A logo of a plant with a magnifying glass

AI-generated content may be incorrect.

**Cereals disease detection**

EfficientNet-B0 Based Image Classification

**Version**

1.0

**Authors**

Sayf Rourou

Table of Contents

[**1. Introduction** 3](#_Toc190682769)

[**2. System components** 3](#_Toc190682770)

[2.1 Dataset preparation 3](#_Toc190682771)

[2.2 Model architecture 3](#_Toc190682772)

[2.3 Training configuration 4](#_Toc190682773)

[2.4 Early stopping mechanism 4](#_Toc190682774)

[**3. Training and evaluation** 5](#_Toc190682775)

[3.1 Training 5](#_Toc190682776)

[3.2 Testing and evaluation 5](#_Toc190682777)

[**4. Results** 5](#_Toc190682778)

[4.1 Test metrics 5](#_Toc190682779)

[4.2 Classification report 6](#_Toc190682780)

[4.3 Strengths 7](#_Toc190682781)

[4.4 Weaknesses 7](#_Toc190682782)

# **1. Introduction**

The goal of this project is to classify images into predefined categories using deep learning techniques including:

* **Data preprocessing** : augmentation techniques such as random cropping, flipping, color jittering, and rotation.
* **Model architecture** : EfficientNet-B0 pre-trained on ImageNet.
* **Training strategy** : mixed precision training, learning rate scheduling, and early stopping.
* **Evaluation metrics** : accuracy, classification report, and confusion matrix visualization.

# **2. System components**

## 2.1 Dataset preparation

The dataset is split into three subsets: **training** , **validation** , and **testing**

**Steps:**

1. Load the dataset from the specified directory (**dataset\_dir**).
2. Apply transformations for data augmentation.
3. Split the dataset into training (70%), validation (15%), and testing (15%) subsets.
4. Save the subsets into separate directories (**train\_folder**, **val\_folder**, **test\_folder**) for reproducibility.

## 2.2 Model architecture

The model uses **EfficientNet-B0**, a lightweight and efficient convolutional neural network architecture. The classifier layer is replaced with a custom linear layer to match the number of classes in the dataset.

## 2.3 Training configuration

**Hyperparameters:**

* **Image Size** : 224x224 pixels
* **Batch Size** : 32
* **Number of Epochs** : 50
* **Learning Rate** : 0.001
* **Weight Decay (L2 Regularization)** : 0.0001
* **Dropout Rate** : 0.3
* **Patience for Early Stopping** : 5 epochs
* **Minimum Validation Accuracy** : 80%

**Loss function and optimizer:**

* **Loss function** : Cross-Entropy Loss (**nn.CrossEntropyLoss**)
* **Optimizer** : AdamW (**optim.AdamW**)
* **Learning rate scheduler** : ReduceLROnPlateau (reduces learning rate when validation loss plateaus)

**Mixed precision training:**

Mixed precision training is enabled using PyTorch's **torch.cuda.amp** module to accelerate computation and reduce memory usage.

## 2.4 Early stopping mechanism

An early stopping mechanism is implemented to prevent overfitting by halting training if the validation accuracy does not improve for a specified number of epochs (**PATIENCE\_EPOCHS**).

# **3. Training and evaluation**

## 3.1 Training

The training loop consists of:

1. Training the model on the training set.
2. Evaluating the model on the validation set.
3. Logging metrics such as training loss, validation loss, and accuracy.
4. Applying early stopping if validation accuracy does not improve.

## 3.2 Testing and evaluation

After training, the model is evaluated on the test set to assess its performance. Metrics include:

* Test Loss
* Test Accuracy
* Classification Report
* Confusion Matrix Visualization

# **4. Results**

The final results are presented in terms of:

* **Test accuracy** : Percentage of correctly classified images in the test set.
* **Classification report** : Precision, recall, F1-score, and support for each class.
* **Confusion matrix** : A visual representation of prediction accuracy across classes.

## 4.1 Test metrics

* **Test loss** : 0.2854
* **Test accuracy** : 91.69%

## 4.2 Classification report

1. **High performance classes** : Classes like **army\_worm**, **healthy\_leaf**, and **powdery\_mildew\_leaf** achieved high precision, recall, and F1-scores (>0.90).
2. **Low performance classes** : **leaf\_scab** and **viral\_disease** had lower F1-scores (0.72 and 0.67, respectively), likely due to fewer samples or class imbalance.
3. **Overall accuracy** : The model achieved an impressive test accuracy of 91.69%, with a weighted average F1-score of 0.92.

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| army\_worm | 0.98 | 0.95 | 0.97 | 60 |
| black\_rust | 0.96 | 0.96 | 0.96 | 23 |
| brown\_rust | 0.86 | 0.92 | 0.89 | 13 |
| common\_rust | 0.91 | 0.93 | 0.92 | 56 |
| healthy\_leaf | 0.98 | 0.96 | 0.97 | 53 |
| leaf\_blight | 0.83 | 0.97 | 0.89 | 35 |
| leaf\_scab | 0.88 | 0.61 | 0.72 | 23 |
| powdery\_mildew\_leaf | 0.92 | 0.95 | 0.93 | 37 |
| viral\_disease | 0.64 | 0.70 | 0.67 | 10 |
| yellow\_rust | 0.93 | 0.93 | 0.93 | 15 |
| **Accuracy** |  |  | **0.92** | **325** |
| **Macro Avg** | **0.89** | **0.89** | **0.88** | **325** |
| **Weighted Avg** | **0.92** | **0.92** | **0.92** | **325** |

A screenshot of a graph

AI-generated content may be incorrect.

## 4.3 Strengths

1. **High accuracy** : the model performed exceptionally well on most classes, achieving over 90% accuracy on the test set.
2. **Mixed precision training** : enabled faster convergence and reduced memory usage.
3. **Early stopping** : prevented overfitting by halting training when validation accuracy plateaued.

## 4.4 Weaknesses

1. **Class imbalance** : some classes like **leaf\_scab** and **viral\_disease** had significantly fewer samples, leading to lower performance.