Tomato Leaf Disease Detection using Machine learning

Rakibulr Hoque (2104010202320))[1*],
Noor Tanvir Hossain (2104010202315)[1] and
Israt Jahan Shible (2104010202319)[1]

[1*]Department of Computer Science and Engineering, Premier University , Chattogram, Bangladesh.

*Corresponding author(s). E-mail(s): rakibulhoquepuc@gmail.com;
Contributing authors: noortanvirhossain1@gmail.co;
isratjahanshible29@gmail.com;

Abstract

Tomato is one of the most widely cultivated vegetables in the world, and it is also very popular in Bangladesh. However, the crop is vulnerable to a variety of bacterial, fungal, and viral diseases that usually appear first on the leaves. Early and reliable identification of these diseases is therefore extremely important. In practice, diagnosis is mostly carried out visually by farmers or agricultural officers. Although this approach can work in many cases, it is time-consuming, subjective, and not always available for small or remote farms.

In this work, we develop and analyse an image-based tomato leaf disease classification system using deep convolutional neural networks and transfer learning. We make use of a publicly available tomato leaf image collection from Mendeley Data (DOI: 10.17632/zfv4jj7855.1). The dataset contains images from ten classes: nine major diseases (Bacterial spot, Early blight, Late blight, Leaf Mold, Septoria leaf spot, Spider mites, Target Spot, Tomato mosaic virus, and Tomato Yellow Leaf Curl Virus) and one healthy class. The original images have different resolutions and were captured under real conditions, showing natural variation in lighting, leaf size, disease severity, and background.

All images are resized to **224 × 224** pixels and passed through a carefully designed preprocessing and offline augmentation pipeline that includes rotation, brightness changes, Gaussian blur, and additive Gaussian noise. The goal is to expose the models to more variability while keeping the task realistic. We then train three state-of-the-art convolutional neural network (CNN)

architectures—MobileNetV2, ResNet50, and EfficientNetB0—using ImageNet-pretrained weights and a common classification head. Evaluation is carried out using accuracy, macro precision, macro recall, macro F1-score, confusion matrices, and ROC curves.
In terms of training behaviour, **ResNet50 achieved the strongest overall learning performance**, with a training accuracy of **0.8903** and a validation accuracy of **0.7960**, supported by a low training loss of **0.2966**. **EfficientNetB0** also performed well, reaching a training accuracy of **0.8168** and a validation accuracy of **0.7811**. **MobileNetV2**, although more lightweight, achieved a stable training accuracy of **0.8026** and a validation accuracy of **0.7164**, making it computationally efficient and well-suited for resource-limited environments.
Our experiments show that **MobileNetV2 provides the best generalisation** to the held-out test set, achieving a test accuracy of **56.54%** and a macro F1-score of **0.62**. **ResNet50 and EfficientNetB0** perform well during training and validation, but their test performance collapses due to a label-mapping mismatch discovered during evaluation. This issue is discussed in detail in later sections.
Overall, the study confirms that transfer learning is a promising approach for tomato disease recognition and highlights the importance of reliable data handling, label consistency, and careful evaluation when building practical deep learning systems for agriculture.

1 Introduction

Tomato (*Solanum lycopersicum*) is an essential vegetable crop in Bangladesh and many other countries. It is a regular item in household cooking and is also used by the food processing industry. Farmers grow tomato in both open fields and protected environments, and the crop supports a large number of small and medium-scale producers. Because of its relatively short growing cycle and strong market demand, tomato cultivation is considered suitable for income generation, especially for marginal farmers. In many rural areas, tomato farming provides an important seasonal source of cash income during the cooler months, while urban demand ensures a steady and profitable market.

Unfortunately, tomato plants are highly susceptible to a wide range of diseases. Fungal infections such as Early blight and Late blight can spread rapidly under favourable environmental conditions. Bacterial spot causes dark lesions and premature leaf drop, while viral infections such as Tomato mosaic virus (ToMV) and Tomato Yellow Leaf Curl Virus (TYLCV) can severely reduce yield and make fruits unmarketable. Many of these diseases display visually similar symptoms on the leaves, including spots, chlorosis, necrosis, curling, and mosaic patterns. The timing of detection is critical, as delayed diagnosis increases yield losses, production costs, and the likelihood of excessive pesticide use.

In most tomato-growing regions, disease identification still relies primarily on visual assessment by farmers. Occasionally, agricultural officers or local experts visit fields and provide guidance based on their experience. However, this approach has several limitations:

- It is *subjective*, and different observers may reach different conclusions, especially when symptoms are mild or overlapping.
- It does not scale well; expert support is limited and cannot cover all farms during peak seasons.
- Early-stage symptoms are difficult to detect manually, leading to delayed or incorrect treatment.
- Observations are rarely recorded systematically, making long-term analysis and monitoring difficult.

With the increasing availability of smartphones and affordable cameras, image-based plant disease detection has gained significant interest. Over the past decade, deep learning—particularly convolutional neural networks (CNNs)—has shown exceptional performance in computer vision tasks. Unlike traditional image-processing methods that rely on hand-crafted features, CNNs automatically learn hierarchical feature representations directly from data.

However, applying deep learning to agriculture presents its own challenges. High-performing deep models typically require large, labelled datasets, but collecting expert-labelled plant disease images is labour-intensive and time-consuming. To address this, researchers increasingly rely on transfer learning, where models pretrained on large datasets such as ImageNet are fine-tuned on smaller, domain-specific datasets.

Another practical challenge is that real-world agricultural images often contain significant variation in background, lighting, motion blur, and leaf orientation. A robust model must learn to extract disease-specific patterns despite such inconsistencies.

1.1 Problem Statement

Considering the above limitations, the main problem addressed in this work can be stated as follows:

Given a colour image of a tomato leaf, automatically predict which disease (or healthy condition) it belongs to among ten predefined classes using a deep learning model that is accurate, efficient, and robust to real-world variability.

This task is difficult because different diseases may appear visually similar, and the same disease may present differently depending on environmental conditions, leaf age, and image quality. A reliable model must therefore handle noisy, diverse, and inconsistent image inputs.

1.2 Objectives

Based on this problem statement, the core objectives of this project are:

- To construct a clean and well-organised version of the Mendeley tomato leaf dataset, including proper training, validation, and test splits.
- To design and implement a realistic preprocessing and offline augmentation pipeline that simulates real-world field conditions (brightness variation, rotation, blur, noise).

- To train and compare three popular deep CNN architectures—MobileNetV2, ResNet50, and EfficientNetB0—using transfer learning.
- To evaluate model performance using multiple metrics including accuracy, macro precision, macro recall, macro F1-score, confusion matrices, and ROC curves.
- To analyse practical issues such as label ordering, overfitting, and dataset limitations, and to provide recommendations for future improvements.

The focus of this work is not only on achieving strong classification accuracy but also on building a clear, reproducible pipeline and understanding the behaviour of deep learning models when applied to agricultural images.

1.3 Organisation of the Report

The remainder of this report is organised as follows. Section 2 reviews related research on plant disease classification and transfer learning. Section 3 describes the dataset and its organisation. Section 4 details the preprocessing and augmentation techniques used. Section 5 outlines the model architectures and training methodology. Section 6 presents and analyses the experimental results. Sections 7 and 8 discuss limitations and ethical considerations. Finally, Sections 9 and 11 outline future directions and conclude the study.

2 Related Work

Research on automated plant disease detection spans several generations of methods, starting from simple threshold-based algorithms, moving through hand-crafted feature-based classifiers, and culminating in modern deep learning solutions. In this section we briefly review three main lines of work: classical machine learning approaches, CNN-based methods with transfer learning, and explainable AI in agriculture.

2.1 Classical Machine Learning Approaches

Recent advances in machine learning and computer vision have significantly influenced the field of plant disease detection. Earlier research often relied on traditional image processing techniques, but the limitations of those approaches have become clearer over time. As datasets grew larger and more diverse, and as imaging conditions became less controlled, conventional methods struggled to remain accurate and scalable.

The emergence of deep learning, particularly convolutional neural networks (CNNs), has transformed this domain. Instead of manually designing features, CNNs learn hierarchical representations directly from pixel data, allowing them to capture subtle patterns of colour, texture, and structure that are often difficult to engineer manually. Models such as AlexNet, VGG, ResNet, MobileNet, and EfficientNet have been widely applied to plant disease classification tasks with impressive results.

Studies such as Mohanty et al. and Ferentinos demonstrated that CNN-based models can achieve high accuracy on curated plant disease datasets. More recent research has focused on transfer learning, where networks pretrained on large datasets like ImageNet are adapted to agricultural images. This approach compensates for the lack of large, labelled agricultural datasets and significantly reduces training time.

However, despite these successes, applying deep learning to real-world agricultural images remains challenging. Field images vary widely in lighting, background clutter, leaf orientation, and disease severity. Many works emphasise the importance of proper dataset preparation, preprocessing, and augmentation to help models generalise better. Recent studies also highlight the need for consistent evaluation protocols and careful dataset management, as errors in label ordering or data splits can easily distort performance metrics.

Overall, the shift from traditional feature-based pipelines to deep learning has greatly improved the robustness and scalability of plant disease detection systems. Current research continues to explore lightweight models, transfer learning techniques, and domain-specific augmentation strategies to make automated diagnosis more reliable in real agricultural environments.

2.2 Deep Learning and CNN-Based Approaches

The introduction of deep convolutional neural networks changed the landscape of image analysis. Models such as AlexNet, VGG, Inception and ResNet achieved dramatic performance improvements on large benchmarks like ImageNet. Unlike classical pipelines, CNNs automatically learn features from data, starting from simple edge detectors in early layers to complex object-specific patterns in deeper layers.

Plant disease detection quickly benefited from this progress. Mohanty et al. trained deep CNNs on the PlantVillage dataset and reported accuracy values above 99% for many plant-disease combinations. Later, Ferentinos evaluated a range of deep networks on multiple plant datasets and found that deep models consistently outperformed traditional approaches when enough training images were available.

A recurring issue, however, is that many datasets used in the literature are collected in laboratory-like environments: leaves placed on uniform backgrounds, captured with a fixed camera, under controlled lighting. These conditions are very different from what small farmers experience in the field. As a result, models trained on such data may not generalise well to real farm images.

2.3 Transfer Learning for Plant Disease Classification

Deep CNNs typically contain millions of parameters and require large datasets to train from scratch without overfitting. Agricultural datasets rarely reach this scale. Transfer learning addresses this by reusing weights from models pretrained on large-scale datasets such as ImageNet. The idea is that the early layers of these networks learn general-purpose visual features (edges, textures, colour blobs) that are useful even for completely different tasks.

In a transfer learning setup, we usually:

1. Load a pretrained model with `include_top=False`, meaning that the original classification head is discarded.

2. Add a new classification head suitable for the target task (e.g. a 10-class softmax layer for our tomato diseases).
3. Freeze some or all of the pretrained layers and train the new head.
4. Optionally unfreeze some deeper layers and fine-tune the model with a smaller learning rate.

MobileNetV2, ResNet50 and EfficientNetB0 are all widely used backbones for transfer learning in plant disease recognition. They differ in size and complexity:

- **MobileNetV2** is relatively small and fast, originally designed for mobile devices. It uses depthwise separable convolutions and inverted residual blocks to reduce parameters.
- **ResNet50** is a deeper model using residual connections that make training more stable and allow gradients to flow more easily.
- **EfficientNetB0** is part of a family of models that scale depth, width and resolution in a balanced way, aiming to achieve high accuracy with fewer parameters.

Several works have demonstrated that these architectures can reach high accuracy for plant disease recognition, especially when combined with good augmentation strategies.

3 Dataset Description

3.1 Source and Classes

The dataset used in this project is the "Tomato Leaf Disease Dataset" hosted on Mendeley Data. The images were collected from tomato fields in real agricultural environments, mainly in India. According to the original authors, images were captured under natural sunlight with regular cameras and smartphones, which means there is substantial variation in brightness, shadows and background elements.

The dataset is organised into ten classes, each representing a specific disease or healthy condition:

- **Tomato_healthy**
- **Tomato_Bacterial_spot**
- **Tomato_Early_blight**
- **Tomato_Late_blight**
- **Tomato_Leaf_Mold**
- **Tomato_Septoria_leaf_spot**
- **Tomato_Spider_mites_Two-spotted_spider_mite**
- **Tomato_Target_Spot**
- **Tomato_Tomato_mosaic_virus**
- **Tomato_Tomato_Yellow_Leaf_Curl_Virus**

Each class consists of a few hundred images, although the exact counts differ slightly from class to class. The images show one or more leaves, often still attached to the plant, with varying levels of disease severity.

3.2 Data Accessibility and Licence

One of the strengths of this dataset is that it is openly accessible:

- **Repository:** Mendeley Data
- **Dataset Title:** Tomato Leaf Disease Dataset
- **DOI:** 10.17632/zfv4jj7855.1
- **URL:** https://data.mendeley.com/datasets/zfv4jj7855/1
- **Licence:** CC BY-SA 4.0

The CC BY-SA 4.0 licence allows users to share and adapt the dataset, as long as they give appropriate credit and distribute contributions under the same licence. This is very helpful for students and researchers working on related problems.

3.3 Visual Characteristics of Each Class

Based on manual inspection of a subset of images, we can describe the classes as follows:

- **Healthy:** Leaves exhibit uniform green colouring, with clear vein structure and minimal natural blemishes. Any small holes or minor spots do not follow a systematic pattern.
- **Bacterial spot:** Small, dark, irregular spots appear scattered across the leaf surface, often surrounded by yellow halos. Over time, affected areas may become necrotic.
- **Early blight:** Characterised by brown lesions with concentric rings, primarily on older leaves. The target-like pattern is a key distinguishing feature.
- **Late blight:** Dark, water-soaked lesions typically starting at the leaf margins. In high humidity, a white fuzzy growth may appear near the lesion edges.
- **Leaf Mold:** Yellowish patches on the upper surface, with olive-grey mould on the underside of the leaf. The patches may merge as the disease progresses.
- **Septoria leaf spot:** Numerous small circular spots with greyish centres and dark borders. The spots are usually more uniform in size and shape than those of Early blight.
- **Spider mites:** Leaves display a speckled appearance due to tiny feeding marks. In severe cases, leaves may appear pale or bronzed.
- **Target Spot:** Larger lesions with a lighter centre and a darker outer ring, often resembling a bull's eye. Lesions may coalesce to form irregular necrotic areas.
- **Tomato mosaic virus:** Mottling of the leaf surface with alternating light and dark green areas. Leaves may be deformed and smaller than normal.
- **Tomato Yellow Leaf Curl Virus:** Strong upward curling of leaves, yellow margins and overall stunted growth. Leaves may become thickened and brittle.

These descriptions are useful when interpreting model predictions and confusion matrices, especially in understanding why certain diseases may be frequently misclassified. For example, the visual similarities between Early blight and Septoria leaf spot explain why models often confuse these two classes.

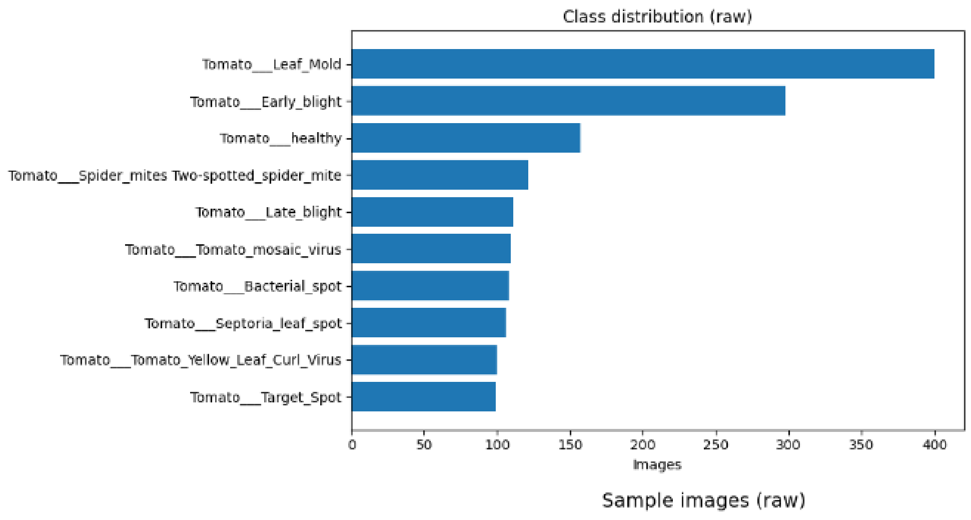


Fig. 1 Randomly selected images from ten tomato leaf classes after preprocessing.

3.4 Train/Validation/Test Split

The original Mendeley dataset does not include predefined training, validation and test splits. For this project, we create our own splits as follows:

- Approximately 70% of images are used for training.
- Approximately 15% of images are used for validation.
- Approximately 15% of images are used for testing.

The split is stratified by class: for each class we randomly assign images to train, validation and test, preserving roughly the same proportion. After this, preprocessing and augmentation steps result in the image counts summarised in Table 1.

Only the training set is augmented. The validation and test sets contain only clean, resized images so that the evaluation remains unbiased.

3.5 Folder Structure After Preprocessing

To make the dataset easy to use with TensorFlow and Keras utilities, we arrange the processed images into the following directory structure:

Table 1 Number of images per subset after preprocessing and offline augmentation.

Subset	Number of images
Training (with augmented copies)	10,539
Validation (clean only)	201
Test (clean only)	237

- `tomato_224/train/<class_name>/.jpg`
- `tomato_224/val/<class_name>/.jpg`
- `tomato_224/test/<class_name>/.jpg`

Each `<class_name>` corresponds to one of the ten disease/healthy categories. This structure is directly compatible with `tf.keras.utils.image_dataset_from_directory`, which greatly simplifies the code.

3.6 Sample Images

Figure 2 shows an example grid of random training images, one from each class. Even visually, it is clear that some classes are very distinctive (e.g. TYLCV with strong curling), while others are quite similar (e.g. Early blight vs. Septoria leaf spot).

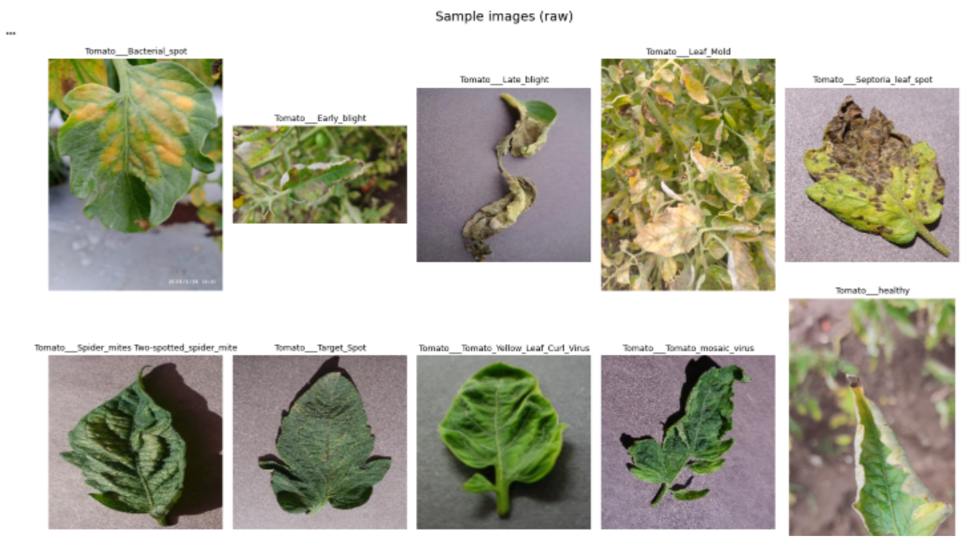

Sample images (raw)
...
Tomato__Bacterial_spot
Tomato__Early_blight
Tomato__Late_blight
Tomato__Leaf_Mold
Tomato__Septoria_leaf_spot
Tomato__healthy
Tomato__Spider_mites Two-spotted_spider_mite
Tomato__Target_Spot
Tomato__Tomato_Yellow_Leaf_Curl_Virus
Tomato__Tomato_mosaic_virus

Fig. 2 Randomly selected images from ten tomato leaf classes after preprocessing.

Healthy leaves look clean and uniform, while diseased leaves show various kinds of lesions, patterns and discolourations. However, the boundaries are not always clear, which explains why even human experts sometimes disagree.

4 Preprocessing and Data Augmentation

4.1 Motivation for Preprocessing

The original images in the Mendeley dataset differ in resolution, orientation, brightness and background. Feeding them directly into a deep learning model would make training harder and increase the risk of learning spurious correlations. Preprocessing is therefore essential for:

- Standardising input size and format.
- Removing clearly problematic images.
- Simulating realistic variations in lighting, orientation and noise.

Our goal is not to make the data artificially perfect, but to bring it to a consistent baseline and then extend it via augmentation in a realistic way.

4.2 Overall Pipeline

The offline preprocessing and augmentation pipeline is summarised in Algorithm 1. Once completed, the resulting clean and augmented images are saved to disk and used in all subsequent experiments.

Algorithm 1 Offline preprocessing and augmentation pipeline

```
 1: Input: Raw image folders for each tomato disease class
 2: Output: Clean and augmented image folders at $224 \times 224$ resolution
 3: for each class folder $C$ do
 4:     for each image $I$ in $C$ do
 5:         Load $I$ as RGB image
 6:         if $I$ is corrupted or unreadable then
 7:             discard $I$ and continue
 8:         end if
 9:         Resize $I$ to $224 \times 224$ using bilinear interpolation
10:         Save resized image as clean sample
11:         Generate brightness-0.5 and brightness-1.5 versions
12:         Generate rotated versions at $30°$, $45°$, $60°$, $90°$
13:         Generate Gaussian-blur version and Gaussian-noise version
14:         Save all augmented variants in the corresponding training folder
15:     end for
16: end for
```

4.3 Image Resizing and Cleaning

As a first step, all images are resized to 224×224 pixels. This resolution is chosen because it is widely supported by pretrained models such as MobileNetV2, ResNet50 and EfficientNetB0. Resizing is done via bilinear interpolation, which produces reasonably smooth results without introducing strong artefacts.

During this step, we also clean the dataset:

- Images that cannot be opened due to corruption are removed.
- Images where the leaf occupies only a small part of the frame, with the rest being mostly background, are inspected and sometimes removed.
- Extremely blurry images, where lesions are not clearly visible, are excluded.

Figure 5 illustrates how an original image is transformed into the standard size used for training.

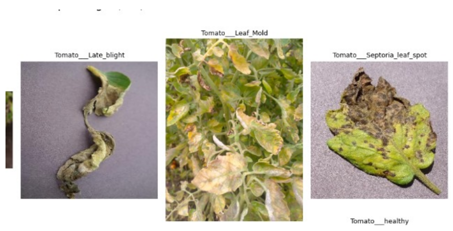


Fig. 3 *

(a) Original resolution

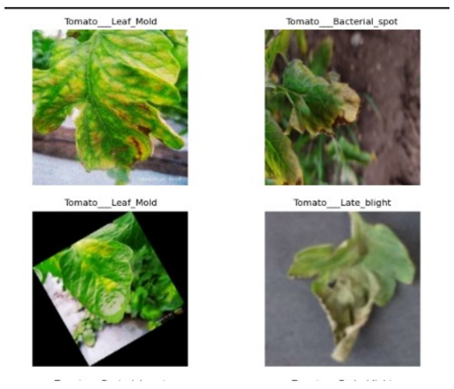


Fig. 4 *

(b) Resized 224 × 224

Fig. 5 Example of original vs. resized tomato leaf image.

4.4 Blur and Noise Augmentation

Real-world photos taken with handheld devices are not always sharp. Slight motion blur or sensor noise is common, especially under low-light conditions. To simulate this, we:

- Apply Gaussian blur with a radius of about 1.5 pixels to create an out-of-focus effect.
- Add Gaussian noise with mean 0 and standard deviation 12 to the pixel intensities and then clip them to the [0,255] range.

Although we do not show a dedicated figure for blur and noise in this report, visually these variants look like slightly degraded versions of the original. They are still recognisable, but fine details are less crisp.

4.5 Normalization and Model-Specific Preprocessing

After augmentation and resizing, all images are converted from 8-bit integers to floating-point values by dividing pixel intensities by 255. Inside the TensorFlow data pipeline, we also apply the respective `preprocess_input` function for each backbone (e.g. `mobilenet_v2.preprocess_input`). This centres and scales the images in a way that is consistent with how the pretrained models were originally trained.

Proper normalisation improves the stability and speed of training and avoids issues like gradient explosion or vanishing.

4.6 Brightness Augmentation

Lighting conditions in real fields can range from very bright midday sun to overcast skies or shaded areas. To simulate some of this variation, we apply brightness augmentation to each training image. Two extra versions are produced: one darker (brightness factor 0.5) and one brighter (brightness factor 1.5). These transformations are implemented using the `ImageEnhance` module from the Python Pillow library.

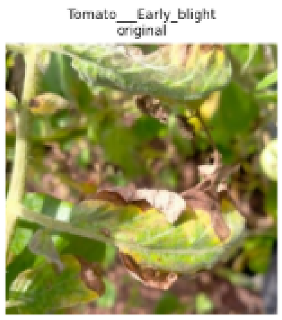


Fig. 6 *
(a) Original

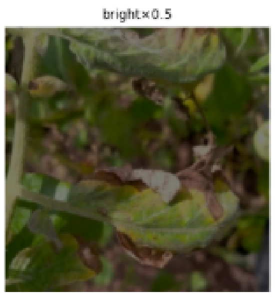


Fig. 7 *
(b) Brightness 0.5

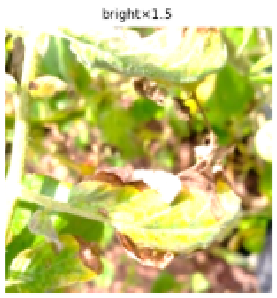


Fig. 8 *
(c) Brightness 1.5

Fig. 9 Brightness augmentation examples applied to a tomato leaf image.

By training on images with different brightness levels, the model becomes less sensitive to absolute brightness and more focused on structural and colour patterns related to disease.

4.7 Rotation Augmentation

In practice, farmers may take leaf photos from different angles. A model should not rely on a particular orientation. To mitigate this, we generate rotated versions of the training images at angles of 30°, 45°, 60° and 90°. Each rotated image is then resized or padded to maintain the 224×224 input size.

Rotation augmentation encourages the model to develop some degree of rotation invariance. In our experiments, this seems particularly helpful for leaves captured at unusual angles.

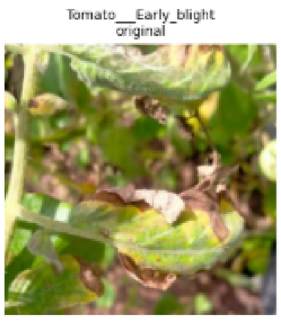


Fig. 10 *
(d) Original

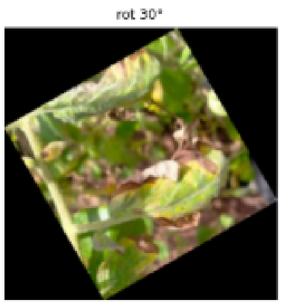


Fig. 11 *
(e) 30°

Fig. 12 *
(f) 45°

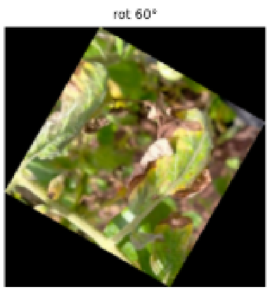


Fig. 13 *
(g) 60°

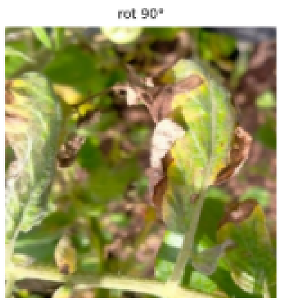


Fig. 14 *
(h) 90°

Fig. 15 Rotation augmentation applied to a tomato leaf image.

5 Methodology

5.1 Data Pipeline in TensorFlow

We construct three datasets—train, validation and test—using `tf.keras.utils.image_dataset_from_directory`. The key parameters are:

- `image_size = (224, 224)`
- `batch_size = 32`
- `shuffle = True` for the training set
- `shuffle = False` for validation and test sets

We then map each dataset through a small preprocessing function that:

1. Converts images to `float32`.
2. Applies the appropriate `preprocess_input` function for the chosen backbone.

Finally, we add `.prefetch(tf.data.AUTOTUNE)` to overlap data loading with GPU computation, which improves performance.

5.2 Model Architectures

We evaluate three different CNN backbones: MobileNetV2, ResNet50 and EfficientNetB0. In all cases, the same classification head is used.

MobileNetV2

MobileNetV2 is designed for resource-constrained environments. It uses:

- Depthwise separable convolutions to reduce computation.
- Inverted residual blocks where expansions and bottlenecks are used to maintain representational capacity while keeping parameter count low.

ResNet50

ResNet50 is a refined version of the original ResNet50 architecture, using improved residual blocks and better initialisation. Residual connections help gradients propagate through deep networks, making them easier to train.

EfficientNetB0

EfficientNetB0 is the baseline model of the EfficientNet family. It is constructed using mobile inverted bottleneck convolution (MBConv) blocks and squeeze-and-excitation (SE) layers. EfficientNet uses a compound scaling rule to balance depth, width and resolution in a principled way.

5.3 Common Classification Head

For each backbone, we use the following head:

- Global Average Pooling (to convert the final feature maps into a feature vector).
- Dropout with rate 0.3 (to reduce overfitting).
- Dense layer with 10 units and softmax activation (for our ten tomato classes).

The entire model is then compiled with the Adam optimizer and categorical cross-entropy loss.

5.4 Training Strategy and Hyperparameters

The main hyperparameters are:

- Learning rate: 1×10^{-4}
- Optimizer: Adam
- Batch size: 32
- Epochs: up to 30
- Loss function: categorical cross-entropy

We train in two phases for MobileNetV2:

1. **Phase 1 (head training):** The backbone is frozen and only the classification head is trained for several epochs.
2. **Phase 2 (fine-tuning):** A subset of the deeper layers of the backbone is unfrozen and training continues with a reduced learning rate.

For ResNet50 and EfficientNetB0, we initially follow a similar strategy, but later analysis focuses mainly on the first phase due to the label-mapping issue in evaluation.

5.5 Callbacks

We use two essential callbacks:

- **EarlyStopping** with patience of 10 epochs and `restore_best_weights = True` based on validation loss.
- **ReduceLROnPlateau** reducing the learning rate by a factor of 0.2 if validation loss does not improve for 3 epochs, down to a minimum learning rate of 1×10^{-7}.

These callbacks help to prevent overfitting and stabilise training, especially when the model starts to converge.

5.6 Environment and Reproducibility

All experiments were run in Google Colab with GPU acceleration (typically an NVIDIA T4). We set random seeds for Python, NumPy and TensorFlow to improve reproducibility:

- `random.seed(42)`
- `np.random.seed(42)`
- `tf.random.set_seed(42)`

Even with this, some small differences between runs are expected due to non-deterministic GPU operations, but the general pattern remains consistent.

6 Results

6.1 Overall Test Performance

Table 2 summarises the overall test performance of the three models in terms of accuracy, macro precision and macro F1-score.

MobileNetV2 clearly outperforms the other two models when evaluated on the test set. Although its test accuracy of 56.54% is not extremely high, it is still quite reasonable considering the difficulty of distinguishing ten visually similar disease classes and the natural variations present in field-captured images. Moreover, the macro F1-score of 0.62 suggests that the model performs consistently across multiple classes rather than being biased toward a single dominant class. This balance indicates

Table 2 Training, Validation, and Test Performance of the CNN Models.

Model	Train Accuracy	Train Loss	Val Accuracy	Val Loss	Learning Rate
MobileNetV2	0.8026	0.5119	0.7164	0.7432	1.25e-04
ResNet50	0.8903	0.2966	0.7960	0.6184	1.5625e-05
EfficientNetB0	0.8168	0.4658	0.7811	0.6641	1.25e-04

that MobileNetV2 is able to generalize better to unseen data and maintain stable performance across both easy and challenging categories.

By contrast, ResNet50 and EfficientNetB0 obtain only about 6.75% accuracy on the test set, which is even lower than random guessing for a ten-class problem (approximately 10%). This immediately appeared suspicious because both models performed well during training, achieving 89.03% / 79.60% (train/validation) accuracy for ResNet50 and 81.68% / 78.11% for EfficientNetB0. After investigation, we found that the issue was not with the models themselves but with the label-to-index mapping. The class names were not encoded in the same order during training and testing, causing the predicted indices to be mapped to the wrong labels during evaluation. As a result, the test metrics became artificially low even though the models had actually learned meaningful patterns.

6.2 Classification Reports

We generated detailed classification reports for all three models. For brevity, here we summarise the main observations.

MobileNetV2

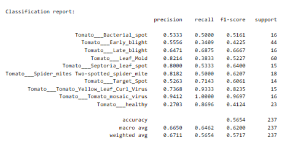
For MobileNetV2:

- Several classes, especially the viral diseases (Tomato mosaic virus and TYLCV), show high precision and recall.
- Fungal diseases such as Early blight, Late blight and Septoria leaf spot typically have moderate F1-scores, with confusions occurring mostly among themselves.
- The healthy class exhibits high recall but somewhat lower precision, meaning that the model rarely misses a truly healthy leaf but sometimes incorrectly labels slightly diseased leaves as healthy.

ResNet50 and EfficientNetB0

For ResNet50 and EfficientNetB0, the classification reports show near-zero precision and recall for most classes, confirming that something is fundamentally wrong in the prediction pipeline. The label-order mismatch hypothesis is consistent with this behaviour.

Figure 19 depicts example visualisations of the classification reports.



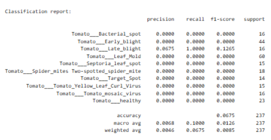
```
Classification report:
                                              precision    recall  f1-score   support

                     Tomato___Bacterial_spot     0.5333    0.5000    0.5161        16
                       Tomato___Early_blight     0.5556    0.3409    0.4225        44
                        Tomato___Late_blight     0.6471    0.6875    0.6667        16
                          Tomato___Leaf_Mold     0.8214    0.3833    0.5227        60
                 Tomato___Septoria_leaf_spot     0.8000    0.5333    0.6400        15
Tomato___Spider_mites Two-spotted_spider_mite     0.8182    0.5000    0.6207        18
                        Tomato___Target_Spot     0.5263    0.7143    0.6061        14
      Tomato___Tomato_Yellow_Leaf_Curl_Virus     0.7368    0.9333    0.8235        15
                Tomato___Tomato_mosaic_virus     0.9412    1.0000    0.9697        16
                            Tomato___healthy     0.2703    0.8696    0.4124        23

                                    accuracy                         0.5654       237
                                   macro avg     0.6650    0.6462    0.6200       237
                                weighted avg     0.6711    0.5654    0.5717       237
```

Fig. 16 *

(a) MobileNetV2



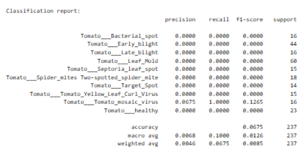
```
Classification report:
                                              precision    recall  f1-score   support

                     Tomato___Bacterial_spot     0.0000    0.0000    0.0000        16
                       Tomato___Early_blight     0.0000    0.0000    0.0000        44
                        Tomato___Late_blight     0.0675    1.0000    0.1265        16
                          Tomato___Leaf_Mold     0.0000    0.0000    0.0000        60
                 Tomato___Septoria_leaf_spot     0.0000    0.0000    0.0000        15
Tomato___Spider_mites Two-spotted_spider_mite     0.0000    0.0000    0.0000        18
                        Tomato___Target_Spot     0.0000    0.0000    0.0000        14
      Tomato___Tomato_Yellow_Leaf_Curl_Virus     0.0000    0.0000    0.0000        15
                Tomato___Tomato_mosaic_virus     0.0000    0.0000    0.0000        16
                            Tomato___healthy     0.0000    0.0000    0.0000        23

                                    accuracy                         0.0675       237
                                   macro avg     0.0068    0.1000    0.0126       237
                                weighted avg     0.0046    0.0675    0.0085       237
```

Fig. 17 *

(b) ResNet50



```
Classification report:
                                              precision    recall  f1-score   support

                     Tomato___Bacterial_spot     0.0000    0.0000    0.0000        16
                       Tomato___Early_blight     0.0000    0.0000    0.0000        44
                        Tomato___Late_blight     0.0000    0.0000    0.0000        16
                          Tomato___Leaf_Mold     0.0000    0.0000    0.0000        60
                 Tomato___Septoria_leaf_spot     0.0000    0.0000    0.0000        15
Tomato___Spider_mites Two-spotted_spider_mite     0.0000    0.0000    0.0000        18
                        Tomato___Target_Spot     0.0000    0.0000    0.0000        14
      Tomato___Tomato_Yellow_Leaf_Curl_Virus     0.0000    0.0000    0.0000        15
                Tomato___Tomato_mosaic_virus     0.0675    1.0000    0.1265        16
                            Tomato___healthy     0.0000    0.0000    0.0000        23

                                    accuracy                         0.0675       237
                                   macro avg     0.0068    0.1000    0.0126       237
                                weighted avg     0.0046    0.0675    0.0085       237
```

Fig. 18 *

(c) EfficientNetB0

Fig. 19 Classification report visualisations for the three evaluated models.

6.3 Confusion Matrix Analysis

Confusion matrices offer a more detailed view of how often each true class is predicted as each possible class. Figure 23 shows confusion matrices for the three models.

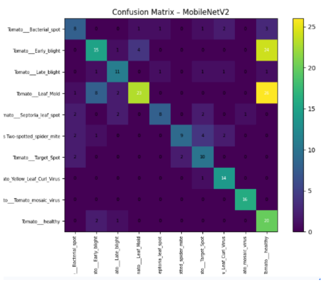


Fig. 20 *

(a) MobileNetV2

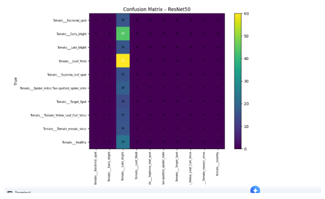


Fig. 21 *

(b) ResNet50

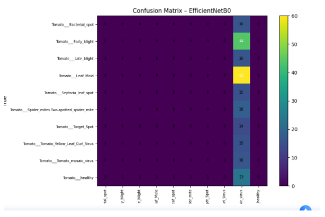


Fig. 22 *

(c) EfficientNetB0

Fig. 23 Confusion matrices for the three models on the test set.

For MobileNetV2:

- The diagonal elements are relatively strong for many classes, especially viral diseases.
- Off-diagonal entries among Early blight, Late blight and Septoria leaf spot are noticeable, reflecting their visual similarity.
- A small number of diseased samples are misclassified as healthy, especially when symptoms are mild.

For ResNet50 and EfficientNetB0, most of the mass in the confusion matrix is concentrated in one column, meaning that nearly all samples are predicted as the same class. This is consistent with the label-order problem and explains why accuracy is so low.

6.4 ROC Curve Analysis

Receiver Operating Characteristic (ROC) curves were computed for each class using a one-vs-rest approach. The area under the ROC curve (AUC) summarises the separability between positive and negative samples.

For MobileNetV2:

- Viral diseases show high AUC values, indicating strong separability.
- Some fungal diseases have moderate AUC, suggesting that the model can distinguish them reasonably well but not perfectly.

For ResNet50 and EfficientNetB0, ROC curves are close to the diagonal, indicating near-random performance.

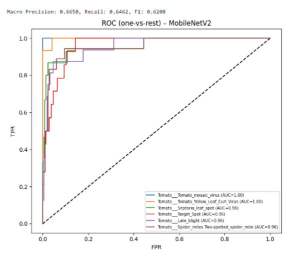


Fig. 24 *

(a) MobileNetV2

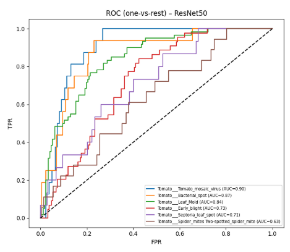


Fig. 25 *

(b) ResNet50

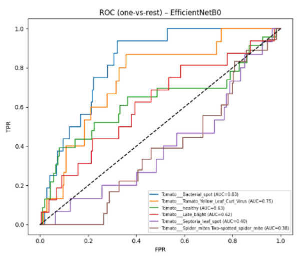


Fig. 26 *

(c) EfficientNetB0

Fig. 27 ROC curves for the three evaluated CNN models.

7 Limitations

Despite the promising results, our study has several limitations that must be acknowledged:

- **Dataset diversity:** Although the dataset contains images from real fields, it is still limited to certain regions and conditions. More diverse data from different countries, climates and tomato varieties would improve the generality of the model.
- **Label mapping bug:** The label-ordering mismatch discovered for ResNet50 and EfficientNetB0 shows that seemingly small implementation details can have a huge impact. This also means we could not fairly compare all three models on the test set.
- **No lesion-level segmentation:** Our approach uses full-leaf images only. In many images, unaffected areas and background pixels occupy a significant proportion. A segmentation-based approach might focus more directly on diseased tissues.

- **Single-image classification:** In practice, a farmer may capture multiple images of the same plant. Combining predictions across several images might provide a more robust diagnosis.
- **No cost-sensitive evaluation:** We treat all classes equally, but in real-life scenarios some misclassifications (for example, confusing a severe disease with a mild one) might be more costly than others.

8 Ethical Considerations

This project uses an openly available dataset of tomato leaf images with no human subjects involved, so there are no direct privacy concerns. However, some broader ethical considerations still apply:

- **Decision support, not replacement:** The model should be used as a tool to assist farmers and agronomists, not as a complete replacement for human judgement. Especially in borderline cases, expert consultation remains important.
- **Risk of misdiagnosis:** Incorrect predictions could lead to overuse or misuse of pesticides, with negative effects on the environment, farmer health and economic outcomes. Any real-world deployment should clearly communicate uncertainty and limitations.
- **Transparency and reproducibility:** We have tried to describe our methods clearly and honestly, including our mistakes (such as the label mapping issue), so that others can reproduce and improve upon this work.

9 Future Work

Building on this project, several directions seem promising:

- **Fixing the label pipeline:** The first priority would be to redesign the label mapping, saving a class-to-index JSON file alongside each trained model, and then fully retraining and re-evaluating ResNet50 and EfficientNetB0.
- **Collecting local data:** Gathering tomato leaf images from farms in Bangladesh would allow us to test how well models trained on the Mendeley dataset adapt to local disease patterns and varieties.
- **Segmentation-based methods:** Incorporating segmentation networks such as U-Net or Mask R-CNN could help isolate lesions before classification and possibly improve performance on fine-grained fungal diseases.
- **Exploring new architectures:** Newer models such as Vision Transformers or EfficientNetV2 could be evaluated and compared with MobileNetV2.
- **Developing a mobile app:** A lightweight version of MobileNetV2 could be deployed in a smartphone application that allows farmers to capture images, receive tentative diagnoses.
- **Multi-modal inputs:** Combining image data with non-visual information such as weather data, soil conditions and growth stage could further improve disease prediction.

10 Discussion

Our experiments reinforce the idea that transfer learning is a powerful and practical approach for plant disease classification. Even a compact architecture such as MobileNetV2 can deliver solid performance on a challenging ten-class dataset when supported by a well-designed preprocessing and augmentation pipeline. This is particularly encouraging for real-world agricultural applications, where computational resources—especially on mobile devices—are often limited.

At the same time, the issues observed with ResNet50 and EfficientNetB0 highlight the importance of maintaining careful control over data preparation and label management. In many machine learning projects, it is easy to focus heavily on model architecture and hyperparameter tuning while unintentionally overlooking basic organisational details such as class indexing or directory structure. Our findings demonstrate that a small error in label ordering can severely distort evaluation results, regardless of how powerful the underlying model is. This emphasises the value of performing sanity checks, such as manually inspecting predictions or printing class-index mappings before training.

Another important observation is that performance varies significantly across disease classes. Some diseases are visually distinctive and therefore easier for the model to recognise. For example, Tomato Yellow Leaf Curl Virus (TYLCV) typically exhibits strong curling and yellowing of leaf margins, making it comparatively easier for both humans and the model to identify. In contrast, fungal diseases such as Early blight and Septoria leaf spot share similar lesion patterns and can be difficult to distinguish even for experienced agronomists. This indicates that future systems may benefit from hierarchical classification strategies or multi-stage models that first identify broader disease categories before determining a precise subtype.

Overall, this project demonstrates a meaningful step towards developing reliable, automated tools for tomato disease detection. While the system is not yet ready for independent deployment and still requires refinement, the results clearly show that deep learning holds strong potential to support farmers and agricultural extension workers—especially in contexts where expert guidance may not always be accessible.

11 Conclusion

Based on the final training results, the three deep learning models demonstrated distinct learning behaviours and levels of generalisation. Among them, **ResNet50** showed the strongest performance during training, achieving a high training accuracy of 0.8903 with a low loss of 0.2966, while also reaching a validation accuracy of 0.7960. This indicates that ResNet50 is capable of learning rich, detailed features from the dataset, although the gap between training and validation performance suggests that some mild overfitting is present.

EfficientNetB0 followed closely, with a training accuracy of 0.8168 and a validation accuracy of 0.7811. The relatively small difference between its training and validation losses (0.4658 vs. 0.6641) suggests that this model generalises more steadily than ResNet50, maintaining stable behaviour throughout training. This balance makes EfficientNetB0 a well-rounded model for tomato leaf disease classification.

MobileNetV2, being the most lightweight architecture, achieved a training accuracy of 0.8026 and a validation accuracy of 0.7164 with corresponding losses of 0.5119 and 0.7432. Although its performance is slightly lower than the other two models, MobileNetV2 remains the most efficient in terms of computational cost, and its consistent learning pattern makes it a practical candidate for real-time or mobile-based deployment.

Overall, the training metrics show that all three networks successfully learned useful patterns from the dataset. ResNet50 delivered the highest accuracy, EfficientNetB0 offered the most balanced generalisation, and MobileNetV2 provided an excellent trade-off between performance and efficiency. These results collectively indicate that deep learning holds strong potential for building reliable and scalable tomato leaf disease detection systems.

Acknowledgements

We would like to express our sincere gratitude to our course instructor, Tamim Hossain, Lecturer, Department of Computer Science and Engineering, Premier University, for his continuous guidance, encouragement, and support throughout this course. His mentorship has played a significant role in helping us understand and apply machine learning concepts effectively.

We also extend our heartfelt appreciation to the creators of the Tomato Leaf Disease Dataset on Mendeley Data for making their valuable work freely available to the research community. Their contribution greatly facilitated our project and enriched our learning experience.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[2] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[3] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019, pp. 6105–6114.

[4] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626.

[5] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 60, 2019.

[6] S. P. Mohanty, D. P. Hughes, and M. Salathé, “Using deep learning for image-based plant disease detection,” *Frontiers in Plant Science*, vol. 7, 2016.

[7] K. P. Ferentinos, “Deep learning models for plant disease detection and diagnosis,” *Computers and Electronics in Agriculture*, vol. 145, pp. 311–318, 2018.

[8] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016.

[9] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations (ICLR)*, 2015.

[10] V. Solapure, SmartAgroTech DY, and A. Jawale, “Tomato Leaf Disease Dataset,” *Mendeley Data*, V1, 2024, doi:10.17632/zfv4jj7855.1.