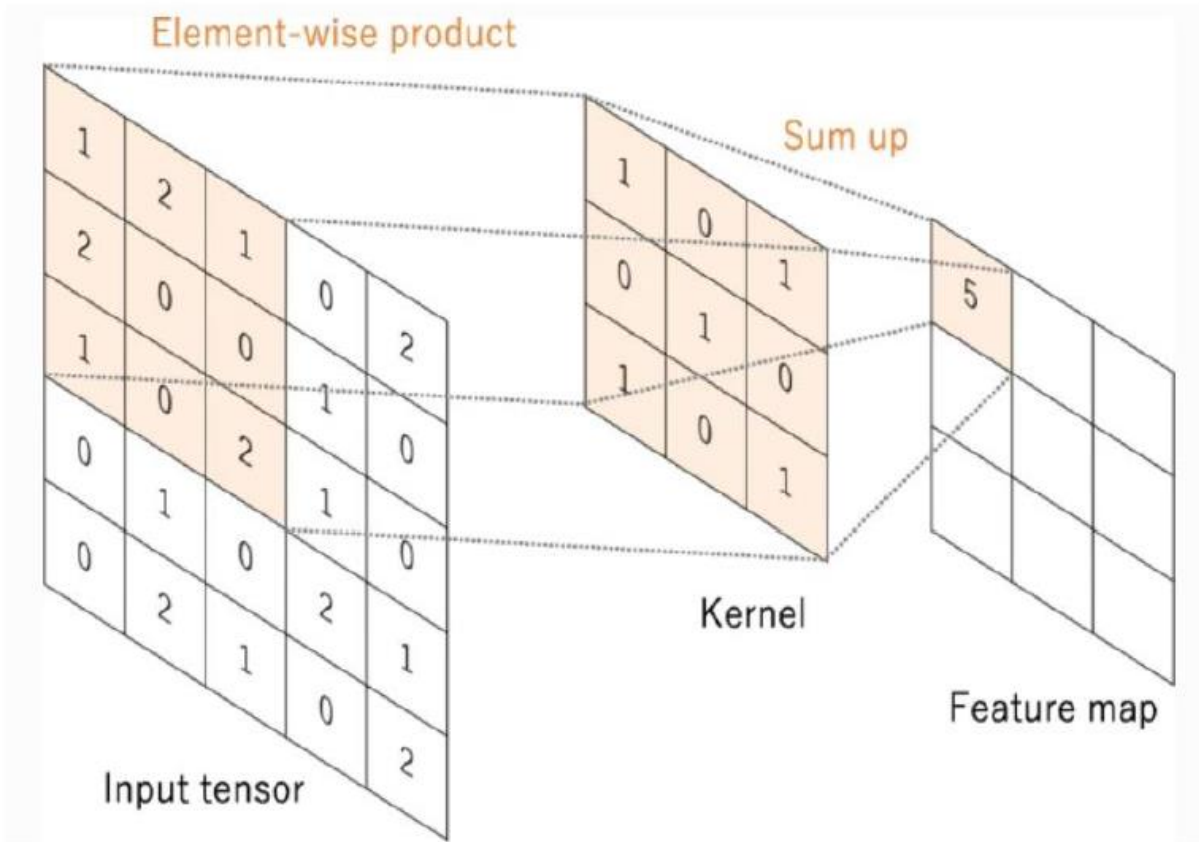


Fig -3.36 Figure showing convolution operation

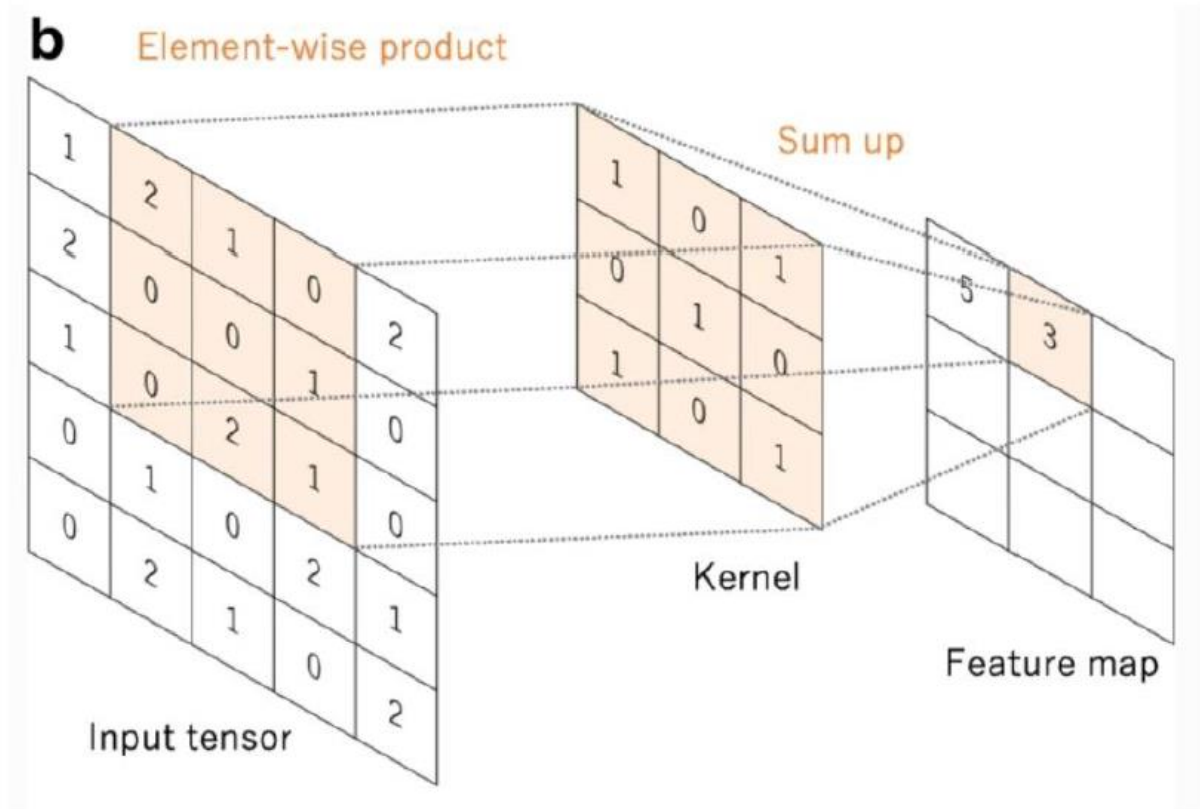


Fig -3.37 Figure showing element-wise product

Above figure is showing a convolution operation on 5x5 image. Filter size is 3x3 without padding and stride equal to 1. If there is no padding, then it will reduce its size as shown in the figure. To maintain same image size padding is used which will add extra pixels across image to maintain its size.

To calculate image size:

$$= (N + 2P - F) / S + 1$$

Where **N** is the image size,

P is the padding,

F is the kernel size,

S is the stride.

Figure with padding: **8**.

Figure with padding:

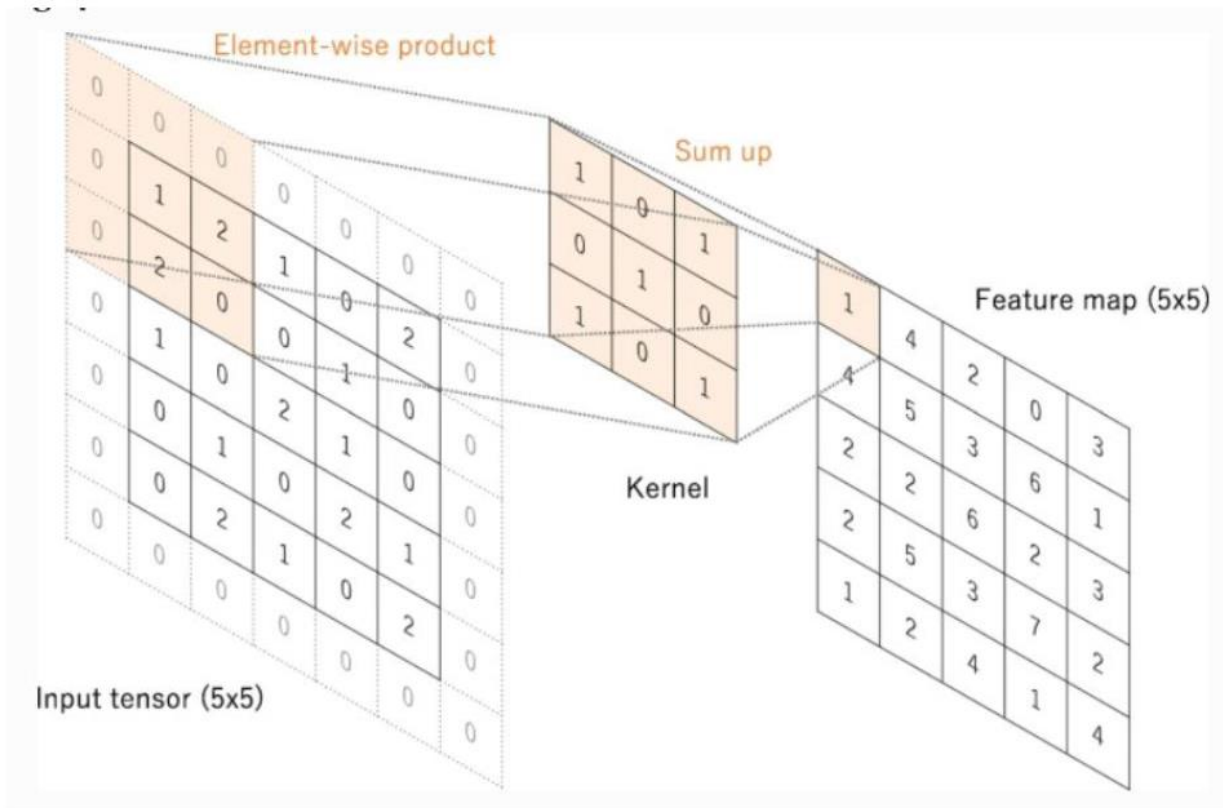


Fig -3.38 Figure showing padding

Activation Function:

Feature maps which are obtained after applying convolution are passed through an activation function. These activation functions are non-linear to introduce non-linearity in the network. Activation functions such as sigmoid, tangent are not used in hidden layers as it may result in vanishing gradient problem. Highly used activation function is Relu function.

$$F(x) = \max(0,x)$$

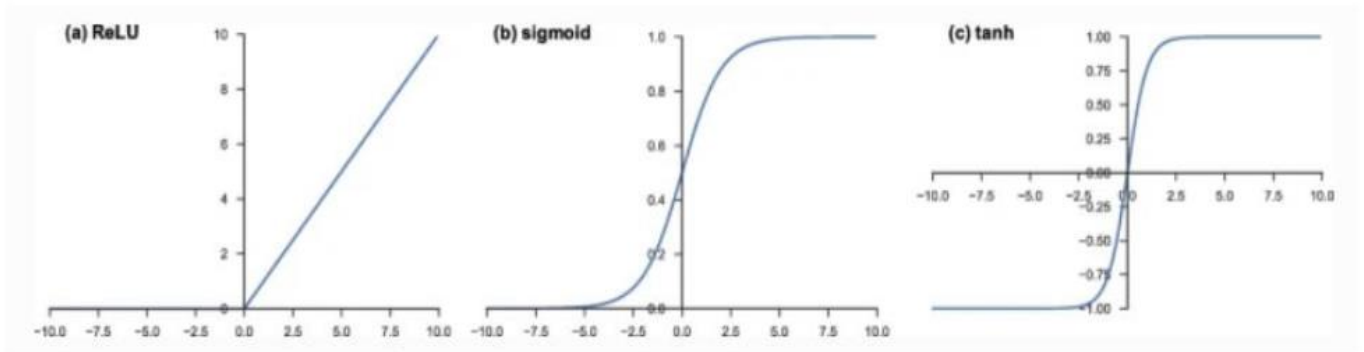


Fig -3.39 Figure showing different activation function used.

Pooling Layer

This layer is used to extract the most important features. Its main task is to reduce the learnable parameters. It reduces the height and width of feature map but depth remains same.

Pooling layers are edge in between back-to-back convolutional layers of a CNN. The admixture of convolutional & pooling layer aid create a hierarchy of progressively abstract and immutable features, allowing the network to learn complex define of the input data.

Max Pooling

In this layer a kernel is applied across an image which will extract maximum value from each patch. Mainly 2x2 filter size is used with stride equal to 2 over an image. Below is the figure showing max pooling operation.

The max-pooling layer plays an important role in spatial down sampling and helps bring a hierarchical representation of attributes in CNN.

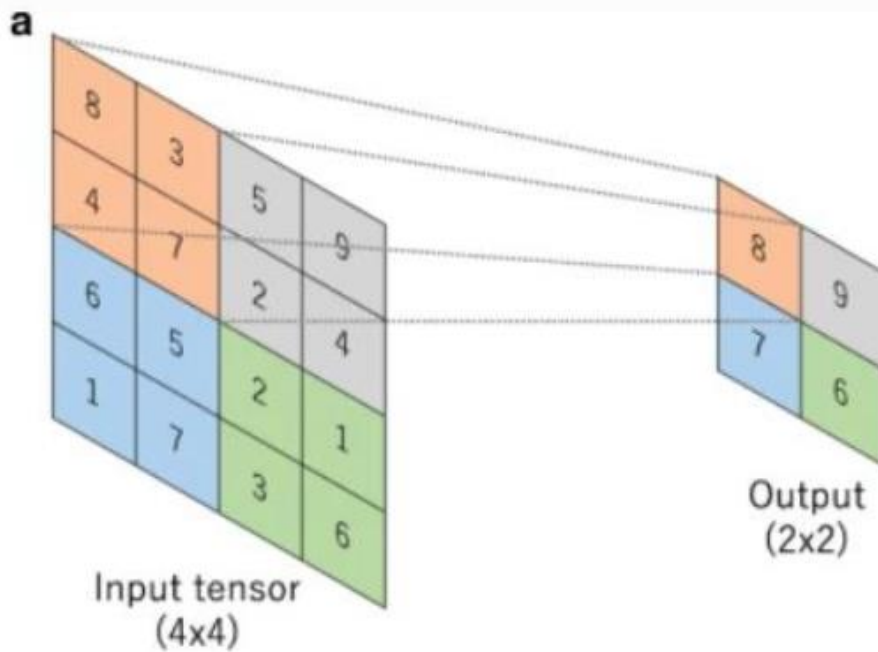


Fig -3.40 Figure showing Pooling layer.

Fully Connected Dense Layer

After taking the feature maps from the max-pooling layer, feature maps are leveled and passed to the completely connected dense layer. Feature maps are first converted to a one-dimensional array and passed to a dense layer. There can be many dense layers in-between. Final output layer mainly contains the number of output classes. Every hidden dense layer contains an activation function. Most commonly used activation function is Relu.

Activation Function at Last Layer

Activation function which is used at last output layer is different from the activation functions that has been used in hidden dense layers as their main task is to introduce non-linearity. Activation function at last layer is used to categorize that means it gives probability of each class. If there are two classes, then sigmoid function is used. But in this project, there are more than two categories. So, in this situation softmax function is used which will normalize in such a way that each probability value lies in range of 0-1 and final sum will be 1. Below is the figure showing output of a sigmoid function.

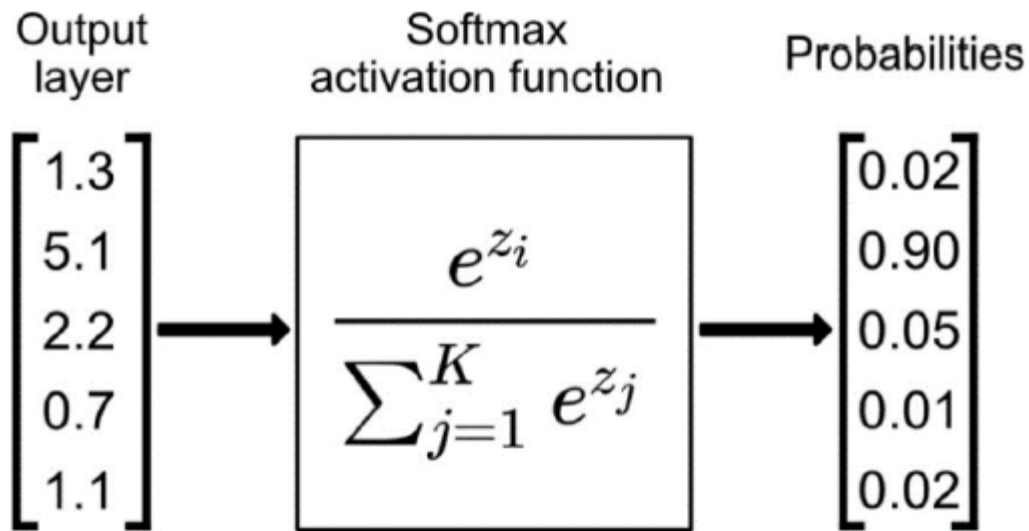


Fig -3.41 Figure showing softmax activation function.

In forward propagation all the weights are filtered & bias are calculated. Then they are validated by the cost function which may give error. Error is the change b/w definite label and projected label. In back propagation weights, filters & bias updation occurs. This updation is carried out by an optimizer which is an algorithm. Its task is to find values of weights, filters and bias is such a way that it will reduce its loss. There are many optimizers available. One which I am using is adam.

Loss Function

The loss function is used to verify the results. Shows the change b/w the actual & predicted labels. If chronologies are passed in batches or all records, the loss function is called the Cost Function. Different types of cost functions like Multiclass cross-entropy & sparse category cross-entropy. Disparate loss functions gives disparate results. Therefore, you should choose the loss function depending on your use case.

Binary Cross Entropy

$$\text{Loss} = -\frac{1}{\text{output size}} \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

Fig -3.42 Figure showing Binary cross entropy loss equation.

Multi Class Cross Entropy

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

Fig -3.43 Figure showing Multi cross entropy loss equation.

Optimizers

Optimizers are algorithms which are used to minimize the loss by updating weights and bias. Optimizers are of different types. The optimizer which we are using is referred to as Adam optimizer.

Adam Optimizer

The Adam optimizer is an advanced procedure. Gradient speed rises.

This is very effectual when working with very large data sets. It uses both momentum concepts. Like Exponentially weighted average and dynamically varying learning rate. It's very fast and resource efficient. It combines gradient descent with momentum & root mean square propagation.

Momentum Concept

$$w_{t+1} = w_t - \alpha m_t$$

where,

$$m_t = \beta m_{t-1} + (1 - \beta) \left[\frac{\delta L}{\delta w_t} \right] \quad \dots\dots\dots 1$$

$m_{\{t\}}$ = aggregate of gradients at time t [current] (initially, $m_{\{t\}} = 0$)

$m (t - 1)$ = aggregate of gradients at time t - 1 [previous]

$W_{\{t\}}$ = weights at time t

$W (t + 1)$ = weights at time t + 1

$a_{\{t\}}$ = learning rate at time t

partial L = derivative of Loss Function

partial $W_{\{t\}}$ = derivative of weights at time t

β = Moving average parameter

RMS Prop

$$w_{t+1} = w_t - \frac{\alpha_t}{(v_t + \epsilon)^{1/2}} * \left[\frac{\delta L}{\delta w_t} \right]$$

where,

$$v_t = \beta v_{t-1} + (1 - \beta) * \left[\frac{\delta L}{\delta w_t} \right]^2 \quad \dots\dots\dots 2$$

On Combining 1 and 2, we get

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \left[\frac{\delta L}{\delta w_t} \right] \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left[\frac{\delta L}{\delta w_t} \right]^2$$

MATHEMATICAL

Mathematics involved in Back Propagation

Backpropagation attempts to reduce the value of the loss function by updating the weights and biases. We try to find gradient descent for a function in an optimized way so that the function reaches the global minimum early and does not get stuck at local minima or saddle points. Update the weights and biases using a set of derivatives. Parameter updates in backpropagation are mainly done through derivatives.

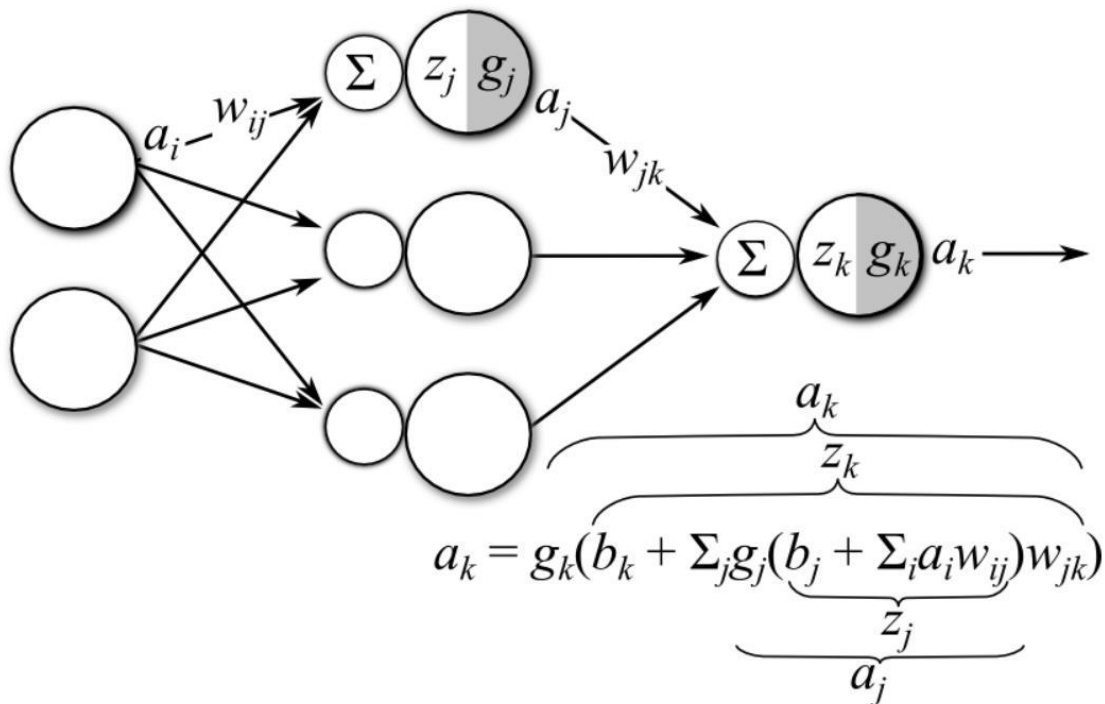


Fig- 3.44 Represents Back propagation Mathematics.

Chaining Rule If you have a function where the variable is also a function, in this case the chaining rule is used to calculate the derivative. Suppose we want to differentiate a function C with respect to W. However, C is a function of Z, and Z is a function of W.

In such cases, the chain rule of differentiation is used as shown below.

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial z} \frac{\partial z}{\partial w} \dots\dots\dots 1$$

Suppose if C is a function of number of variables Z. Those variables are function of variable W then in such situation chain rule becomes additive as shown below.

$$\frac{\partial C}{\partial w} = \sum_{i=1}^N \frac{\partial C}{\partial z_i} \frac{\partial z_i}{\partial w} \dots\dots\dots 2$$

If you want to train a model using gradient descent, you need to know the derivative of each parameter with respect to the loss function.

$$\frac{\partial C}{\partial W^m} \text{ and } \frac{\partial C}{\partial b^m}$$

3.6 KEY CHALLENGES

1. Imbalanced dataset:

Challenge: Limited number of late blight cases compared to healthy plants can lead to biased model predictions.

Solution: Use data augmentation techniques, during training, use methods such as oversampling, or weighted loss functions to balance the class representation.

2. Annotated Data Limitations:

Challenge: Interpreting large datasets with expert labels can be resource rigorous.

Solution: Effort with subject matter experts to ensure precise annotations.

3. Environmental Factors:

Challenge: Changes in environmental conditions, lighting, and background can cause noise.

Solution: Apply robust preprocessing techniques to standardize image conditions &

use image magnification to simulate different environmental scenarios.

4. Compute Resources:

Challenge: Training a deep CNN can require significant computational power.

Solution: Optimize model architecture for efficiency & leverage cloud-based solutions for scalable processing.

5. Field Deployment:

Challenges: When implementing a model in a real-world scenario, you may encounter issues related to hardware limitations and user acceptance.

Solution: Develop a lightweight model suitable for edge computing & it offers a user-friendly interface & clear documentation.

TESTING

4.1 TESTING STRATEGIES

The testing strategy for potato late blight classification using convolutional neural networks (CNN) includes evaluation of model performance, generalization, and robustness.

Here are some key testing strategies you can consider.

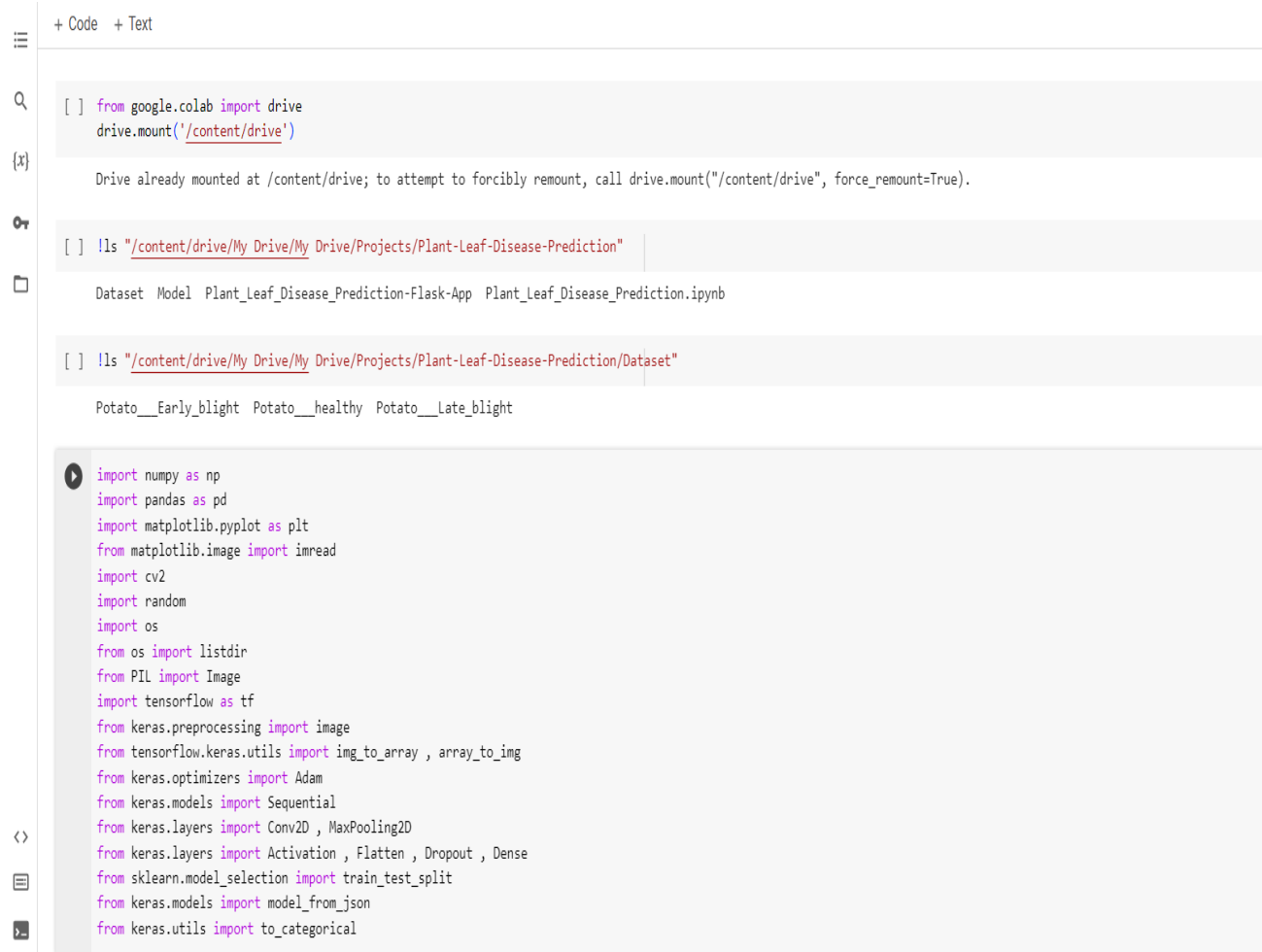
1. **Data Split:** Split the data set into training set, validation set, and test set.

The training set is used to train the model, the validation set is used to tune hyperparameter, & the test set is used to evaluate the concluding performance of the model.

2. **Data Augmentation:** This techniques during training to artificially increase the diversity of the training set. This may also include random rotations, reflections, & zooms. Evaluate how well the model generalizes to the unknown amplified data.
3. **Transfer Learning:** Using a CNN model pertained on a large dataset (such as ImageNet) as a starting point. Optimize the retrained model for the potato blight dataset. This can also potentially improve performance, especially when labeled data is limited.
4. **Tuning hyperparameter:** Trial with different hyperparameter, counting learning rate, batch size, & model architecture. Use the validation set to tune these hyperparameter to evade overfitting.
5. **Confusion Matrices & Metrics:** Confusion metrices is to understand how well your model classifies different categories.
6. **Cross-validation:** If your data set is limited, consider using k-fold cross-validation to evaluate model performance. This involves splitting the dataset into k folds and training/evaluating the model k times, each time using a different fold for validation.
7. **Adversarial Testing:** Introduce adversarial samples to test the robustness of the model. These are inputs that can easily be distorted to mislead the model. Evaluate how well the model

handles such disturbances.

8. **Ensemble Method:** Ensemble of multiple CNN models with different architectures or initializations. Combine predictions to improve overall performance and robustness.
9. **Real-World Testing:** Appraise model in real-world scenarios by capturing images in different lighting, weather conditions, & environments, evaluating performance in different conditions.



```
+ Code + Text

[ ] from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

[ ] !ls "/content/drive/My Drive/My Drive/Projects/Plant-Leaf-Disease-Prediction"

Dataset Model Plant_Leaf_Disease_Prediction-Flask-App Plant_Leaf_Disease_Prediction.ipynb

[ ] !ls "/content/drive/My Drive/My Drive/Projects/Plant-Leaf-Disease-Prediction/Dataset"

Potato__Early_blight Potato__healthy Potato__Late_blight

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.image import imread
import cv2
import random
import os
from os import listdir
from PIL import Image
import tensorflow as tf
from keras.preprocessing import image
from tensorflow.keras.utils import img_to_array, array_to_img
from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Flatten, Dropout, Dense
from sklearn.model_selection import train_test_split
from keras.models import model_from_json
from keras.utils import to_categorical
```

Fig -4.1 Figure showing drive mounted on google collaboratory & importing necessary python libraries

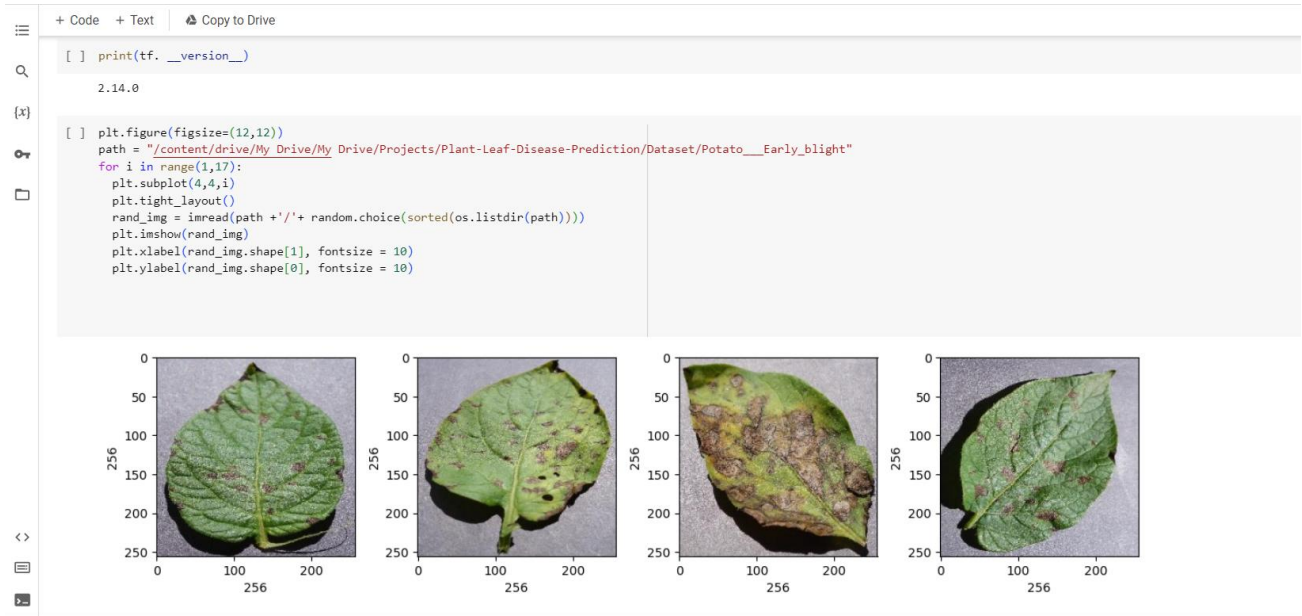


Fig -4.2 Figure showing converting image to array.

```
+ Code + Text Copy to Drive

[ ] dir = "/content/drive/My Drive/My Drive/Projects/Plant-Leaf-Disease-Prediction/Dataset"
image_list , label_list = [] , []
all_labels = ['Potato-Early_blight' , 'Potato-healthy' , 'Potato-Late_blight']
binary_labels = [0,1,2]
temp = -1

for directory in ['Potato__Early_blight' , 'Potato__healthy' , 'Potato__Late_blight']:
    plant_image_list = listdir(f"{dir}/{directory}")
    temp += 1
    for files in plant_image_list:
        image_path = f"{dir}/{directory}/{files}"
        image_list.append(convert_image_to_array(image_path))
        label_list.append(binary_labels[temp])

[ ] label_counts = pd.DataFrame(label_list).value_counts()
label_counts.head()

2    1000
0     174
1     152
dtype: int64

[ ] image_list[0].shape

(256, 256, 3)

[ ] x_train, x_test, y_train, y_test = train_test_split(image_list, label_list, test_size=0.2, random_state = 10)

x_train = np.array(x_train, dtype=np.float16) / 255.0
x_test = np.array(x_test, dtype=np.float16) / 255.0
x_train = x_train.reshape(-1, 256,256,3)
```

Fig -4.3 Figure showing the label counts and train/test dataset.

```

x_train = np.array(x_train, dtype=np.float16) / 255.0
x_test = np.array(x_test, dtype=np.float16) / 255.0
x_train = x_train.reshape(-1, 256,256,3)

x_test=x_test.reshape(-1, 256,256,3)

[ ] y_train = to_categorical(y_train)

y_test = to_categorical(y_test)

[ ] model = Sequential()

model.add(Conv2D(32, (3, 3), padding="same", input_shape=(256,256,3), activation="relu"))

model.add(MaxPooling2D(pool_size=(3, 3)))

model.add(Conv2D(16, (3, 3), padding="same", activation="relu"))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(8, activation="relu"))
model.add(Dense(3, activation="softmax"))

model.summary()

Model: "sequential"

```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 256, 256, 32)	896

Fig -4.4 Figure showing model summary.

```

[ ]
Layer (type)                Output Shape                Param #
-----
conv2d (Conv2D)              (None, 256, 256, 32)       896
max_pooling2d (MaxPooling2D) (None, 85, 85, 32)         0
conv2d_1 (Conv2D)            (None, 85, 85, 16)         4624
max_pooling2d_1 (MaxPooling2D) (None, 42, 42, 16)         0
flatten (Flatten)            (None, 28224)              0
dense (Dense)                (None, 8)                  225800
dense_1 (Dense)              (None, 3)                  27
-----
Total params: 231347 (903.70 KB)
Trainable params: 231347 (903.70 KB)
Non-trainable params: 0 (0.00 Byte)

[ ] model.compile(loss = 'categorical_crossentropy', optimizer = Adam (0.0001), metrics=['accuracy'])

[ ] x_train, x_val, y_train, y_val= train_test_split(x_train, y_train, test_size = 0.2, random_state = 10)

[ ] epochs = 50

batch_size = 128

history = model.fit(x_train, y_train, batch_size = batch_size, epochs = epochs, validation_data = (x_val, y_val))

```

Fig -4.5 Figure showing compiling the model.



```
+ Code + Text Copy to Drive

epochs = 50

batch_size = 128

history = model.fit(x_train, y_train, batch_size = batch_size, epochs = epochs, validation_data = (x_val, y_val))

Epoch 1/50
7/7 [=====] - 47s 7s/step - loss: 1.0964 - accuracy: 0.6498 - val_loss: 1.0903 - val_accuracy: 0.7594
Epoch 2/50
7/7 [=====] - 47s 7s/step - loss: 1.0827 - accuracy: 0.7535 - val_loss: 1.0672 - val_accuracy: 0.7594
Epoch 3/50
7/7 [=====] - 48s 7s/step - loss: 1.0518 - accuracy: 0.7535 - val_loss: 1.0227 - val_accuracy: 0.7594
Epoch 4/50
7/7 [=====] - 43s 6s/step - loss: 0.9994 - accuracy: 0.7535 - val_loss: 0.9536 - val_accuracy: 0.7594
Epoch 5/50
7/7 [=====] - 48s 7s/step - loss: 0.9228 - accuracy: 0.7535 - val_loss: 0.8650 - val_accuracy: 0.7594
Epoch 6/50
7/7 [=====] - 46s 7s/step - loss: 0.8379 - accuracy: 0.7535 - val_loss: 0.7765 - val_accuracy: 0.7594
Epoch 7/50
7/7 [=====] - 46s 7s/step - loss: 0.7610 - accuracy: 0.7535 - val_loss: 0.7252 - val_accuracy: 0.7594
Epoch 8/50
7/7 [=====] - 48s 7s/step - loss: 0.7382 - accuracy: 0.7535 - val_loss: 0.7194 - val_accuracy: 0.7594
Epoch 9/50
7/7 [=====] - 44s 7s/step - loss: 0.7375 - accuracy: 0.7535 - val_loss: 0.7210 - val_accuracy: 0.7594
Epoch 10/50
7/7 [=====] - 47s 7s/step - loss: 0.7378 - accuracy: 0.7535 - val_loss: 0.7189 - val_accuracy: 0.7594
Epoch 11/50
7/7 [=====] - 46s 7s/step - loss: 0.7341 - accuracy: 0.7535 - val_loss: 0.7179 - val_accuracy: 0.7594
Epoch 12/50
7/7 [=====] - 46s 7s/step - loss: 0.7339 - accuracy: 0.7535 - val_loss: 0.7191 - val_accuracy: 0.7594
Epoch 13/50
7/7 [=====] - 49s 7s/step - loss: 0.7342 - accuracy: 0.7535 - val_loss: 0.7194 - val_accuracy: 0.7594
Epoch 14/50
7/7 [=====] - 46s 7s/step - loss: 0.7341 - accuracy: 0.7535 - val_loss: 0.7190 - val_accuracy: 0.7594
Epoch 15/50
7/7 [=====] - 47s 7s/step - loss: 0.7340 - accuracy: 0.7535 - val_loss: 0.7182 - val_accuracy: 0.7594
```



```
+ Code + Text Copy to Drive

Epoch 40/50
7/7 [=====] - 47s 7s/step - loss: 0.7313 - accuracy: 0.7535 - val_loss: 0.7180 - val_accuracy: 0.7594
Epoch 41/50
7/7 [=====] - 47s 7s/step - loss: 0.7314 - accuracy: 0.7535 - val_loss: 0.7180 - val_accuracy: 0.7594
Epoch 42/50
7/7 [=====] - 49s 7s/step - loss: 0.7309 - accuracy: 0.7535 - val_loss: 0.7175 - val_accuracy: 0.7594
Epoch 43/50
7/7 [=====] - 49s 7s/step - loss: 0.7309 - accuracy: 0.7535 - val_loss: 0.7174 - val_accuracy: 0.7594
Epoch 44/50
7/7 [=====] - 51s 7s/step - loss: 0.7309 - accuracy: 0.7535 - val_loss: 0.7174 - val_accuracy: 0.7594
Epoch 45/50
7/7 [=====] - 48s 7s/step - loss: 0.7306 - accuracy: 0.7535 - val_loss: 0.7174 - val_accuracy: 0.7594
Epoch 46/50
7/7 [=====] - 47s 7s/step - loss: 0.7306 - accuracy: 0.7535 - val_loss: 0.7174 - val_accuracy: 0.7594
Epoch 47/50
7/7 [=====] - 49s 7s/step - loss: 0.7303 - accuracy: 0.7535 - val_loss: 0.7175 - val_accuracy: 0.7594
Epoch 48/50
7/7 [=====] - 47s 7s/step - loss: 0.7304 - accuracy: 0.7535 - val_loss: 0.7175 - val_accuracy: 0.7594
Epoch 49/50
7/7 [=====] - 47s 7s/step - loss: 0.7302 - accuracy: 0.7535 - val_loss: 0.7170 - val_accuracy: 0.7594
Epoch 50/50
7/7 [=====] - 47s 6s/step - loss: 0.7302 - accuracy: 0.7535 - val_loss: 0.7167 - val_accuracy: 0.7594
7/7 [=====] - 58s 9s/step - loss: 0.7303 - accuracy: 0.7535 - val_loss: 0.7170 - val_accuracy: 0.7594

[ ] model.save("/content/drive/My Drive/My Drive/Projects/Plant-Leaf-Disease-Prediction/Model/plant_disease_model.h5")

/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using `saving_api.save_model`

[ ] plt.figure(figsize=(12, 5))
plt.plot(history.history['accuracy'], color='r')
```

Fig -4.6 Figure showing 50 epochs.

OUTCOME:

Model Accuracy - 75.18%.

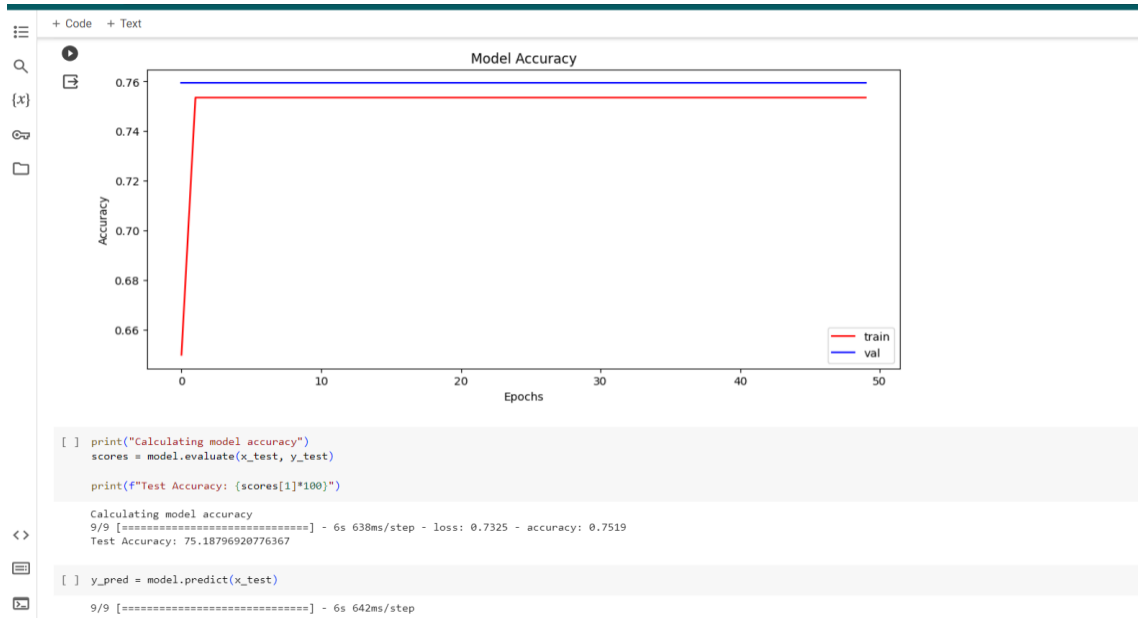


Fig -4.7 Figure shows accuracy with plotted graph.



Fig -4.8 Figure shows original and predicted label.

RESULTS AND EVALUATION

```
def predict(model,img):
    img_array=tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    img_array=tf.expand_dims(img_array,0)
    predictions = model.predict(img_array)
    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100*(np.max(predictions[0])),2)
    return predicted_class,confidence

plt.figure(figsize=(15,15))
for images,labels in test_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3,3,i+1)
        plt.imshow(images[i].numpy().astype("uint8"))
        predicted_class,confidence = predict(model,images[i].numpy())
        actual_class = class_names[labels[i]]
        plt.title(f"Actual:{actual_class},\n Predicted: {predicted_class}.\n Confidence:{confidence}%")
        plt.axis("off")
```

Fig -5.1 Figure shows model predicting the outcome with confidence.

RESULT:

```
1/1 [=====] - 0s 58ms/step
1/1 [=====] - 0s 39ms/step
1/1 [=====] - 0s 50ms/step
1/1 [=====] - 0s 58ms/step
1/1 [=====] - 0s 56ms/step
1/1 [=====] - 0s 62ms/step
1/1 [=====] - 0s 41ms/step
1/1 [=====] - 0s 59ms/step
1/1 [=====] - 0s 47ms/step
```

Actual:Potato_Early_blight,
 Predicted: Potato_Early_blight.
 Confidence:100.0%



Actual:Potato_Late_blight,
 Predicted: Potato_Late_blight.
 Confidence:100.0%



Actual:Potato_Late_blight,
 Predicted: Potato_Late_blight.
 Confidence:100.0%



Predictions which we got after applying predict function are shown below:



Fig -5.2 Figure shows Predictions which we got after applying predict function.

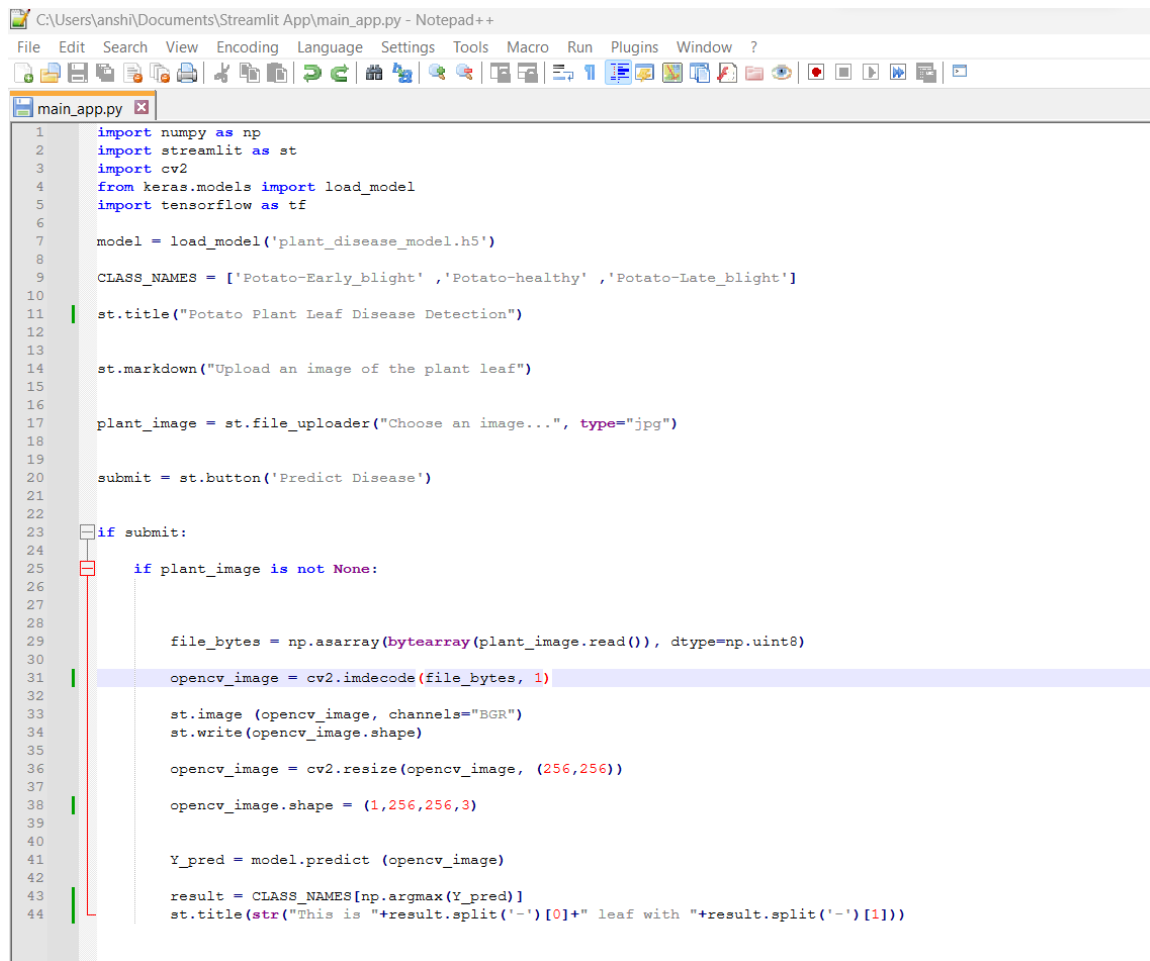
Here, above is the output which shows that what was actual image, and what image is classified into after the prediction is done, along with the confidence.

Below is the code required for making the Streamlit application along with procedure done on Anaconda prompt to get the desired web application where we can drop/drag image and can predict the respective disease a potato plant leaf is having.

Streamlit App python file-

For creating the web application using streamlit python code has been used which will manage backend of the web applicaion.

Below is the implemented code:



```
1 import numpy as np
2 import streamlit as st
3 import cv2
4 from keras.models import load_model
5 import tensorflow as tf
6
7 model = load_model('plant_disease_model.h5')
8
9 CLASS_NAMES = ['Potato-Early_blight', 'Potato-healthy', 'Potato-Late_blight']
10
11 st.title("Potato Plant Leaf Disease Detection")
12
13
14 st.markdown("Upload an image of the plant leaf")
15
16
17 plant_image = st.file_uploader("Choose an image...", type="jpg")
18
19
20 submit = st.button('Predict Disease')
21
22
23 if submit:
24
25     if plant_image is not None:
26
27
28
29         file_bytes = np.asarray(bytearray(plant_image.read()), dtype=np.uint8)
30
31         opencv_image = cv2.imdecode(file_bytes, 1)
32
33         st.image(opencv_image, channels="BGR")
34         st.write(opencv_image.shape)
35
36         opencv_image = cv2.resize(opencv_image, (256,256))
37
38         opencv_image.shape = (1,256,256,3)
39
40
41         Y_pred = model.predict(opencv_image)
42
43         result = CLASS_NAMES[np.argmax(Y_pred)]
44         st.title(str("This is "+result.split('-')[0]+" leaf with "+result.split('-')[1]))
```

Fig- 5.3 Streamlit App python file

Anaconda prompt for writing the commands to run the application.


```
(base) C:\Users\anshi>cd C:\Users\anshi\Documents\Streamlit App

(base) C:\Users\anshi\Documents\Streamlit App>pip install pipreqs
Requirement already satisfied: pipreqs in c:\users\anshi\anaconda3\lib\site-packages (0.4.13)
Requirement already satisfied: yarg in c:\users\anshi\anaconda3\lib\site-packages (from pipreqs) (0.1.9)
Requirement already satisfied: doctest in c:\users\anshi\anaconda3\lib\site-packages (from pipreqs) (0.6.2)
Requirement already satisfied: requests in c:\users\anshi\anaconda3\lib\site-packages (from yarg->pipreqs) (2.27.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\anshi\anaconda3\lib\site-packages (from requests->yarg->pipreqs) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\anshi\anaconda3\lib\site-packages (from requests->yarg->pipreqs) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\anshi\anaconda3\lib\site-packages (from requests->yarg->pipreqs) (2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\anshi\anaconda3\lib\site-packages (from requests->yarg->pipreqs) (1.26.9)
```

Fig -5.4 Anaconda prompt for writing the commands

```
(base) C:\Users\anshi\Documents\Streamlit App>pipreqs
WARNING: requirements.txt already exists, use --force to overwrite it

(base) C:\Users\anshi\Documents\Streamlit App>pip install -r requirements.txt
Requirement already satisfied: keras==2.12.0 in c:\users\anshi\anaconda3\lib\site-packages (from -r requirements.txt (line 1)) (2.12.0)
Requirement already satisfied: numpy==1.23.5 in c:\users\anshi\anaconda3\lib\site-packages (from -r requirements.txt (line 2)) (1.23.5)
Requirement already satisfied: streamlit==1.22.0 in c:\users\anshi\anaconda3\lib\site-packages (from -r requirements.txt (line 3)) (1.22.0)
Requirement already satisfied: opencv_python==4.6.0.66 in c:\users\anshi\anaconda3\lib\site-packages (from -r requirements.txt (line 4)) (4.6.0.66)
Requirement already satisfied: tensorflow==2.12.0 in c:\users\anshi\anaconda3\lib\site-packages (from -r requirements.txt (line 5)) (2.12.0)
Requirement already satisfied: click>=7.0 in c:\users\anshi\anaconda3\lib\site-packages (from streamlit==1.22.0->-r requirements.txt (line 3)) (8.0.4)
Requirement already satisfied: python-dateutil in c:\users\anshi\anaconda3\lib\site-packages (from streamlit==1.22.0->-r requirements.txt (line 3)) (2.8.2)
Requirement already satisfied: pyarrow>=4.0 in c:\users\anshi\anaconda3\lib\site-packages (from streamlit==1.22.0->-r requirements.txt (line 3)) (14.0.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\anshi\anaconda3\lib\site-packages (from streamlit==1.22.0->-r requirements.txt (line 3)) (9.0.1)
Requirement already satisfied: tenacity<9,>=8.0.0 in c:\users\anshi\anaconda3\lib\site-packages (from streamlit==1.22.0->-r requirements.txt (line 3)) (8.0.1)
Requirement already satisfied: protobuf<4,>=3.12 in c:\users\anshi\anaconda3\lib\site-packages (from streamlit==1.22.0->-r requirements.txt (line 3)) (3.20.3)
Requirement already satisfied: rich>=10.11.0 in c:\users\anshi\anaconda3\lib\site-packages (from streamlit==1.22.0->-r requirements.txt (line 3)) (13.7.0)
Requirement already satisfied: tzlocal>=1.1 in c:\users\anshi\anaconda3\lib\site-packages (from streamlit==1.22.0->-r requirements.txt (line 3)) (5.2)
Requirement already satisfied: pydeck>=0.1.dev5 in c:\users\anshi\anaconda3\lib\site-packages (from streamlit==1.22.0->-r requirements.txt (line 3)) (0.8.1)
Requirement already satisfied: packaging>=14.1 in c:\users\anshi\anaconda3\lib\site-packages (from streamlit==1.22.0->-r requirements.txt (line 3)) (21.3)
Requirement already satisfied: pypmpler>=0.9 in c:\users\anshi\anaconda3\lib\site-packages (from streamlit==1.22.0->-r requirements.txt (line 3)) (1.0.1)
Requirement already satisfied: pandas<3,>=0.25 in c:\users\anshi\anaconda3\lib\site-packages (from streamlit==1.22.0->-r requirements.txt (line 3)) (1.4.2)
Requirement already satisfied: gitpython=3.1.19 in c:\users\anshi\anaconda3\lib\site-packages (from streamlit==1.22.0->-r requirements.txt (line 3)) (3.1.19)
Requirement already satisfied: toml in c:\users\anshi\anaconda3\lib\site-packages (from streamlit==1.22.0->-r requirements.txt (line 3)) (0.10.2)
```

Fig-5.5 Installing -r requirement.txt.

```
>tensorboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow==2.12.0->-r requirements.txt (line 5)) (3.2.2)
Requirement already satisfied: tzdata in c:\users\anshi\anaconda3\lib\site-packages (from tzlocal>=1.1->streamlit==1.22.0->-r requirements.txt (line 3)) (2023.3)

(base) C:\Users\anshi\Documents\Streamlit App>streamlit run main_app.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://172.16.117.37:8501

1/1 [=====] - 0s 439ms/step
1/1 [=====] - 0s 170ms/step
2023-11-30 10:27:19.081 Uncaught app exception
Traceback (most recent call last):
```

Fig-5.6 Running the web application.

WEBSITE DEVELOPMENT:

STEP -1:

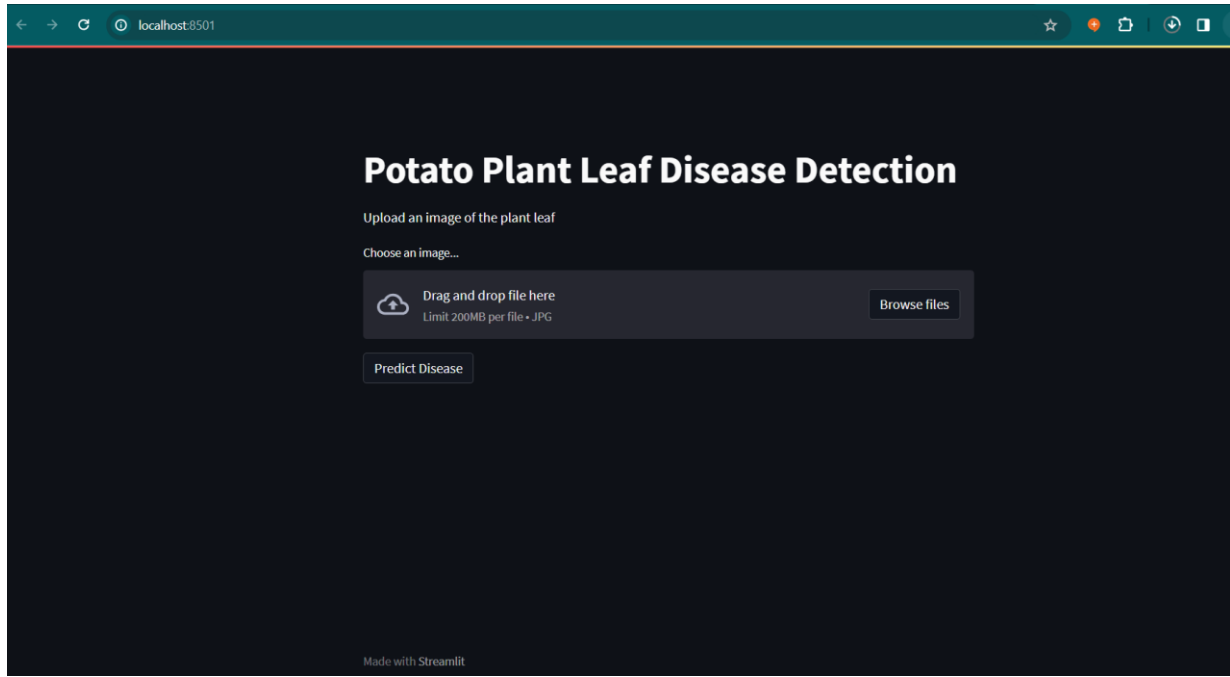


Fig-5.7 cover page of website.

STEP -2: Select the image for which you want to predict the disease.

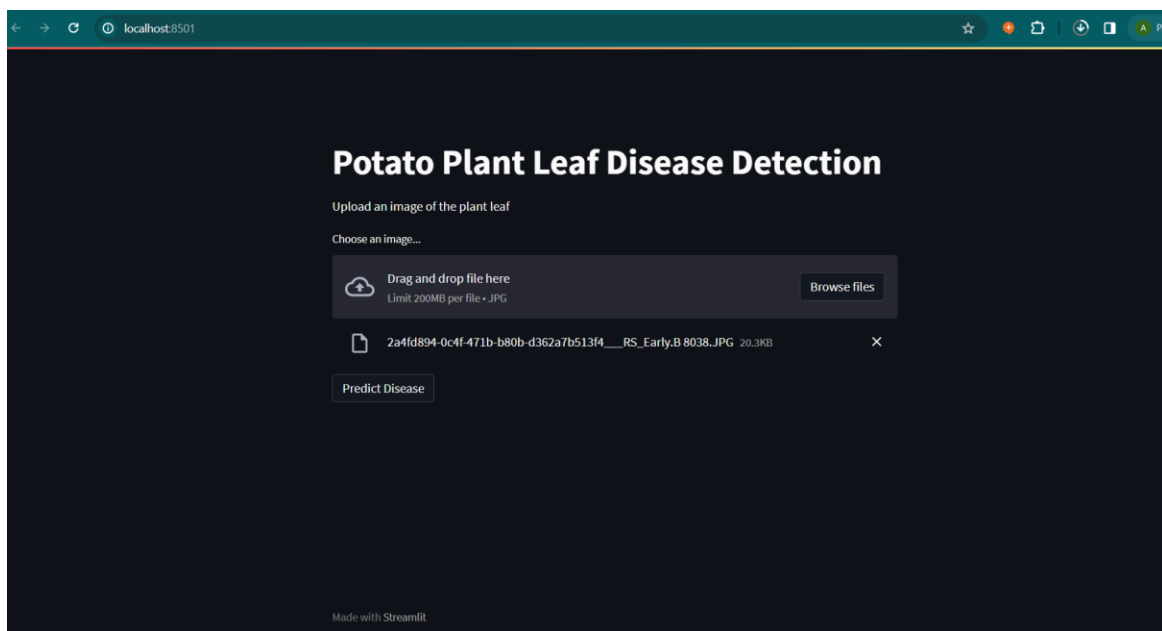


Fig-5.8 Selecting the image.

RESULTS:

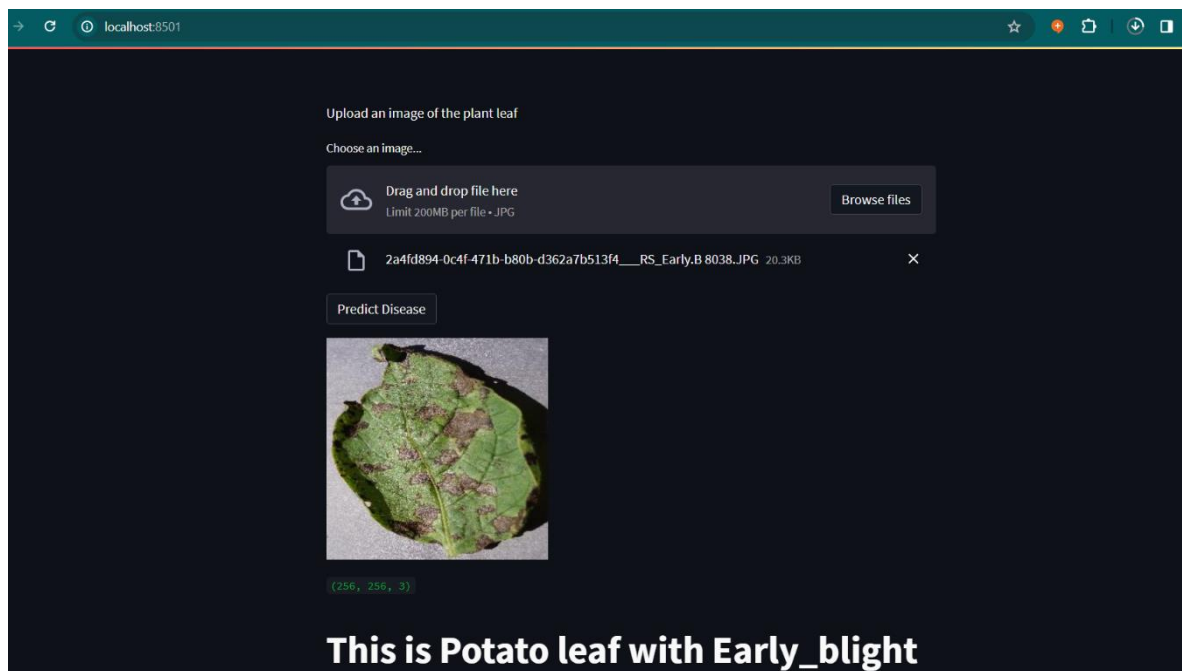


Fig- 5.9 Potato leaf is classified with disease Early Blight.

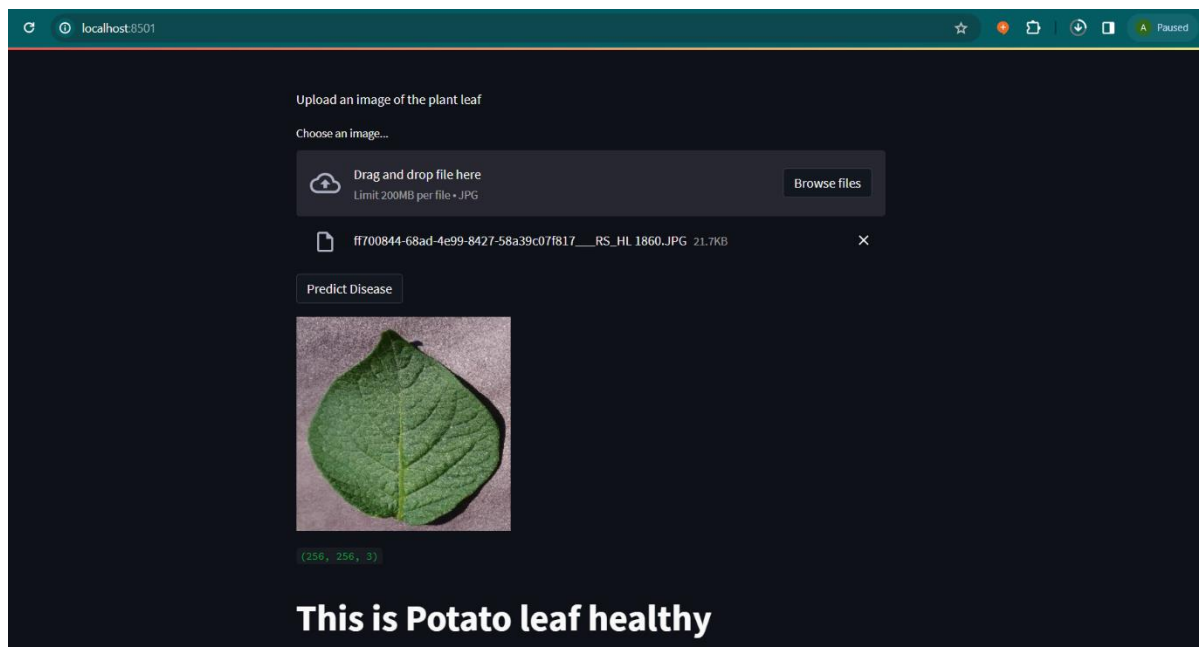


Fig- 5.9 Potato leaf is classified as healthy.

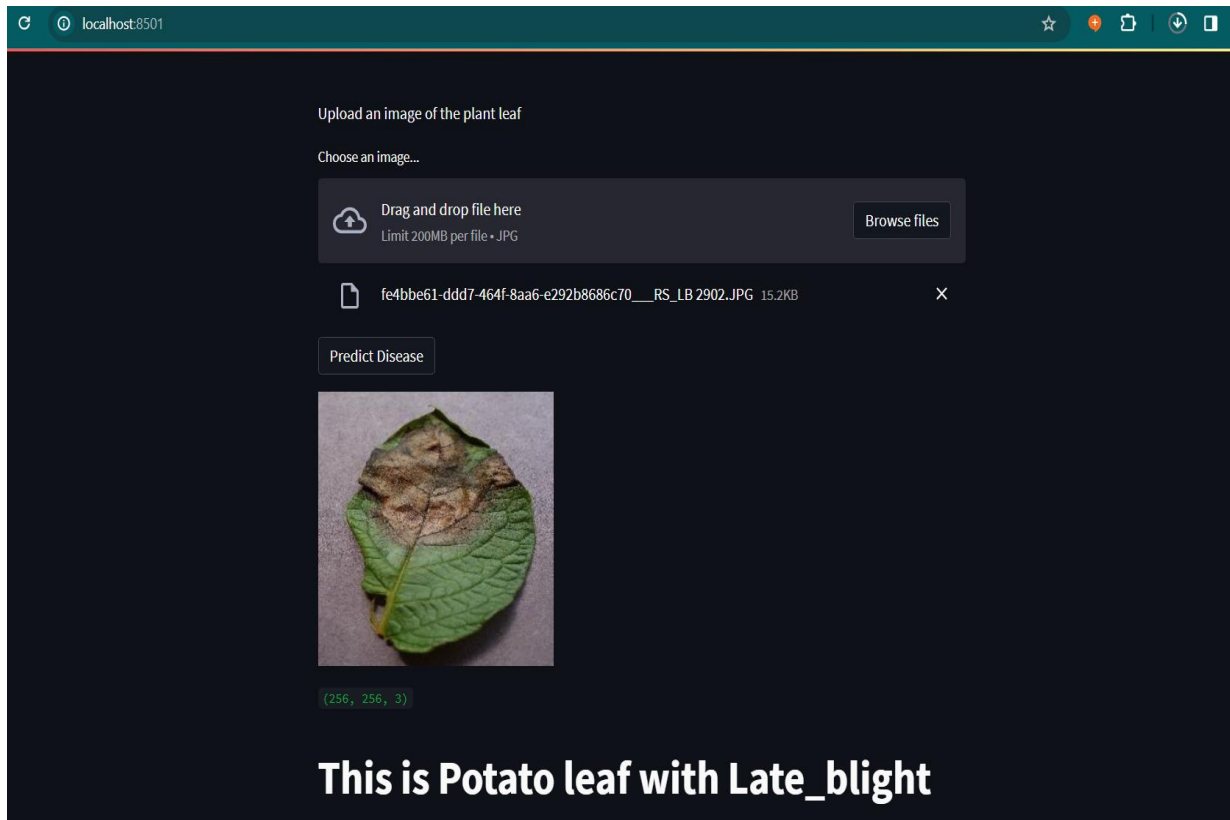



Fig- 5.10 Potato leaf is classified with disease Late Blight.

APP DEVELOPMENT:

STEP -1:

 **Namaste!**
Select your Planti language

उत्तराखण्ड भाषा चिंच धेडीघाडी

मराठी
स्वतःच्या भाषेत शेती

বাংলা
চাষাবাদের কথা আপনার ভাষায়

اردو
آپ کی زبان میں کاشتکاری

ಕನ್ನಡ
ನಿಮ್ಮ ಭಾಷೆಯಲ್ಲಿ ಕೃಷಿ

हिन्दी
खेती आपकी भाषा में

English

Accept

I read and accept the [terms of use](#) and the [privacy policy](#).



Instant Disease Detection

...

Next

i)

ii)

Fig- 5.11 Planti application initial stages (Choose preferable languages).

STEP -2:

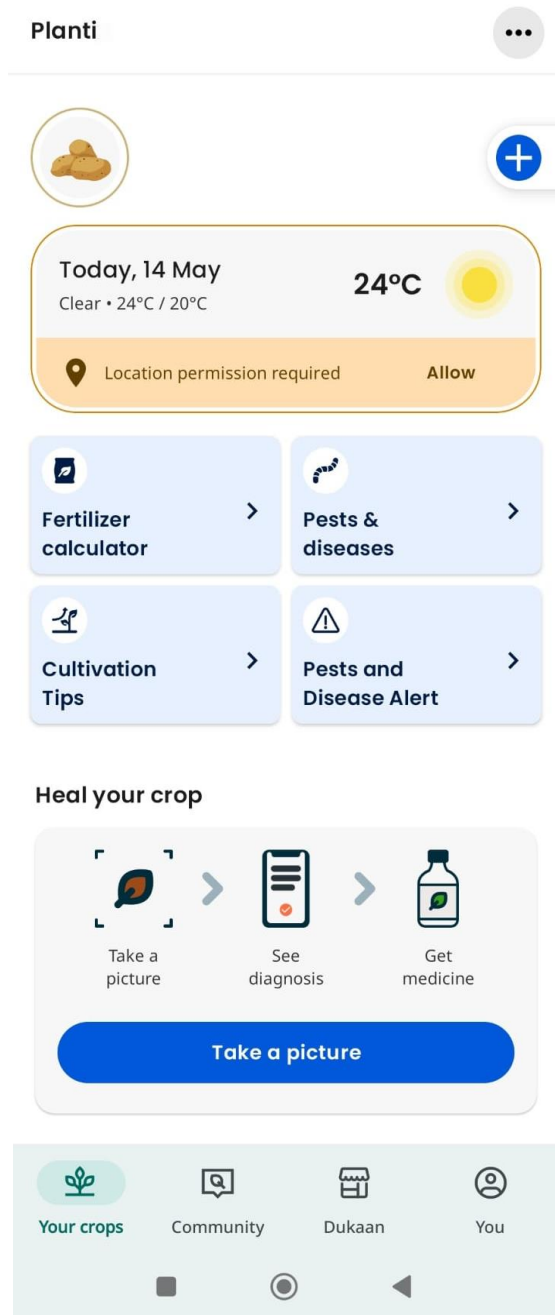


Fig- 5.12 Planti application cover page.

STEP -3:

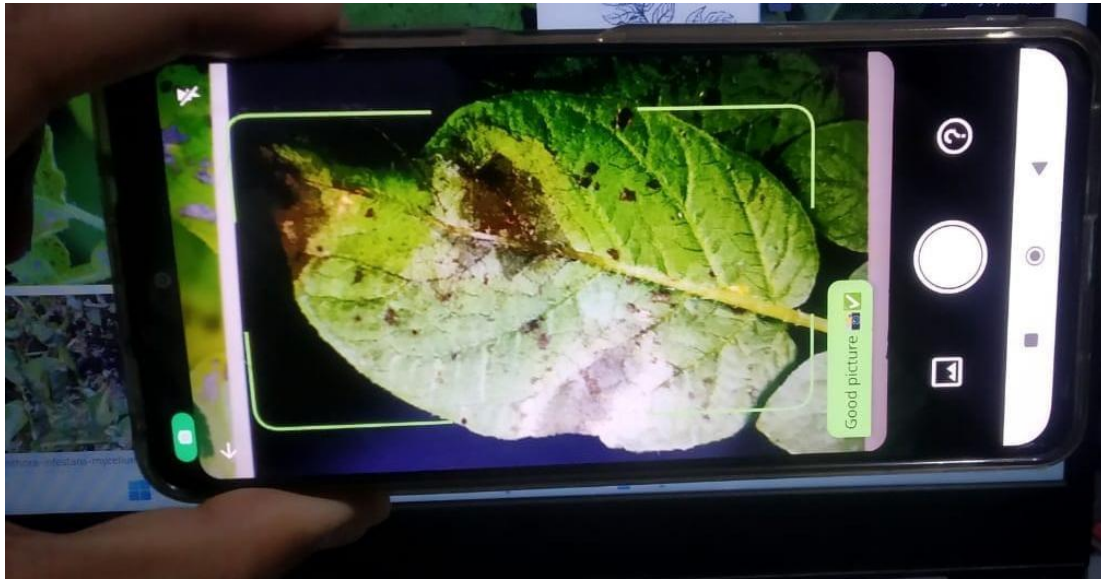


Fig- 5.13 Use option “Take a picture” and scan leaf.

STEP -4: For late blight leaf

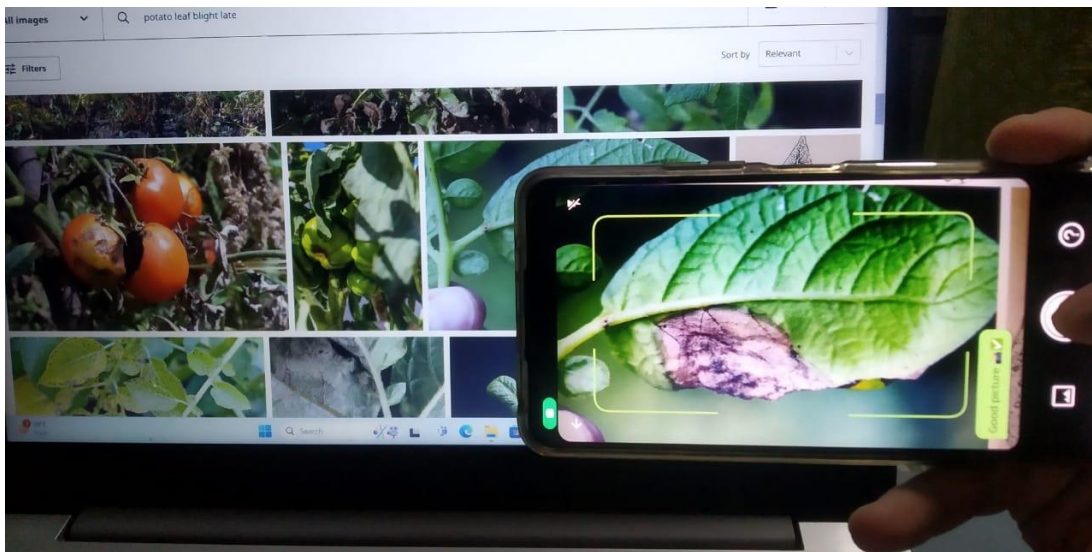


Fig- 5.14 Scan photo of late blight leaf.

RESULT :

The screenshot displays a mobile application interface for diagnosing a plant disease. On the left, a card titled "Potato Late Blight" is labeled "Fungus" and features a photograph of a potato leaf with brown necrotic spots. Below this, a "Symptoms" section lists: "Dark brown spots on leaf tips and margins," "Spots turn into transparent wounds," "White fungus covers the underside of leaves," "Leaves wilt and die off," and "Grayish-blue spots on potato tubers." A descriptive paragraph states: "Dark brown spots develop on the leaves starting at the tip or the leaf margins. In humid climates, these spots become water-soaked lesions. A white fungal covering can be seen on the underside of the leaves."

The right side of the interface is divided into two main sections:

- 1 Diagnosis result:** A card showing the identified disease "Potato Late Blight Fungus" with a right-pointing arrow.
- 2 Recommended products:** A section titled "Recommended by Plantix" with a note "Based on active ingredient and availability". It includes a filter for "Potato" and three product cards: "Antracol by Bayer" (marked as a "Plantix Pick"), "ROMILO by Dharmaj Crop Guar...", and "Jat by C".

A yellow warning banner at the bottom of the product section reads: "⚠ Select and apply ONLY ONE of the products to your crops."

Fig- 5.15 See diagnosis and Get medicine.

STEP -5: For healthy leaf



Fig- 5.16 Scan photo of healthy leaf.

RESULT:

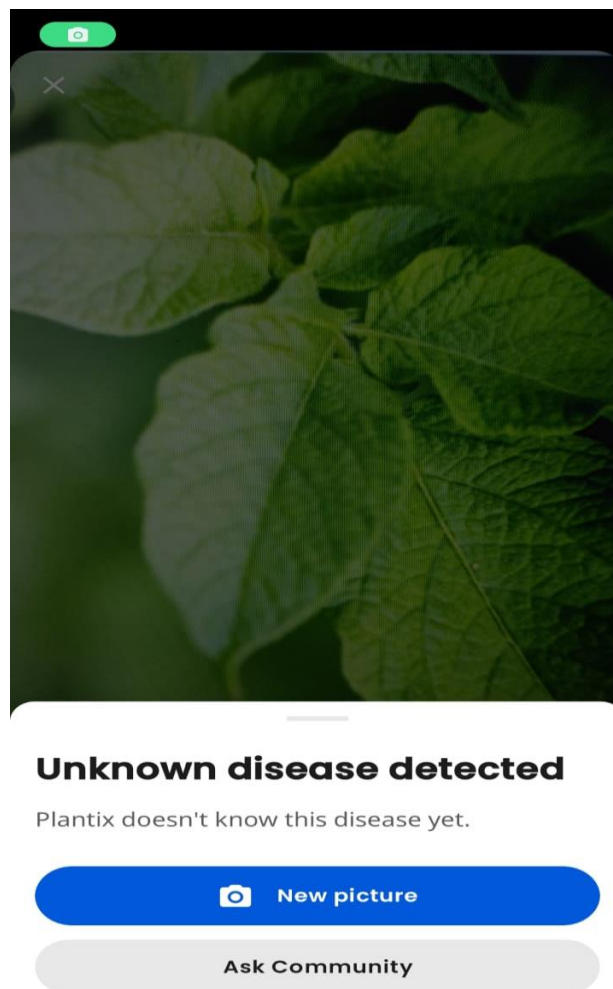


Fig- 5.17 No disease predicted.

STEP -6: For early blight leaf

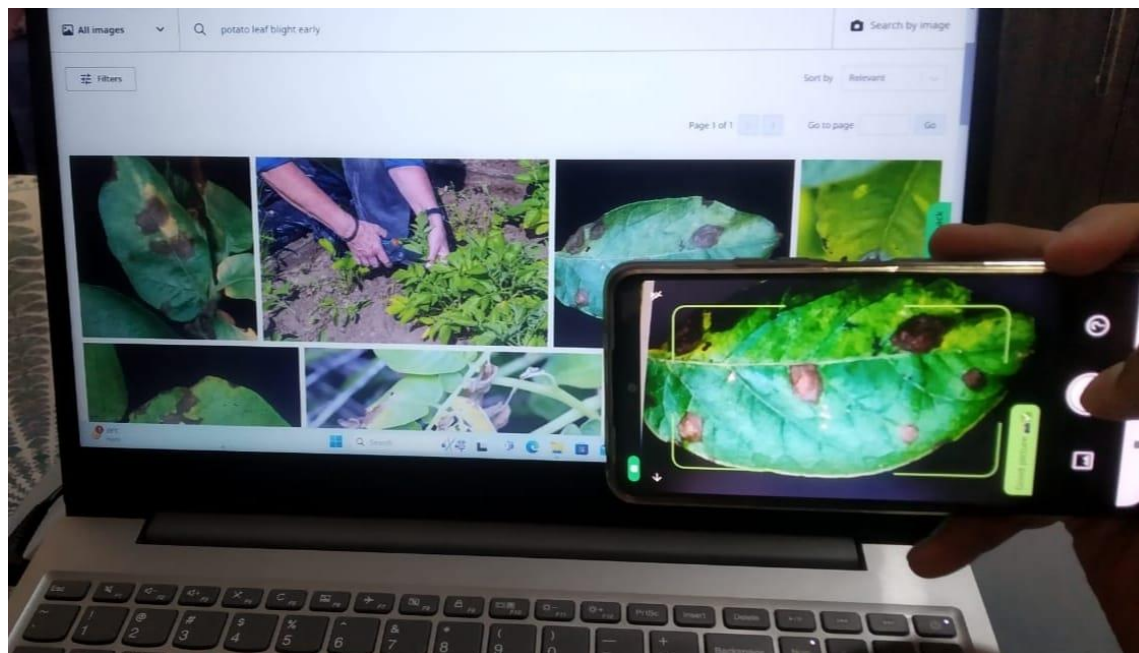



Fig- 5.18 Scan photo of early blight leaf.

RESULT:

← Diagnosis

Early Blight

Fungus



1 Diagnosis result

Early Blight
Fungus

2 Recommended products

Recommended by Planti
Based on active ingredient and availability

See product information on Potato

Symptoms

- Dark spots on leaves with circles around them and yellow rings around the spots.
- Leaves dry out and fall off.
- Fruits start to rot and fall off the plant.

If a plant has early blight, it will show symptoms on its older leaves, stem, and fruits. There will be gray or brown spots on the leaves that get bigger in circles around a clear center, which looks like a target (the characteristic “bullseye” formation). The spots are also surrounded by a bright yellow ring.

Products:

- KITAZU** by *Industries Ltd.*
- ISHAAN** by Tata India Pvt. Ltd.
- Jatayu** by Coromandel

⚠ Select and apply **ONLY ONE** of the products to your crops.

Fig- 5.19 See diagnosis and Get medicine.

CONCLUSION AND FUTURE SCOPE

6.1 SUMMARY

In summary, this study aimed to classify potato late blight using CNN. This is an critical step for early & accurate detection of diseases in agricultural environments. The results obtained with our model establish its effectiveness in differentiating between healthy & disease-affected potato plants. The CNN architecture demonstrated confide accuracy and highlighted the potential of deep learning approach in identifying plant diseases. The use of a well-curated dataset containing cases with a variety of epidemic severity greatly contributed to the fit of the model.

In this project, convolutional neural network architecture is used in the model. So, that it can be used to classify for web application. Image of a potato leaf will be given as an input then it will be able to classify whether plant is vigorous or effected by late blight & early blight. With little modifications, this project can prove very beneficial for farmers. Without monitoring on our own, we can easily identify the plant is infected or healthy. It will reduce the production cost as well as preventive measures can be taken early. This project can be modified so that it can be used for disease identification for other species as well. This can be done in real time and in a better way by using computer vision. The model accuracy we get is 75.18%.

This project highlights the implicit of deep learning to revolutionize disease administration in agriculture, and further advancement & validation is expected to pave over the way for actionable application in this field.

LIMITATIONS OF THE PROJECT

Recognizing the drawbacks of potato blight classification using CNNs is important to fully understand the scope of the project & areas for advancement. Limitations to consider are:

- 1. Limited environmental variation:** Model performance can be affected by specific environmental conditions in the training data set. If the training data is not diverse, it may be

difficult for the model to generalize to different conditions.

2. **Data imbalance:** Data set imbalance causes one class to be significantly more represented than the other resulting in a biased model. Ensuring a balanced dataset is important for training fair & accurate models.
3. **Lack of real-world validation:** Model performance in a controlled environment may not be fully applicable to real agricultural conditions. Field testing & validation in different geographic locations & climates is essential to ensure real-world effectiveness.
4. **Computational resource requirements:** CNNs can have high computational requirements, especially if they are deep and complex. This might pose challenges when implemented in resource-limited environments, such as field agriculture applications.
5. **Image quality dependence:** Performance of the model is affected by the excellence of the input images. Factors like lighting conditions, camera resolution, and angle can affect the model's ability to accurately classify potato late blight.
6. **Scalability Issues:** Scalability issues can occur when implementing large models that cover large agricultural areas. For practical applications, it is important to ensure the efficiency of the model when processing large amounts of data.

CONTRIBUTION OF THE PROJECT

The benefaction of the potato late blight classification project using CNN lies in its potential to transform agricultural practices, especially the early discovery and control of potato late blight.

Some of the key contributions are:

1. **Early detection and intervention:** This project will contribute to the early detection of late potato blight, allowing farmers to identify affected crops at an early stage. Early intervention can significantly reduce disease spread and deprecate crop losses.
2. **Precision Agriculture:** By using CNN, this project takes into account the principles of precision agriculture. Farmers can target specific areas affected by the disease, optimize the use of resources such as pesticides.
3. **Increasing crop yields & food security:** Correct classification of late potato blight could enable timely & targeted responses, potentially preserving crop yields. This contributes to food security by reducing losses & assure a more calculable potato harvest.
4. **Technology Integration in Agriculture:** This project demonstrates the integration of developed

technologies, particularly deep learning with CNNs, into classical agricultural practices.

5. **Data-driven decision-making:** This project elevates data-driven decision-making in agriculture. By providing reliable disease detection tools, farmers can make informed decisions based on the real-time status of their crops
6. **Reducing environmental impacts:** Applying targeted interventions based on model predictions can lead to reductions in pesticide use. This not only deprecate environmental impact but is also consistent with sustainable farming practices.
7. **Empowering farmers:** This project provides technology-driven solutions to farmers by providing automated disease detection tools. This can increase confidence in plant health management & enhance overall farm productivity.
8. **Contribution to agricultural innovation:** This project represents a contribution to proceeding innovation in agriculture. This shows the implicit for artificial intelligence, & deep learning in particular, to bring about revolutionary changes in the way plant health is monitored and managed.

APPLICATION OF THE PROJECT

Potato late blight classification using convolutional neural networks (CNN) has several valuable applications in agriculture and crop production.

Some of the most important applications are listed below.

- Early detection of disease.
- Plant monitoring.
- Yield improvement.
- Reduce environmental impact.
- Resource efficiency.
- Education & Outreach Activities.
- Research & Disease Surveillance.
- World food security.
- Customized treatment.
- Smart Agriculture.

6.2 FUTURE SCOPE

The future of potato late blight classification using CNNs includes increasing model versatility, absorb transfer learning, enabling real-time monitoring, exploring developed imaging techniques, leveraging crowdsourced data, and integrating into precision agriculture & fostering global cooperation for criterion and all-around in diverse agricultural environments.

We will try to improve factors like accuracy. A mobile application can be created which will make a call to FastAPI using cross origin resource sharing and will give the results.

REFERENCES

- [1]. Arshad, Fizzah, et al. "PLDPNet: End-to-end hybrid deep learning framework for potato leaf disease prediction." *Alexandria Engineering Journal* 78 (2023): 406-418.
- [2]. Y. P. Wasalwar, K. S. Bagga, V. K. Joshi and A. Joshi, "Potato Leaf Disease Classification using Convolutional Neural Networks," 2023 11th *International Conference on Emerging Trends in Engineering & Technology - Signal and Information Processing (ICETET - SIP)*, Nagpur, India, 2023, pp. 1-5, doi: 10.1109/ICETET-SIP58143.2023.10151591.
- [3]. Qi, Chao, et al. "In-field classification of the asymptomatic biotrophic phase of potato late blight based on deep learning and proximal hyperspectral imaging." *Computers and Electronics in Agriculture* 205 (2023): 107585.
- [4]. Al-Adhaileh, Mosleh Hmoud, et al. "Potato Blight Detection Using Fine-Tuned CNN Architecture." *Mathematics* 11.6 (2023): 1516.
- [5]. Chakraborty, Kulendu Kashyap, et al. "Automated recognition of optical image based potato leaf blight diseases using deep learning." *Physiological and Molecular Plant Pathology* 117 (2022): 101781.
- [6]. Baron, Marco Javier, Angie Lizeth Gomez, and Jorge Enrique Espindola Diaz. "Supervised Learning-Based Image Classification for the Detection of Late Blight in Potato Crops." *Applied Sciences* 12.18 (2022): 9371.
- [7]. Abdul Muiz, et al. "Leaf Blights Detection and Classification in Large Scale Applications." *Intelligent Automation & Soft Computing* 31.1 (2022).
- [8]. Johnson, J., G. Sharma, and S. Srinivasan. "Enhanced field-based detection of potato blight in complex backgrounds using deep learning." *Plant Phenomics* 2021: 1–13." (2021).
- [9]. Asif, Md Khalid Rayhan, Md Asfaqur Rahman, and Most Hasna Hena. "CNN based disease detection approach on potato leaves." *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*. IEEE, 2020.
- [10]. Rozaqi, Abdul Jalil, and Andi Sunyoto. "Identification of disease in potato leaves using Convolutional Neural Network (CNN) algorithm." *2020 3rd International Conference on Information and Communications Technology (ICOIACT)*. IEEE, 2020.

- www.youtube.com
- www.google.com
- www.Wikipedia.com
- www.kaggle.com

G16_Report.docx

ORIGINALITY REPORT

9%	6%	5%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	link.springer.com Internet Source	1%
2	Submitted to The University of Wolverhampton Student Paper	1%
3	drpress.org Internet Source	1%
4	Submitted to University of Bedfordshire Student Paper	1%
5	icas.lincoln.ac.uk Internet Source	<1%
6	Submitted to University of East London Student Paper	<1%
7	www.frontiersin.org Internet Source	<1%
8	www.researchgate.net Internet Source	<1%
9	Submitted to CSU Northridge Student Paper	<1%

10	www.sciencegate.app Internet Source	<1 %
11	zuscholars.zu.ac.ae Internet Source	<1 %
12	www.research-collection.ethz.ch Internet Source	<1 %
13	Abhinav Baranwal, Mehul Mishra, Aryan Goyal, Yogesh. "Potato Plant Disease Classification Through Deep Learning", 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON), 2022 Publication	<1 %
14	Submitted to Deakin University Student Paper	<1 %
15	Submitted to SP Jain School of Global Management Student Paper	<1 %
16	github.com Internet Source	<1 %
17	journal.50sea.com Internet Source	<1 %
18	www.ijisae.org Internet Source	<1 %
19	downloads.hindawi.com Internet Source	<1 %

		<1 %
20	0-www-mdpi-com.brum.beds.ac.uk Internet Source	<1 %
21	beta.vu.nl Internet Source	<1 %
22	dspace.lib.ntua.gr Internet Source	<1 %
23	www.journal.esrgroups.org Internet Source	<1 %
24	Kulendu Kashyap Chakraborty, Rashmi Mukherjee, Chandan Chakraborty, Kangkana Bora. "Automated recognition of optical image based potato leaf blight diseases using deep learning", Physiological and Molecular Plant Pathology, 2021 Publication	<1 %
25	iieta.org Internet Source	<1 %
26	neptjournal.com Internet Source	<1 %
27	qtanalytics.in Internet Source	<1 %
28	sciencedocbox.com Internet Source	<1 %

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 16/05/24

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: ANSHIT GOEL & ADITYA SHARMA Department: CSE & IT Enrolment No. 2011598201551

Contact No. 8053623345 E-mail: anshittab7@gmail.com

Name of the Supervisor: Dr. AMIT KUMAR, ASSISTANT PROFESSOR (SG)

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): POTATO BLIGHT CLASSIFICATION USING CNN

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 72
- Total No. of Preliminary pages = 5
- Total No. of pages accommodate bibliography/references = 6

Anshit Goel
(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found Similarity Index at 09 (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

[Signature]
(Signature of Guide/Supervisor)

[Signature]
Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
Report Generated on	<ul style="list-style-type: none"> • All Preliminary Pages • Bibliography/ma ges/Quotes • 14 Words String 	<u>09</u>	Word Counts	<u>6171</u>
			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck_juit@gmail.com