Plant Disease Classification

Team Name – Mlxmasters

GitHub - https://github.com/Shreyash1001/Plant-Disease-Classification

Mohd. Talib

Harshit Raizada

Tejus Kavishwar

Nandini Shukla

Shreyas Tare

G. Naveen

Abstract

Plant diseases pose significant threats to agricultural productivity, food security, and the overall health of plant populations. The implementation of efficient disease control methods depends on the prompt and correct detection of plant diseases. Recent developments in computer vision and machine learning approaches have demonstrated tremendous promise for automating the classification of plant diseases. This abstract presents an overview of the research and development in the field of plant disease classification using machine learning algorithms. The primary objective of this research is to develop automated systems that can accurately classify plant diseases based on visual symptoms observed on plant leaves, stems, fruits, or other plant parts. The development of plant disease classification systems holds great potential in assisting farmers and agricultural experts in early disease detection, accurate diagnosis, and timely interventions. Such systems can enable the implementation of targeted treatment strategies, reduce the excessive use of pesticides, and ultimately contribute to sustainable agriculture and food production.

Problem Statement

Global agriculture faces considerable obstacles from plant diseases, which influence crop productivity, quality, and total food output. Effective disease management and prompt application of control measures depend on the early identification and precise classification of plant diseases. However, manually identifying plant diseases through visual inspection can be laborious, arbitrary, and mistake prone. Therefore, it is imperative to create automated systems that can correctly categorise plant diseases based on their visual characteristics, facilitating quick and accurate diagnosis. The creation of a reliable and effective plant disease classification system that can correctly identify and categorise plant illnesses based on visual signals seen in photographs of plant parts, such as leaves, stems, or fruits, is the key issue addressed in this research.

Objectives

- **Early Detection:** Detecting plant diseases at an early stage is crucial for timely intervention and effective disease management. By accurately classifying diseases, plant disease classification systems can aid in early detection, allowing farmers and agronomists to implement appropriate control measures and prevent further spread.

- **Disease Management:** A thorough classification of plant diseases offers useful information for disease control measures. Farmers can use tailored treatments, such as picking the right fungicides or implementing cultural practises, to control the specific disease and reduce crop losses by identifying the exact disease that is infecting a plant.

- **Yield Protection:** Plant diseases can cause significant yield losses if not addressed promptly. The objective of plant disease classification is to protect crop yields by accurately identifying diseases and enabling farmers to take preventive or remedial actions to mitigate the impact on crop production.

- **Sustainable Agriculture:** Plant disease classification contributes to promoting sustainable agricultural practices. By precisely identifying diseases, farmers can implement targeted control measures, reducing the reliance on broad-spectrum pesticides and minimizing the environmental impact of agricultural practices.

Data Description

The dataset used for this project is the "New Plant Diseases Dataset" available on Kaggle. It contains a large collection of images of plant leaves classified into various categories such as healthy, infected with different diseases, and seemingly infected. The dataset provides a diverse range of plant diseases, allowing for robust training and evaluation of the classification model. This dataset consists of about 87K RGB images of healthy and diseased crop leaves which is categorized into 38 different classes. The total dataset is divided into 80/20 ratio of training and validation set preserving the directory structure.

Link - https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset

Data Pre-Processing

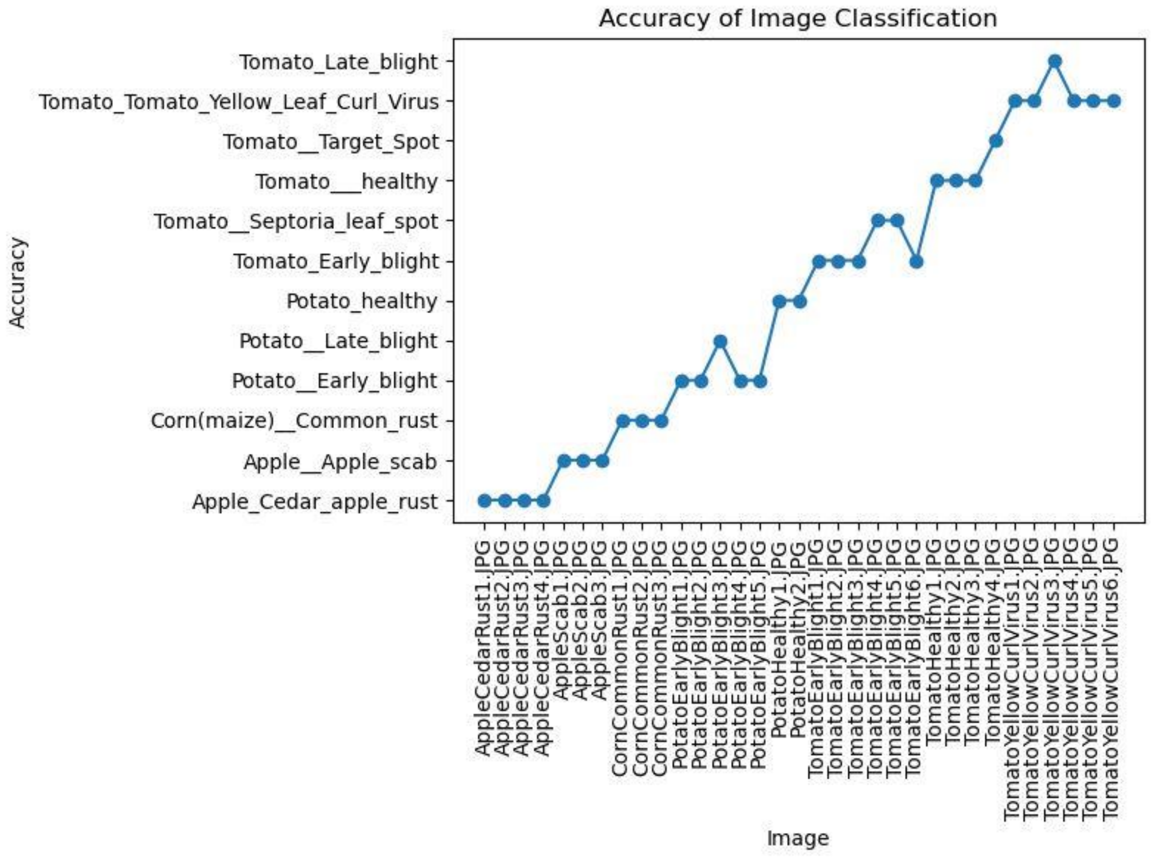
- **Image Resizing:** Resizing the plant images to a standard size helps in reducing computational complexity and ensures uniformity across the dataset. It also minimizes the impact of variations in image resolutions.

- **Image Enhancement:** Increasing the clarity and contrast of photos can make disease signs more visible. Image attributes that are important for illness identification can be improved using methods such adaptive histogram equalisation, contrast stretching, and histogram equalisation.

- **Image Augmentation:** Image Augmentation techniques are used to expand the dataset's diversity and size, reducing overfitting, and enhancing the model's capacity for generalisation. Rotation, flipping, cropping, zooming, and adding noise to the photos are common augmentation techniques.

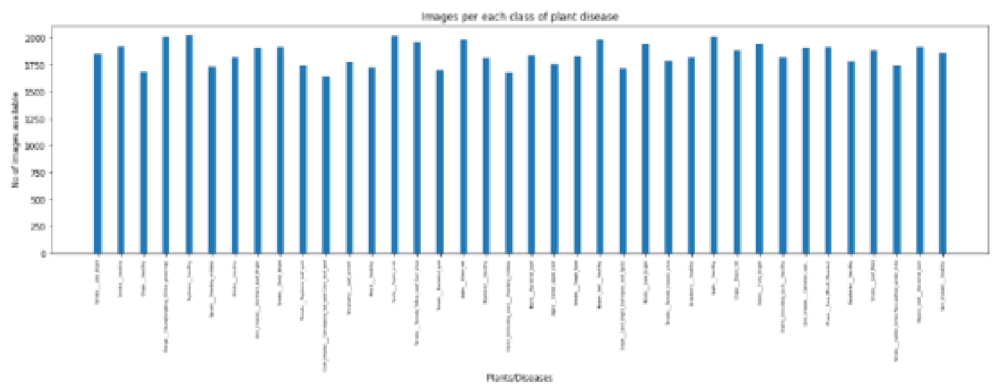
- **Normalization:** Normalizing the pixel values of the images helps in standardizing the input data. It involves scaling the pixel values to a specific range (e.g., 0 to 1 or -1 to 1). Normalization ensures that the features contribute equally to the learning process.

Exploratory Data Analysis

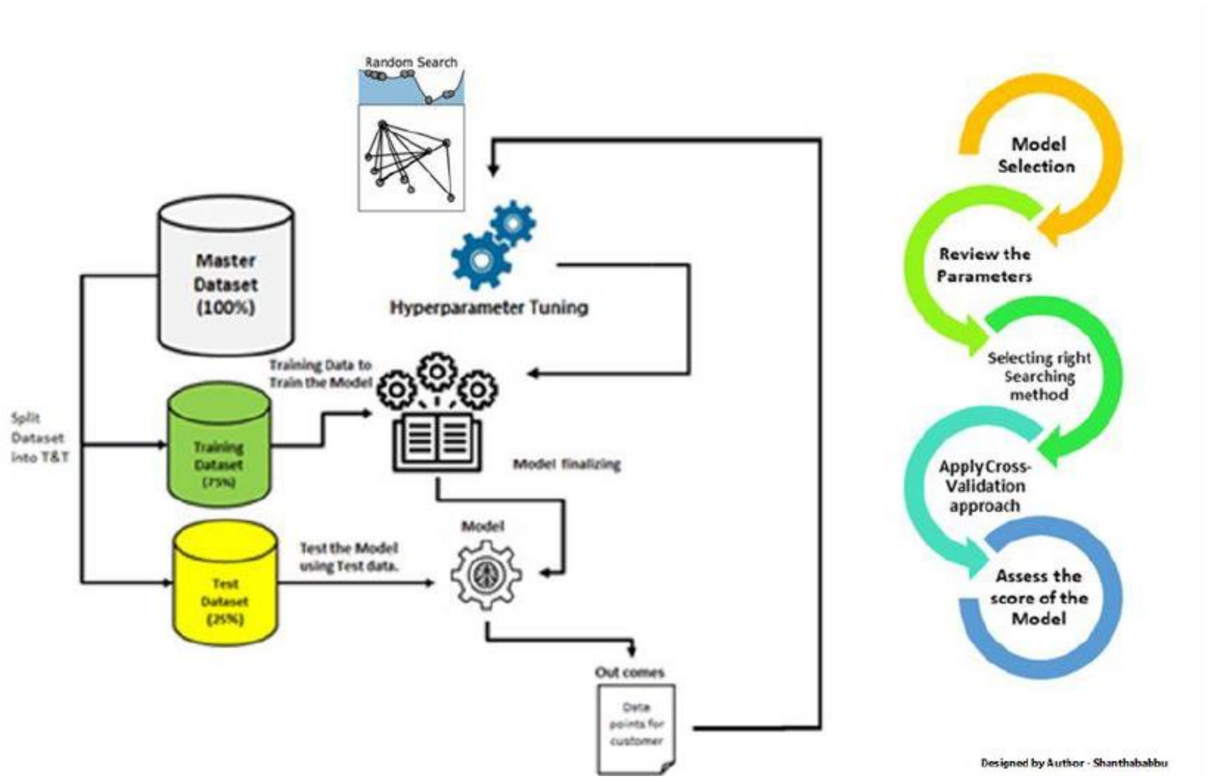


```
In [11]:
# plotting number of images available for each disease
index = [n for n in range(38)]
plt.figure(figsize=(20, 5))
plt.bar(index, [n for n in nums.values()], width=0.3)
plt.xlabel('Plants/Diseases', fontsize=10)
plt.ylabel('No of images available', fontsize=10)
plt.xticks(index, diseases, fontsize=5, rotation=90)
plt.title('Images per each class of plant disease')
```

```
Out[11]:
Text(0.5, 1.0, 'Images per each class of plant disease')
```

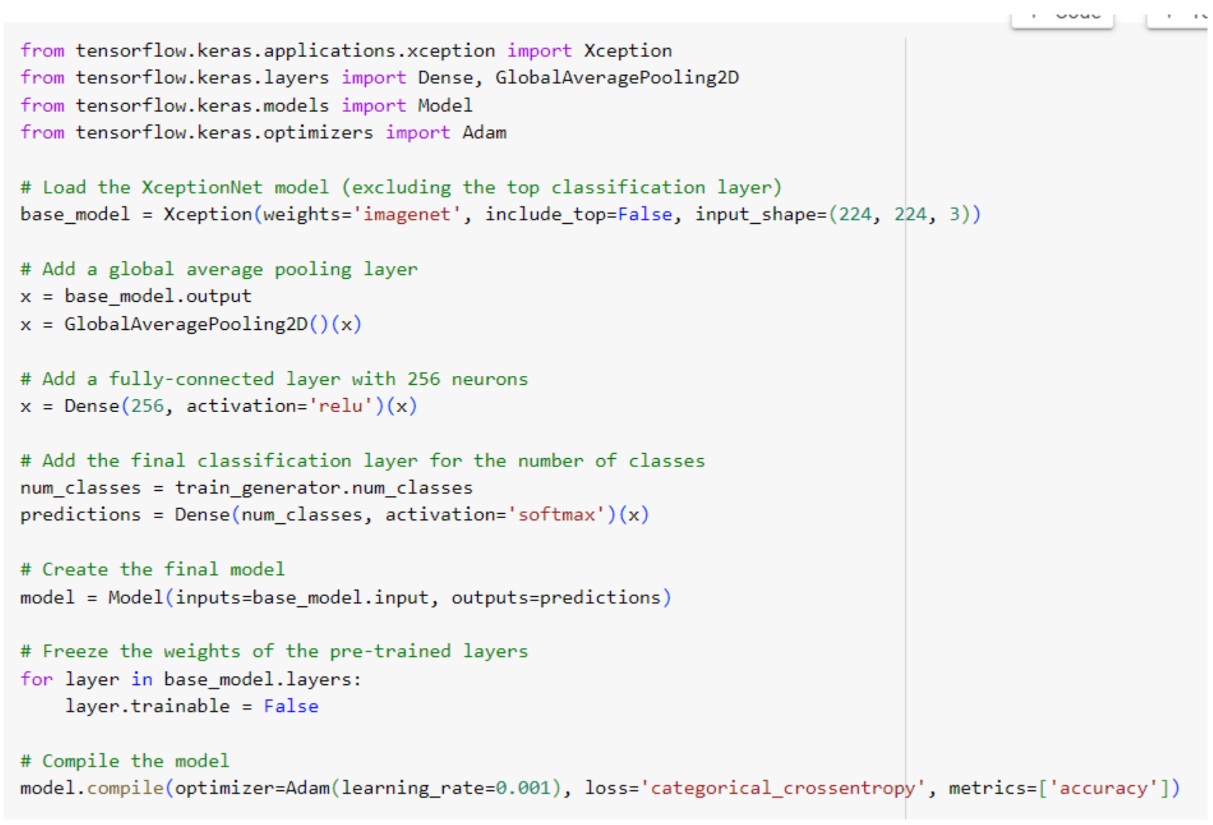


Flowchart

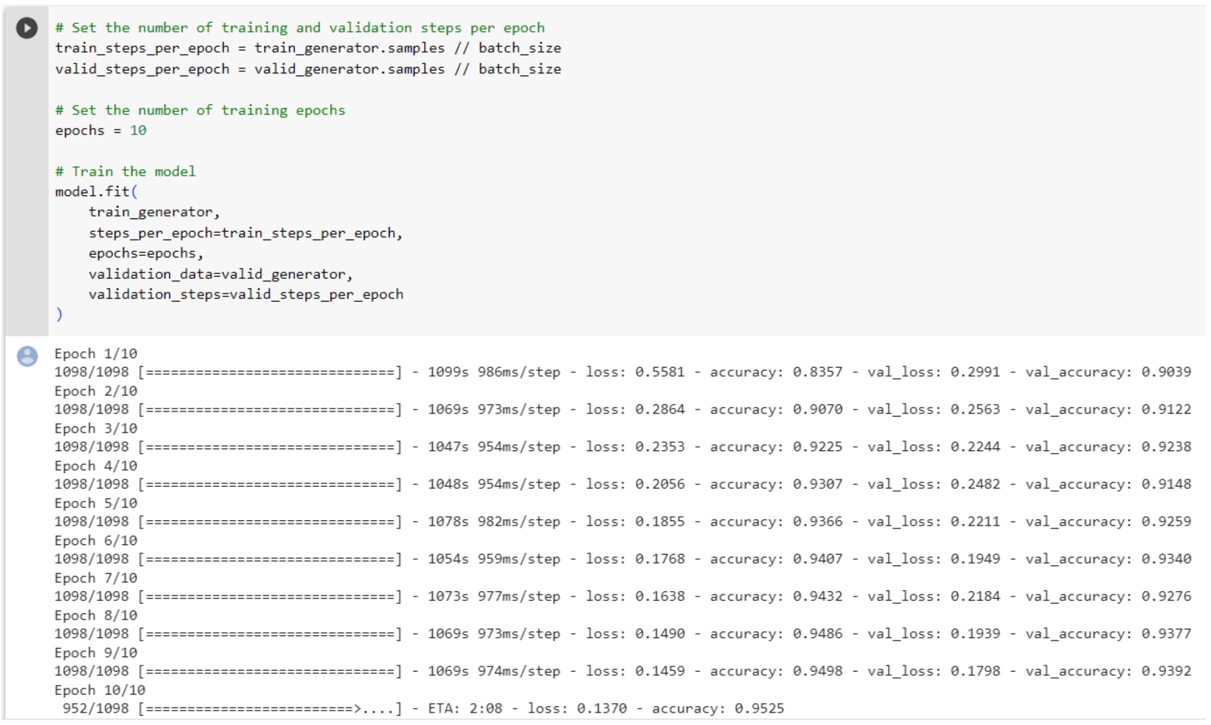


Model Development

XceptionNet: It is a deep convolutional neural network architecture introduced by François Chollet in 2016. It is based on the concept of the Inception module but aims to improve efficiency and performance by introducing depth wise separable convolutions. In XceptionNet, the network architecture consists of multiple stacked depth wise separable convolutional blocks, interleaved with max pooling, and skip connections. Each block comprises a depth wise convolution, followed by a pointwise convolution. This architecture reduces the number of parameters and computational complexity compared to traditional convolutions, making it more efficient. XceptionNet has shown impressive results in various computer vision tasks, including image classification, object detection, and semantic segmentation. It has achieved state-of-the-art performance on benchmark datasets and has become a popular choice in the field of deep learning for its efficiency and accuracy.



```python
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

# Load the XceptionNet model (excluding the top classification layer)
base_model = Xception(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Add a global average pooling layer
x = base_model.output
x = GlobalAveragePooling2D()(x)

# Add a fully-connected layer with 256 neurons
x = Dense(256, activation='relu')(x)

# Add the final classification layer for the number of classes
num_classes = train_generator.num_classes
predictions = Dense(num_classes, activation='softmax')(x)

# Create the final model
model = Model(inputs=base_model.input, outputs=predictions)

# Freeze the weights of the pre-trained layers
for layer in base_model.layers:
    layer.trainable = False

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

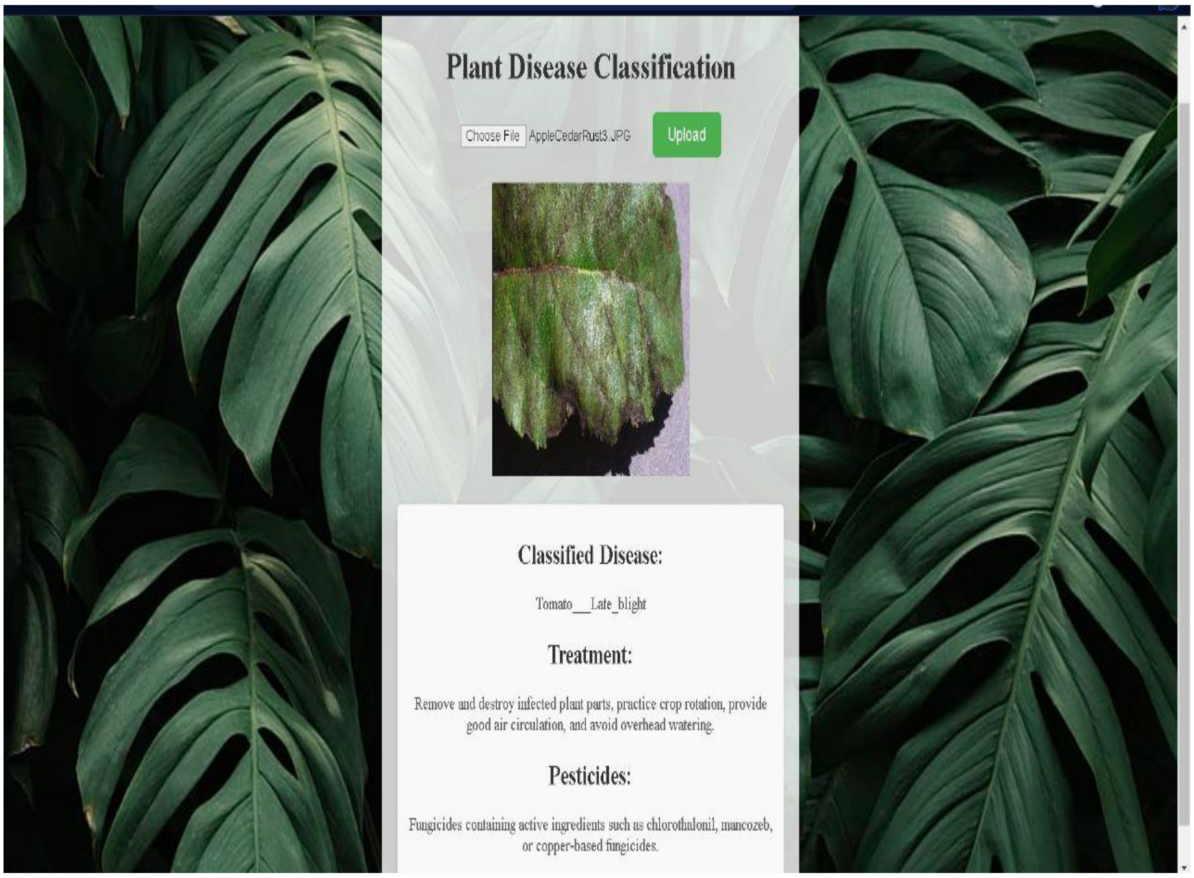


```python
# Set the number of training and validation steps per epoch
train_steps_per_epoch = train_generator.samples // batch_size
valid_steps_per_epoch = valid_generator.samples // batch_size

# Set the number of training epochs
epochs = 10

# Train the model
model.fit(
    train_generator,
    steps_per_epoch=train_steps_per_epoch,
    epochs=epochs,
    validation_data=valid_generator,
    validation_steps=valid_steps_per_epoch
)
```

```
Epoch 1/10
1098/1098 [==============================] - 1099s 986ms/step - loss: 0.5581 - accuracy: 0.8357 - val_loss: 0.2991 - val_accuracy: 0.9039
Epoch 2/10
1098/1098 [==============================] - 1069s 973ms/step - loss: 0.2864 - accuracy: 0.9070 - val_loss: 0.2563 - val_accuracy: 0.9122
Epoch 3/10
1098/1098 [==============================] - 1047s 954ms/step - loss: 0.2353 - accuracy: 0.9225 - val_loss: 0.2244 - val_accuracy: 0.9238
Epoch 4/10
1098/1098 [==============================] - 1048s 954ms/step - loss: 0.2056 - accuracy: 0.9307 - val_loss: 0.2482 - val_accuracy: 0.9148
Epoch 5/10
1098/1098 [==============================] - 1078s 982ms/step - loss: 0.1855 - accuracy: 0.9366 - val_loss: 0.2211 - val_accuracy: 0.9259
Epoch 6/10
1098/1098 [==============================] - 1054s 959ms/step - loss: 0.1768 - accuracy: 0.9407 - val_loss: 0.1949 - val_accuracy: 0.9340
Epoch 7/10
1098/1098 [==============================] - 1073s 977ms/step - loss: 0.1638 - accuracy: 0.9432 - val_loss: 0.2184 - val_accuracy: 0.9276
Epoch 8/10
1098/1098 [==============================] - 1069s 973ms/step - loss: 0.1490 - accuracy: 0.9486 - val_loss: 0.1939 - val_accuracy: 0.9377
Epoch 9/10
1098/1098 [==============================] - 1069s 974ms/step - loss: 0.1459 - accuracy: 0.9498 - val_loss: 0.1798 - val_accuracy: 0.9392
Epoch 10/10
 952/1098 [========================>....] - ETA: 2:08 - loss: 0.1370 - accuracy: 0.9525
```

Deployment

Flask has been used for web development wherein we have clearly stated the disease along with treatments that can be performed. The type of Pesticides to be used in those cases also has been discussed. An About section has been created where we have mentioned details about the project mentor and team members.



Plant Disease Classification

Classified Disease:

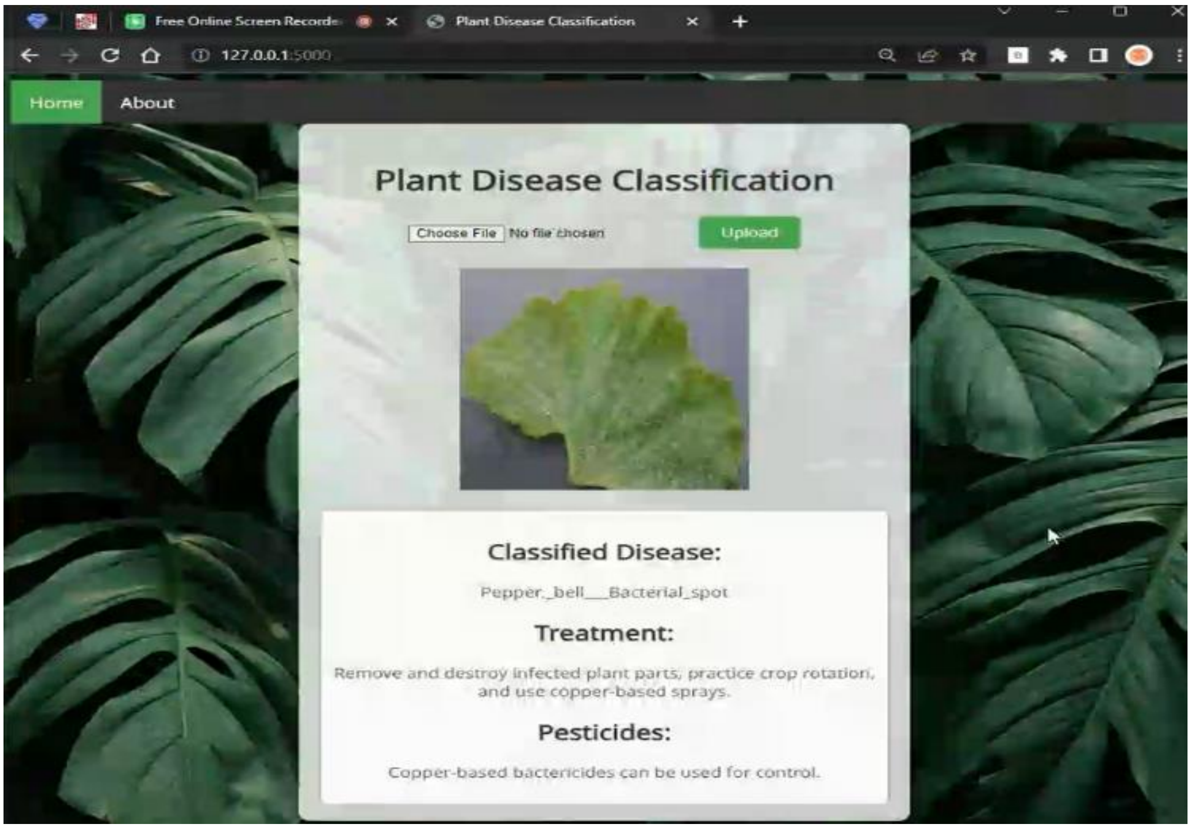
Tomato___Late_blight

Treatment:

Remove and destroy infected plant parts, practice crop rotation, provide good air circulation, and avoid overhead watering.

Pesticides:

Fungicides containing active ingredients such as chlorothalonil, mancozeb, or copper-based fungicides.



Plant Disease Classification

Classified Disease:

Pepper,_bell___Bacterial_spot

Treatment:

Remove and destroy infected plant parts, practice crop rotation, and use copper-based sprays.

Pesticides:

Copper-based bactericides can be used for control.

References

https://in.mathworks.com/help/deeplearning/ref/xception.html

https://keras.io/api/applications/xception/

https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset

https://www.mdpi.com/2223-7747/8/11/468

https://ieeexplore.ieee.org/abstract/document/8728415/