Advanced International Journal for Research (AIJFR)
E-ISSN: 3048-7641 ● Website: www.aijfr.com ● Email: editor@aijfr.com

A Comparative Study of Deep Transfer Learning Architectures for Multi-Class Plant Leaf Disease Detection

Dr. Joseph Deril K S [1], Fathimathu Safna C S[2]

[1] Associate Professor, Department of Computer Applications, MES College Marampally
[2] MSc CS Student, Department of Computer Applications, MES College Marampally

Abstract
Plant Leaf Diseases represent a significant risk to global agricultural production. Crops ranging from Peppers and Tomatoes to Potatoes are affected by these diseases. Traditional methods of identifying leaf diseases based primarily on visual inspection have historically been slow and relatively inaccurate. A deep learning-based solution for the automated identification of plant leaf diseases utilizes Convolutional Neural Networks (CNNs) as the primary methodology. To identify leaf images into 15 disease categories, both pre-trained models such as VGG16, EfficientNetB3, Inception V3 and a custom CNN model were used. Using pre-trained models to allow for the use of Transfer Learning helps to mitigate some of the issues associated with computational resource limitations and data limitations in providing faster convergence rates and higher accuracy when compared to training a model from scratch. The Plant Village image collection which contains over 20,000 images of different plant leaf diseases was utilized for training and testing purposes. Each model's performance was evaluated based on its accuracy, loss and generalization capabilities. Additionally, each model was fine-tuned through hyperparameter optimization. As a result, the model that achieved the highest validation accuracy rate of 95% was the EfficientNetB3 model while the second highest accuracy rate was achieved by the Inception V3 model at 92%. This methodology provides an excellent answer to addressing early disease detection, enabling farmers to take the necessary actions quickly to reduce their losses and maximize their harvest.

Keywords: Deep Learning, CNN, Transfer Learning, Image Classification, Custom CNN, VGG-16, EfficientNet.

1. Introduction

There are various ways that cause plant leaf disease. Common dis eases include fungal infections, bacterial blights, and viral infections. Each type of disease can manifest in different symptoms, such as discoloration, wilting, or spots on the leaves, which can significantly affect the health of the plant that affects the plants. The disease is caused by microorganisms, fungi, air pollution, soil contamination, water pollutants, temperature fluctuations, insects, etc. Plant leaf diseases that widely affect farmers in producing a better yield. The traditional method is to find the disease using the naked eye, which is not an effective method. Because of the lack of knowledge of farmers about this, and false predictions that result in a loss. So this paper focuses on accurately diagnosing disease-infected leaves. Deep learning techniques mainly use convolutional neural networks (CNNs), such as pre-trained models like InceptionV3, VGG16,

EfficientNet, and custom CNNs. Early disease detection, which profitably helps the farmers, results in a big yield in production. The existing methods have many drawbacks, including interpretability, computational resources, data dependency, and generalization

2. Literature Review

Geetharamani et al. (1) analyzed 39 classes of plant diseases and used six types of data augmentation. They achieved 96.46% classi- fication accuracy using a Transfer Learning approach.

Wasi Ullah et al. (2) proposed the timely detection of apple leaf diseases using lightweight devices. They introduce AppViT, a hybrid model combining convolution blocks and multi-head self- attention to enhance local and global feature extraction. AppViT which achieves 96.38% precision on the Plant Pathology 2021—FGVC8 dataset, outperforming ResNet-50 by 11.3% and EfficientNet-B3 by 4.3%. The model also achieves a precision of 0.967, a recall of 0.959, and an F1-score of 0.963, demonstrating its efficiency in apple leaf disease detection.

Geetabai S. Hukkeri et al. (3) used image classification techniques to analyze 38 different classes of leaf images. Their study incorporated pre-trained CNN models such as VGG16, ResNet50, InceptionV3, MobileNetV2, AlexNet, and EfficientNet for image classification. The comparison showed that the EfficientNet model achieved the highest accuracy, 97.5%.

Muhammad Umar et al. (4) proposed an improved YOLOv7 model enhanced with SimAM, DAiAM, and an optimized MPConv structure for the detection of seven major tomato leaf diseases. The SIFT technique segments images to extract key regions for CNN classification, and the model achieved an impressive 98.8% accu- racy.

D. S. Joseph et al. (5) evaluated eight deep learning models, such as Xception (0.9580) and MobileNet (0.9464) in maize, MobileNetV2 (0.9632) and MobileNet (0.9628) in wheat, and Xception (0.9728) and InceptionV3 (0.9620) in rice. The proposed model achieved excellent testing accuracies of 0.9704, 0.9706, and 0.9808 on the maize, rice, and wheat datasets, respectively.

Zhang et al. (6) introduced an improved dark channel prior (DCP) dehazing algorithm and proposed the TDR-Model, which is based on the MobileNetV3 architecture. The model is optimized for mobile devices and achieves real-time detection capabilities with 98% accuracy.

Prajwala et al. (7) used the Plant Village dataset, which contains 54,306 images across 14 crops and 26 diseases, including 18,160 images for tomato leaf disease detection. They implemented a modified LeNet architecture with standard image preprocessing and achieved 95% accuracy.

Vijai Singh (8) proposed a technique that utilized a Particle Swarm Optimization (PSO) algorithm, achieving an accuracy of about 98%. The advantages of PSO include its ease of implementation and minimal boundaries for parameter changes. PSO performs better than Genetic Algorithms (GA) in terms

of computational efficiency, although its application may not be as complex due to its simple characteristics.

Rubina Rashid et al. (9) and Waqar Aslam et al. introduced multi-model approaches that integrate heterogeneous data, improving classification accuracy. Their Multi-Model Fusion Network (MMF-Net) achieved an impressive accuracy of about 99.23%. Feature fusion techniques further enhanced the performance and precision.
Eunice et al. (10) used a CNN-based training method for plant leaf disease detection. They employed pre-trained models such as DenseNet-121, ResNet-50, VGG-16, and Inception V4 on a dataset containing fifty thousand images from different plants. The model achieved an impressive accuracy of 99.81%.

3. Methodology

Convolutional Neural Networks are better for image classification. But CNN also has certain drawbacks for better classification. To extract the complex and low-level features, deep CNNs are required; still, we can face an increase in the time complexity during the training period. So here we used the transfer learning approaches, which use pre-trained networks that solve certain problems.

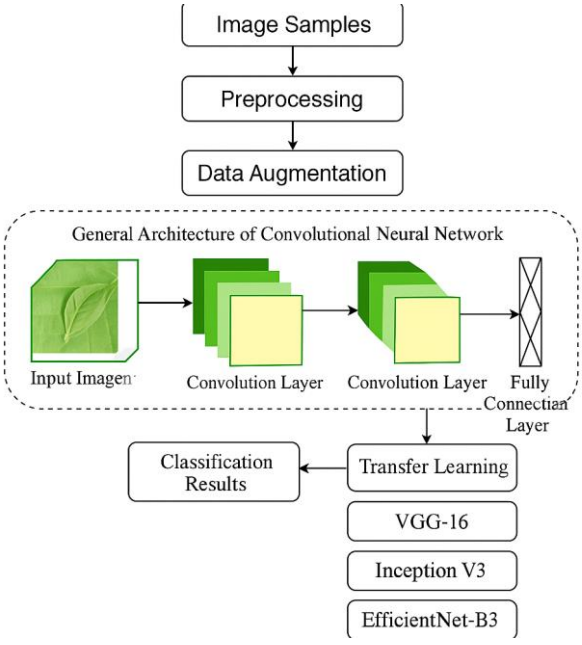

Figure 1. Overall workflow diagram.

3.1 Multi-class classification

The plant disease dataset contains multiple images of diseased and healthy plant samples. This contains the samples of Bell Pepper, Tomato, and Potato leaves from healthy and diseased plants. Consider some examples of plant leaf disease, which are namely "Pepper Bell Bacterial spot, Pepper bell healthy,

Potato Early blight, Potato Late blight, Potato healthy, Tomato Bac- terial spot, Tomato Early blight, Tomato Late blight, Tomato Leaf Mold, Tomato Septoria leaf spot, Tomato Spider mites Two spot- ted spider mite, Tomato Target Spot, Tomato YellowLeaf Curl Virus, Tomato mosaic virus, Tomato healthy. When a sample of one particular disease is fetched as input after training with 15 classes, the testing phase output will classify the exact label of the disease from among the 15 categories mapped under particular classes.

3.2 Transfer Learning

Transfer learning is a machine learning approach that involves reusing a pre-trained model on a new problem. Training [10] and building a model from scratch are time-consuming. A CNN model built from scratch with a publicly available plant disease dataset seemed to attain 25% accuracy in 200 epochs, whereas using a pre-trained CNN model using a transfer learning approach attained
63% accuracy in almost half the number of iterations (over 100 epochs).

3.3 Custom CNN

Custom CNN is termed as the created CNN layer from scratch. Here we use PyTorch classes for a custom convolutional neural network architecture called FasterCNN, which uses a unique Depth wise Separable Convolution approach to improve efficiency. A conventional convolution is divided into two stages by the Depth wise Separable Conv class, which implements a depthwise separable convolution layer. The first stage involves applying a distinct filter to each input channel, and the second step is a pointwise convolution that combines the filtered channels.
This approach rescues the computational complexity compared to standard convolutions. The FasterCNN class is a complete neural network architecture designed for image classification. It uses a sequence of depthwise separable convolution layers alternating with ReLU activation functions and batch normalization, followed by max pooling to progressively extract and down sample features. The network concludes with a global average pooling layer and a fully connected layer that outputs predictions for a specified number of classes. The forward method demonstrates how input images are processed through the convolutional layers, compressed via global average pooling, and then classified by the final fully connected layer.

3.4 Inception V3

Inception V3 is a deep convolutional neural network architecture that was designed for efficient image classification, with 48 layers deep. It is an improved version of earlier Inception models. It is used for factorized convolutions, multiple feature extraction paths, and auxiliary classifiers to efficiently process images and reduce computational complexity while maintaining high accuracy in image classification tasks through its unique multi-scale inception modules and intelligent network design.

3.5 VGG-16

The VGG-16 [11] network model, also known as the Very Deep Convolutional Network for Large-Scale Image Recognition, was built by the Visual Geometry Group from Oxford University. The depth is pushed to 16–19 weight layers and 138 M trainable parameters. The depth of the model is also expanded by reducing the convolution filter size to 3x3. This model requires more training time and occupies more disk space.

3.6 EfficientNet-B3

EfficientNet-B3 is part of the EfficientNet family of convolutional neural networks. EfficientNet-B3 is a third variant offering a balance between accuracy and speed. It is a high-performance, lightweight convolutional neural network that uses compound scaling, which helps to uniformly grow depth, width, and resolution, MBConv blocks for efficient bottleneck layers; Swish activation, a smooth nonlinear function for better learning; and squeeze-and-excitation modules help to focus on important features and achieve high performance with fewer parameters.

4. Experiments

In this section, we describe the details of the dataset, preprocessing and augmentation details, hyperparameters, and network architecture model, which provides more about the evaluation of the model and helps to improve the performance.

4.1 Dataset

The Plant Village (11) dataset is a publicly available dataset with different categories of plant diseases. This dataset consists of 15 classes with 20639 images. For our experiment, we split the dataset into training samples, testing samples, and validation samples. The model was trained with 80% of the Plant Village dataset, and 20% was used for validation and testing. All these train, test, and validation sets include all 15 classes of the different plant diseases. The details of the datasets are presented in Table 1.

Sl. No	Type of Leaf Disease	No. of Images
1	Pepper bell Bacterial spot	997
2	Pepper bell healthy	1478
3	Potato Early blight	1000
4	Potato Late blight	1000
5	Potato healthy	152
6	Tomato Bacterial spot	2127
7	Tomato Early blight	1000
8	Tomato Late blight	1909

9	Tomato Leaf Mold	952
10	Tomato Septoria leaf spot	1771
11	Tomato Spider mites (Two-spotted spider mite)	1676
12	Tomato Target Spot	1404
13	Tomato Yellow Leaf Curl Virus	3209
14	Tomato mosaic virus	373

Table 1: Types of Leaf Disease and Number of Images

4.2 Preprocessing and Augmentation

The dataset held 15 classes with 3 species of crop. We used the color images from the PlantVillage dataset. The images were down-scaled to 224 x 224 pixels as a standardized format that we used for pre-trained models. For VGG-16 and EfficientNet, the input size is 224 x 224, whereas Inception uses 299 x 299 pixels. Preprocessing refers to transforming the image into a suitable format for model training, such as resizing and normalization. To increase the generality and resilience of the model, augmentation techniques such as rotation, zoom, random cropping, horizontal flipping, and brightness change are used.

4.3 Fine-Tuning of Hyperparameters in Pre-Trained Models

Transfer learning enables faster training, requires less data, reduces overfitting, lowers computational cost, and achieves higher accuracy compared to training models like custom CNNs, VGG-16, or Inception from scratch. The details of hyperparameter tuning are listed in Table 2.

Hyperparameter	Value
Dropout	0.5
Epochs	50
Activation	ReLU
Optimizer	Adam optimizer
Learning rate	0.001
Output classes	38/15

Table 2: Hyperparameter Specifications

4.4 Network Architecture Model

The pre-trained network models were picked because they could be used to classify plant diseases. The details of the model architecture are given in 3. Each network has different filter sizes for extracting specific features from feature maps. Filters play a key role in feature extraction. Further, each filter, when convolved with the input, will extract different features from it, and the specific feature extraction from the feature maps depends on the specific values of the filters. In our experiments, we used the actual pre-trained network models with the actual combinations of convolution layers and actual filter sizes used for each network model.

4.4.1 Custom CNN Tuning Details

Tuning the model includes a Dropout rate of 0.5 applied after the third DepthwiseSeparableConv layer to improve generalization, MaxPooling layers with a stride of 2 to reduce spatial dimensions, DepthwiseSeparableConv layers utilizing 3×3 filters for depth wise convolutions and 1×1 filters for pointwise convolutions, and a Fully Connected (Dense) layer with 256 input features and num_classes output neurons.
Together, these contribute to approximately 1.2 million trainable parameters for a dataset with 15 classes.

Sl. No	Component	Custom FasterCNN	VGG16	EfficientNetB3	InceptionV3
1	Total Layers	16	16	16	22
2	MaxPooling Layers	2	5	1	5
3	Dense Layers	1	2	2	2
4	Dropout Layers	1	2	2	2
5	Flatten Layers	Global Average Pooling			
6	Filter Sizes 3	x3, 1x1	3x3	3x3, 5x5, 1x1	1x1, 3x3, 5x5
7	Stride	2(MaxPooling in all models)			
8	Trainable Parameters	1.2M	15.2M	1.8M	2.3M

Table 3: Detailed Breakdown of Network Architectures

4.4.2 VGG-16 Tuning Details

The model can be fine-tuned by unfreezing the final four convolutional layers of the VGG16 base model, using a custom classification head with two Dense layers (1024 neurons with ReLU activa- tion and 15 neurons with Softmax activation), adding two Dropout layers with a regularization rate of 0.5, and reducing the number of spatial dimensions with MaxPooling layers with a stride of 2. It has 15.2 million trainable parameters as a result.

4.4.3 EfficientNetB3 Tuning Details

The model is tuned by freezing all layers of the EfficientNetB3 base model while training only the custom classification head, which includes two Dense layers (1024 neurons with ReLU and 15 neurons with Softmax), two Dropout layers with a 0.5 rate for regularization, and a GlobalAveragePooling2D layer, resulting in approximately 1.8 million trainable parameters.

4.4.4 Inception Tuning Details

The model is tuned by freezing all layers of the InceptionV3 base model while training only the custom classification head, which consists of two Dense layers (1024 neurons with ReLU and 38 or 15 neurons with Softmax), two Dropout layers with a 0.5 rate for regularization, and a GlobalAveragePooling2D layer, resulting in approximately 2.3 million trainable parameters.

4.5 Results and Discussion

After my training on the PlantVillage dataset, I verified the model's accuracy on the test data, and it scored 85-90 percent. Over time, the training loss decreased, and accuracy improved, thus confirming that the early stopping rule was engaged with no validation accuracy improvement for five rounds. It was trained on the Adam op- timizer with a 0.0001 learning rate. CrossEntropyLoss, along with accuracy, was utilized during training. There was a maximum number of 50 epochs, each with a batch size of 64 and data augmentation alongside the model. The best model.pth was saved, alongside a graph which visualizes the training loss and accuracy of the model over time.

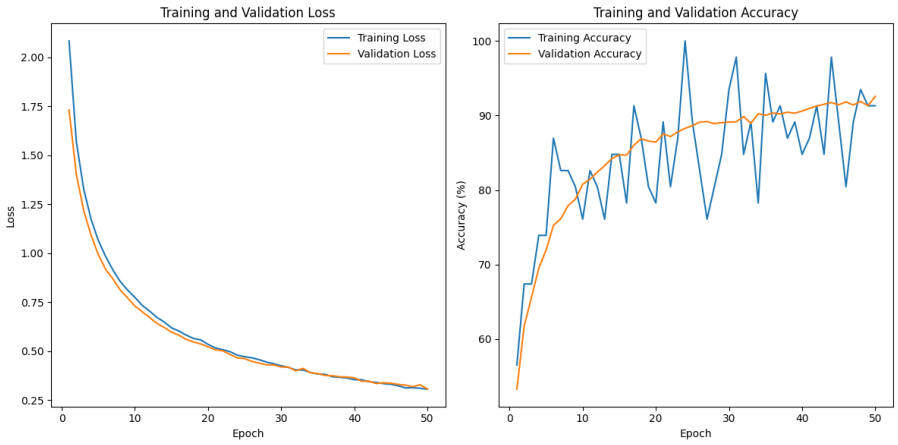

Figure 2.CNN model training and validation loss and accuracy curves.

The VGG16-based model using the same dataset achieved a test accuracy of 85-90 percent based on the size of the dataset and class distribution. The training loss during the process also continuously reduced, and there was a steady increase in training accuracy, further proving the learning of the model. In order to avoid overfitting, early stopping was implemented by halting the model training when it could not improve the validation accuracy over 5 epochs. It was trained on Adam optimizer with a 0.0001 learning rate, and the loss function used was CrossEntropyLoss.There was a maximum of 50 epochs, each with a batch size of 64 and data augmentation alongside the model. The model that achieved the best result was saved as the VGG16 model. keras, along with a graph visualizing the model's accuracy and loss over time.

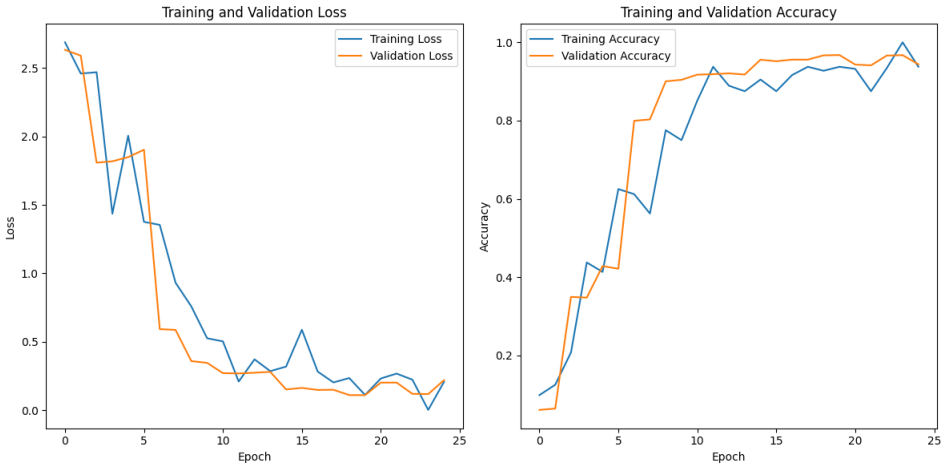

Figure 3: VGG16 model training and validation loss and accuracy curves.

The EfficientNetB3-based model achieved a validation accuracy of 85-90 percent based on the size of the dataset and class distribution, and training accuracy of approximately 90-95 percent. The training loss during the process also continuously reduced and there was a steady increase in training accuracy, further proving the learning of the model. In order to avoid overfitting, early stopping was implemented by halting the model training when it could not improve the validation accuracy over 5 epochs. It was trained on Adam optimizer with a 0.0001 learning rate, and the loss function used was CrossEntropyLoss.There was a maximum number of 50 epochs, each with a batch size of 64 and data augmentation alongside the model. The model that achieved the best result was saved as the best model. keras, along with a graph visualizing the model's accuracy and loss over time.

Figure 4: EfficientNetB3 training and validation loss and accuracy curves.

The fourth experiment was taken with the Inception V3 model. The same method was applied in the evaluation of model loss and accuracy. It achieved an accuracy of approximately 90%. It was trained on the Adam optimizer with a 0.0001 learning rate, and the loss function used was CrossEntropyLoss. The training process showed a continuous decrease in loss and steady improvement in validation accuracy. The model that achieved the best result was saved as an inception_model.keras, along with a graph visualizing the model's accuracy and loss over time.

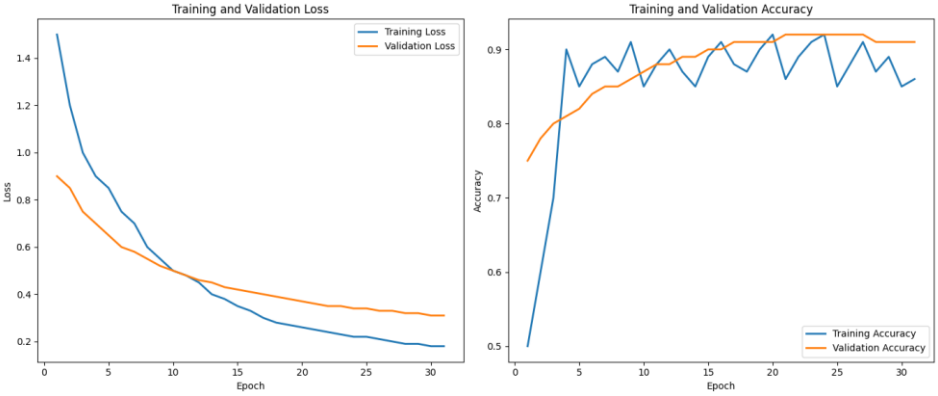

Figure 4: Inception V3 training and validation loss and accuracy curves.

Model	Train Accuracy	Val Accuracy	Train Loss	Val Loss
VGG16	83%	88%	0.75	0.37
Custom CNN	85%	91-92%	0.18	0.32
Inception v3	92%	92%	0.30	0.30
EfficientNet	95%	95%	0.20	0.20

Table 4: Model Performance Metrics

The graphs for VGG-16 (Fig. 2), Custom CNN (Fig. 1), Inception V3 (Fig. 3) and EfficientNet B3 (Fig. 4) . But it's EfficientNet B3 that really stands out. By epoch 30, its training loss has dropped to about 0.18—and stays there. Validation loss stabilizes around 0.32 at the same point. Accuracy-wise, training accuracy reaches about 85%, while validation accuracy climbs steadily to around 91-92%. That's a very stable progression with minimal fluctuation, indicating good generalization capability.

Custom CNN Fig. 1 shows the opposite. It displays significant fluctuations in training metrics with a highly volatile training accuracy and loss curve. Meanwhile, validation metrics show more stability, with validation accuracy gradually increasing to around 88-89% by epoch 30. The validation loss decreases more smoothly than the training loss, settling around 0.37. There seems to be a huge difference in performance on what the model learns versus what it's really capable of. It looks like the training process might not be stable.

That volatility suggests potential problems with training stability.

Inception V3 (Fig. 3) was trained for 50 epochs. Both loss functions seemed to plateau steadily over time and eventually met near 0.3. Training accuracy spikes above 90% and sometimes reaches 100%—while validation accuracy shows a steadier upward trend to about 92%. That convergence of training and validation loss in later epochs is a good sign of generalization after sufficient training.

EfficientNet B3 (Fig. 4) demonstrates rapid performance improvement. Both training and validation loss drop quickly from initial values above 2.5 to below 0.5 by epoch 10. Then they both keep going down to around 0.20. Accuracy curves show rapid improvement, too, with validation accuracy surpassing training accuracy early on and reaching approximately 95% by the end of training. The metrics for final training and validation are now very closely bunched up, which means the model is performing really well at generalizing and does great at leaving the training data behind. There are few, if any, signs of overfitting, either because it can't get too close to performance on just the training data.

That's why EfficientNet B3 stands out. Its rapid convergence, high validation accuracy (95% and excellent generalization make it the most reliable choice. It gets really good at losing almost nothing, and then every stage in the training goes along really smoothly while not getting too wedded just to the training data it got to see.

4.6 Conclusion

This article delves into the use of Convolutional Neural Networks (CNNs) to spot diseases in plant leaves with high precision. We tested both off-the-shelf models and tailor-made CNN designs to get top-notch results in disease identification. Our research covered various CNN models like VGG16, EfficientNetB3, Inception V3, and a unique model called FasterCNN. The study shows that these deep learning methods

help farmers a lot. They excel at spotting and grouping different plant diseases, with success rates between 85% and 95%. One smart move was to use transfer learning with prerained models. This led to great outcomes while saving on resources and training time. EfficientNetB3 stood out among the models for its accuracy and thrifty use of resources making it a good fit for real-time farm disease detection. While the custom CNN model had some ups and downs in its results, it still gave us useful insights into the good and bad points of building custom models for specific jobs. These findings highlight how crucial it is to catch diseases to boost crop yields and lessen the blow of plant diseases on farming.As deep learning keeps getting better in making models more productive and flexible, systems for spotting plant diseases could shake up how farming is done. These systems can provide farmers with timely and dependable ways to sustain crop health. In the future, researchers could explore the possibility of implementing these models into other practical devices that anyone could use. These devices could support farmers in monitoring their crops for diseases while they farm, and also guide them to effectively manage resources and protect their crops.

REFERENCES

1. G. Geetharamani et al., "Identification of plant leaf diseases using a nine-layer deep convolutional neural network," *Computers and Electronics in Agriculture*, vol. 157, pp. 41-49, 2019. Available: https://www.sciencedirect.com/science/article/abs/pii/S0045790619300023.
2. Wasi Ullah, et al., "Timely detection of apple leaf diseases using lightweight devices: AppViT," Journal Name, vol. Volume Number, no. Is- sue Number, pp. Page Numbers, Year. Available: https://link. springer.com/article/10.1007/s43621-024-00307-1.
3. Geetabai S. Hukkeri, et al., "Image classification for leaf disease detection using pre-trained CNN models," Journal Name, vol. Vol- ume Number, no. Issue Number, pp. Page Numbers, 2024. DOI:10.2174/0118743315305194240408034912.
4. M. Umar, S. Altaf, S. Ahmad, H. Mahmoud, A. S. N. Mohamed, and R. Ayub, "Precision agriculture through deep learning: Tomato plant multiple diseases recognition with CNN and improved YOLOv7," *IEEE Access*, vol. 12, pp. 49167-49183, 2024. doi: 10.1109/ACCESS.2024.3383154.
5. D. S. Joseph, P. M. Pawar, and K. Chakradeo, "Real-time plant disease dataset development and detection of plant disease using deep learning," IEEE Access, vol. 12, pp. 16310-16333, 2024. doi: 10.1109/ACCESS.2024.3358333. Available: https://ieeexplore.ieee.org/author/37089899660.
6. Z. Zhang, T. Liu, J. Gao, M. Yang, W. Luo, and F. Lin, "TDR-Model: Tomato disease recognition based on image dehazing and improved MobileNetV3 model," IEEE Access, vol. 13, pp. 852-865, 2025. doi: 10.1109/ACCESS.2024.3522101.
7. Prajwala et al., "Tomato leaf disease detection using the Plant Village dataset and a modified LeNet architecture," in Proceedings of [Conference or Journal Name], 2019.
8. V. Singh, "Particle swarm optimization for plant disease detection," Artificial Intelligence in Agriculture*, vol. 3, pp. 1-10, 2019. doi:10.1016/j.aiia.2019.09.002.
9. R. Rashid, W. Aslam, R. Aziz, and G. Aldehim, "An early and smart detection of corn plant leaf diseases using IoT and deep learning multi-models," IEEE Access, vol. 12, pp. 23149-23162,

2024. doi:10.1109/ACCESS.2024.3357099. Available: https://ieeexplore.ieee.org/author/37088923128.

10. E. Eunice, J. J., D. E. Popescu, M. K. Chowdary, and J. Hemanth, "Deep learning-based leaf disease detection in crops using images for agricultural applications," Agronomy, vol. 12, no. 10, p. 2395, 2022. doi: 10.3390/agronomy12102395. Available: https://doi.org/10.3390/agronomy12102395.
11. E. Marrex, "Plant Disease Dataset," Kaggle, Available: https://www.kaggle.com/datasets/emmarex/ plantdisease. [Accessed: 03- Apr-2025].