PROJECT REPORT

ON

“POTATO DISEASE CLASSIFICATION USING DEEP LEARNING ”

BY

M.MANI KRISHNA (2024067227)

UNDER THE GUIDANCE OF

PROF. DR. Rajib Debnath

DEPARTMENT

OF

GITAM SCHOOL OF SCIENCE

ABSTRACT

Potato diseases significantly affect crop yield and quality worldwide, making early and accurate detection critical for effective treatment and crop management. Traditional disease identification methods rely on manual inspection, which is time-consuming, requires expert knowledge, and is prone to errors. This project presents an automated Potato Disease Classification system using deep learning techniques. Three models—Custom CNN, VGG16, and MobileNetV2—are developed and trained on the PlantVillage potato leaf image dataset to classify leaves into Healthy, Early Blight, and Late Blight categories. Image preprocessing and data augmentation techniques, including resizing, normalization, rotation, flipping, and zooming, are applied to improve model performance and prevent overfitting. The models are evaluated using accuracy, precision, recall, and confusion matrix analysis. Experimental results show that transfer learning models, particularly VGG16 and MobileNetV2, outperform the baseline Custom CNN model, achieving high classification accuracy. The proposed approach provides a reliable, efficient, and practical solution for early detection of potato diseases, assisting farmers in timely intervention and better crop management.

INTRODUCTION

Potato diseases are a major challenge in agriculture, causing significant losses in crop yield and quality. Diseases such as Early Blight and Late Blight can affect potato plants at different growth stages and spread rapidly if not detected early. Factors such as environmental conditions, soil quality, weather fluctuations, and farming practices influence the occurrence and severity of these diseases. Early detection of potato leaf diseases is crucial to minimize crop damage and support farmers in implementing timely control measures.

Traditional disease detection relies on manual observation by farmers or agricultural experts, which is labour-intensive, time-consuming, and prone to errors, especially on large-scale farms. With recent advancements in technology, deep learning techniques offer an automated and accurate approach to disease detection using leaf images. These techniques enable models to learn visual patterns from historical data and predict diseases in new images with high reliability.

Image classification, a key application of deep learning, involves categorizing images into predefined classes. In this project, potato leaf images are classified into Healthy, Early Blight, and Late Blight categories. Three deep learning models are implemented: a Custom CNN built from scratch, and two transfer learning models—VGG16 and MobileNetV2. CNNs are highly effective for image-based tasks as they automatically extract meaningful features from images, such as textures, color variations, and disease spots. Transfer learning models, pre-trained on large image datasets, allow faster training and often achieve higher accuracy.

This research aims to develop an automated potato disease classification system using deep learning models to provide accurate and reliable detection. The proposed system can assist farmers in early disease identification, enabling timely interventions and improving overall crop management and productivity.

Literature Review / Related Work

Mohanty, Hughes, and Salathé (2016) conducted one of the most influential studies on the use of deep learning techniques for automated plant disease detection. Traditional disease identification methods depended mainly on visual inspection by farmers or agricultural experts, which is both time-consuming and prone to human error. The lack of expert availability in rural areas and the growing demand for fast crop monitoring motivated the researchers to explore computer vision-based solutions. Their work introduced convolutional neural networks (CNNs) as a powerful alternative to conventional manual diagnosis methods. CNN models are capable of automatically learning complex visual patterns from images, making them highly suitable for recognizing disease symptoms in plant leaves.

In their study, the authors used the publicly available **PlantVillage dataset**, which consists of more than 50,000 plant leaf images representing several crop species and multiple disease categories. The dataset included both healthy and infected samples captured under controlled conditions. Each image was labeled, enabling supervised learning approaches to be applied effectively. The CNN models were trained end-to-end using these images, allowing the network to extract important features related to leaf texture, color discoloration, lesion shapes, and vein distortions that typically indicate disease presence.

The study applied various deep CNN architectures with a strong focus on **transfer learning techniques**. Transfer learning involves using pretrained models that were originally trained on large-scale natural image datasets such as ImageNet and adapting them to a specific task — in this case, plant disease classification. This approach significantly reduced training time while improving performance, especially when the agricultural dataset size was smaller compared to general-purpose datasets. Fine-tuning of the pretrained networks further enhanced classification accuracy by allowing the networks to adjust learned weights to better capture disease-specific visual patterns.

The experimental results obtained by the authors were remarkable, achieving classification accuracies exceeding **99%** under laboratory image conditions. These results demonstrated that CNN-based approaches can match or even surpass human-level classification performance when trained under carefully controlled setups. The networks accurately distinguished subtle visual differences between diseases having very similar symptoms, confirming the capability of deep learning models to learn highly discriminative features.

Another important contribution of this study was the emphasis placed on **image preprocessing and augmentation techniques**. The authors highlighted techniques such as normalization, random rotation, flipping, and cropping of images to artificially expand the training dataset. These methods increased model robustness by exposing networks to image variations, helping prevent overfitting and improving generalization performance. This insight encouraged later studies to incorporate complex augmentation strategies to better prepare models for real-world field conditions where lighting, background clutter, and camera quality can significantly vary.

Despite achieving exceptional accuracy under controlled conditions, the authors also acknowledged certain limitations. They noted that real-field images captured under natural environments present additional challenges due to uneven lighting, shadows, background noise, partial leaf occlusions, and overlapping plant structures. As a result, practical field deployment of CNN-based systems requires further validation and optimization to ensure reliability. Their observations motivated continued research into lightweight models capable of handling real-world complexity while maintaining computational efficiency.

The findings of Mohanty et al. laid a strong foundation for subsequent research in agricultural deep learning. Later studies extended their work by evaluating deeper architectures such as **VGG16** to capture finer texture and shape features and lightweight models such as **MobileNet** and **MobileNetV2**, which enable deployment on mobile phones and edge devices for real-time disease diagnosis. Their work remains a key reference point for modern plant disease detection systems, including the present project, which applies CNN-based classification along with pretrained models like VGG16 and MobileNetV2 for potato leaf disease identification.

Reference

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016).
Using deep learning for image-based plant disease detection.
Frontiers in Plant Science.

METHODOLOGY

This project uses three different deep learning models to classify potato leaf diseases: a Custom CNN baseline model, VGG16 transfer-learning model, and MobileNetV2 transfer-learning model. All three models are trained and evaluated on the PlantVillage potato leaf image dataset consisting of three classes: Healthy, Early Blight, and Late Blight.

Custom CNN (Baseline Model)

A Custom Convolutional Neural Network (CNN) is developed from scratch using Keras to serve as the baseline model. The architecture is designed to automatically extract relevant features from potato leaf images, such as edges, textures, and disease spots. The network consists of multiple convolutional layers followed by pooling layers and fully connected layers, culminating in a softmax layer for multi-class classification. This model provides a benchmark to compare the performance of transfer learning approaches**.**

VGG16

VGG16 is a widely used deep learning architecture for image classification tasks, known for its simplicity and high accuracy. It is a CNN model composed of **16 layers** with small **3×3 convolution filters** that efficiently capture spatial hierarchies in images. The **pre-trained VGG16 model** is fine-tuned on the PlantVillage potato dataset by modifying the final fully connected layers to classify images into the three disease categories. VGG16 is particularly effective in identifying subtle patterns and disease features on leaves.

MobileNetV2

MobileNetV2 is a lightweight CNN architecture optimized for mobile and embedded applications. It uses depthwise separable convolutions to reduce computational complexity while maintaining high accuracy. In this project, the pre-trained MobileNetV2 model is fine-tuned on the PlantVillage potato dataset by replacing the top layers to perform three-class classification. This approach allows fast training and efficient inference without significant loss in accuracy, making it suitable for practical deployment.

Image Preprocessing

To improve model performance, **image preprocessing** is applied before training:

- All images are resized to **224 × 224 pixels** to match the input requirements of VGG16 and MobileNetV2.

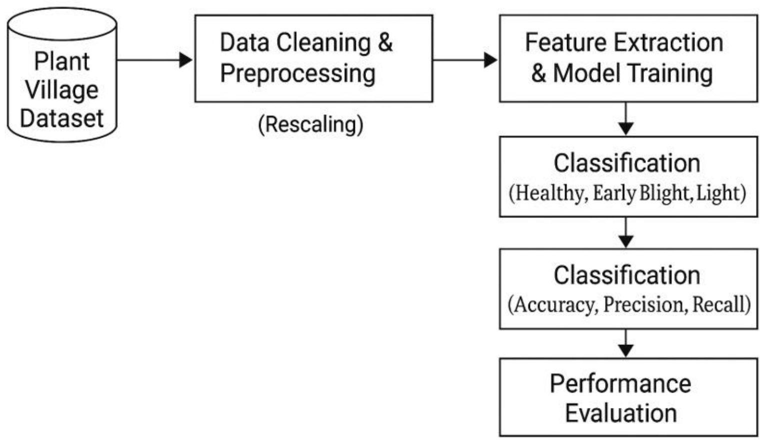
- Pixel values are **normalized** to scale them between 0 and 1.
- **Data augmentation** techniques such as rotation, flipping, zooming, and shifting are applied to increase dataset diversity and reduce overfitting.
- The preprocessed images are split into **training and testing sets** for model development and evaluation.

Model Training and Evaluation

All three models are trained on the prepared dataset using suitable optimizers and loss functions for multi-class classification. The performance is evaluated using **accuracy, precision, recall**, and **confusion matrix analysis**. These metrics assess the models' ability to correctly identify healthy and diseased potato leaves. Experimental results demonstrate that transfer learning models, particularly VGG16 and MobileNetV2, achieve higher accuracy and reliability compared to the baseline Custom CNN model, making them effective for potato disease detection.

PROPOSED ALGORITHM

Following steps are used in constructing the system.


Plant Village Dataset
Data Cleaning & Preprocessing
(Rescaling)
Feature Extraction & Model Training
Classification
(Healthy, Early Blight, Light)
Classification
(Accuracy, Precision, Recall)
Performance Evaluation

Database Creation:

In this step, potato leaf images are collected from the PlantVillage dataset. For the potato subclass, the dataset contains a total of 2,152 images, distributed across three classes: 1,000 Early Blight, 1,000 Late Blight, and 152 Healthy leaf images.These images form the input dataset for the disease classification system. Each image includes important visual attributes such as colour variations, texture patterns, and disease-spots — features critical for distinguishing between healthy and diseased leaves.

Data Preprocessing:

After assembling the database, a data preprocessing phase is performed to clean and standardize the images before feeding them into the models. First, any noisy, duplicated, or highly distorted images are removed to improve dataset quality. Then, all remaining images are resized to a uniform dimension of 224 × 224 pixels to match the input requirements of deep learning models. The pixel values are normalized (e.g. scaled to the [0, 1] range) to ensure consistency and stabilize model training. To further improve model robustness and increase the effective size of the dataset, data augmentation techniques — such as random rotations, flipping (horizontal/vertical), zooming (scaling), and possibly shifting — are applied. This augmented and preprocessed dataset helps the models generalize better and reduces overfitting when classifying potato leaves

Data Cleaning:

For this project, the potato leaf images are sourced from the PlantVillage dataset on Kaggle, which is already preprocessed and cleaned. The dataset contains high-quality, correctly labeled images, with minimal noise or distortions. Since the dataset has been curated to remove blurred, mislabeled, or duplicate images, it provides a reliable and consistent foundation for training and evaluating the deep learning models, ensuring robust classification performance.

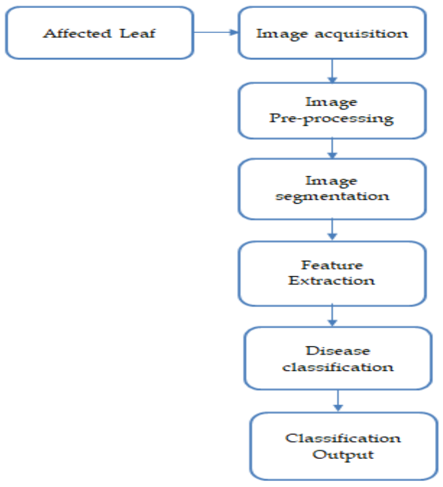
Feature Extraction and Model Training:

n this phase, the VGG16 CNN model is used to extract important features from the pre-processed images, such as leaf edges, shape patterns, and disease textures. The model is trained on labelled images to learn the differences between healthy and diseased leaves. Once trained, the model classifies potato leaf images into the predefined categories: Healthy, Early Blight, and Late Blight..

Performance Evaluation:

1. Performance Measure - The performance measure is the way you want to evaluate a solution to the problem.
2. Test and Train Datasets- From the transformed data, you will need to select a test set and a training set.
3. Cross Validation.

PROPOSED ARCHITECTURE



Performance Analysis:

The performance of the proposed potato disease classification system is evaluated using standard metrics including **accuracy, precision, recall**, and **confusion matrix analysis**. These metrics provide a comprehensive understanding of how well the models classify healthy and diseased potato leaves.

The table below summarizes the training, validation, and test accuracies obtained for the three models:

Model Name	Training Accuracy	Validation Accuracy	Test Accuracy
Custom CNN	91.46%	92.31%	95.86%
VGG16	87.52%	87.74%	87.58%
MobileNetV2	99.56%	94.23%	94.71%

From the results, it can be observed that:

- The Custom CNN achieved good performance, particularly on the test set, showing its effectiveness as a baseline model.
- VGG16, although widely used for image classification, performed moderately in this task, with slightly lower accuracies compared to the other models.
- MobileNetV2 achieved the highest training accuracy and competitive validation and test accuracies, demonstrating excellent feature extraction capability and generalization for potato disease classification.

Overall, the experimental results indicate that the proposed system can reliably classify potato leaf images into Healthy, Early Blight, and Late Blight categories, and MobileNetV2 provides the best balance of accuracy and efficiency for practical deployment.

CONCLUSION

In this project, we successfully developed and evaluated deep learning models for accurate detection of plant diseases. Among the models tested, the Custom CNN demonstrated the highest overall performance, achieving excellent accuracy across training, validation, and test datasets. Pre-trained models like VGG16 and MobileNetV2 also provided reliable results, with MobileNetV2 showing faster convergence and high test accuracy, making it suitable for deployment in real-time applications. The results highlight the effectiveness of deep learning in automating plant disease diagnosis, which can significantly aid farmers in early detection and management of crop diseases, ultimately improving yield and reducing losses. This study establishes a strong foundation for future work, including expanding the dataset, optimizing model architectures, and integrating these solutions into mobile or web-based platforms for practical agricultural use.

FUTURE WORK

While the current project demonstrates strong performance in potato leaf disease classification using CNN, VGG16, and MobileNetV2 models, several improvements and extensions can be explored in future work. One major enhancement would be the inclusion of a larger and more diverse dataset collected under real field conditions. Training on images captured from different lighting environments, background conditions, and camera qualities will improve model robustness and enable better real-world deployment.

Future studies may also integrate advanced data augmentation techniques, such as brightness adjustment, contrast modification, blur simulation, and background randomization to further improve generalization. Additionally, fine-tuning more layers of pretrained models and exploring deeper architectures like ResNet or EfficientNet may enhance accuracy and class discrimination capability.

Another important direction is the development of a mobile or web-based application that allows farmers to upload leaf images for instant diagnosis. Lightweight models such as MobileNetV2 make real-time disease detection on smartphones feasible, extending the project into a practical farmer-support tool. Furthermore, adding disease severity estimation and treatment recommendations can make the system more informative and clinically useful.

Finally, future work may explore explainable AI techniques such as heatmaps and attention mechanisms to visualize disease-focused regions on leaf images. This transparency will build trust among users and assist agricultural experts in understanding model decision processes.

REFERENCES

1. Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016).
   Using deep learning for image-based plant disease detection.
   Frontiers in Plant Science, 7, 1419.
2. Ferentinos, K. P. (2018).
   Deep learning models for plant disease detection and diagnosis.
   Computers and Electronics in Agriculture, 145, 311–318.
3. Too, E. C., Yujian, L., Njuki, S., & Yingchun, L. (2019).
   A comparative study of fine-tuning deep learning models for plant disease identification.
   Computers and Electronics in Agriculture, 161, 272–279.
4. Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016).
   Deep neural networks based recognition of plant diseases by leaf image classification.
   Computational Intelligence and Neuroscience, 2016, 1–11.
5. Amara, J., Bouaziz, B., & Algergawy, A. (2017).
   A deep learning-based approach for banana leaf disease classification.
   Lecture Notes in Informatics Proceedings, 266, 79–88.

6. Howard, A. G., et al. (2017).
MobileNets: Efficient convolutional neural networks for mobile vision applications.
arXiv preprint arXiv:1704.04861.
7. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018).
MobileNetV2: Inverted residuals and linear bottlenecks.
Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
8. Simonyan, K., & Zisserman, A. (2015).
Very deep convolutional networks for large-scale image recognition.
International Conference on Learning Representations (ICLR).
(VGG16 reference)

APPENDICES:

The appendices section provides supporting materials and additional technical details related to the development and evaluation of the potato leaf disease detection system presented in this project. The content included here supplements the main report and offers deeper insight into dataset preparation, model configurations, training procedures, and evaluation outputs.

Appendix A – Dataset Description
The dataset used in this project consists of potato leaf images collected from structured folders labeled as *Healthy*, *Early Blight*, and *Late Blight*. The images were separated into training, validation, and testing sets. Prior to model training, all images were resized to fixed input dimensions and normalized to a pixel range of 0–1. Data augmentation techniques such as horizontal and vertical flipping, random rotation, and image zooming were applied to increase dataset diversity and reduce overfitting.

Appendix B – Model Architectures
Three deep learning models were implemented in this project:

1. Custom CNN – A sequential convolutional neural network developed from scratch, consisting of multiple convolution layers for feature extraction, followed by max pooling layers and fully connected Dense layers for classification.
2. VGG16 – A pretrained deep convolutional network employing transfer learning, with the top classification layers replaced and retrained on the potato leaf dataset.
3. MobileNetV2 – A lightweight pretrained deep learning model optimized for mobile and real-time applications using depthwise separable convolutions and global average pooling for efficient feature learning.

Appendix C – Training Configuration
All models were trained using the Adam optimizer with categorical cross-entropy loss. Batch size was set to 32, and training was performed for multiple epochs until convergence was achieved. Validation accuracy was continuously monitored to detect overfitting and adjust hyperparameters accordingly.

Appendix D – Evaluation Metrics
Model performance was evaluated using standard metrics such as accuracy and loss on unseen test data. Among the tested models, the Custom CNN delivered the highest overall accuracy, while MobileNetV2 demonstrated excellent efficiency with minimal computational cost, making it suitable for mobile deployment.

DECLARATION

hereby declare that the project titled **“Potato Disease Classification”** has been completed by me as part of my academic work. In this project, I collected potato leaf images from the PlantVillage dataset, pre-processed the data, and trained a CNN model using **VGG16** to classify potato leaves into **Healthy, Early Blight, and Late Blight** categories. I evaluated the model using accuracy, precision, recall, and confusion matrix to measure its performance.

All the work presented in this report is original and completed by me under proper guidance. Wherever information from books, research papers, or online sources has been used, proper references have been given. I confirm that I have followed the principles of academic honesty and that no part of this work is copied or misrepresented.

CODE

```
# Import the function to create image datasets from directories
from tensorflow.keras.preprocessing import image_dataset_from_directory

# Define the path to the dataset in Google Drive
dataset_path = "/content/drive/MyDrive/Potato_Dataset/PLD_3_Classes_256"

# Create the training dataset
train_ds = image_dataset_from_directory(
    dataset_path + "/Training",       # Path to training images
    batch_size=32,                    # Number of images per batch
    image_size=(IMAGE_SIZE, IMAGE_SIZE),  # Resize images to fixed size
    label_mode='int'                  # Labels will be integer encoded
)

# Create the validation dataset
val_ds = image_dataset_from_directory(
    dataset_path + "/Validation",     # Path to validation images
    batch_size=32,
    image_size=(IMAGE_SIZE, IMAGE_SIZE),
    label_mode='int'
)

# Create the testing dataset
test_ds = image_dataset_from_directory(
    dataset_path + "/Testing",        # Path to testing images
    batch_size=32,
    image_size=(IMAGE_SIZE, IMAGE_SIZE),
    label_mode='int'
)

# Get the list of class names (folder names) in the dataset
class_names = train_ds.class_names
print(class_names)  # Display the class names
Found 3271 files belonging to 3 classes.
Found 416 files belonging to 3 classes.

Found 435 files belonging to 3 classes.

['Early_Blight', 'Healthy', 'Late_Blight']
```

Custom Convolutional Neural Network (CNN)

```
# Define input shape and number of classes
input_shape = (IMAGE_SIZE, IMAGE_SIZE, CHANNELS)  # Input image dimensions
n_classes = len(class_names)                    # Number of output classes

# Preprocessing: resize and scale pixel values
resize_and_rescale = tf.keras.Sequential([
    layers.Resizing(IMAGE_SIZE, IMAGE_SIZE),     # Resize input images
    layers.Rescaling(1.0/255)                  # Normalize pixel values
])

# Data augmentation pipeline
data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal_and_vertical"),  # Random horizontal & vertical flip
    layers.RandomRotation(0.2)                  # Random rotation
])

# Define Custom CNN model
model_cnn = models.Sequential([
    layers.Input(shape=input_shape),  # Input layer
    resize_and_rescale,              # Apply resizing & rescaling
    data_augmentation,               # Apply data augmentation

    layers.Conv2D(32, (3,3), activation="relu"),  # Conv layer 1
    layers.MaxPooling2D(2,2),                  # Max pooling 1

    layers.Conv2D(64, (3,3), activation="relu"),  # Conv layer 2
    layers.MaxPooling2D(2,2),                  # Max pooling 2

    layers.Conv2D(64, (3,3), activation="relu"),  # Conv layer 3
    layers.MaxPooling2D(2,2),                  # Max pooling 3

    layers.Flatten(),                          # Flatten feature maps
    layers.Dense(64, activation="relu"),        # Fully connected layer
    layers.Dense(n_classes, activation="softmax")# Output layer
])

# Compile model
model_cnn.compile(
    optimizer="adam",                          # Optimizer
    loss="sparse_categorical_crossentropy",      # Loss function
    metrics=["accuracy"]                       # Evaluation metric
)

# Display model summary
model_cnn.summary()  # Print model architecture
```

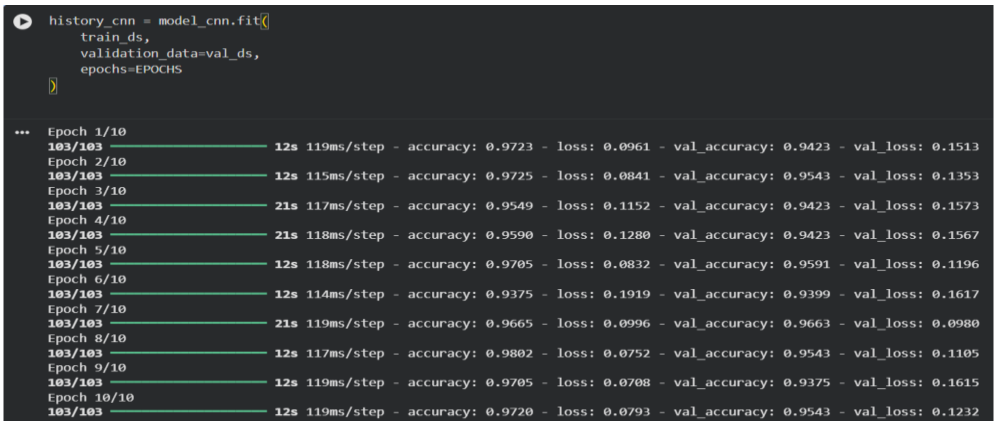
```
# Train the Custom CNN model
history_cnn = model_cnn.fit(
    train_ds,            # Training dataset
    validation_data=val_ds,  # Validation dataset
    epochs=EPOCHS        # Number of training epochs
)
# Evaluate the Custom CNN on the test dataset
loss_cnn, acc_cnn = model_cnn.evaluate(test_ds)  # Returns test loss and accuracy
print("CNN Test Accuracy:", acc_cnn)            # Display test accuracy

# Save the trained CNN model
model_cnn.save("/content/drive/MyDrive/Potato_CNN.keras")  # Save model to Google Drive
```

# Model 2 — VGG16

```
resize_vgg = tf.keras.Sequential([
    layers.Resizing(224,224),
    layers.Rescaling(1.0/255)
])

import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import VGG16
from tensorflow.keras.callbacks import ReduceLROnPlateau

# ---------------------------
# Hyperparameters
# ---------------------------
IMAGE_SIZE = 224
CHANNELS = 3
NUM_CLASSES = 3
EPOCHS = 25
BATCH_SIZE = 32

# ---------------------------
# Data Augmentation
# ---------------------------
data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal_and_vertical"),
    layers.RandomRotation(0.3),
    layers.RandomZoom(0.2),
    layers.RandomContrast(0.2),
    layers.RandomBrightness(0.2)
])

# ---------------------------
# Image resizing and rescaling
# ---------------------------
resize_vgg = tf.keras.Sequential([
    layers.Resizing(IMAGE_SIZE, IMAGE_SIZE),
    layers.Rescaling(1.0/255)
])

# ---------------------------
# Load VGG16 base
# --------------------------
vgg_base = VGG16(
    weights="imagenet",
    include_top=False,
    input_shape=(IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
)

# Freeze all layers first
for layer in vgg_base.layers:
    layer.trainable = False

# Fine-tune last 8 layers
for layer in vgg_base.layers[-8:]:
    layer.trainable = True

# --------------------------
# Build VGG16 Model
# --------------------------
vgg_model = models.Sequential([
    resize_vgg,
    data_augmentation,
    vgg_base,
    layers.GlobalAveragePooling2D(),
    layers.Dense(256, activation="relu"),   # more neurons
    layers.Dropout(0.5),                    # regularization
    layers.Dense(NUM_CLASSES, activation="softmax")
])

# --------------------------
# Compile Model
# --------------------------
vgg_model.compile(
    optimizer=tf.keras.optimizers.Adam(1e-4),  # higher LR for fine-tuning
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)

vgg_model.summary()

# --------------------------
# Callbacks
# --------------------------
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, verbose=1)

# --------------------------
# Train Model
# --------------------------
history = vgg_model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    callbacks=[reduce_lr]
```

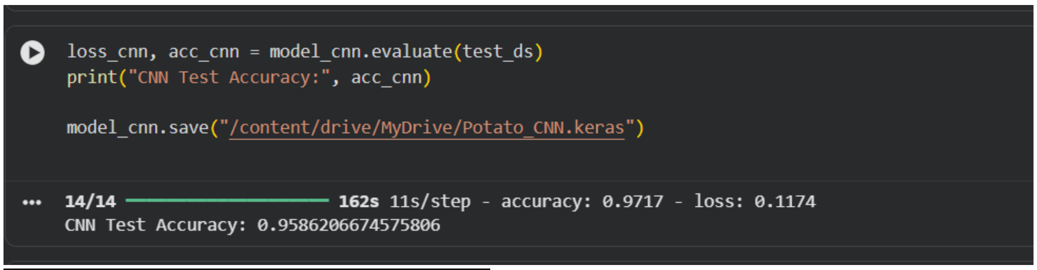
```
)

# Evaluate the pre-trained VGG16 model on the test dataset
vgg_loss, vgg_acc = vgg_model.evaluate(test_ds)  # Returns test loss and accuracy
print("✅ VGG16 Test Accuracy:", vgg_acc)        # Display VGG16 test accuracy


# TRAIN MOBILENETV2
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import MobileNetV2

# -------------------------------
# 1. Image Preprocessing
# -------------------------------
resize_and_rescale = tf.keras.Sequential([
    layers.Resizing(224, 224),     # Resize input images to 224x224
    layers.Rescaling(1.0/255)      # Normalize pixel values
])

# -------------------------------
# 2. Data Augmentation
# -------------------------------
data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal_and_vertical"),  # Random horizontal & vertical flip
    layers.RandomRotation(0.2),                 # Random rotation
    layers.RandomZoom(0.1),                     # Random zoom
])

# -------------------------------
# 3. Load Pre-trained MobileNetV2
# -------------------------------
mobile_base = MobileNetV2(
    weights="imagenet",      # Use ImageNet pre-trained weights
    include_top=False,       # Exclude default classification layer
    input_shape=(224, 224, 3)
)
mobile_base.trainable = False  # Freeze base layers initially

# -------------------------------
# 4. Build the Model
# -------------------------------
mobile_model = models.Sequential([
    resize_and_rescale,      # Preprocessing
    data_augmentation,       # Data augmentation
    mobile_base,             # Pre-trained MobileNetV2 base
    layers.GlobalAveragePooling2D(),  # Pool feature maps
    layers.Dropout(0.3),     # Dropout to prevent overfitting
    layers.Dense(128, activation="relu"),  # Fully connected layer
    layers.Dropout(0.3),     # Dropout
    layers.Dense(3, activation="softmax")  # Output layer (3 classes)
])

# -------------------------------
# 5. Compile the Model
# -------------------------------
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)  # Small LR for fine-tuning
mobile_model.compile(
    optimizer=optimizer,
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)

mobile_model.summary()  # Display model architecture

# -------------------------------
# 6. Optional: Fine-tuning
# -------------------------------
# Uncomment to fine-tune top layers after initial training
# mobile_base.trainable = True
# for layer in mobile_base.layers[:-30]:  # Freeze first layers, fine-tune last 30
#     layer.trainable = False
EPOCHS = 20

history = mobile_model.fit(
    train_ds,               # your training dataset
    validation_data=val_ds,    # your validation dataset
    epochs=EPOCHS,
    batch_size=32
)

# Step 1: Evaluate on test set
test_loss, test_acc = mobile_model.evaluate(test_ds)
print(f"Test Accuracy: {test_acc*100:.2f}%")
```

OUTPUT:



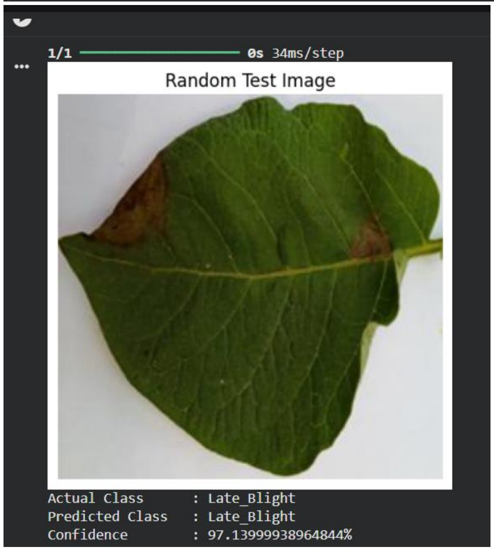
```
history_cnn = model_cnn.fit(
    train_ds,
    validation_data=val_ds,
    epochs=EPOCHS
)
```

```
Epoch 1/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 12s 119ms/step - accuracy: 0.9723 - loss: 0.0961 - val_accuracy: 0.9423 - val_loss: 0.1513
Epoch 2/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 12s 115ms/step - accuracy: 0.9725 - loss: 0.0841 - val_accuracy: 0.9543 - val_loss: 0.1353
Epoch 3/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 21s 117ms/step - accuracy: 0.9549 - loss: 0.1152 - val_accuracy: 0.9423 - val_loss: 0.1573
Epoch 4/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 21s 118ms/step - accuracy: 0.9590 - loss: 0.1280 - val_accuracy: 0.9423 - val_loss: 0.1567
Epoch 5/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 12s 118ms/step - accuracy: 0.9705 - loss: 0.0832 - val_accuracy: 0.9591 - val_loss: 0.1196
Epoch 6/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 12s 114ms/step - accuracy: 0.9375 - loss: 0.1919 - val_accuracy: 0.9399 - val_loss: 0.1617
Epoch 7/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 21s 119ms/step - accuracy: 0.9665 - loss: 0.0996 - val_accuracy: 0.9663 - val_loss: 0.0980
Epoch 8/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 12s 117ms/step - accuracy: 0.9802 - loss: 0.0752 - val_accuracy: 0.9543 - val_loss: 0.1105
Epoch 9/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 12s 119ms/step - accuracy: 0.9705 - loss: 0.0708 - val_accuracy: 0.9375 - val_loss: 0.1615
Epoch 10/10
103/103 ━━━━━━━━━━━━━━━━━━━━ 12s 119ms/step - accuracy: 0.9720 - loss: 0.0793 - val_accuracy: 0.9543 - val_loss: 0.1232
```



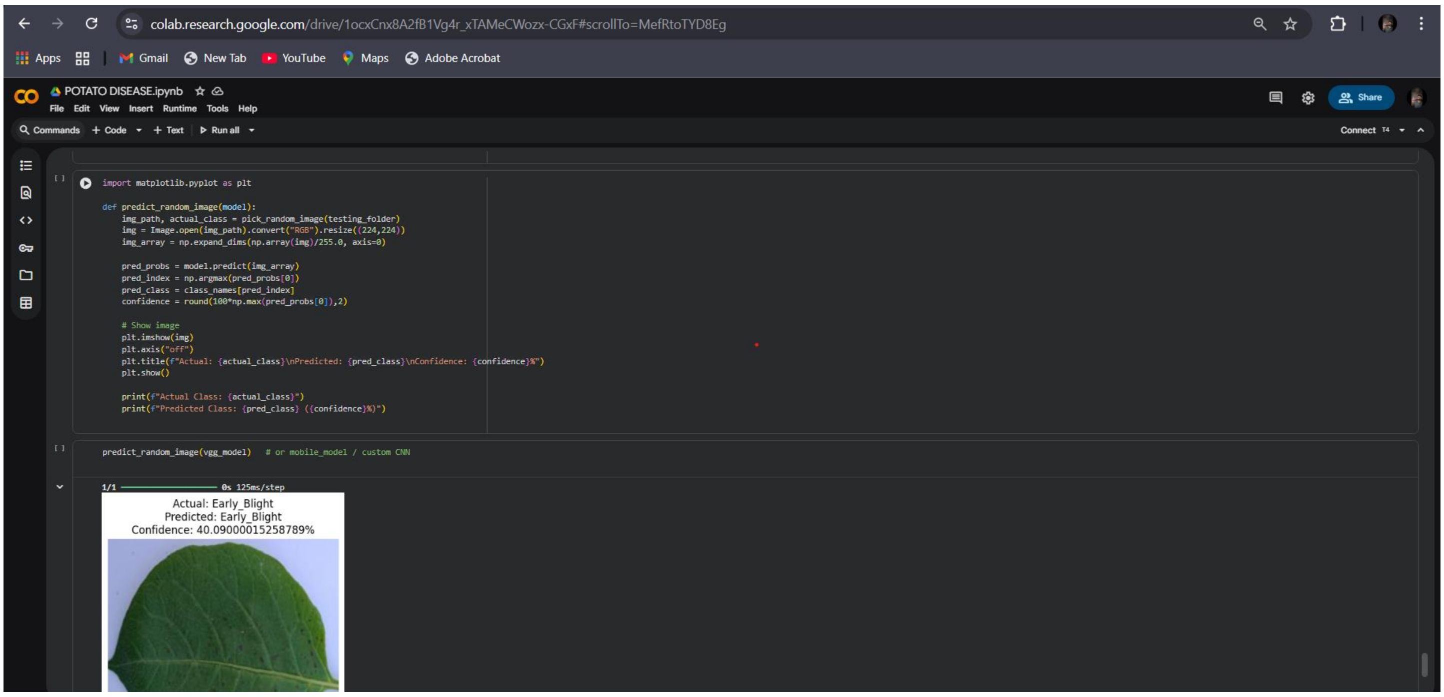
```
loss_cnn, acc_cnn = model_cnn.evaluate(test_ds)
print("CNN Test Accuracy:", acc_cnn)

model_cnn.save("/content/drive/MyDrive/Potato_CNN.keras")
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 162s 11s/step - accuracy: 0.9717 - loss: 0.1174
CNN Test Accuracy: 0.9586206674575806
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
```

```
Actual Class      : Late_Blight
Predicted Class   : Late_Blight
Confidence        : 97.13999938964844%
```



```
import matplotlib.pyplot as plt

def predict_random_image(model):
    img_path, actual_class = pick_random_image(testing_folder)
    img = Image.open(img_path).convert("RGB").resize((224,224))
    img_array = np.expand_dims(np.array(img)/255.0, axis=0)

    pred_probs = model.predict(img_array)
    pred_index = np.argmax(pred_probs[0])
    pred_class = class_names[pred_index]
    confidence = round(100*np.max(pred_probs[0]),2)

    # Show image
    plt.imshow(img)
    plt.axis("off")
    plt.title(f"Actual: {actual_class}\nPredicted: {pred_class}\nConfidence: {confidence}%")
    plt.show()

    print(f"Actual Class: {actual_class}")
    print(f"Predicted Class: {pred_class} ({confidence}%)")
```

```
predict_random_image(vgg_model)   # or mobile_model / custom CNN
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 125ms/step
```