Crop Disease Prediction

Sumana M - E0321023
Jayavarshitha D - E0321039
Aadithya Prabha R - E0321008

Content

Introduction

01

By 2050 the yield of crops needs to be doubled to satisfy the growing population. Diseases majorly hamper productivity outcome.

It is estimated that between 20% to 40% of yearly crop production is lost due to plant diseases

02

03

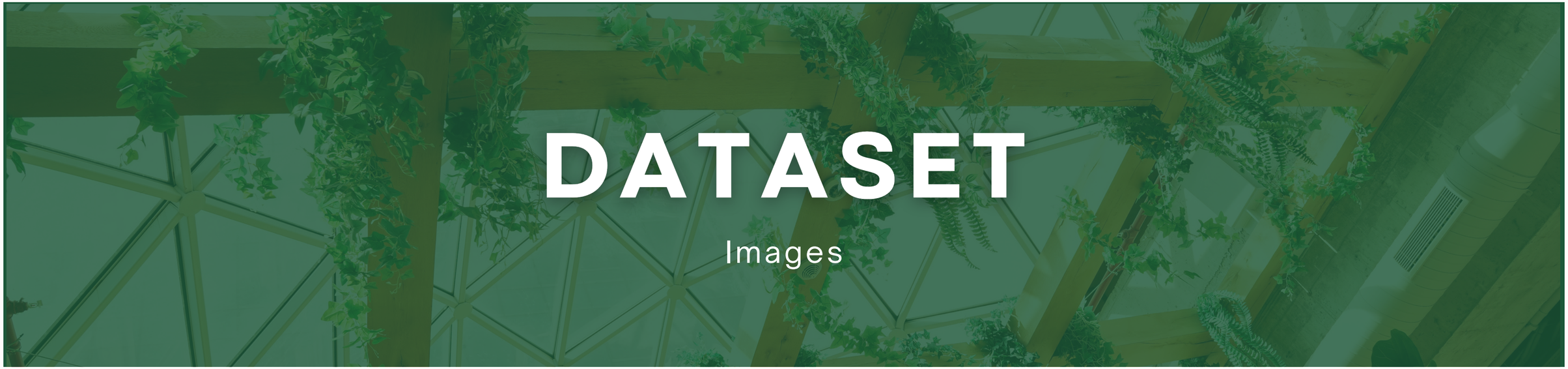
Traditional plant disease detection relies on manual observation by experts, a time-consuming and skill-intensive process, making it impractical for monitoring large farms.

An efficient system can alleviate the drawbacks of manual detection, contributing significantly to reducing the overuse of pesticides and chemicals.

04

Problem Statement

“ To detect crop diseases early in an accurate manner so that effective treatment can be administered “



DATASET

Images

Maize Common Rust

Maize rust is a fungal disease that primarily affects maize (corn) crops. The disease is often characterized by orange to reddish-brown pustules on the leaves, stems, and other plant parts. Maize rust is a significant concern for farmers as it can lead to yield losses if not managed effectively.

Potato Early Blight

Potato early blight is a common fungal disease caused by the pathogen Alternaria solani. This disease affects the leaves, stems, and tubers of potato plants. Early symptoms include dark lesions with concentric rings on the leaves, which can lead to premature defoliation if left untreated.

Tomato Bacterial Spot

Tomato bacterial spot is a bacterial disease. It affects various parts of the tomato plant, including leaves, fruit, and stems. The symptoms include small, dark lesions with a water-soaked appearance on leaves and circular spots on fruit. In severe cases, the disease can lead to defoliation and reduced fruit quality.

Methodology

Data Preprocessing

Data Visualization

Splitting data into train and test sets

Encoding target variable

Create , Compile, Fit model

Make Predictions on Testing Data

Visualizing the original and predicted label

Data Preprocessing

Data Preprocessing

01

Visualizing and resizing the images

Pick random images and plot as 256 x 256 pixels

02

Convert images to NumPy array and normalize them

Give binary labels to each directory and form an array so that it can be given as input

Data Visualization

Data Visualization

01

Visualize the class count and check for imbalance

Observe the different number of images in each class

Splitting data into train and test sets

Splitting data into train and test sets

01

Split the dataset

Testing - 20%
Training -80%

02

Normalize

Divide each image pixel with 255

Encoding the target variable

Encoding the target variable

01

One Hot Encoding

To make sure the label 2 does not get preference over 0 and 1

Create model architecture, compile model , fit it using training data

Create model architecture, compile model , fit it using training data

01

CNN Model

Sequential

Conv2D - Feature Extraction

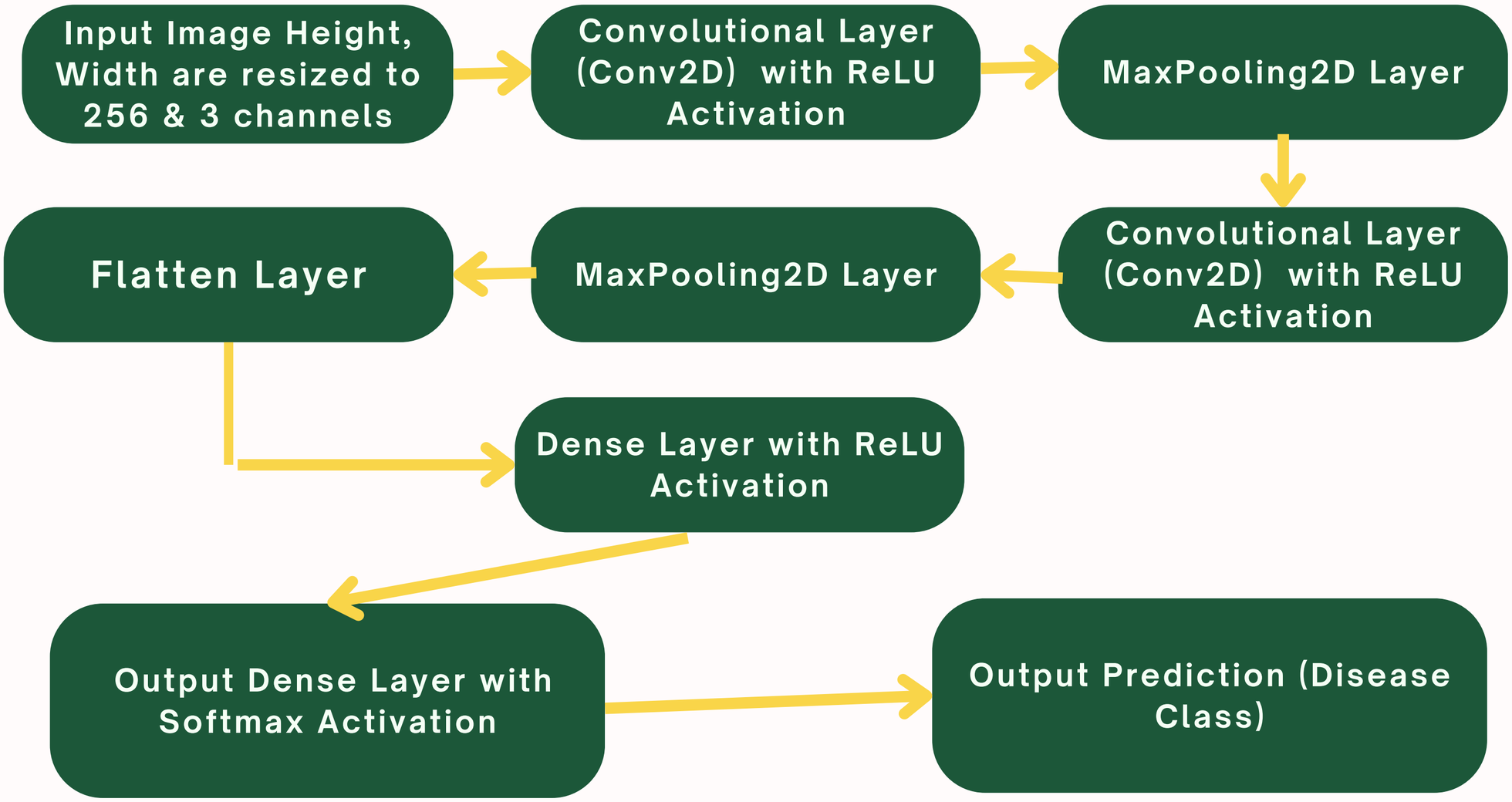
MaxPooling2D - Downsampling technique to retain important features

Flatten - Flattens input to 1D output

Dense - Dense layer for classification

Softmax activation in the output layer for multi-class classification

ARCHITECTURE



IMPLEMENTATION

Code & Output

Step 1: Mount the google drive on Google Collab Notebook and import the data set

```
# Load the Drive helper and mount
from google.colab import drive
# This will prompt for authorization.
drive.mount ('/content/drive')
```

[2] Pyth

```
Mounted at /content/drive
```

After mounting the drive, locate the folder where the data is stored to use it in this collab notebook.

```
# After executing the cell above, Drive files will be present in the following directory
!ls "/content/drive/My Drive/plant_disease_prediction_project"
```

[7] Pyth

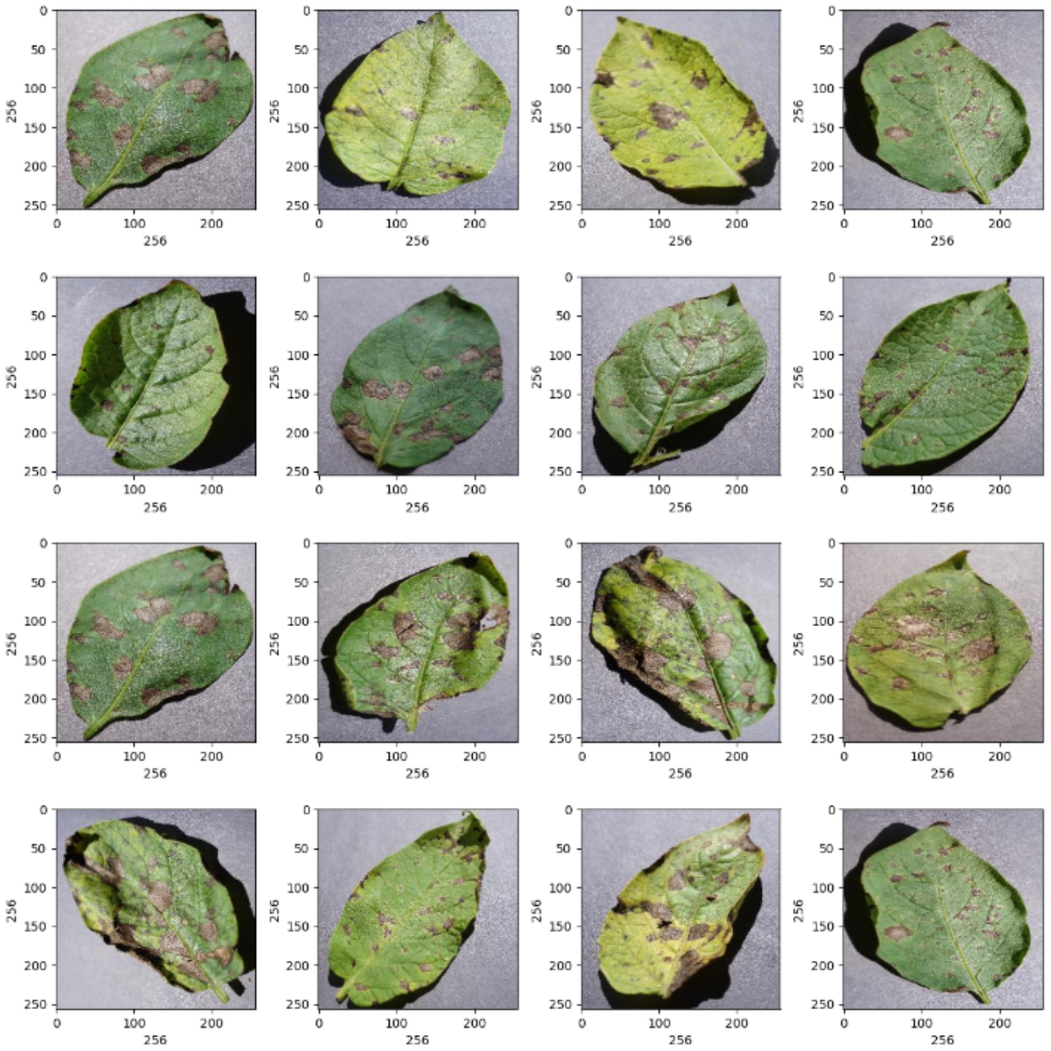
```
data  Model
```

```
!ls "/content/drive/My Drive/plant_disease_prediction_project/data"
```

[8] Pyth

```
Bacterial_Spot  Common_Rust  Early_Blight
```

```
# Plotting 12 images to check dataset
plt. figure(figsize=(12,12))
path = "/content/drive/My Drive/plant_disease_prediction_project/data/Early_Blight"
for i in range(1,17):
  plt. subplot(4,4,i)
  plt.tight_layout ()
  rand_img = imread(path +'/'+ random.choice(sorted(os.listdir(path))))
  plt.imshow(rand_img)
  plt.xlabel(rand_img.shape[1], fontsize = 10)#width of image
  plt.ylabel(rand_img.shape[0], fontsize = 10)#height of image
```



Step 4: COnvert the images into a Numpy array and normalize them.

```
#Converting Images to an array
def convert_image_to_array(image_dir):
  try:
    image = cv2.imread(image_dir)
    if image is not None:
      image = cv2.resize(image, (256,256))
      return img_to_array(image)
    else :
      return np.array([])
  except Exception as e:
    print(f"Error : {e}")
    return None
```

Now convert images into numpy array.

```
dir = "/content/drive/My Drive/plant_disease_prediction_project/data"
image_list, label_list = [], []
all_labels = ['Bacterial_Spot', 'Early_Blight', 'Common_Rust']
binary_labels = [0,1,2]
temp = -1
# Reading and converting image to numpy array
for directory in ['Bacterial_Spot', 'Early_Blight', 'Common_Rust']:
  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])
```

Step 5: Visualize the class count and check for class imbalance

```
# Visualize the number of classes count
label_counts = pd.DataFrame(label_list).value_counts()
label_counts.head()
```

```
0    300
1    300
2    300
dtype: int64
```

```
image_list[0].shape
```

```
(256, 256, 3)
```

```
from sklearn.utils.class_weight import compute_sample_weight

# Assuming `y_train` is your training labels
sample_weights = compute_sample_weight('balanced', y_train)
```

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

```
# Split the dataset without sample weights
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, random_state=10)

# Recalculate sample weights based on the training labels
sample_weights_train = compute_sample_weight('balanced', y_train)
sample_weights_val = compute_sample_weight('balanced', y_val)
```

```
# Train the model
epochs = 50
batch_size = 128
history = model.fit(
    x_train, y_train,
    batch_size=batch_size,
    epochs=epochs,
    validation_data=(x_val, y_val),
    sample_weight=sample_weights_train
)
```

Step 6: Splitting the dataset into train,validate and test sets.

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

Step 7: Performing one-hot encoding on target variable.

```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

Step 8: Creating the model architecture, compile the model and then fit it using the training data.

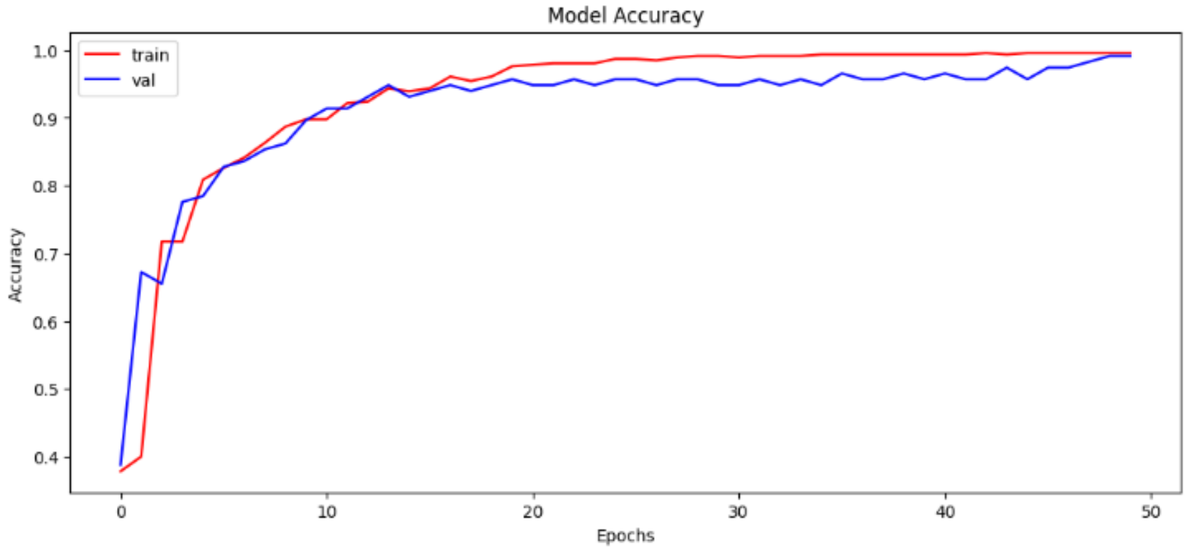
```
model = Sequential()
model.add(Conv2D(32, (3, 3), padding="same", input_shape=(256,256,3), activation="relu"))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Conv2D(16, (3, 3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(8, activation="relu"))
model.add(Dense(3, activation="softmax"))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 256, 256, 32)      896

 max_pooling2d (MaxPooling2D  (None, 85, 85, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 85, 85, 16)        4624

 max_pooling2d_1 (MaxPooling  (None, 42, 42, 16)       0
 2D)

 flatten (Flatten)           (None, 28224)             0

 dense (Dense)               (None, 8)                 225800

 dense_1 (Dense)             (None, 3)                 27

=================================================================
Total params: 231,347
Trainable params: 231,347
Non-trainable params: 0
```

```
[24] model.save("/content/drive/My Drive/plant_disease_prediction_project/Model/plant_disease_model2.h5")
```

Step 9: Plot the accuracy and loss against each epoch

```
[25] #Plot the training history
     plt.figure(figsize=(12, 5))
     plt.plot(history.history['accuracy'], color='r')
     plt.plot(history.history['val_accuracy'], color='b')
     plt.title('Model Accuracy')
     plt.ylabel('Accuracy')
     plt.xlabel ('Epochs')
     plt.legend(['train','val'])
     plt.show()
```




```
[26] # Evaluate the model
     print("Calculating model accuracy")
     scores = model.evaluate(x_test, y_test)
     print(f"Test Accuracy: {scores[1]*100}")
```

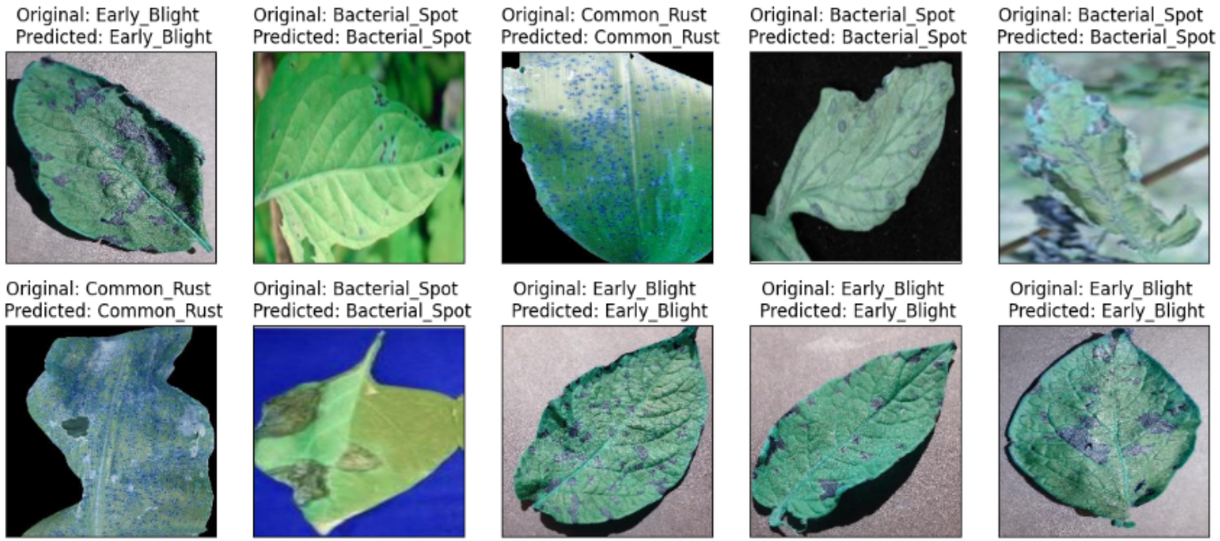
```
Calculating model accuracy
6/6 [==============================] - 5s 940ms/step - loss: 0.0803 - accuracy: 0.9778
Test Accuracy: 97.77777791023254
```

Step 10:Make Predictions on testing data.

```
y_pred = model.predict(x_test)
```

```
6/6 [==============================] - 3s 461ms/step
```

```
# Visualizing the original and predicted labels for the test images
fig, axes = plt.subplots(2, 5, figsize=(15, 6), subplot_kw={'xticks':[], 'yticks':[]}, gridspec_kw=dict(hspace=0.3, wspace=0.1))

for i, ax in enumerate(axes.flat):
    ax.imshow(array_to_img(x_test[i]))
    ax.set_title(f'Original: {all_labels[np.argmax(y_test[i])]} \n Predicted: {all_labels[np.argmax(y_pred[i])]}')

plt.show()
```



Step 11: Visualizing the original and predicted labels for the test images.

```
#plotting image to compare
img = array_to_img(x_test[11])
img
```

```
# Finding max value from predition list and comparing original value vs predicted
print("Original Label: ",all_labels[np.argmax(y_test[11])])
print("Predicted Label: ",all_labels[np.argmax(y_pred[4])])
print(y_pred[2])
```

```
Original Label:  Bacterial_Spot
Predicted Label:  Bacterial_Spot
[7.6856650e-04 2.0632572e-03 9.9716812e-01]
```

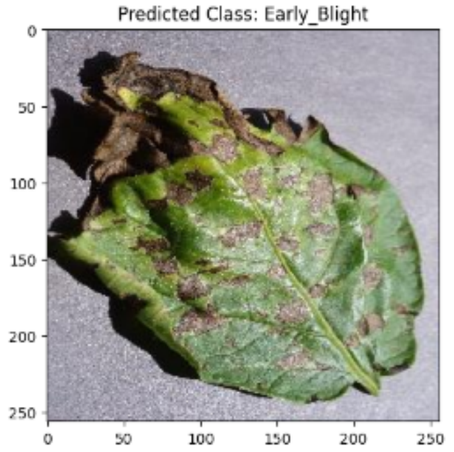
```
for i in range(50):
  print(all_labels[np.argmax(y_test[i])], "-", all_labels[np.argmax(y_pred[i])])
```

```
Early_Blight - Early_Blight
Bacterial_Spot - Bacterial_Spot
Common_Rust - Common_Rust
Bacterial_Spot - Bacterial_Spot
Bacterial_Spot - Bacterial_Spot
Common_Rust - Common_Rust
Bacterial_Spot - Bacterial_Spot
Early_Blight - Early_Blight
Early_Blight - Early_Blight
Early_Blight - Early_Blight
```

```python
import numpy as np
from tensorflow.keras.preprocessing import image  # Updated import statement
from keras.models import load_model
import matplotlib.pyplot as plt
from keras.applications.inception_v3 import preprocess_input

# Load the trained model
model = load_model("/content/drive/My Drive/plant_disease_prediction_project/Model/plant_disease_model.h5")

# Function to preprocess an image for prediction
def preprocess_image(image_path):
    img = image.load_img(image_path, target_size=(256, 256))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0  # Normalize pixel values to the range [0, 1]
    return img_array

# Specify the path to a random image
random_image_path = "/content/drive/My Drive/plant_disease_prediction_project/data/Bacterial_Spot/Bs43.JPG"

# Preprocess the image
preprocessed_image = preprocess_image(random_image_path)

# Make a prediction
predictions = model.predict(preprocessed_image)

# Decode the prediction to get the class label
class_label = np.argmax(predictions)

# Print the predicted class label
print("Predicted Class Label:", class_label)

# Map class label to class name (replace with your actual class names)
class_names = ['Bacterial_Spot', 'Early_Blight', 'Common_Rust']
predicted_class_name = class_names[class_label]

# Display the image
img = image.load_img(random_image_path, target_size=(256, 256))
plt.imshow(img)
plt.title(f'Predicted Class: {predicted_class_name}')
plt.show()
```

```python
# Function to preprocess an image for prediction
def preprocess_image(image_path):
    img = image.load_img(image_path, target_size=(256, 256))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0  # Normalize pixel values to the range [0, 1]
    return img_array

# Specify the path to a random image
random_image_path = "/content/drive/My Drive/plant_disease_prediction_project/data/Early_Blight/ffef8a3b-3e8a-4c24-a27e-64c41708f7a0___RS_Early.B 7132.JPG"

# Preprocess the image
preprocessed_image = preprocess_image(random_image_path)

# Make a prediction
predictions = model.predict(preprocessed_image)

# Decode the prediction to get the class label
class_label = np.argmax(predictions)

# Print the predicted class label
print("Predicted Class Label:", class_label)

# Map class label to class name (replace with your actual class names)
class_names = ['Bacterial_Spot', 'Early_Blight', 'Common_Rust']
predicted_class_name = class_names[class_label]

# Display the image
img = image.load_img(random_image_path, target_size=(256, 256))
plt.imshow(img)
plt.title(f'Predicted Class: {predicted_class_name}')
plt.show()
```

```
1/1 [==============================] - 2s 2s/step
Predicted Class Label: 1
```



REFERENCE PAPERS

- **Plant Disease Detection using Convolutional Neural Network**
  Jahnavi Kolli; Dhara Mohana Vamsi; V. M. Manikandan

Development of an efficient machine learning model based on Convolutional Neural Networks (CNN) for identifying diseased crops using technologies like IoT devices, mobile phones, and drones. The model achieved a high accuracy of 94.87% by training on a plant village dataset and utilizing image processing with OpenCV.

- **LEAF DISEASE DETECTION USING CNN**
  Reshmi A.M, Prameeja Prasidhan M.Sc Scholar, Assistant Professor

The project utilizes an efficient Convolutional Neural Network (CNN) algorithm for real-time disease detection in plants, achieving an impressive accuracy of 99.5% through the stages of dataset gathering, training, segmentation, feature extraction, testing, and classification.

THANK YOU