Tomato Disease Classification using Different Deep Learning Approaches

Biplove Pokhrel[1], Raj Kiran Chhatkuli[1*], Roshan Subedi[1], Suresh Timilsina[1], Santosh Panth[2]

[1]Department of Electronics and Computer Engineering. Pashchimanchal Campus, Institute of Engineering, Tribhuvan University, Pokhara, Nepal

[2] Department of Software and Computer Engineering, Gandaki College of Engineering and Science, Pokhara, Nepal

*rajkiran@ioepas.edu.np

(Manuscript Received: 1st February, 2026; Revised: 13th April, 2026; Accepted:18th April, 2026)

Abstract

Accurate identification of tomato diseases from leaf images is a complex task that presents significant challenges even for agricultural experts. This study addresses this issue by developing a deep learning-based classification system using Convolutional Neural Networks (CNNs). Specifically, we investigated the performance of various MobileNet architectures (V2, V3 Small, and V3 Large) trained from scratch on a standardized dataset of 224x224 pixel images. The models were evaluated based on their ability to classify leaf images into distinct disease categories. Experimental results demonstrated that MobileNetV3 Large achieved the highest test accuracy of 96.9%, outperforming MobileNetV3 Small (96.34%) and MobileNetV2 (93.8%). Through hyper parameter tuning and comparative evaluation, the MobileNetV3 Large model was selected as the optimal classifier for deployment. The findings suggest that efficient Mobile Net architectures provide a robust solution and light weight model for automated plant disease detection, offering a viable tool for precision agriculture.

Keywords: Convolution Neural Network, Hyper Parameter Tuning, MobileNet, Precision Agriculture

1. Introduction

1.1 Background

Agriculture is a vital global industry essential for food security, with tomatoes ranking among the most widely consumed crops worldwide. However, tomato production is frequently threatened by a variety of pathogens that significantly diminish both crop yield and quality. Effective disease management and treatment rely fundamentally on early and accurate diagnostics; delayed identification often results in the rapid spread of infection and substantial agricultural loss.

Traditionally, disease detection has depended on manual visual inspections conducted by experts. While effective to a degree, these existing approaches present significant drawbacks: they are labor-intensive, time-consuming, and often lack the precision required for large-scale farming. Consequently, there is a pressing need for automated systems that can provide rapid and reliable diagnostics.

Recent advancements in computer vision and deep learning have paved the way for automated plant disease identification. Image classification models, particularly those based on Convolutional Neural Networks (CNNs), are capable of automatically extracting key features and patterns from images to assign labels to unseen data. This capability is critical for applications ranging from medical image analysis to object detection. In the realm of image classification, two primary paradigms exist: supervised and unsupervised learning. Unsupervised classification is typically utilized when pre-defined labels are unavailable, relying on algorithms to cluster data based on inherent structures. However, when labeled data is available, supervised image classification is the standard approach. In this method, the model is trained on a dataset where every input is associated with a known output label, enabling the algorithm to learn the patterns necessary to predict labels with high precision.

This study focuses on the development of a robust, automated model for classifying tomato diseases using a supervised deep learning approach. Specifically, we evaluate the performance of three efficient MobileNet architectures: MobileNet V2, MobileNet V3 Small, and MobileNet V3 Large to determine which provides the highest classification accuracy. The model is designed to identify and categorize eleven distinct tomato diseases commonly encountered by farmers: Late Blight, Early Blight, Septoria Leaf Spot, Tomato Yellow Leaf Curl Virus, Bacterial Spot, Target Spot, Tomato Mosaic Virus, Leaf Mold, Two-spotted Spider Mites, and Powdery Mildew.

1.2 Related Work

Several researchers have turned their attention towards the MobileNet architecture because it is known to be efficient, which is important in the mobile environment. Zaki et al. (2020) adopted a Computer Vision technique based on MobileNet V2 to classify three tomato diseases. They had their model trained on a smaller portion of PlantVillage data with 4,671 images, and the accuracy of their fine-tuned model was greater than 90%, which shows that smaller deep learning models are capable of achieving great accuracy.

In this work, Verma et al. (2021) used MobileNetV2 for a wider set of 38 different plant species including healthy and infected plants. They tested their model, which was found to be more accurate than other competitive models, and also more efficient in computational cost, which makes it very suitable to be used in network training and simulation in the plant classification process.

Efficient as MobileNet is, other works have tried to compare it with heavier architectures to find the best accuracy-to-speed tradeoff. Trivedi, Pandey, Singh and Jaiswal (2021) suggested the development of a custom High Performance Deep Neural Network (HPDNN) for the early detection of tomato leaf diseases. They optimized hyperparameters like learning rate and epochs and evaluated their proposed model with some pre-trained models like MobileNet, Inception V3 and VGG16. They found that the custom CNN model achieved higher rates of accuracy than the pre-trained CNN models.

Likewise, Rangarajan et al. (2018) performed a comparative research work that classifies tomato crop diseases with AlexNet and VGG16. They optimized the parameters of the AlexNet (learning rate, bias, and epochs) and compared the results with VGG16 using their dataset to find that the former achieved higher accuracy. They discovered that the choice of architecture indeed plays a major role in the performance.

Researchers have started to combine attention mechanisms with the standard architectures to further improve feature extraction and reduce noise. To classify diseases of different types in the 12 different fruits, Wang et al. (2023) created a framework using a model called Attention MobileNet V2. They used an Otsu algorithm based on Naive Bayes model to separate the leaf area from the background. The results of the study showed that the dual attention mechanism adopted in the MobileNet framework helps to identify diseases quickly and accurately.

All these studies demonstrate the promise of D.L. for agricultural diagnostics. While MobileNet V2 has been extensively investigated, however, newer versions like MobileNet V3 are yet to be explored and could have the potential of improving detection accuracy and efficiency.

2. Methodology

2.1 System model Block Diagram

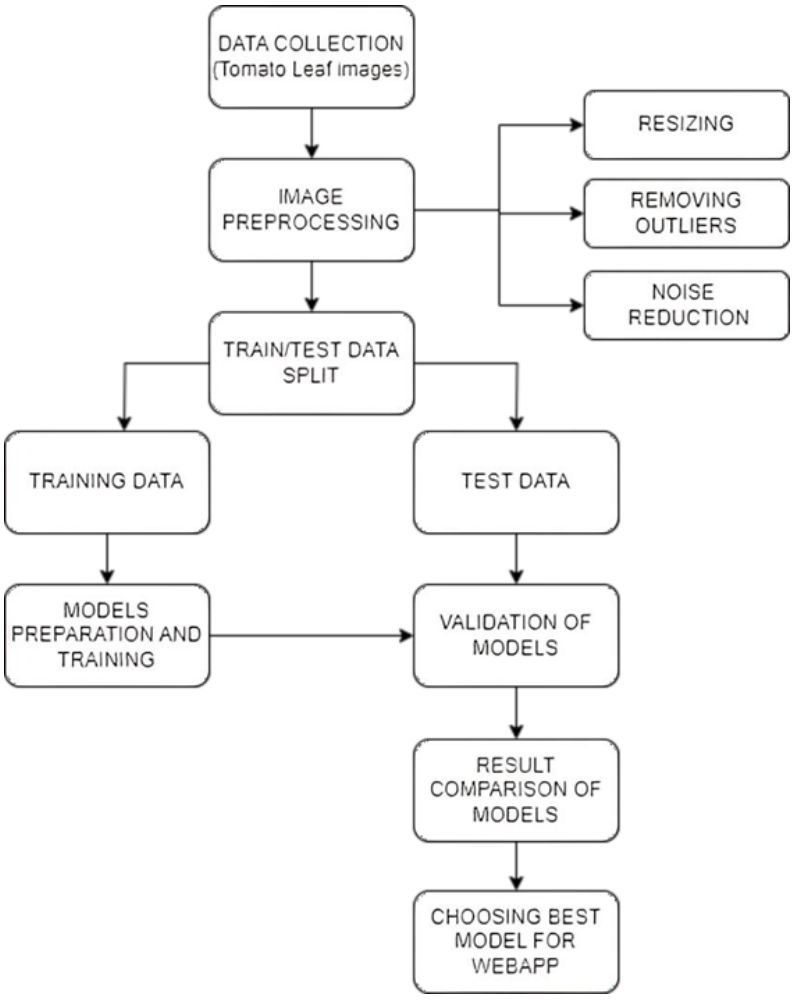

Fig. 1: System Model Block diagram

2.2 Data Collection

Over 20k images of tomato leaves with 10 diseases and 1 healthy class. Images are collected from both lab scenes and in-the-wild scenes. The goal is to develop a lightweight model that can predict tomato leaf disease & deploy it offline on a mobile app.

Classes:

- Late_blight
- healthy
- Early_blight
- Septoria_leaf_spot
- Tomato_Yellow_Leaf_Curl_Virus
- Bacterial_spot
- Target_Spot
- Tomato_mosaic_virus
- Leaf_Mold
- Spider_mites Two-spotted_spider_mite
- Powdery Mildew

The data has been augmented offline using multiple advanced techniques like image flipping, Gamma correction, noise injection, PCA color augmentation, rotation, and scaling. Some recent images were generated offline with GANs. The subset of images containing Taiwan tomato leaves was augmented using rotations at multiple angles, mirroring, reducing image brightness, etc.

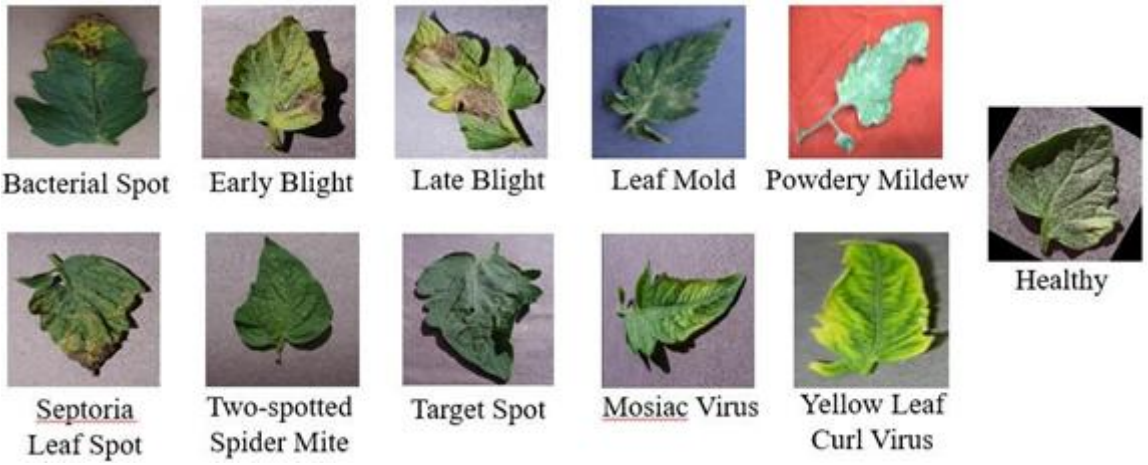

Fig. 2: Dataset of Tomato Leaves

2.3 Image Processing

For tomato disease classification, which is considered an image classification problem, the image needs to be preprocessed in order to be used to train a classification model. This process brings in the uniformity, clarity and relevance of the images provided to the model, thereby enhancing the model's performance.

Resizing is one of the most basic image processing operations, especially for deep learning networks requiring a specific image size as input. All images were resized to 224×224 pixels here, as this is the size required by many pre-trained models, such as ResNet, VGG and MobileNet. This standardization is important because the image size will be uniform, allowing the same model to be used for processing all images in the dataset. Bilinear Interpolation was used to resize the image, which involves taking the weighted average of the four nearest pixels in the original image to determine the new one. This technique has the advantage of maintaining smoothness and detail in the image, especially when compared to more straightforward techniques such as nearest-neighbor interpolation, which can produce jagged edges. Resizing it also helps to reduce the computational load because it makes the input image small enough to be trained and still have enough detail to identify healthy and diseased tomato leaves (Kadhim & Ghathwan, 2022). Resizing may produce a slight loss of fine detail in an image, affecting the accuracy of its classification since small elements important for classification can become obscured.

To rectify that, the images were sharpened by using Unsharp Masking technique (Shaziya, 2020). Unsharp masking is accomplished by subtracting a blurred version of the image from the original, thus increasing the contrast of edges and fine detail. This is particularly beneficial for tomato disease classification as some important attributes such as leaf spots, vein, etc., are highlighted and the model is trained to see the most relevant part of the plant to help distinguish disease from healthy plants. The sharpening process guarantees that information that is important for the model to learn from the image remains crisp and distinct, even if the image is downscaled, increasing its probability of accurate diagnosis. By focusing on the fine details of the images, this step enhances the dataset's quality, making it more informative for the training process.

The third stage of image preprocessing is to eliminate the picture with the text information by Optical Character Recognition (OCR) (Mukkamala & Hein, 2017). This is achieved by using the Tesseract OCR engine to find and extract any text that is contained within the images. The image is flagged and removed from the dataset if any text is found. This step aims to make sure that the model is trained to learn visual features, instead of being biased by irrelevant textual features, e.g. disease symptoms on tomato leaves. Deleting these images will enhance the model's generalization skill and

minimize the possibility of bias in the model, which will ultimately result in better performance in identifying diseases on tomato leaves, using just the visual attributes.

2.4 Algorithms

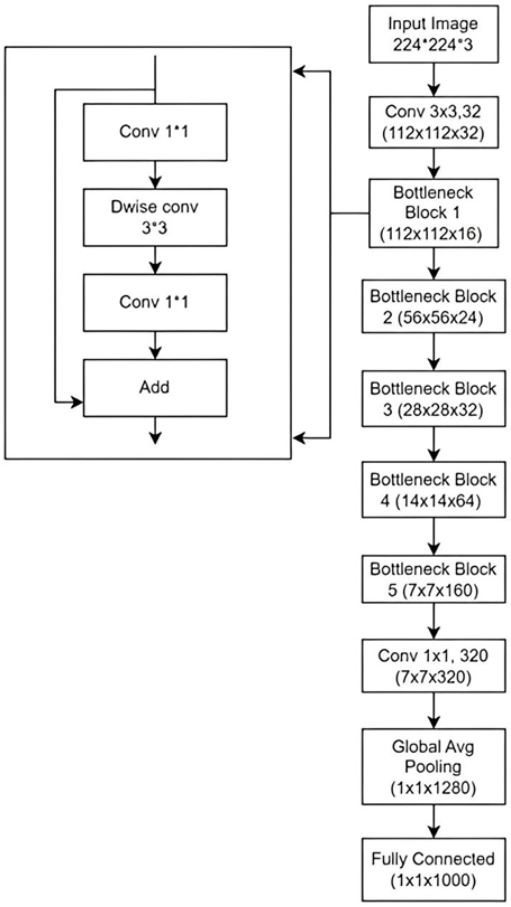

Fig. 3: Mobilenet V2 Architecture

2.4.1 MobilenetV2

MobileNetV2 is based on inverted residual structure, and it mainly includes convolutional layer sequence, depthwise separable convolutions (1-step and 3-steps), linear bottlenecks (bottleneck block + shortcut detour) schemes. These components work together and help to reduce the number of parameters as well as computation while still allowing it to learn complex features. MobileNetV2 when using for the first layer goes through this approach of a depthwise separable convolution, which is only one part of the main mobile net architecture. It breaks standard convolution into two different operations termed as depthwise convolution and pointwise one. This separation allows for far fewer computations that need to be made which makes the model more efficient. One of the key component of the MobileNetV2 model is inverted residuals to further improve its accuracy. They employ a bottleneck design and they first widen the number of channels before depthwise separable convolutions are applied. The model can now learn more complex features and be able to represent those features. The bottleneck design in MobileNetV2 further reduces the computational cost by using 1×1 convolutions to reduce the number of channels before applying depthwise separable convolutions. This design choice helps maintain a good balance between model size and accuracy. Linear bottlenecks are introduced in MobileNetV2 to address the issue of information loss during the bottleneck process. By using linear activations instead of non-linear activations, the model preserves more information and improves its ability to capture fine-grained details. Squeee-and-excitation (SE) blocks are added to MobileNetV2 to

enhance its feature representation capabilities. These blocks adaptively recalibrate the channel-wise feature responses, allowing the model to focus on more informative features and suppress less relevant ones.

2.4.2 MobilenetV3 small and MobilenetV3 large

MobileNetV3 is an advanced version of the MobileNet architecture, designed to further enhance efficiency and performance on mobile and edge devices. Building on the foundation of MobileNetV2, which introduced the concept of Inverted Residuals and Linear Bottlenecks to balance computational efficiency and accuracy, MobileNetV3 incorporates several significant improvements. One of the key advancements in MobileNetV3 is its use of Neural Architecture Search (NAS), which automatically optimizes the model design for better performance and efficiency. This is complemented by the integration of Squeeze-and-Excitation (SE) blocks, which enhance the network's ability to focus on important features by recalibrating channel-wise weights. Additionally, MobileNetV3 employs the Hard-Swish activation function, providing smoother and more effective activation compared to the previous version. These updates make MobileNetV3 not only more efficient but also more accurate, with two main variants MobileNetV3 Large and MobileNetV3 Small catering to different computational requirements and application needs. Overall, MobileNetV3 represents a significant leap forward in optimizing deep learning models for mobile and edge environments.

3. Results

3.1 Using MobilenetV2

The accuracy of MobileNetV2 model was evaluated by modifying the hyperparameters. The hyperparameter namely batch-size, learning and activation function in hidden layer were optimized to find final accuracy of the model. The table below provides different accuracy obtained by using different combinations of hyperparameters used and below it is confusion matrix and per class precision, recall and f-1 score of the optimized model:

Table 1: Accuracy using different combinations for MobilenetV2

Activation function	Batch size	Learning Rate	Accuracy	Remarks
Relu6	16	0.001	93.8%	Maximum
	32	0.001	93.18%	
	64	0.001	92.8%	
Relu	32	0.001	92%	Minimum
	32	0.0001	93.3%	
	32	0.00001	92.36%	

Table 2: Perclass precision, recall and f1-score of optimized model

Classes	Precision	Recall	F1-score
Bacterial spot	0.97	0.88	0.92
Early Blight	0.89	0.89	0.89
Late Blight	0.88	0.93	0.91
Leaf mold	0.95	0.94	0.94
Septoria_leaf_spot	0.89	0.88	0.89
Two-spotted spider mite	0.95	0.9	0.92
Target_spot	0.88	0.91	0.89
Tomato_Yellow_Leaf_Curl_Virus	0.98	0.97	0.97
Tomato_mosiac_virus	0.95	0.97	0.96
Healthy	0.96	0.98	0.97
Powdery mildew	0.94	0.96	0.95

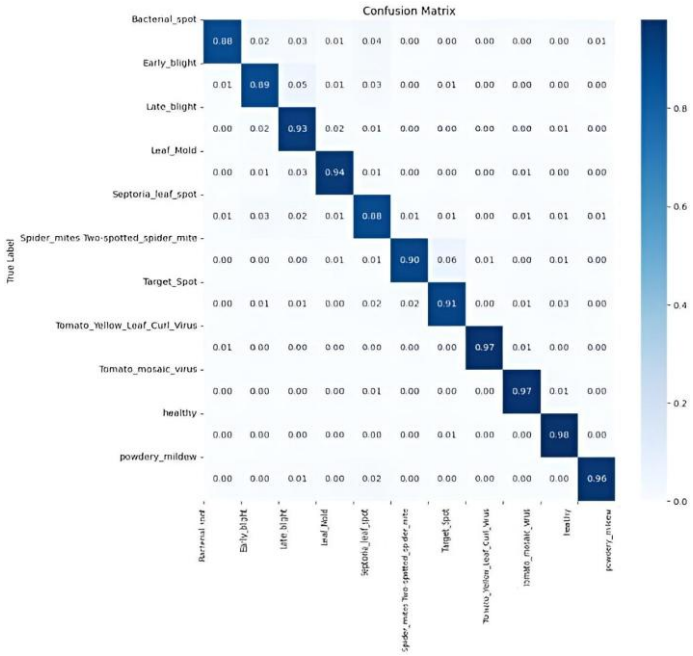

Fig. 4: Confusion matrix

3.2 Using MobilenetV3 Small

The accuracy of MOBILENETV3 Small model was evaluated by modifying the hyperparameters. The hyperparameter namely batch-size, learning were optimized to find final accuracy of the model. Here, activation function Hard-swish was used for all combinations as Hard-swish comes under the architecture of MobileNetV3. The table below provides different accuracy obtained by using different combinations of hyperparameters used and below it is confusion matrix and per class per class precision, recall and f-1 score of the optimized model:

Table 3: Accuracy using different combinations for MobilenetV3

Batch size	Learning Rate	Accuracy	Remarks
32	0.001	94.9%	
32	0.0001	95.9%	Maximum
32	0.00001	95.3%	
16	0.0001	95.46%	
64	0.0001	95%	

Table 4: Perclass precision, recall and f1-score of optimized model

Classes	Precision	Recall	F1-score
Bacterial_spot	0.95	0.98	0.96
Early_Blight	0.95	0.94	0.95
Late_Blight	0.96	0.97	0.97
Leaf_mold	0.98	0.97	0.97
Septoria_leaf_spot	0.95	0.92	0.94
Two-spotted spider_mite	0.97	0.94	0.96
Target_spot	0.92	0.96	0.94
Tomato_Yellow_Leaf_Curl_Virus	0.99	0.98	0.99
Tomato_mosiac_virus	0.99	0.98	0.98
healthy	0.98	0.98	0.98
powder_mildew	0.97	0.96	0.96

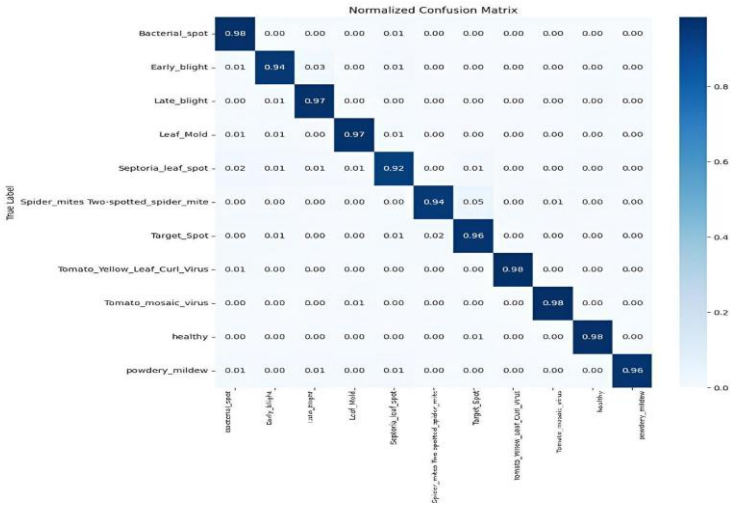

Fig. 5: Normalized Confusion matrix for MobilenetV2

3.3 Using MobilenetV3 Large

The accuracy of MOBILENETV3 Large model was evaluated by modifying the hyperparameters. The hyperparameter namely batch-size, learning were optimized to find final accuracy of the model. Here, activation function Hard-swish was used for all combinations as Hard-swish comes under the architecture of MOBILENETV3. The table below provides different accuracy obtained by using different combinations of hyperparameters used and below it is confusion matrix and per class per class precision, recall and f-1 score of the optimized model :

Table 5: Accuracy using different combinations for MobilenetV3 Large

Batch size	Learning Rate	Accuracy	Remarks
32	0.001	96.36%	Minimum
32	0.0001	96.72%	
32	0.00001	95.36%	
16	0.0001	96.9%	Maximum
64	0.0001	96.5%	

Table 6: Perclass precision, recall and f1-score of optimized model

Classes	Precision	Recall	F1-score
Bacterial_spot	0.98	0.97	0.97
Early_Blight	0.95	0.95	0.95
Late_Blight	0.96	0.97	0.96
Leaf_mold	0.98	0.98	0.98
Septoria_leaf_spot	0.95	0.94	0.94
Two-spotted spider_mite	0.95	0.97	0.96
Target_spot	0.95	0.94	0.94
Tomato_Yellow_Leaf_Curl_Virus	0.99	0.99	0.99
Tomato_mosiac_virus	0.99	0.99	0.99
healthy	0.98	0.99	0.99
powder_mildew	1	0.97	0.98

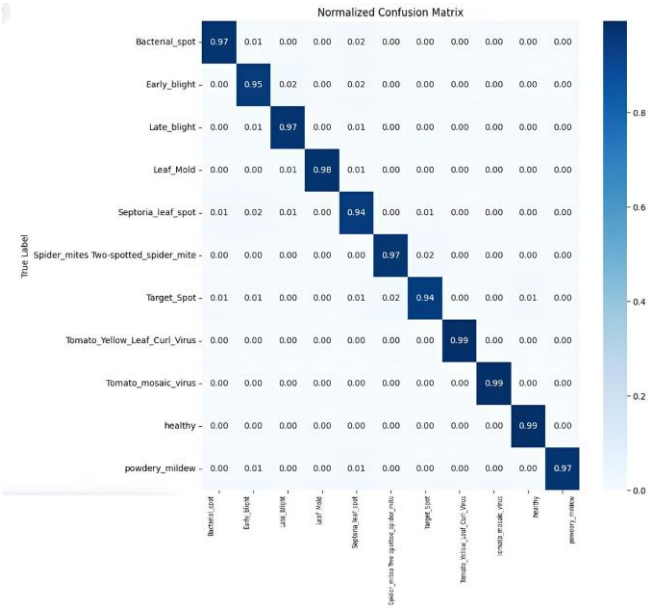

Fig. 6: Normalized Confusion Matrix for MobileNetV3 Large

4. Discussions and Conclusion

The state-of-the-art architectures-MobileNetV2, MobileNetV3 Small, and MobileNetV3 Large-for disease classification in tomato leaves using a publicly available dataset from Kaggle was adopted. The total number of images in the dataset is 32,500, with 11 classes, of which there are 10 categories regarding diseases and one healthy class. We have utilized an extensive preprocessing pipeline to standardized images as input for these models. Examples of hyperparameters that have been tuned in search of maximum accuracy include batch size, learning rate, and activation functions.From the table given below, we can see that MobileNetV2 was much more efficient with respect to computational cost, while accuracy was as good as up to 93.8% using ReLU6 for activation and with a learning rate of 0.001. As we advanced from that toward better architectures, including MobileNetV3 Small and MobileNetV3 Large, there was quite a difference in the improvements. MobileNetV3 Small reached 95.9% using the Hard-swish activation, while both had a learning rate of 0.0001. On the other hand, MobileNetV3 Large outperformed them with the maximum accuracy of 96.9% for the same Hard-swish activation function but an optimized learning rate of 0.0001. Hence, MobileNet V3 Large is used for deployment of the Web application.

Table 7: Final Accuracy and Computational time for MobileNet versions.

Algorithms	Final Accuracy	Computational Time(in sec)
Mobile Net V2	93.8%	2.18
Mobile Net V3 Small	95.9%	4.37
Mobile Net V3 Large	96.9%	7.63

These results underline how powerful the models of MobileNetV3 are, particularly in resource-constrained setups such as mobile and edge devices, where computational efficiency with high accuracy is regarded as key. Specifically, an introduction of a combination of inverted residual blocks, squeeze-and-excitation modules, and Hard-swish activation in MobileNetV3 provides significant improvement both in model performance and accuracy compared to its earlier versions such as MobileNetV2.

Conflicts of Interest

The authors declare no conflict of interest.

References

Siti Zulaikha Muhammad Zaki, et al. "Classification of Tomato Leaf Diseases Using MobileNet V2." *IAES International Journal of Artificial Intelligence (IJ-AI)*, June 2020, www.researchgate.net/publication/341798630_Classification_of_tomato_leaf_diseases_using_MobileNet_v2.

Devvret Verma, et al. "Plant Leaf Disease Detection Using MobileNetV2." *Webology*, 2021, pp. 3241–3246, https://doi.org/10.29121/web/v18i5/60.

Trivedi, Naresh K., et al. "Early Detection and Classification of Tomato Leaf Disease Using High-Performance Deep Neural Network." *Sensors*, vol. 21, no. 23, 30 Nov. 2021, p. 7987, https://doi.org/10.3390/s21237987.

Aravind Krishnaswamy Rangarajan, et al. "Tomato Crop Disease Classification Using Pre-Trained Deep Learning Algorithm." *Procedia Computer Science*, Jan. 2018, www.researchgate.net/publication/326538543_Tomato_crop_disease_classification_using_pre-trained_deep_learning_algorithm.

Wang, Huan, et al. "A Plant Disease Classification Algorithm Based on Attention MobileNet V2." *Algorithms*, vol. 16, no. 9, 1 Sept. 2023, p. 442, https://doi.org/10.3390/a16090442.

Kadhim, H. Abdullah, and K. Ghathwan. "Artificial Neural Network Hyperparameters Optimization: A Survey." *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 18, pp. 59–87, Dec. 2022, DOI: 10.3991/ijoe.V18i15.34399.

H. Shaziya. "A Study of the Optimization Algorithms in Deep Learning." Mar. 2020, DOI: 10.1109/ICISC44355.2019.9036442.

M. C. Mukkamala and M. Hein. "Variants of RMSProp and Adagrad with Logarithmic Regret Bounds." 2017, arXiv: 1706.05507 [cs.LG].

S. Sharma, S. Sharma, and A. Athaiya. "Activation Functions in Neural Networks." *International Journal of Engineering Applied Sciences and Technology*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:225922639.

A. Pal. "Different Activation Functions for Deep Neural Networks You Should Know." *Medium*, 12 Apr. 2021. [Online]. Available: https://medium.com/geekculture/different-activation-functions-for-deep-neural-networks-you-should-know-ea5e86f51e84.

A. Kumar. "Hardswish Activation Function." *Medium*, 21 Jan. 2022. [Online]. Available: https://medium.com/@akp83540/hardswish-activation-function-62fb36870bf4.

"Hard-Swish." *Papers with Code*, 2023. [Online]. Available: https://paperswithcode.com/method/hard-swish.