Tomato Leaf Diseases Detection Classification Using CNN and Transfer Learning

YAOXIAO
Computer Science
Universiti Sains Malaysia
xiaoyao99@student.usm.my

***Abstract*—Crop losses due to pests and diseases pose a significant threat to agricultural productivity, particularly in tomato cultivation, where early detection is crucial for effective management. Traditional methods of diagnosing plant diseases rely heavily on expert judgment, which can be subjective and limited by human cognitive capabilities. In this study, we address these challenges by developing a robust Convolutional Neural Network (CNN) model combined with transfer learning techniques to accurately detect and classify tomato leaf diseases from high-resolution images.**

Our approach employs pre-trained models such as VGG16, InceptionV3, and DenseNet121, enhanced with custom layers tailored to our specific dataset. The dataset, consisting of tomato leaf images affected by various diseases, is augmented using techniques like rotation, flipping, and color adjustment to improve the model's generalization capabilities. The architecture of our CNN is meticulously designed to extract deep and complex features, ensuring precise identification of disease characteristics. Experimental results demonstrate that the DenseNet121 model, optimized with the Adam optimizer, achieves superior performance, with a validation accuracy of 96%. Additional metrics such as precision, recall, F1-score, and ROC curve analyses further validate the model's efficacy.

This research highlights the transformative potential of deep learning and transfer learning techniques in agricultural diagnostics. By enabling early and accurate detection of tomato leaf diseases, our approach provides a scalable solution for farmers to implement timely preventative measures, reducing crop losses and promoting sustainable farming practices. The significant improvements in diagnostic accuracy underscore the value of integrating modern AI technologies into agriculture.

***Keywords*: Tomato Leaf Diseases, Convolutional Neural Network, Transfer learning, VGG16, InceptionV3, DenseNet121**

I. INTRODUCTION

According to a report by the Food and Agriculture Organization of the United Nations, pests and diseases can lead to up to 40% of crop losses, significantly affecting crop yields[1]. Traditionally, the diagnosis of plant leaf diseases has primarily relied on experts and experienced agricultural producers who determine disease types by observing the differences in lesions caused by various diseases. However, certain types of disease lesions exhibit very subtle visual differences, making accurate diagnosis challenging even for experienced experts. Furthermore, manual judgment is often limited by cognitive capabilities.

Tomatoes, being nutrient-rich and widely cultivated vegetables, have an annual global consumption of approximately 16 million tons[2]. The growth of tomatoes is significantly impacted by parasitic insects and diseases, making research on field crop disease diagnosis crucial. Tomato yield is threatened by different types of diseases on tomato leaves. However, early diagnosis of these diseases can help farmers take preventive actions and save their crops.

With advancements in computer vision and artificial intelligence technologies, their application in crop disease identification has become increasingly prevalent. If these technologies can detect the subtle characteristic differences in plant leaf lesions and accurately diagnose disease types at an earlier stage, it would be of great significance for preventing the spread of pests and diseases and reducing agricultural losses.

This study focuses on the identification of tomato leaf diseases by constructing a recognition model utilizing a Convolutional Neural Network (CNN) architecture. The CNN structure has demonstrated commendable performance in image classification

tasks[3]. The model's design takes into account the depth and complexity of feature extraction to ensure effective identification of disease characteristics from high-resolution tomato leaf images. To enhance the model's generalization ability, data augmentation techniques were employed. By applying a series of random transformations to the training images, such as rotation, flipping, scaling, and color adjustment, the diversity of the samples was successfully increased. This approach aids the model in learning more robust feature representations and reduces its dependency on specific samples.

In the design of the feature extraction network, the concept of transfer learning was introduced. Transfer learning enhances the generalization capability of the feature extraction network, helping to reduce the interference of background noise on the recognition task, and better extract key characteristic information of the disease-affected regions[4]. Consequently, this improves the accuracy of the model's recognition capabilities.

II. Related Work

Tomato Leaf Disease Detection Using Convolutional Neural Networks by Prajwala Tm[5], the methodology encompasses three main stages: Data Acquisition, Data Pre-processing, and Classification. Utilizing the Plant Village dataset with around 18,160 images across 10 tomato leaf disease classes, the model leverages data augmentation techniques such as random rotations, horizontal flipping, and shifting to enhance robustness. Implemented with Keras, the model employs the Adam optimizer and categorical cross-entropy loss function, with a batch size of 20 and training over 30 epochs. The dataset is divided into 13,360 training images and 4,800 testing images. The model achieves a highest validation accuracy of 94.8% and a training accuracy of 99.3%, demonstrating its effectiveness in classifying tomato leaf diseases accurately.

Too and Yujian et al. [6]fine-tuned and evaluated VGG 16, Inception V4, ResNet with 50, 101, and 152 layers, and DenseNet121 deep convolutional neural networks on the Plant Village dataset. The experiments demonstrated that the DenseNet network achieved an experimental accuracy of 99.75%, which is higher than that of the other networks.

Brahimi et al. [7]implemented a Convolutional Neural Network (CNN) to classify diseases in tomato leaves, achieving a high accuracy of 99.18%. The study utilized a dataset of 14,828 images covering nine types of diseases, significantly larger than those used in previous studies. The CNN's ability to automatically extract features from raw images allowed for a more efficient and accurate classification process compared to traditional methods requiring manual feature extraction.

The study employed pre-trained AlexNet and VGG16 architectures[8]. With 13,262 images, VGG16 achieved a classification accuracy of 97.29%, while AlexNet reached 97.49%. The performance evaluation involved adjusting the number of images, minibatch sizes, and learning rates for weights and biases. Results indicated that the number of images greatly influenced model performance, with the highest accuracy observed using 373 images. In AlexNet, altering the minibatch size showed no clear link to accuracy, whereas in VGG16, accuracy decreased as minibatch size increased.

III. Proposed Work

In this part, my proposed work focuses on creating a effective CNN model to detect and classify diseases in tomato leaves. This involves essential steps like preprocessing the dataset, applying data augmentation techniques, and using transfer learning with pretrained models such as VGG16, InceptionV3, and DenseNet121. By enhancing the model’s accuracy and its ability to generalize, the goal is to enable early and precise detection of diseases, helping to reduce agricultural losses and support farmers.

As illustrated in Fig 1. The images capture a range of leaf conditions, showcasing various stages of disease manifestation. This diversity is crucial for training a robust machine learning model, particularly a CNN, which relies on varied inputs to learn discriminative features effectively.

A. Dataset preprocessing

Before input images are fed into the model, they must undergo an essential step: image preprocessing. This involves filtering the raw data samples by resizing and reshaping the images to ensure uniformity and compatibility with the model architecture.

In the task of image recognition using convolutional neural networks (CNNs), certain adjustments to the input images can enhance the model's performance without compromising its accuracy. These adjustments, often referred to as data augmentation techniques, help the model avoid local optima and improve its robustness and generalization capabilities. By introducing variations in the training data, data augmentation reduces the risk of overfitting and enables the model to learn more invariant features, ultimately leading to better performance on unseen data.

During the preprocessing phase of the dataset, images are resized to 224x224 pixels to ensure uniformity and compatibility with the model architecture. Following this, data augmentation techniques are applied to enhance the dataset's diversity. Specifically, a data augmentation function is employed, incorporating horizontal flipping and slight rotations using TensorFlow's tf.keras.Sequential. This step not only enriches the training data by introducing variability but also helps the model generalize better, as illustrated in Fig 2.

Splitting the data into two parts is a crucial step in training a predictive model. The complete dataset is divided into a training set and a validation set, typically with an 80/20 split. As shown in Fig 3. The training set, comprising 80% of the data, is used to teach the model, while the validation set, comprising 20%, evaluates the model's performance. The following code demonstrates this process, where the `image_dataset_from_directory` function creates the datasets, and a subsequent mapping normalizes the image pixel values to a range of 0 to 1 using `lambda x, y: (x / 255.0, y)`. This ensures the model receives consistently scaled inputs, enhancing its learning efficiency and accuracy.

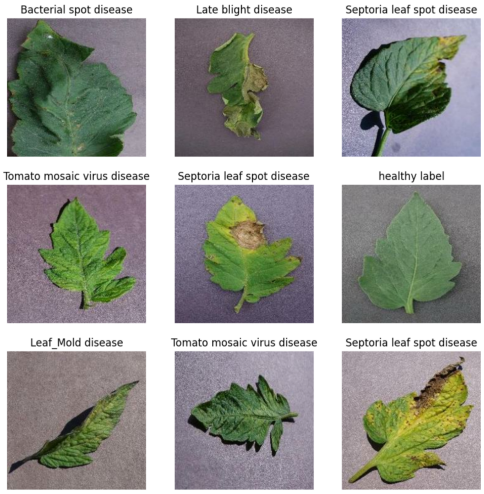


Fig 1.Dataset example

Fig 2. Augmented Image

```
Found 10000 files belonging to 10 classes.
Using 8000 files for training.
Found 10000 files belonging to 10 classes.
Using 2000 files for validation.
```

Fig 3. Number of samples considered

B. CNN model

Convolutional Neural Networks (CNNs) were first introduced in 1989 by Yann LeCun, Léon Bottou, and their colleagues through the LeNet architecture, which achieved remarkable success in recognizing handwritten digit images. CNNs are a class of deep neural networks that have become the cornerstone of modern computer vision tasks. They consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers apply filters to the input data to extract essential features, while the pooling layers reduce the dimensionality of the data, preserving the most critical information and improving computational efficiency. Fully connected layers then integrate these features to perform high-level reasoning and classification. Additionally, CNNs utilize weights and activation functions to introduce non-linearity and enhance the network's learning capabilities. The core operations of CNNs—convolutions and pooling—are pivotal in enabling the network to learn hierarchical representations of data, making them highly effective for a wide range of image recognition tasks.

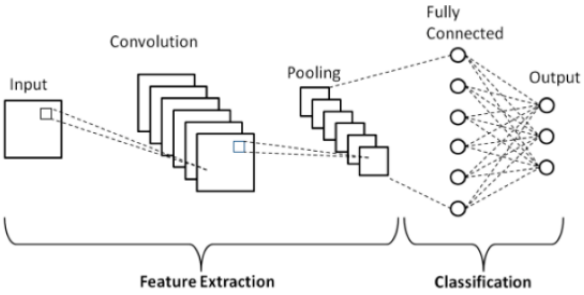


Fig 4. CNN Architecture

Here is my project modeling part, where I have developed a sophisticated Convolutional Neural Network (CNN) for detecting tomato leaf diseases. The model is designed to classify images into one of ten categories, each representing a distinct disease, with a dataset comprising 10,000 images (1,000 per category).

The model architecture starts with an input layer that processes images resized to 224x224 pixels. To enhance generalization and robustness, data augmentation techniques, including random horizontal flipping and rotation, are applied. The network architecture consists of multiple convolutional layers, each followed by batch normalization, max pooling, and dropout layers to mitigate overfitting. The convolutional layers are responsible for extracting hierarchical features, while the fully connected dense layers at the end perform the final classification into one of the ten categories.

The first diagram offers a comprehensive layer-by-layer breakdown, showcasing the sequential arrangement and dimensions of each layer. The second diagram provides a 3D perspective of the network, illustrating the depth and complexity of the model.

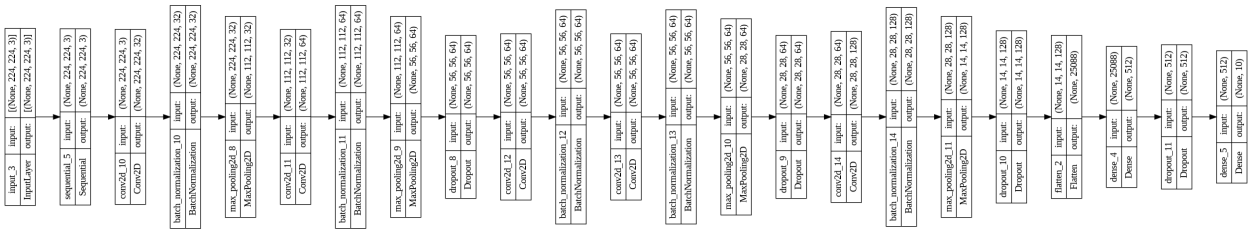


Fig 5. Model Diagram

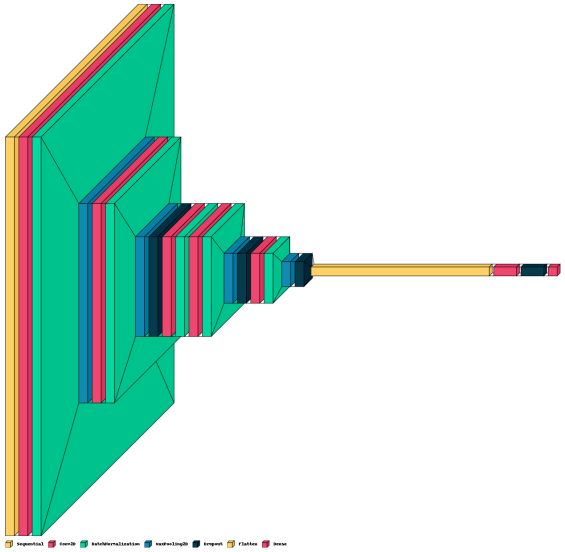


Fig 6. 3D diagram

Modeling a CNN in this structured manner for tomato leaf disease detection offers several key benefits. Firstly, the use of convolutional layers helps in automatically extracting important features from images, such as edges, textures, and patterns, which are crucial for distinguishing between different disease types. BatchNormalization layers are included to standardize the inputs to each layer, improving training stability and accelerating the convergence of the model. Dropout layers are strategically added to prevent overfitting by randomly deactivating a subset of neurons during each training iteration, ensuring the model generalizes well to unseen data.

C. Pretrained model

In my project on Tomato Leaf Diseases Detection Classification, I leverage three pretrained models: VGG16, InceptionV3, and DenseNet121. These models, pretrained on ImageNet, serve as the backbone for my classification tasks, providing powerful feature extraction capabilities.

Using pretrained models is an effective strategy for small datasets. By integrating advanced architectures, I augment the models with custom layers tailored specifically to our dataset, achieving good accuracy and enhances model performance and generalization.

a). VGG16

Due to the limited number of images in the tomato leaf disease dataset used in this experiment, achieving high accuracy requires deepening the network model, which in turn requires a large number of training parameters. However, the small experimental dataset cannot satisfy this training requirement. Therefore, transfer learning was employed to utilize the pre-trained VGG16 model. [9]Originally proposed by K. Simonyan and A. Zisserman from Oxford University, the VGG16 model is renowned for its performance in the ILSVRC-2014, achieving a top-5 accuracy of 92.7% on ImageNet. The model's architecture, involving very deep convolutional layers with small (3×3) filters, allows for high performance and adaptability to various image recognition tasks. The VGG16 model diagram and the VGG network structure diagram are shown in Fig 7 and Fig 8:

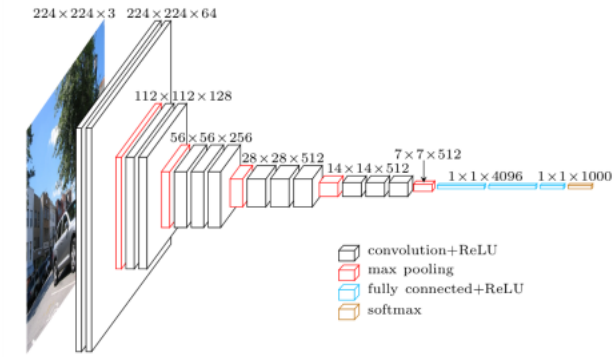


Fig 7. VGG16 Architecture[9]

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig 8. VGG16 Network Structure Diagram[9]

As shown in Fig 8, the VGG-16 architecture, known for its simplicity and effectiveness, consists of 13 convolutional layers using 3x3 filters, five max-pooling layers, and three fully connected layers, culminating in a soft-max classifier. The input image size is 224x224. The use of small filters (3x3) throughout the network allows for deep architecture with fewer parameters, enhancing feature learning capabilities. In the context of my project on tomato leaf disease detection, the dataset includes 10 distinct labels, each representing a specific condition affecting tomato plants, so the last layer output layer is set to 10. As shown in Fig 12.

b). Inception V3

In Inception V3, the methodology employed by VGGNet is utilized, where large convolutional kernels are replaced with multiple smaller convolutional kernels. The core structure of InceptionV3 consists of Inception Modules. Fig.. illustrates the Inception Architecture and Inception Module respetively.

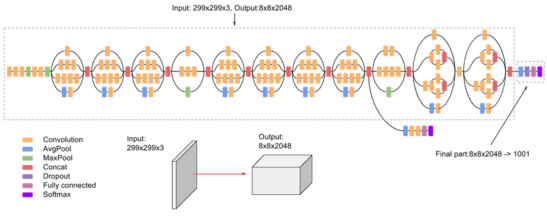


Fig 9. Inception V3 Architecture[10]

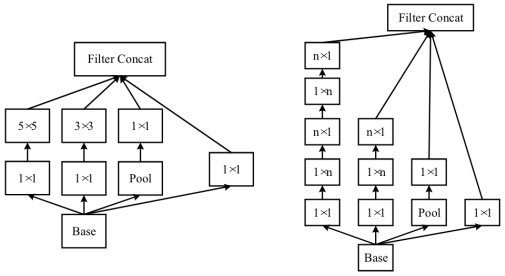


Fig 10. Inception Module[10]

The InceptionV3 architecture diagram illustrates modular designs for optimizing convolutional neural networks. [10]The left Inception module integrates multiple filter sizes (5x5, 3x3, 1x1) and pooling layers, merging their outputs for diverse feature extraction. The right module demonstrates advanced factorization, decomposing convolutions into smaller operations (e.g., n×1 followed by 1×n) and utilizing parallel structures. This approach reduces computational costs while preserving high representational power, enhancing the network's efficiency and performance in complex tasks.

In our project, I employ the InceptionV3 architecture, pre-trained on ImageNet, as the foundational model for our classification task. I extend it with custom layers: input, flatten, dense (512 units, ReLU), another dense (512 units, ReLU), and an output layer (10 units, softmax). This setup leverages InceptionV3's advanced modular design for enhanced performance and efficiency. As shown in Fig 13.

c). DenseNet121

DenseNet, introduced by Gao Huang et al., revolutionizes deep learning by connecting each layer to every other layer in a feed-forward fashion, ensuring maximum information flow and mitigating the vanishing gradient problem. This architecture promotes feature reuse and significantly reduces the number of parameters compared to traditional convolutional networks.

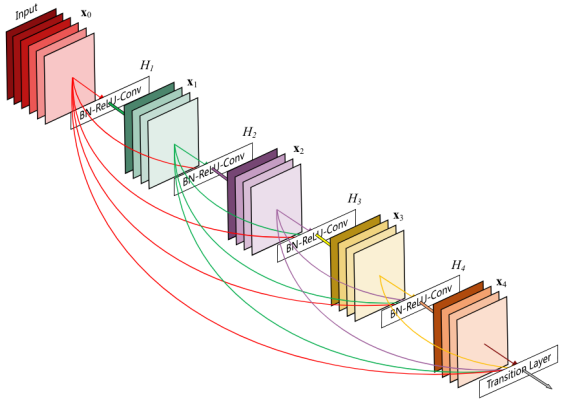


Fig 11. DenseNet Architecture[11]

The diagram from DenseNet illustrates a densely connected convolutional network where each layer receives inputs from all preceding layers and passes its output to all subsequent layers. [11]This connectivity pattern ensures maximum information flow between layers, mitigates the vanishing gradient problem, and promotes feature reuse. The network consists of Batch Normalization (BN), ReLU activation, and Convolutional (Conv) layers, followed by transition layers for down-sampling.

In our project, I utilize the pre-trained model DenseNet121, as the backbone for feature extraction. I enhance it with custom layers: input, batch normalization, dense (256 units), dropout (0.35), another batch normalization, dense (120 units), and an output layer (10 units, softmax). This combination leverages DenseNet's strengths while improving performance and generalization for our classification task. As shown in Fig 14.

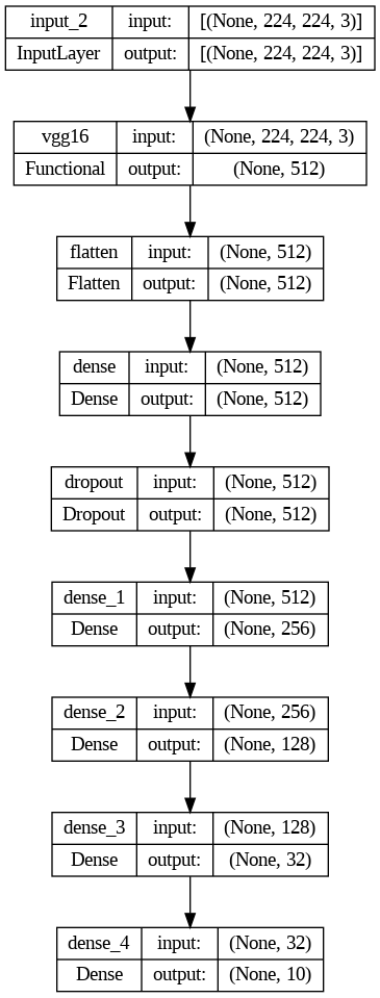


Fig 12. VGG-16

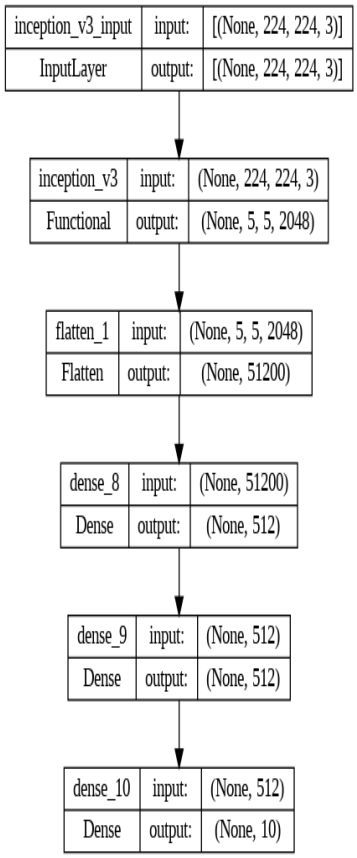


Fig 13. Inception V3

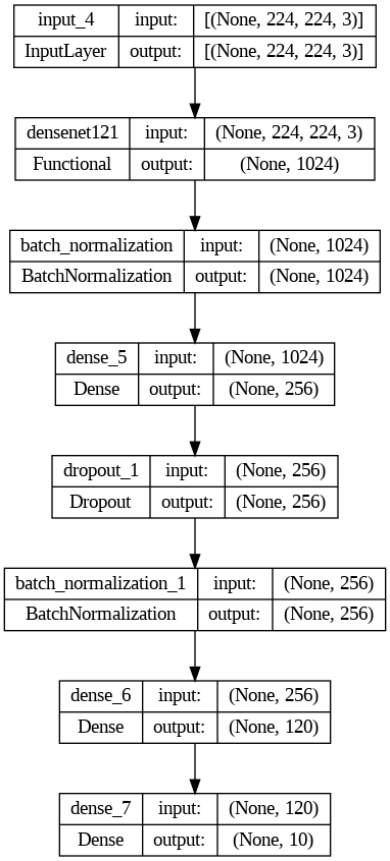


Fig 14. DenseNet121

IV. EXPERIMENT

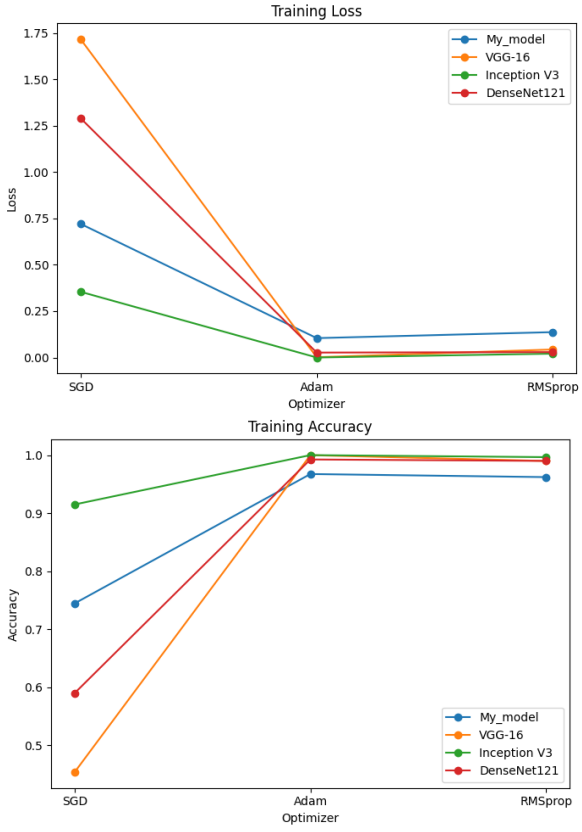
TABLE 1. Comparison of Model PERFORMANCE with Different Optimizers

Learning Rate=0.0001 Epoch=40

Model	***Optimizer***	***Loss***	***Accuracy***	***Val_loss***	***Val_accuracy***	***Trainable params***	***Non-trainable params***
My model	SGD	0.7206	0.7446	1.6410	0.5805	13,018,506	704
My model	Adam	0.1046	0.9674	1.5438	0.8045		
My model	RMSprop	0.1369	0.9621	1.5252	0.8245		
VGG-16	SGD	1.7162	0.4540	1.5905	0.5415	7,510,762	7,635,264
VGG-16	Adam	0.0014	0.9999	0.3341	0.9360		
VGG-16	RMSprop	0.0432	0.9899	0.4476	0.9320		
Inception V3	SGD	0.3543	0.9150	0.5862	0.8085	26,482,698	21,802,784
Inception V3	Adam	1.2589e-04	1.0000	0.5152	0.8720		
Inception V3	RMSprop	0.0208	0.9965	1.7448	0.8220		
DenseNet121	SGD	1.2900	0.5901	1.0932	0.7045	297,010	7,040,064
DenseNet121	Adam	0.0267	0.9925	0.1301	0.9580		
DenseNet121	RMSprop	0.0292	0.9900	0.1234	0.9620		

Table 1 presents the experimental results of four convolutional neural network (CNN) models—My Model, VGG-16, Inception V3, and DenseNet121—using three different optimizers: SGD, Adam, and RMSprop. Each model was trained for 40 epochs with a constant learning rate of 0.0001. The findings reveal that the Adam optimizer consistently delivered the best performance, achieving the lowest training and validation losses and the highest accuracies. For instance, VGG-16 with Adam attained an almost perfect training accuracy (0.9999) and a high validation accuracy (0.9360). RMSprop also demonstrated robust performance, particularly with My Model and DenseNet121, showcasing high accuracy and low loss values. Conversely, the SGD optimizer generally resulted in higher loss values and lower accuracies, indicating its relative inefficacy for these models and datasets. Therefore, the Adam optimizer is recommended for training CNN models on this dataset due to its superior performance metrics.

The following four pictures illustrate these results more clearly, providing a visual comparison of training loss, validation loss, training accuracy, and validation accuracy for each model-optimizer combination.



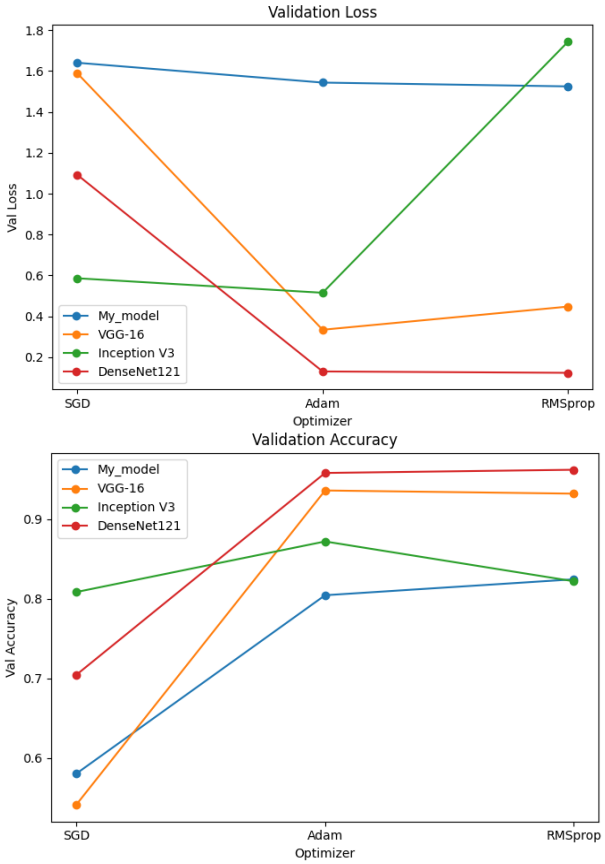


Fig 15. Performance Comparison Plot

As illustrated in Fig 16. The number of trainable and non-trainable parameters for four different models: My model, VGG-16, Inception V3, and DenseNet121. My model and DenseNet121 have significantly fewer non-trainable parameters compared to VGG-16 and Inception V3. Inception V3 has the highest total number of parameters, while DenseNet121 has the fewest trainable parameters. This comparison highlights the trade-offs between model complexity and parameter efficiency in transfer learning for disease detection.

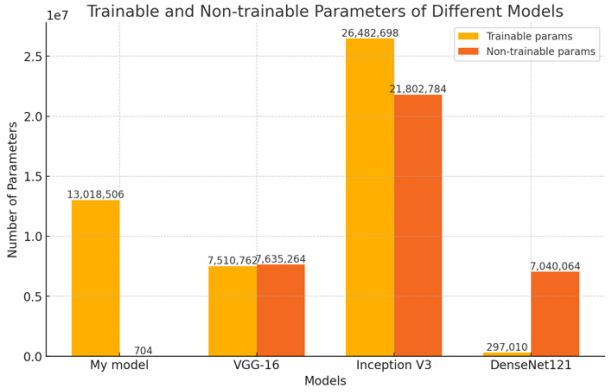


Fig 16. Comparison of number of parameters in various Models

Based on the data and graphs, we observe that the best model and optimizer combination is DenseNet121 with Adam. Consequently, the next step involves running this model again to evaluate additional performance metrics: recall, precision, F1-score, and the ROC curve. This extended evaluation will provide a more comprehensive understanding of the model's performance.

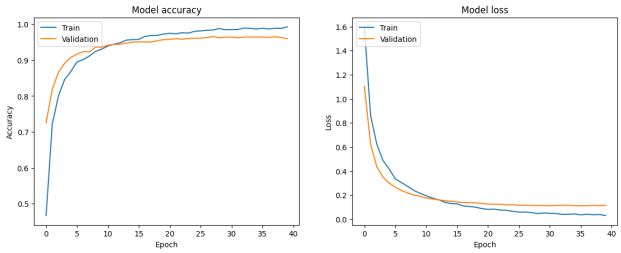


Fig 17. Accuracy and Loss of the Best Model

Recall (Sensitivity, True Positive Rate):

Recall is the ratio of correctly predicted positive observations to all the actual positives. It is calculated as:

$$Recall = \frac{TP}{TP + FN}$$

where TP is the number of true positives, and $FN$is the number of false negatives. High recall indicates that the model captures most of the positive instances.

Precision (Positive Predictive Value):

Precision is the ratio of correctly predicted positive observations to the total predicted positives. It is calculated as:

$$Precision = \frac{TP}{TP + FP}$$

where FP is the number of false positives. High precision indicates that the model has a low false positive rate.

F1-Score:

The F1-score is the harmonic mean of precision and recall, providing a single metric that balances both concerns. It is calculated as:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

A high F1-score indicates that the model maintains a balance between precision and recall.

TABLE 2. Performance Metrics for Tomato Leaf Disease Classification

	Precision	***Recall***	***F1-Score***
Bacterial Spot Disease	0.97	0.97	0.97
Early Blight Disease	0.95	0.90	0.93
Late Blight Disease	0.96	0.96	0.96
Leaf Mold Disease	0.96	0.97	0.97
Septoria Leaf Sport Disease	0.97	0.92	0.94
Two-spotted Spider Mite Disease	0.92	0.95	0.94
Target Spot Disease	0.92	0.91	0.91
Mosaic Virus Disease	0.96	1.00	0.98
Yellow Leaf Curl Virus Disease	0.96	0.98	0.97
Healthy Label	0.98	0.99	0.99
Accuracy			0.96
Macro Avg	0.96	0.96	0.96
Weighted Avg	0.96	0.96	0.96

The results demonstrate that the DenseNet121 model is highly effective in classifying tomato leaf diseases with high precision, recall, and F1-scores across all categories. The overall accuracy of 96% underscores the model's reliability and robustness, making it a valuable tool for early disease detection in agricultural settings.

ROC Curve:

The ROC curve is a graphical representation of a model's diagnostic ability, plotting the true positive

rate (recall) against the false positive rate (1 - specificity) at various threshold settings. The area under the ROC curve (AUC) provides a single scalar value to evaluate the model's overall performance.

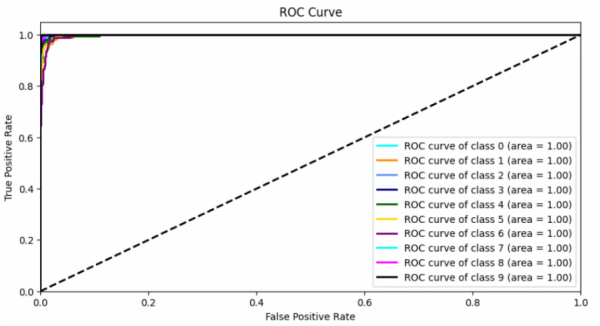


Fig 18. ROC Curve

Each curve corresponds to one of the ten disease categories, plotting the true positive rate (sensitivity) against the false positive rate (1-specificity). Notably, the Area Under the Curve (AUC) for all classes approaches 1.00, demonstrating the model's exceptional performance. This indicates that the model can accurately distinguish between all categories of diseased and healthy leaves, achieving nearly 100% sensitivity and specificity. The near-perfect alignment of the ROC curves along the upper left boundary of the plot underscores the model's superior accuracy and reliability in identifying tomato leaf diseases.

V. CONCLUSION

In this study, I have successfully developed a robust Convolutional Neural Network (CNN) model for the detection and classification of tomato leaf diseases. my approach involved critical steps such as dataset preprocessing to ensure image uniformity, data augmentation to enhance diversity, and the implementation of transfer learning using pretrained models like VGG16, InceptionV3, and DenseNet121. These methodologies collectively contributed to a significant improvement in the model's accuracy and generalization capabilities, enabling it to effectively identify and distinguish various disease features from high-resolution tomato leaf images.

The present research outcomes prove the use of CNN and transfer learning techniques to be a successful tool for agricultural applications. The early and effective identification of tomato leaf diseases may be beneficial for farmers in making decisions about timely preventative measures, thus reducing crop losses and eventually moving towards sustainable farming practices. In this way, with a model able to identify even the slightest dissimilarity in disease symptoms, we can discard the inadequacies of conventional manual diagnosis methods from the consideration that lacks consistency and is void of human judgments, which often lead to errors. Apart from being a great showcase of the potential power of deep learning, in this case, in diagnosis, it is also intended to be a vehicle for introducing modern AI techniques into agriculture.

REFERENCES

[1] SECRETARIAT I,GULLINO M L,ALBAJES R,et al.Scientific review of the impact of climate change on plant pests[M]. FAO on behalf of the lPPC Secretaniat,2021.

[2] P. Baser, J. R. Saini, and K. Kotecha, “TomConv: An improved CNN model for diagnosis of diseases in tomato plant leaves,” Procedia Computer Science, vol. 218, pp. 1825–1833, Jan. 2023, doi: 10.1016/j.procs.2023.01.160.

[3] S. Tammina, “Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images,” International Journal of Scientific and Research Publications, vol. 9, no. 10, p. p9420, Oct. 2019, doi: 10.29322/ijsrp.9.10.2019.p9420.

[4] I. Kandel and M. Castelli, “Transfer Learning with Convolutional Neural Networks for Diabetic Retinopathy Image Classification. A Review,” Applied Sciences, vol. 10, no. 6, p. 2021, Mar. 2020, doi: 10.3390/app10062021.

[5] P. Tm, A. Pranathi, K. SaiAshritha, N. B. Chittaragi and S. G. Koolagudi, "Tomato Leaf Disease Detection Using Convolutional Neural Networks," 2018 Eleventh International Conference on Contemporary Computing (IC3), Noida, India, 2018, pp. 1-5, doi: 10.1109/IC3.2018.8530532.

[6] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, “A comparative study of fine-tuning deep learning models for plant disease identification,” Computers and Electronics in Agriculture, vol. 161, pp. 272–279, Jun. 2019, doi: 10.1016/j.compag.2018.03.032.

[7] M. Brahimi, K. Boukhalfa, and A. Moussaoui, “Deep Learning for Tomato Diseases: Classification and Symptoms Visualization,” Applied Artificial Intelligence, vol. 31, no. 4, pp. 299–315, Apr. 2017, doi: 10.1080/08839514.2017.1315516.

[8] A. K. Rangarajan, R. Purushothaman, and A. Ramesh, “Tomato crop disease classification using pre-trained deep learning algorithm,” Procedia Computer Science, vol. 133, pp. 1040–1047, Jan. 2018, doi: 10.1016/j.procs.2018.07.070.

[9] K. Simonyan and A. Zisserman, “Very deep convolutional networks for Large-Scale image recognition,” arXiv (Cornell University), Jan. 2014, doi: 10.48550/arxiv.1409.1556.

[10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” arXiv (Cornell University), Jan. 2015, doi: 10.48550/arxiv.1512.00567.

[11] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” arXiv (Cornell University), Jan. 2016, doi: 10.48550/arxiv.1608.06993.