**Topic:** Multiple plant disease classification using CNN and Transfer learning

## Introduction

With the rising world population, global food insecurity has also been on a steady rise, with projections indicating that by 2030, around 670 million individuals (8% of the world's population) will experience hunger (Feo et al.,2022). One major factor contributing to food insecurity is pests and disease, which reduces the yield of agricultural production and leads to significant financial loss. According to Allard & Micallef (2019), farmers lose an estimated $ 60 billion in revenue to plant disease. Therefore, to properly manage plant disease, early detection is crucial to help farmers carry out the necessary precautionary measures to reduce the impact on plant yield (Nandhini et al.,2022).

Traditionally, farmers have relied on agricultural specialists to identify and diagnose pests and diseases in their crops. This process typically involves experts physically visiting the farm to visually inspect the symptoms, which can manifest on various parts of the plant (Ahmad et al., 2021). However, this method has its limitations, especially in developing countries where the availability of such experts is limited. Additionally, even in more developed nations, many farmers reside in remote rural areas, posing challenges for professionals to access and effectively diagnose diseased crops (Pawlak & Kołodziejczak, 2020). The lack of readily available expert opinions often leads to delays in taking preliminary steps to mitigate the disease at an early stage, resulting in significant crop yield losses year after year.

Alternatively, artificial intelligence (AI) offers a promising solution for identifying plant diseases. AI-powered systems can use image recognition algorithms and machine learning to detect disease crops accurately from images. This technology can be deployed remotely, enabling farmers worldwide to access timely and reliable crop disease diagnosis (Ahmad et al., 2020; Liu & Wang, 2021). This streamlined approach can reduce crop losses by minimizing the complexities associated with the conventional diagnostic process, typically involving an expert physically visiting the farm.

In this study, we will work with a subset of the Plant Village dataset, specifically focusing on tomato, potato, and pepper plant diseases Fig.1. This subset comprises a total of 20,638 images distributed across 15 distinct classes, with a training set consisting of 16516 images while the testing set consists of 4122 images. A customized CNN model and seven pre-trained models (VGG16, VGG19, ResNet50, DenseNet 121, Inception V4, EfficentNet, and

MobileNet) will be used to classify the images. After the training phase using these diverse models, we will employ standard performance metrics such as accuracy, recall, and precision to comprehensively assess and compare the models performance in classifying the plant disease images.

The research aims to conduct a comparative evaluation of various deep-learning architectures designed to classify multiple plant diseases. The primary aim is to improve early plant disease detection, which helps manage and control pests and diseases. Ultimately, this study will identify the most effective models and techniques for accurate disease classification.

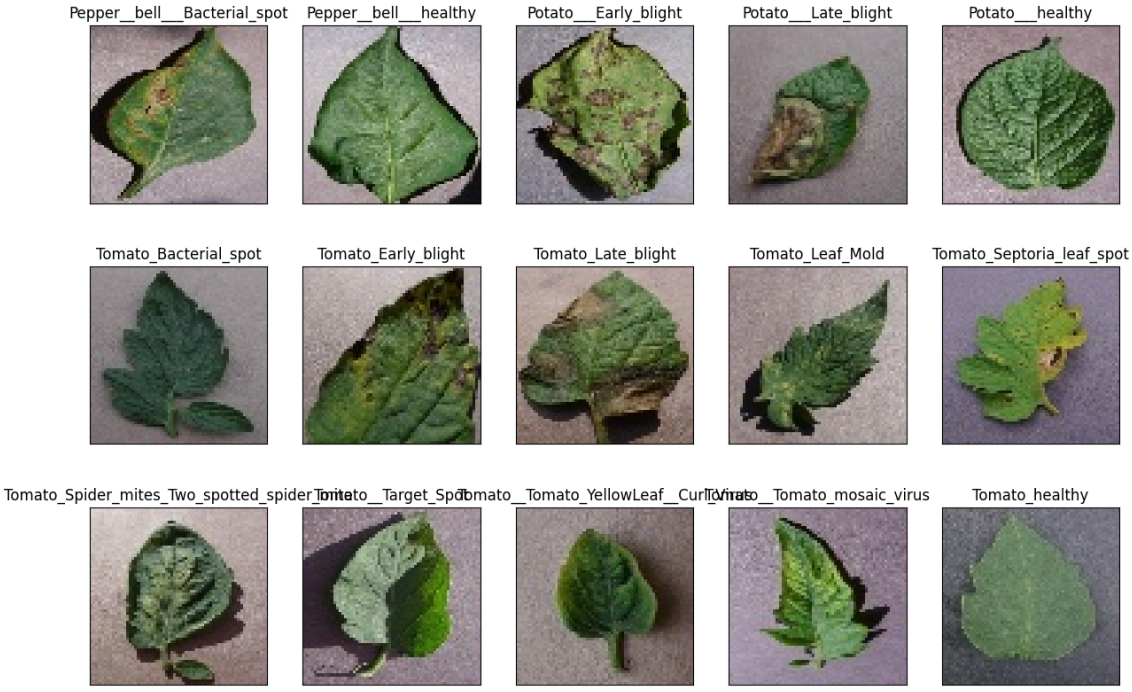


Fig1. Representation of the 15 distinct classes present in our data set.

### 1.1 Literature review

Convolutional neural networks (CNNs) have transformed the field of plant disease recognition in recent years. These modern techniques offer automated classification, enhanced accuracy, and feature extraction, which significantly differ from traditional methodologies such as K-means clustering and support vector machines (SVM) (Geetharamani & Pandian, 2019; Singh et al., 2019). As a result of their superior performance, CNNs have become the preferred choice for automated plant disease detection. Additionally, the transfer learning approach, in which a previously trained model is fine-tuned for a new objective, has emerged as the fundamental approach for attaining high accuracy in plant disease detection. Many studies have been carried out in relation to classifying plant diseases using the Plant Village dataset.

Mohanty et al. (2016) used the Plant Village Dataset to classify 26 plant diseases using transfer learning on AlexNet and GoogleNet. They discovered that the transfer learning trained model performed better than the model taught from scratch. Another study implemented transfer learning on the same dataset using VGG 16 and achieved a test accuracy of 90.4% (Wang et al., 2017).

Saleem et al. (2020) comprehensively evaluated deep learning methods for classifying defective plants. They assessed the performance of 18 deep learning architectures, employing stochastic gradient descent as the optimizer due to its rapid convergence. They further improved the performance of their best model using different optimizers on popular pre-trained models. This approach improves the performance of their model to 99.81% accuracy.

Ahmad et al. (2021) focused on addressing imbalanced data and negative transfer learning issues using simple statistical techniques and a stepwise transfer learning approach. This approach led to fast convergence and reduced overfitting, resulting in an improved accuracy of 99% while using MobileNet.

More recently, Eunice et al. (2022) concentrated on fine-tuning hyperparameters for well-known pre-trained models. Their experiments achieved the highest accuracy of 99.81% by fine-tuning DenseNet-121.

The preceding research underscores the growing importance of transfer learning and custom models in classifying plant diseases. However, Ahmad et al. (2021) report that the accuracy of these models significantly declined when evaluated on different datasets. This decline was attributed to the limited diversity within the training dataset. Furthermore, it is

worth noting that the experiment for plant disease identification occurred under ideal conditions. Consequently, the accuracy is lower on real-world data.

This study will employ various data augmentation techniques to address these limitations to enhance dataset diversity. A comprehensive examination of the performance of different pre-train models coupled with different deep-learning optimizers in classifying multiple plant diseases has yet to be conducted. Thus, this research aims to perform a detailed analysis of multiple deep learning models for the classification of plant diseases, with subsequent assessment of the performance of various optimizers. The ultimate aim of this research is to enhance the early detection of plant diseases, thereby facilitating improved disease management and control strategies.

### 1.2 Aims

The primary objective of this study is to build a customized CNN model specifically tailored for the classification of plant diseases. Subsequently, we aim to assess its performance by subjecting it to a comparative evaluation against seven pre-trained models, such as VGG16, VGG19, DenseNet 121, Inception V4, EfficientNet, ResNet50, and MobileNet.

### 1.3 Objectives.

1. Designing a Customized CNN architecture tailored for plant disease classification, focusing on achieving high accuracy.

2. Conduct a comparative analysis to assess how the performance of the custom-designed CNN architecture measures up against pre-trained models.

3. Thoroughly evaluating the effectiveness of various optimization algorithms when applied to the top-performing models.

## 2. Methodology

### 2.1 Dataset

The study utilized a subset of the Plant Village dataset, consisting of tomato, potato, and pepper plants. This subset comprises 20,638 images distributed across 15 distinct plant classes. 80 % of data was used for training (16516 images), while the remaining 20% was used for (4122 images). Importantly, both the training and testing sets include all 15 classes. Table 1 provides a detailed breakdown of this dataset split. The dataset was sourced from Kaggle (https://www.kaggle.com/datasets/emmarex/plantdisease/code).

Table 1 Details of the dataset after splitting into training and testing.

| Type of plant | Class of disease | Sample total | Training | Testing |
|---|---|---|---|---|
| Pepper | Pepper bell_bacterial_spot | 997 | 798 | 199 |
| | Pepper Bell_healthy | 1478 | 1183 | 295 |
| Potato | Potato_early_blight | 1000 | 800 | 200 |
| | Potato_healthy | 152 | 122 | 30 |
| | Potato_late_blight | 1000 | 800 | 200 |
| Tomato | Tomato_bacterial_spot | 2127 | 1702 | 425 |
| | Tomato_early_blight | 1000 | 800 | 200 |
| | Tomato_healthy | 1591 | 1273 | 318 |
| | Tomato_late_blight | 1909 | 1528 | 381 |
| | Tomato_leaf_mold | 952 | 762 | 190 |
| | Tomato_septoria_leaf_spot | 1771 | 1417 | 354 |
| | Tomato_spider_mites_twospotted_spider_mite | 1676 | 1341 | 335 |
| | Tomato_target_spot | 1404 | 1124 | 280 |
| | Tomato_mosaic_virus | 373 | 299 | 74 |
| | Tomato_yellow_leaf_curl_virus | 3208 | 2567 | 641 |
| Total | | 20638 | 15516 | 4122 |

### 2.2 Data preprocessing and augmentation

For our experimental objectives, we resized the images to 64 x 64 x 3 dimensions and normalized the images by dividing the pixel values by 255. This was done to enhance the stability of the training process and promote improved convergence and generalization of the CNN network. The dataset exhibits class imbalance, as shown in Figure 2. This class imbalance can result in overfitting during the training phase and introduce bias into the model predictions. To address this issue, we implemented data augmentation techniques. These augmentation

techniques were applied to preprocessed images and included horizontal and vertical flipping, clockwise and anticlockwise rotation, zoom intensity, and rescaling. This augmentation strategy serves a dual purpose by preventing overfitting and reducing model loss while enhancing the model's robustness. Additionally, we used macro average to evaluate the performance of our model to avoid bias from the majority class.

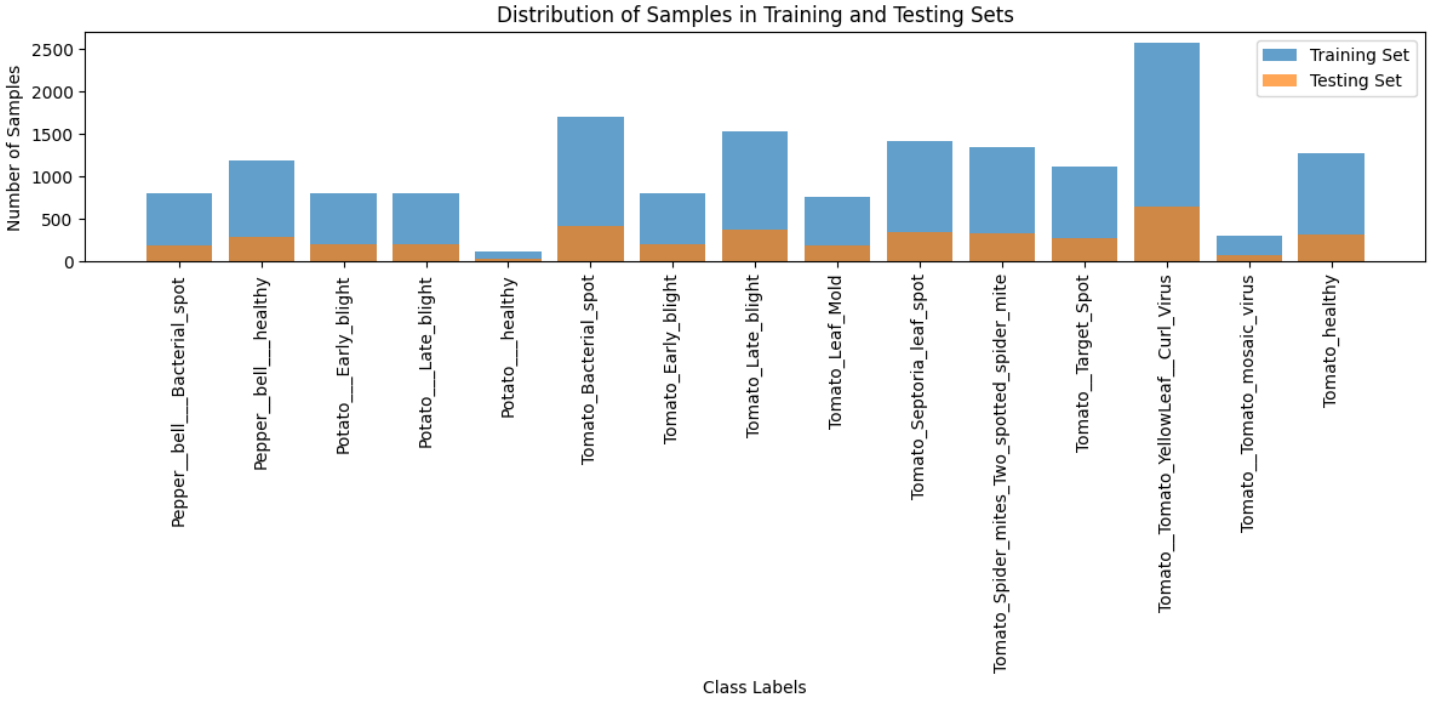


Fig.2. Frequency distribution of samples within each disease category in the dataset.

**Experimental setup**

CNN usually consists of a convolutional layer which is followed by a pooling and fully connected dense layer. The basic architecture of a CNN model is shown in Figure 3. They operate by applying convolutional layers to extract import features from the impute images. These layers are usually followed by a relu activation function, which captures non-linearity. The pooling layers reduce the representation's dimensionality while retaining key information. The fully connected layers perform the classification and, finally, a loss function to measure prediction errors(Lu et al., 2021).

This study assessed the effectiveness of both a custom model and a pre-trained model in classifying plant diseases. The research methodology, illustrated in Figure 3, involved several steps. The Adam optimizer was chosen to train the CNN model due to its capacity to automatically adapt learning rates for faster convergence and enhanced stability, as noted by

Saleem et al. (2020). Subsequently, the custom and pre-trained models were trained and evaluated using various metrics. Ultimately, multiple optimizers (SGD, Rmsprop, & Adamax) were employed to enhance the performance of both the custom and pre-trained models. Every training consisted of 50 epochs, and early stopping was implemented with a patience of four to prevent overfitting.

Transfer learning was performed by freezing the pre-trained model's top layer before connecting it to the fully connected layers of our best-performing custom model. The pre-trained model selection for this study was based on their suitability for the classification of plant diseases, and comprehensive detail of their model architecture is given in Table 2.

Table 2 Pre-trained model properties

| **Model** | **VVG16** | **VGG19** | **Inception V3** | **ResNet 50** | **DenseNet 121** | **MobileNet** | **EfficientNet** |
|---|---|---|---|---|---|---|---|
| **Total layers** | 16 | 19 | 48 | 50 | 121 | 28-53 | 260 |
| **Maxpool layers** | 5 | 5 | varies | 1 | 4 | Varies | Varies |
| **Dense Layers** | 3 | 3 | 2 | 3 | 4 | GAP | Varies |
| **Dropout layers** | 2 | - | Varies | 2 | - | - | - |
| **Flatten layers** | 1 | 1 | 1 | 1 | - | GAP | GAP |
| **Trainable Parameter (Millions)** | 14.7 | 143.67 | 23.8 | 23.6 | 7.05 | 4.2 | 5.3 |

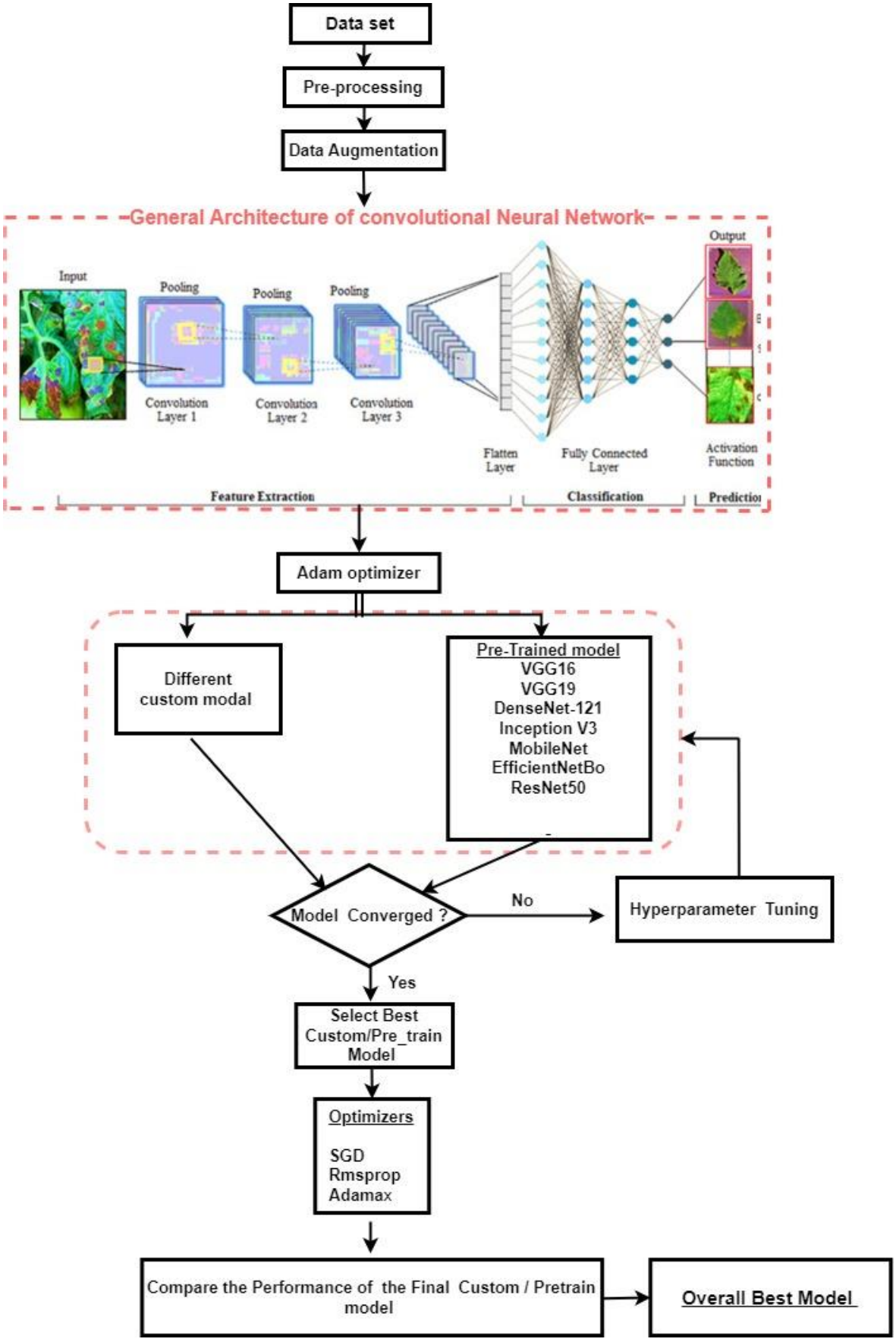


Fig.3. Research workflow

### 2.3 Software and Hardware Specification

The experiment was carried out in DAIM lab workstation using a Jupyter notebook. The feature of the Dell system used in this study is as follows: Core i9-12900 CPU operating at a speed of 2.40 GHz, 32 GB of RAM, and an NVIDIA GeForce RTX 3090 GPU.

### 2.4 Performance evaluation

The research employs evaluation metrics such as accuracy, precision, and recall.

**Accuracy:** It denotes the proportion of correct predictions made on the test dataset.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \qquad eq\ (1)$$

TP = True Positive, TN =True Negative (correct predictions ), FP = False Positive, and FN = False Negative (represent errors).

**Precision:** It quantifies how many positive predictions the model got right.

$$Precision = \frac{TP}{TP + FP} \times 100\% \qquad eq\ (2)$$

**Recall:** It quantifies how well the model identifies positives within the positive class.

$$Recall = \frac{TP}{TP + FN} \times 100\% \qquad eq\ (3)$$

**F1-Score:** It considers the model's precision and recall.

$$F1 = \frac{2Precision \times Recall}{Precision + Recall} \times 100\% \qquad eq\ (4)$$

## 3 Result and discussion

### 3.1 Custom model

The customized model architecture was developed from scratch by carefully fine-tuning various hyperparameters, such as the number of convolutional layers, optimizer choices, and batch size, as shown in Table 3. We applied early stopping with a patience of 4 to mitigate overfitting for all our models.

Table 3. Performance of different Custom model

| Model | Hyperparameter tuned | Convolutional layer | Accuracy | Loss | Epoch before early stopping |
|---|---|---|---|---|---|
| *Model 1* | Adam (0.001), Bc=32 | 3 | 0.864 | 0.445 | 21 |
| *Increasing the number of convolutional layers* | | | | | |
| **Model 2** | **Adam (0.001), Bc=32** | **4** | **0.903** | **0.291** | **26** |
| Model 3 | Adam (0.001), Bc=32 | 5 | 0.880 | 0.384 | 24 |
| *Effect of different optimizers on the best model (Model 2)* | | | | | |
| Model 4 | Rmsprop, Bc=32 | 4 | 0.873 | 0.407 | 18 |
| Model 5 | SGD, Bc=32 | 4 | 0.848 | 0.442 | 30 |
| Model 6 | Adamax, Bc=32 | 4 | 0.872 | 0.436 | 28 |
| *Decreasing the learning rate of the best model (Model 2)* | | | | | |
| Model 7 | Adam (0.0001), Bc=32 | 4 | 0.731 | 0.798 | 18 |
| *Increasing the batch to 64 for the best model (Model 2)* | | | | | |
| Model 8 | Adam (0.001), Bc=64 | 4 | 0.776 | 0.757 | 19 |

Bc =Batch size, * base model *, Best Costom model = Model 2

The initial base model achieved an impressive accuracy of approximately 86%, surpassing the 83% reported by Fuentes et al. et al. (2017) on the same dataset. To enhance the model performance, convolutional layers were increased to 4, improving the base model's accuracy to 90.3%. However, further increasing the convolutional layers to 5 did not improve the model's performance.

To further improve the model performance, the impact of different optimizers on the best-performing model was investigated (Model2). Rmsprop and Adamax exhibited

commendable performance, achieving accuracies of 87.3% and 87.4%, respectively, though they fell short of the Adam optimizer's performance. The SGD optimizer, unfortunately, performed the least effectively, with an accuracy of only 84.8%, trailing behind the base model.

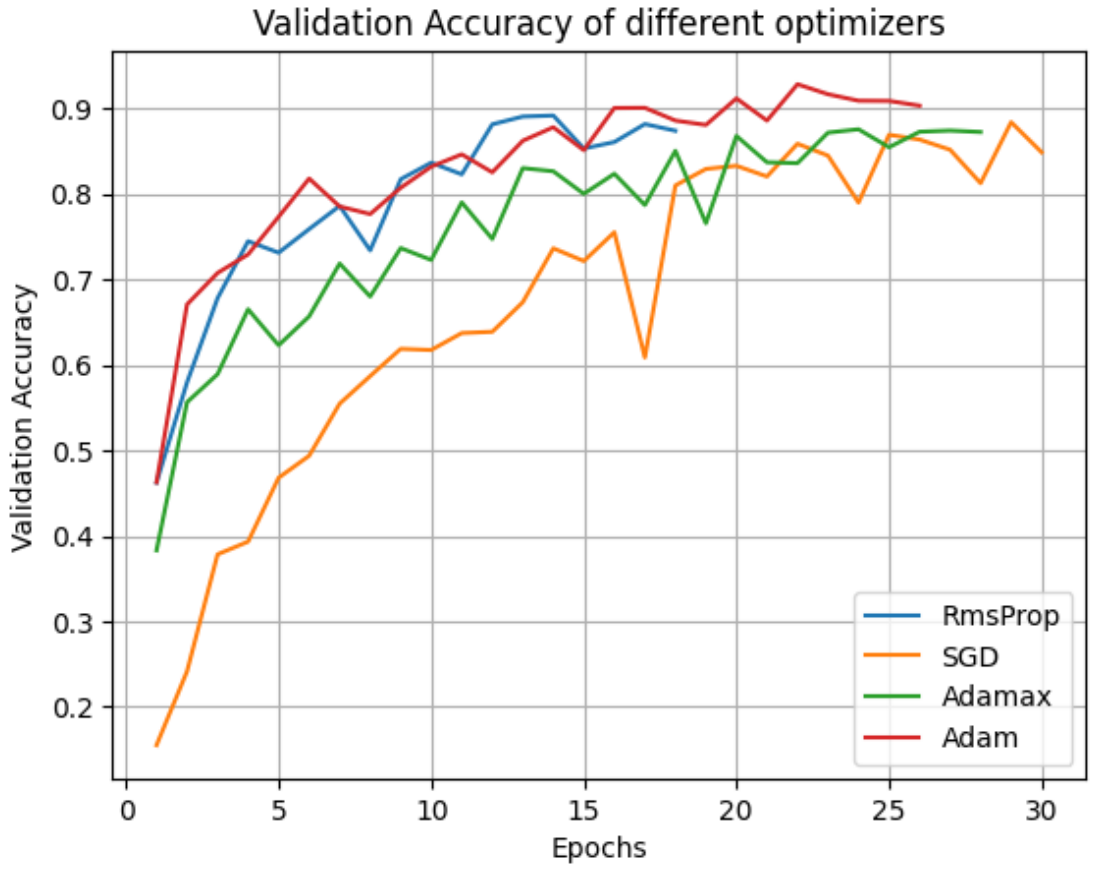


Fig.4 Validation Accuracy of different optimizers

The validation accuracy plot depicted in Figure 4 reveals distinct convergence patterns among these optimizers. The Adam optimizer exhibited rapid convergence, surpassing 80% accuracy within the initial epochs and continuing to improve. In contrast, RMSprop and Adamax showed gradual convergence, eventually plateauing after a few epochs without achieving the same level of accuracy as the Adam optimizer. The SGD optimizer demonstrated a less stable convergence pattern, suggesting that it may necessitate hyperparameter fine-tuning to realize its full convergence potential.

In summary, the Adam optimizer emerged as the most effective choice for the custom model. Other hyperparameter tuning techniques (batch size and learning rate) did not improve

model performance. Thus, the optimal custom model configuration comprises four convolutional layers, the Adam optimizer (0.001).

### 3.2 Transfer learning

The results of leveraging transfer learning for multi-plant disease classification are shown in Table 4.

Table 4. Comparative analysis of the performance of pre-trained model and Custom model

| Model | Optimizer | Accuracy | Precision | Recall | Loss |
|---|---|---|---|---|---|
| VGG16 | Adam | 0.843 | 0.840 | 0.820 | 0.559 |
| VGG19 | Adam | 0.836 | 0.830 | 0.800 | 0.533 |
| DenseNet-121 | Adam | 0.867 | 0.860 | 0.840 | 0.411 |
| Inception V3 | Adam | 0.874 | 0.870 | 0.820 | 0.392 |
| EfficientNetB0 | Adam | 0.849 | 0.840 | 0.820 | 0.446 |
| MobileNet | Adam | 0.825 | 0.820 | 0.790 | 0.548 |
| ResNet-50 | Adam | 0.903 | 0.900 | 0.890 | 0.340 |
| Effect of different optimizers on the best model (ResNet50) | | | | | |
| ResNe-50 | Rmsprop | 0.888 | 0.870 | 0.870 | 0.425 |
| ResNet-50 | SGD | 0.899 | 0.880 | 0.890 | 0.299 |
| ResNet-50 | Adamax | 0.910 | 0.910 | 0.910 | 0.208 |
| Comparison to the best Custom model | | | | | |
| Custom model | **Adam** | 0.903 | 0.890 | 0.900 | 0.291 |

ResNet50 achieved the highest validation accuracy of 90.3%, followed by Inception V3 and DenseNet 121, with validation accuracies of 87.4% and 86.7%, respectively. As expected, this model also exhibited stable and consistent convergence, as shown in Figure 5. The worst-performing transfer learning models were VGG16 and VGG19, attaining 84.3% and 83.6% accuracy, respectively. Ahmad et al. (2021) reported 82.60% and 81.67% accuracy on the same dataset using VGG16 & 19. They attributed the lower accuracy to the depth of architecture of VGG 16 &19, which resulted in the vanishing gradient problem and slowed down the training process. This is why VGG16 & 19 exhibited the slowest convergence, as shown in Figure 5. Further model improvement was carried out by investigating the effect of different optimizers on the best-pretrained model (ResNet50). The Adamax optimizer further improved the

performance of our model and achieved an accuracy of proximally 91 %. Overall, the best transfer learning model is ResNet50 with the Adamax optimizer.

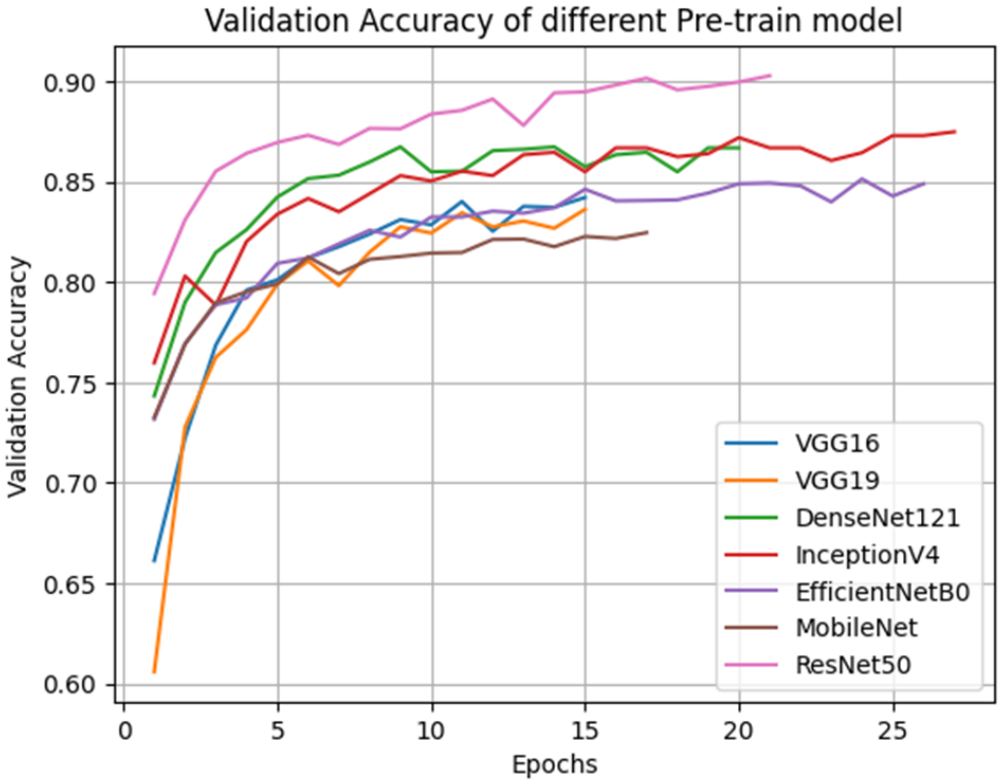


Fig. 5 Validation accuracy of different pre-trained models

### 3.3 Comparative Analysis Between the Best Pretrained and Custom Models

The performance of the best pre-trained and custom models is almost identical, with the custom model achieving an accuracy of 90%, precision of 90%, and recall of 89%. On the other hand, the pre-trained model achieved an accuracy of 91%, precision of 91%, and recall of 91%, along with a lower loss (0.208), implying that the model is more efficient at reducing the loss compared to the custom model. However, the plot of training and validation loss for both pre-trained and custom models in Figure 6 shows that the models generalized well because the training loss is greater than the validation loss, which implies that our model did not overfit. Overall, both models performed satisfactorily in multi-plant disease classification.

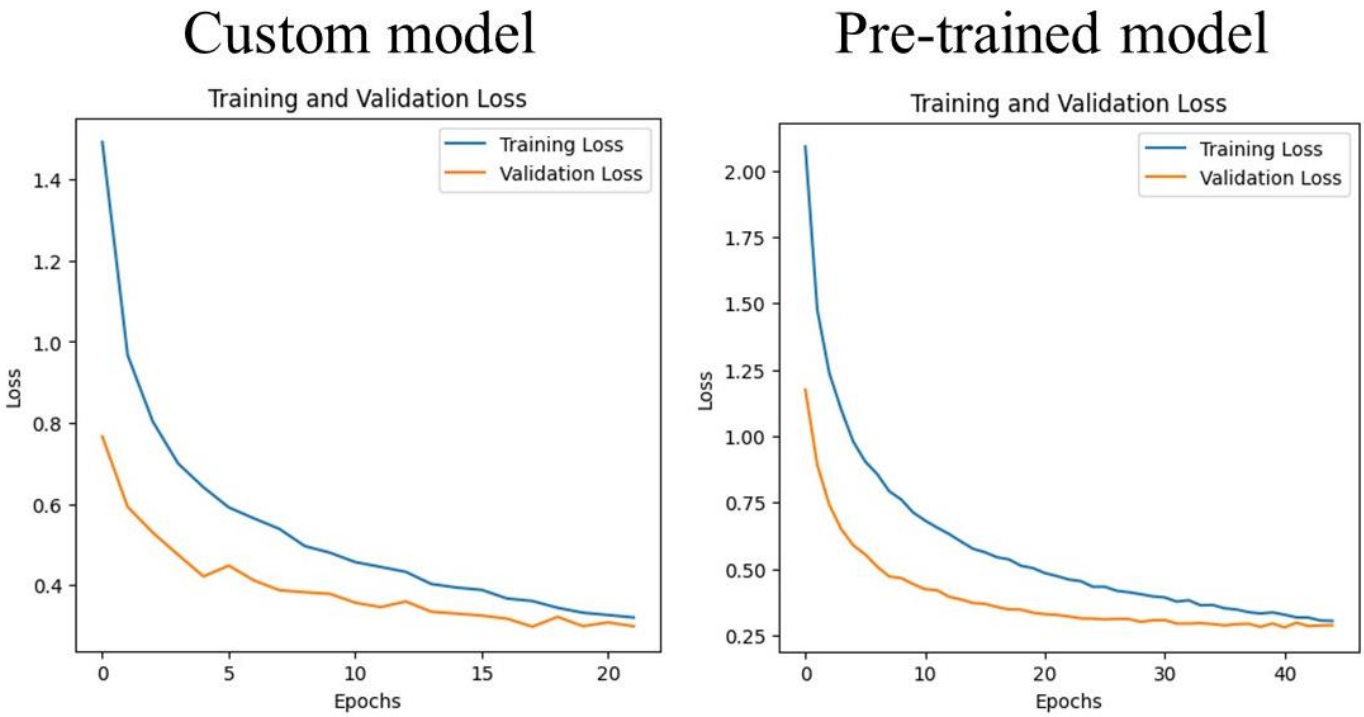


Fig. 6 A plot Validation and training loss for the pre-trained and custom model.

When comparing our results to the work of Ahmed et al. in 2021, who worked on the same dataset, as shown in Table 5, both our pre-trained and custom models performed better. This implies a better balance between model complexity and generalization in our approach. However, there is still room for improvement, and further work should investigate the effect of balancing the dataset using different augmentation techniques on the model. Additionally, the impact of resizing the images to the pre-trained model's actual size is worth exploring. In conclusion, both the custom and pre-trained models performed well in multi-plant disease classification, as demonstrated by the results of the confusion matrices in Figure 7.

Table 5: Comparative analysis of the performance of our model to previous work

| **MODEL** | **Accuracy by Ahmed et al. (2021) %** | **Our accuracy %** |
|---|---|---|
| VGG16 | 82.60 | 84.3 |
| VGG19 | 81.79 | 83.6 |
| ResNet 50 | 82.0 | 91.0 |
| DenseNet 121 | 78.19 | 86.7 |
| MobileNet | 77.72 | 87.4 |
| Inception V3 | 81.20 | 82.5 |

* Custom model accuracy = 90.3%

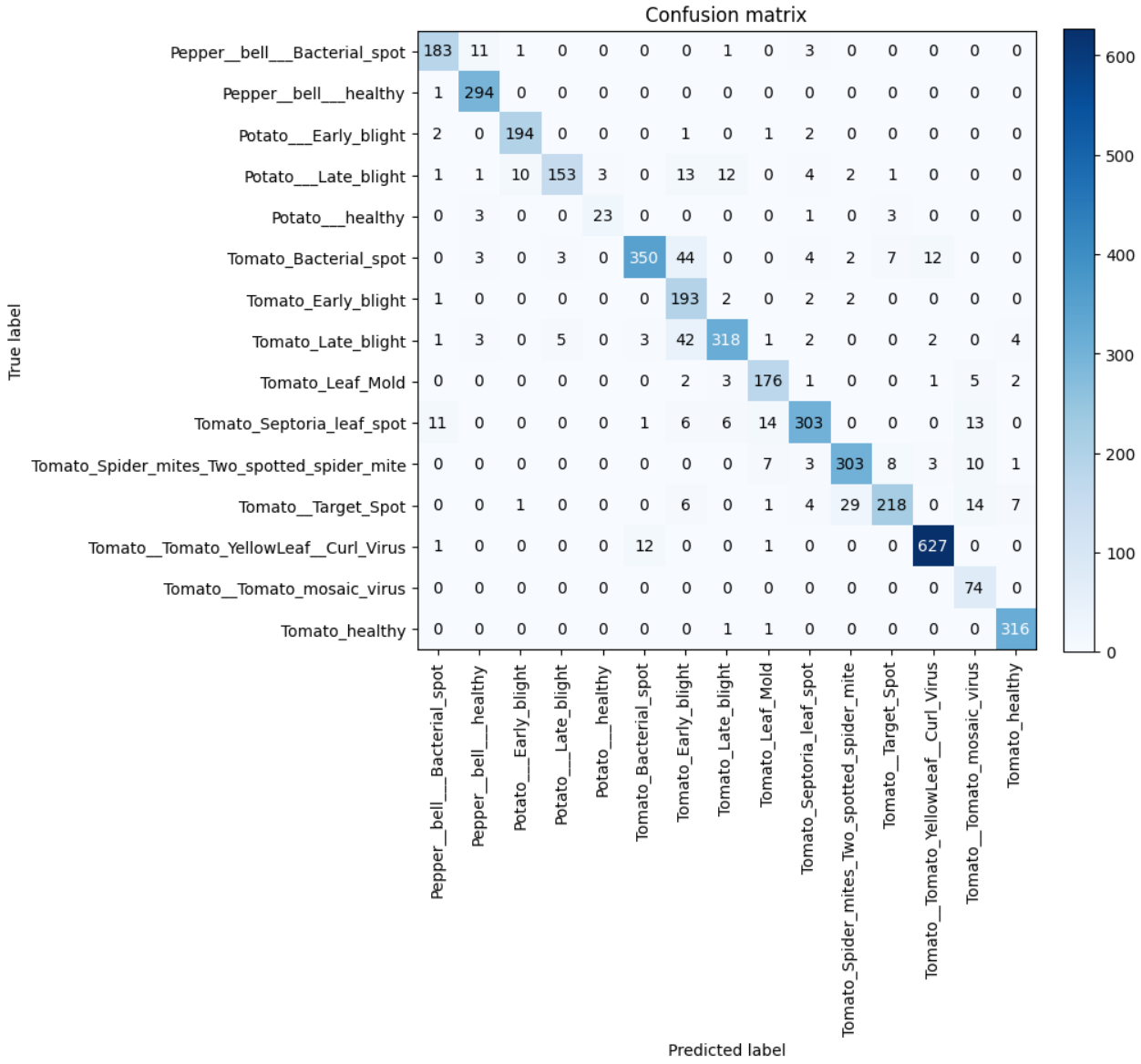


Fig. 7 A typical confusion matrix of our best pre-trained and custom model.

## 4. Summary and Conclusion

In summary, we comprehensively evaluated different pre-trained and custom models designed for multi-plant disease classification to improve early plant diseases, which is vital in pest and disease management. Using a subset of the Plant Village dataset, various hyperparameter tuning techniques were employed, including changing the learning rate, batch size, and optimizer settings, to fine-tune our custom model. This technique resulted in an impressive accuracy of 90.4%.

Subsequently, we compared our custom model to the top-performing pre-trained models. Both models perform very similarly, with a slight 1% advantage observed for the pre-trained model in terms of accuracy. Notably, both models outperformed previous work conducted on the same dataset.

Future research should explore techniques like data balancing and stepwise transfer learning to improve the model's effectiveness further. Optimizers such as improved GoogleNet, Xception, and cascaded AlexNet with GoogleNet should be explored to boost the model's performance, eventually enabling deployment on mobile applications. Such a deployment could significantly assist farmers in efficiently managing plant diseases, thereby contributing to improved crop health and yield.

**Reference**

Ahmad, M., Abdullah, M., Moon, H. & Han, D. (2021) Plant disease detection in imbalanced datasets using efficient convolutional neural networks with stepwise transfer learning. *IEEE Access*, 9, 140565-140580.

Ahmad, J., Jan, B., Farman, H., Ahmad, W. & Ullah, A. (2020) Disease detection in plum using convolutional neural network under true field conditions. *Sensors*, 20(19), 5569.

Allard, S. M. & Micallef, S. A. (2019) The plant microbiome: Diversity, dynamics, and role in food safety, *Safety and practice for organic food*Elsevier, 229-257.

Dubey, S. R. & Jalal, A. S. (2013) Adapted approach for fruit disease identification using images, *Image processing: Concepts, methodologies, tools, and applications*IGI Global, 1395-1409.

Eunice, J., Popescu, D. E., Chowdary, M. K. & Hemanth, J. (2022) Deep learning-based leaf disease detection in crops using images for agricultural applications. *Agronomy*, 12(10), 2395.

Fao, I. F. A. D., UNICEF, WFP and WHO (2022) The State of Food Security and Nutrition in the World 2022. Repurposing Food and Agricultural Policies to Make Healthy Diets More Affordable

Fuentes, A., Yoon, S., Kim, S. C. & Park, D. S. (2017) A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. *Sensors*, 17(9), 2022.

Geetharamani, G. & Pandian, A. (2019) Identification of plant leaf diseases using a nine-layer deep convolutional neural network. *Computers & Electrical Engineering*, 76, 323-338.

Liu, J. & Wang, X. (2021) Plant diseases and pests detection based on deep learning: a review. *Plant Methods*, 17, 1-18.

Lu, J., Tan, L. & Jiang, H. (2021) Review on convolutional neural network (CNN) applied to plant leaf disease classification. *Agriculture*, 11(8), 707.

Li, G., Ma, Z. & Wang, H. (2012) Image recognition of grape downy mildew and grape powdery mildew based on support vector machine, *Computer and Computing Technologies in Agriculture V: 5th IFIP TC 5/SIG 5.1 Conference, CCTA 2011, Beijing, China, October 29-31, 2011, Proceedings, Part III 5*. Springer.

Mohanty, S. P., Hughes, D. P. & Salathé, M. (2016) Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7, 1419

Nandhini, M., Kala, K., Thangadarshini, M. & Verma, S. M. (2022) Deep Learning model of sequential image classifier for crop disease detection in plantain tree cultivation. *Computers and Electronics in Agriculture*, 197, 106915.

Pawlak, K. & Kołodziejczak, M. (2020) The role of agriculture in ensuring food security in developing countries: Considerations in the context of the problem of sustainable food production. *Sustainability*, 12(13), 5488.

Singh, U. P., Chouhan, S. S., Jain, S. & Jain, S. (2019) Multilayer convolution neural network for the classification of mango leaves infected by anthracnose disease. *IEEE access*, 7, 43721-43729.

Wang, G., Sun, Y. & Wang, J. (2017) Automatic image-based plant disease severity estimation using deep learning. *Computational intelligence and neuroscience*, 2017.