A white background with black dots

AI-generated content may be incorrect.

IMAGE PROCESSING

LEAF DISEASE DETECTION PROJECT

Faris Mahmoud Salah 202203798

Zeyad Hani Mahmoud 202202671

Bahaa Mohammed shams 202204811

Ahmed Ramadan Mahmoud 202204930

Osama Khalil Abd-Elfadeel 202202090

Ahmed Mohammed Abd-Elazeem 202203710

PRACTICAL EXAM

# A logo with a house and a sun AI-generated content may be incorrect.

# Leaf Disease Detection

# 1. Overview

This project implements a binary image classification model that uses Convolutional Neural Networks (CNNs) to detect whether a plant leaf is healthy or diseased. It is built using TensorFlow and Keras, with a focus on simple yet effective model architecture and data preprocessing techniques to ensure model robustness.

A diagram of a flatarg

AI-generated content may be incorrect.

# 2. Dataset and Preprocessing

2.1 Dataset Structure

The dataset follows a standard directory layout:

leaf\_dataset/  
├── train/  
│ ├── healthy/  
│ └── diseased/  
├── test/  
 ├── healthy/  
 └── diseased/

**2.2 Preprocessing Techniques**

- Rescaling: All images are resized to 128x128 and pixel values are normalized to [0, 1].  
- Augmentation: Used only during training to improve generalization.

**Example:**

**A screen shot of a computer program

AI-generated content may be incorrect.**

Rescaling: Pixel values are divided by 255 so they fall between 0 and 1, which helps neural networks converge faster.

Augmentation makes training more robust to rotation, lighting, and scale changes, mimicking the variability of real-world images.

**3. CNN Model Architecture**

The architecture is built using Keras' Sequential API:

A screenshot of a computer program

AI-generated content may be incorrect.

**Key Components:**

* **Feature Extraction: Convolutional layers detect visual patterns (edges, blobs, textures) specific to diseased vs. healthy leaves.**
  + - **Conv2D layers:** apply filters to extract low-level (edges) and mid-level (spots, lesions) features.
    - **MaxPooling2D:** down samples spatial dimensions, reducing computation and capturing the most salient features.
* **Flattening: Converts the 2D feature maps into a 1D vector to feed into fully connected layers.**
  + - **Flatten:** The CNN outputs a multi-channel feature map for example ( 14×14×64), which is flattened into a vector so it can be processed by the dense (fully connected) layers.
* **Classification:Dense layers interpret features and use a sigmoid function to predict whether the leaf is healthy or diseased.**
  + - Dense(128) captures combinations of features that correlate to disease.
    - Dropout(0.5) randomly disables neurons during training to prevent overfitting.
    - Dense(1) + sigmoid outputs a probability score between 0 (Diseased) and 1 (Healthy).

# 4. Model Training

Model uses Adam optimizer with binary crossentropy loss:

A screen shot of a computer code

AI-generated content may be incorrect.

# 5. Evaluation and Visualization

To track performance:A screenshot of a computer program

AI-generated content may be incorrect.

# **6. Prediction Logic**

A screen shot of a computer code

AI-generated content may be incorrect.

# **7. Algorithm Discussion and Improvements**

Strengths:  
- Simple, fast training  
- Augmentation enhances generalization

Limitations:  
- Can't classify disease types  
- Sensitive to complex backgrounds

Recommendations:  
- Use MobileNetV2 or EfficientNet (transfer learning)  
- Add multi-class classification  
- Apply attention modules like CBAM  
- Export to TensorFlow Lite for mobile use

# **8. Real-World Applications**

Precision Farming - Detect diseases early  
Agricultural Drones - Survey large areas  
Educational Tools - Teach plant pathology with AI

# **9. Conclusion**

The CNN is an effective baseline for leaf disease detection. For real-world deployment, enhancements like transfer learning, multi-class support, and background preprocessing are necessary.