AI Skills Final Project

Plant Disease Classification

Project Team:

ID	Name
931230343	هيثم محمد محمد أحمد
931230270	محمد مهدي زين العابدين محمد
931230067	اماني محمد امين سفينة
931230223	ليليان شحاته شكرى شحاته
931230386	يوسف يونان نصيف بدري
931230329	ميرا باسم ارنست

1)Project Overview

This project focuses on **classifying plant leaf diseases** using deep learning techniques.
The main goal is to assist in early and accurate agricultural disease diagnosis, which helps farmers improve crop yield and reduce losses.

The system uses convolutional neural networks (**CNN**s) trained on a labeled dataset of plant leaf images to identify various disease types.

Dataset:

- **Dataset Name:** PlantVillage Dataset
  https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset
- **Description:**
  The dataset contains labeled images of healthy and diseased plant leaves across multiple crop types.

2) Models

2.1 EfficientNetB3 Model

2.1.1 Overview:

EfficientNetB3 is an image classification model that achieves high accuracy with low computational cost. It was selected for plant disease classification because it can distinguish between visually similar diseases while remaining lightweight.
Transfer learning was applied using pre-trained **ImageNet** weights, enabling the model to reuse learned image features and adapt them to the **PlantVillage** dataset, which improves performance and reduces training time.

2.1.2 Data Preprocessing and Augmentation:

All image file paths were collected and stored with their labels in a DataFrame containing filepaths and labels (54,305 images). Classes with fewer than three images were removed, resulting in 38 classes.

Labels were encoded into numerical values to be compatible with the model. The dataset was then split into **training** (34,755 images), **validation** (8,689 images), and **test** (10,861 images) sets using stratified sampling to preserve class distribution.

Image preprocessing and augmentation were applied using ImageDataGenerator. The training set included rotations, shifts, shear, zoom, and horizontal flipping, while validation and test sets used only

preprocessing. Data generators were created for each split to feed images efficiently during training and evaluation.

2.1.3 Model Building and Training:

EfficientNetB3 with ImageNet pre-trained weights was used as the base model with the top layer removed. All base layers were frozen, and new layers were added: a dense layer with **256 units**, a dropout layer (0.4), and a softmax output layer.

EarlyStopping and ModelCheckpoint callbacks were used to prevent overfitting and save the best model. The model was compiled using the Adam optimizer (learning rate 0.0005), categorical crossentropy loss, and accuracy metric. Training was performed for 15 epochs, after which the best model was loaded for fine-tuning.

2.1.4 Fine-Tuning EfficientNetB3:

For fine-tuning, only the last 20 layers of the base model were unfrozen, while the remaining layers stayed frozen. The model was re-compiled with a lower learning rate (**1e-5**) and trained for **10 epochs**.

Training and validation accuracy increased steadily, while loss decreased, indicating improved learning of plant disease features. Accuracy and loss curves were plotted to visualize performance, and the best fine-tuned model was loaded for evaluation.

2.1.5 Model Evaluation on Test Dataset:

The fine-tuned model was evaluated on the test dataset, achieving a test accuracy of approximately 97% and a test loss of 0.098. A confusion matrix showed that most classes were correctly classified, with few misclassifications among similar disease types. A classification report was generated, showing strong precision, recall, and F1-scores across classes, with an overall accuracy of 0.97, macro-average F1-score of 0.95, and weighted-average F1-score of 0.97.

2.1.6 Explainability – Grad-CAM Visualization:

To explain the model's predictions and understand which image regions influenced the decision, **Grad-CAM** (Gradient-weighted Class Activation Mapping) was applied to the EfficientNetB3 model.

First, the best fine-tuned EfficientNetB3 model was loaded. To enable Grad-CAM visualization, the model was reconstructed without internal pooling layers, allowing access to spatial feature maps from the last convolutional layer. The original trained weights were then transferred to the new model, ensuring identical performance without retraining.

Grad-CAM works by computing the gradients of the predicted class with respect to the feature maps of the last convolutional layer. These gradients highlight the most important regions in the image that contributed to the model's prediction.

For each input image, the model predicts the disease class and confidence score. A heatmap is generated and overlaid on the original image, visually showing the areas that the model focused on when making its decision.

The Grad-CAM visualizations demonstrate that the model primarily attends to disease-affected regions on the plant leaves, which increases the interpretability and reliability of the classification results.

2.2 ResNet50 Model

2.2.1 Data Preprocessing and Dataset Splitting:

The **PlantVillage** dataset (**54,305** images) was used, organized into 38 plant disease classes. Images were processed using **PyTorch** transforms. For the training set, data augmentation techniques such as random horizontal flipping, random rotation, and color jittering were applied to improve model generalization.
All images were resized and center-cropped to a fixed size and normalized using ImageNet mean and standard deviation.

The dataset was split into training, validation, and test sets using stratified sampling to preserve class distribution. The splits consisted of 70% training (38,013 images), 20% validation (10,861 images), and 10% test (5,431 images). **DataLoaders** were created for each split to load images in batches of 32 during training and evaluation.

2.2.2 Model Building and Fine-Tuning (ResNet50)

A ResNet50 model pre-trained on the ImageNet dataset was used for image classification. All model parameters were initially frozen to preserve the learned low-level and mid-level features. The final fully connected layer was replaced with a new layer matching the number of plant disease classes. To allow fine-tuning, the parameters of the final classification layer and the last convolutional block (layer4) were unfrozen.

The model was trained using CrossEntropyLoss and the Adam optimizer with a low learning rate to ensure stable fine-tuning. A learning rate scheduler (StepLR) was applied to gradually reduce the learning rate every five epochs, improving convergence during training. This setup enables the model to adapt high-level features to the PlantVillage dataset while maintaining the robustness of the pre-trained **ResNet50** features.

2.2.3 Training the ResNet50 Model

The model was trained using a custom training loop with early stopping and full checkpoint saving. Training included both forward and backward passes with gradient updates, while validation was used to monitor performance each epoch.

Early stopping was applied with a patience of 7 epochs to halt training if no improvement in validation accuracy was observed. The best model and full training checkpoint were saved during training.

Training results showed rapid improvement in both training and validation accuracy. Validation accuracy peaked at 0.9962 (~99.6%) and validation loss reached 0.0135. Early stopping was triggered after 17 epochs, ensuring efficient training without overfitting.
This setup allowed the model to effectively learn high-level features for

plant disease classification while retaining the robustness of pre-trained ResNet50 features.

2.2.4 Model Evaluation on Test Dataset:

The model was evaluated on the test dataset to measure its performance and analyze predictions.

- **Test Accuracy:** achieved a high accuracy on unseen data: ~99.6%.
- **Classification Report:** Precision, recall, and F1-score were computed for each class. The report confirms that the model performs consistently well across most plant disease classes.
- **Confusion Matrix:** A confusion matrix was generated to visualize correct and incorrect predictions. Most classes are correctly predicted, with only a few misclassifications in visually similar disease types. The matrix provides insight into which classes are more challenging for the model.
- **Workflow:** Model weights were loaded from best_resnet50_model.pth. Evaluation was performed without gradient updates (eval() mode). Predictions and true labels were used to generate accuracy metrics, classification report, and the confusion matrix.

2.2.5 Explainability – Grad-CAM Visualization:

To interpret the ResNet50 model's predictions and identify which regions of the leaf images influenced its decisions, Grad-CAM (Gradient-weighted Class Activation Mapping) was applied.

- **Model Preparation:** The trained ResNet50 model was loaded for evaluation. Grad-CAM targets the last convolutional block (layer4) to access spatial feature maps.
- **Grad-CAM Mechanism:** Computes gradients of the predicted class with respect to the feature maps. Highlights image regions most responsible for the model's prediction.

- **Visualization:** For each input image, the model predicts the class and confidence. A heatmap is generated and overlaid on the original image. Colored bounding boxes and labels indicate true and predicted classes.
- **Outcome:** The visualizations show that the model focuses on disease-affected regions of leaves. Grad-CAM increases interpretability and reliability of the classification results, helping to understand both correct and misclassified predictions.

2.2.6 Grad-CAM GUI:

To provide an interactive way to visualize Grad-CAM results for ResNet50, a GUI application was created using **Tkinter**.

- **GUI Features:** Users can choose between visualizing images from the **dataset** or an **external image**. The number of dataset images to display can be specified. For external images, users can select the file and immediately see Grad-CAM results.
- **Grad-CAM Implementation in GUI:** Targets the last convolutional block (layer4) of ResNet50. Computes Grad-CAM heatmaps for the predicted class. Displays **side-by-side comparisons** of the original image and the Grad-CAM overlay. Uses colored bounding boxes and labels to indicate true and predicted classes.
- **Interactive Visualization:** The canvas dynamically updates to show multiple images or a single selected image. Prediction labels and Grad-CAM overlays help understand which regions the model focused on. Provides an intuitive tool to interpret model behavior and validate predictions visually.
- **Outcome:** The GUI demonstrates the model’s attention on disease-affected areas of leaves. Enhances interpretability and makes the model explainable to non-technical users.

2.3 MobileNet-V2:

2.2.1 Data Preprocessing and Augmentation:

All image file paths were collected from the dataset directory and stored along with their labels in a **DataFrame** containing **filepaths** and labels (54,284 images). All classes had at least three samples, so no classes were removed, resulting in 38 classes.

Labels were encoded into numerical values to be compatible with the model. The dataset was then split into training (34,741 images), validation (8,686 images), and test (10,857 images) sets using stratified sampling to preserve class distribution.

Image preprocessing and augmentation were applied using **ImageDataGenerator**. The training set included rotations, width and height shifts, shear, zoom, and horizontal flipping, while validation and test sets used only rescaling for normalization. Data generators were created for each split to feed images efficiently during training and evaluation.

2.2.2 Model Architecture and Compilation:

A **MobileNetV2** pre-trained model was used as the base model with **imagenet weights**. The top classification layer was removed (include_top=False) to allow customization for the plant disease classification task. The base model was **frozen** during initial training to leverage the pre-trained features.

On top of the base model, the following layers were added:

- **Global Average Pooling** to reduce the spatial dimensions of the feature maps.
- **Dropout (0.3)** for regularization to reduce overfitting.
- **Dense layer** with **softmax** activation for multi-class classification, where the number of neurons equals the number of classes (**38 classes**).

The model was compiled using:

- **Optimizer:** Adam
- **Loss function:** Categorical Crossentropy
- **Metrics:** Accuracy

A **ModelCheckpoint** callback was defined to save the best model weights during training based on **validation accuracy**. The checkpoint ensures that only the best-performing model is saved to 'best_plant_model.h5'.

2.2.3 Model Training and Checkpointing:

The MobileNetV2-based model was trained using the previously defined **training** and **validation** generators for **25 epochs**. The training process was monitored using the **ModelCheckpoint** callback, which saved the model weights whenever the **validation accuracy improved**.

Training observations:

- The model started with a training accuracy of ~68% in the first epoch and quickly improved.
- Validation accuracy consistently increased during the initial epochs, reaching **~94.6%** at the best epoch.
- Training and validation loss decreased steadily, demonstrating effective learning and generalization.
- The **best-performing model** based on validation accuracy was saved as best_plant_model.h5.

2.2.4 Model Evaluation on Test Set

Test results: **Test Accuracy:** 93.76% **Test Loss:** 0.202

2.2.5 Classification Report and Confusion Matrix:

The confusion matrix showed that the majority of classes were correctly classified, with occasional misclassifications in visually similar disease categories. The classification report demonstrated strong performance across classes, with high precision, recall, and F1-scores.

Overall, the model achieved an **accuracy of 0.938**, a **macro-average F1-score of 0.92**, and a **weighted-average F1-score of 0.94**, indicating robust performance and generalization on unseen plant disease images.

2.1.7 Explainability – Grad-CAM Visualization:

Grad-CAM was applied to interpret the MobileNetV2 model's predictions by highlighting image regions most influential for classification.

The **make_gradcam_heatmap** function computes gradients of the predicted class with respect to the last convolutional layer (**out_relu**) and generates a normalized heatmap. The **overlay_heatmap_on_image** function superimposes this heatmap on the original image for visual explanation.

The **visualize_grad_cam** function supports:

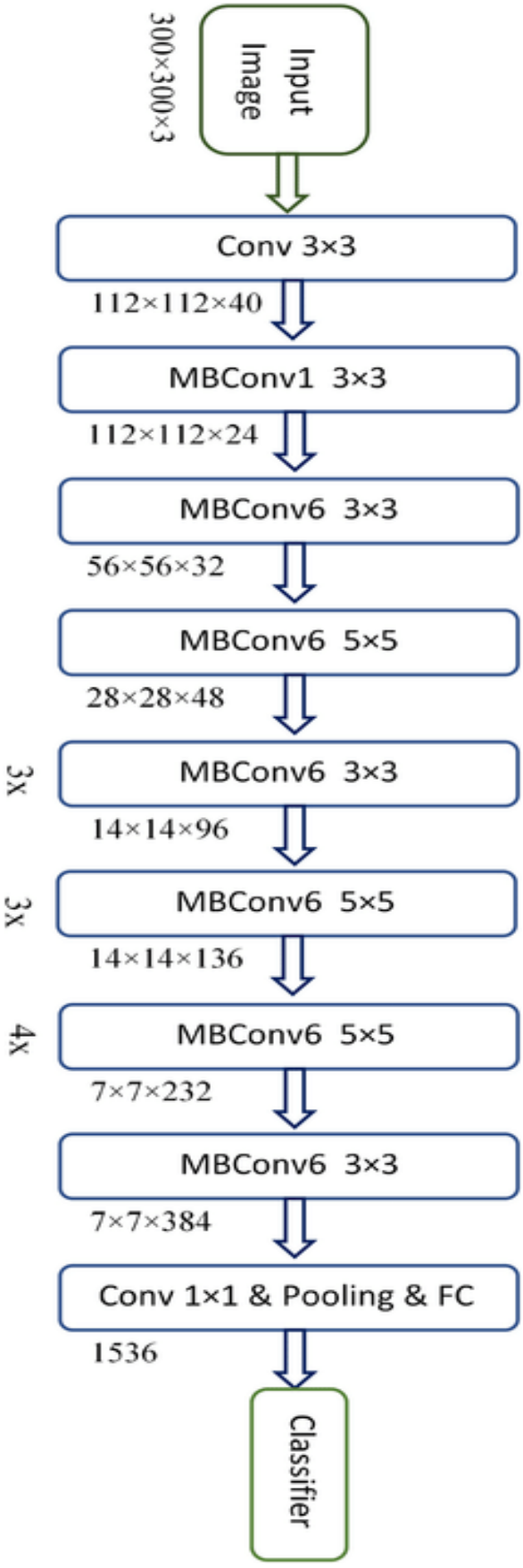
- **External images:** single image Grad-CAM visualization.
- **Dataset generators:** side-by-side original and Grad-CAM images for multiple samples, showing true and predicted labels with confidence.

The heatmaps confirm that the model focuses on disease-affected regions, improving interpretability.

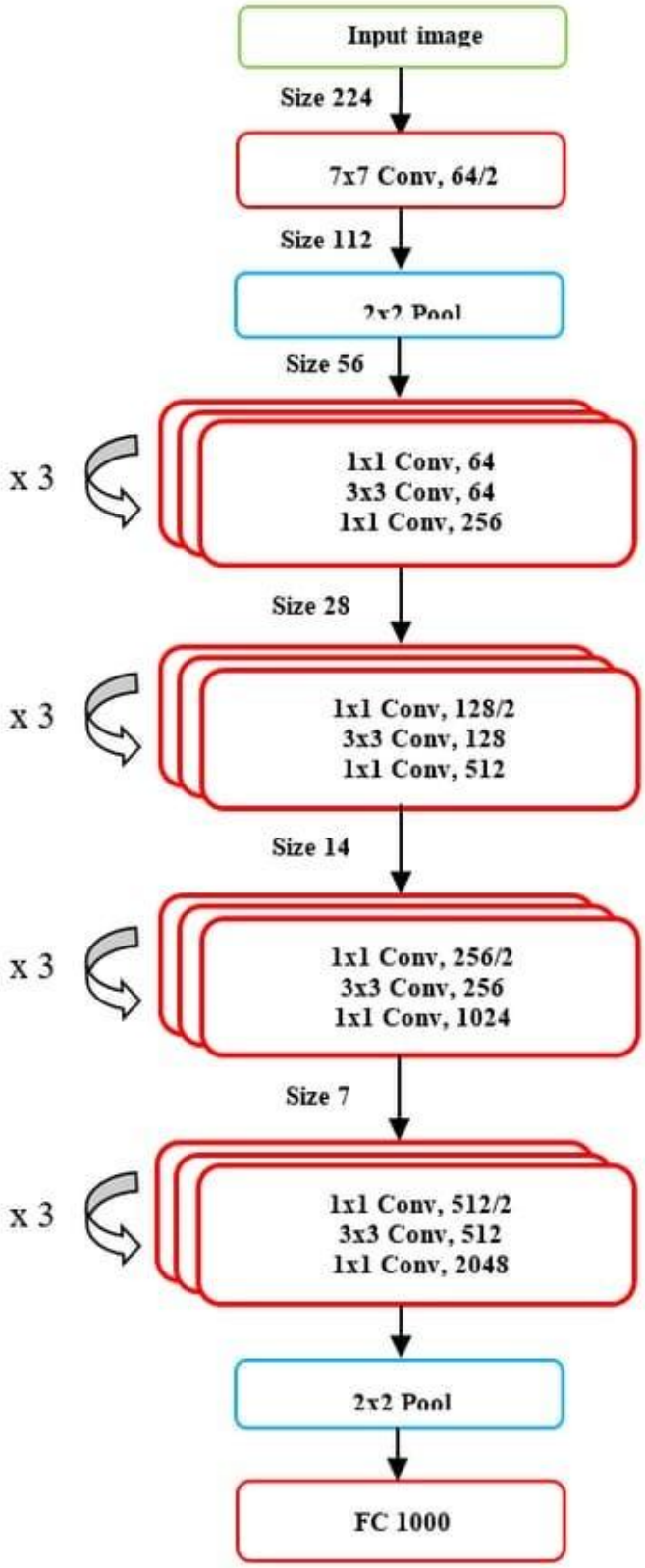
comparison between 3 architectures:

Metric	EfficientNetB3	ResNet50	MobileNetV2
Input Size	300×300	224×224	224×224
Trainable params	3,241,862	23,585,894	3,504,872
Test Accuracy	96%	99.28%	93%
Precision (Weighted)	0.97	0.99	0.94
Recall (Weighted)	0.97	0.99	0.94
F1-score (Weighted)	0.97	0.99	0.94
Training Time	09:15:00	06:30:00	10:00:00
Total params (Model Size)	17,670,499 (67.41 MB)	23,585,894 (~90 MB)	3,538,984 (13.50 MB)

EfficientNet B3 model architecture:



ResNet50model architecture:



MobileNetV2model architecture: