JASTEC

JOURNAL OF APPLIED SCIENCE, TECHNOLOGY AND COMPUTING
e-ISSN: 3036-0250

Vol. 3 No. 1 (2026) 1-11
https://publisher.uthm.edu.my/ojs/index.php/ijie

Lightweight Deep Learning for Plant Disease Detection: A Review of MobileNet Architectures and TensorFlow.js Deployment

Mohammed Fuad Mohammed Ahmed Saif[1]*, Abdikarin Osman Mohamed[2], Rusma Anieza Ruslan[2]

[1] *Studygate Sdn. Bhd, Megan Avenue, 2, Jalan Yap Kwan Seng, Wilayah Persekutuan, 50450 Kuala Lumpur, MALAYSIA*
[2] *Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA*

*Corresponding Author: sayff.contact@gmail.com
DOI: https://doi.org/10.30880/jastec.2026.03.01.001

Article Info

Received: 4 April 2026
Accepted: 19 June 2026
Available online: 30 June 2026

Keywords

Plant disease detection, MobileNet, TensorFlow.js, deep learning, edge computing, agriculture

Abstract

The rapid advancement of deep learning has significantly enhanced automated plant disease detection in agriculture. The advancement enables improved crop monitoring and food security management. However, despite high performance in controlled environments, many deep learning models remain difficult to deploy in real agricultural settings due to high computational cost, limited infrastructure, and poor generalization. This paper presents a systematic review of MobileNet-based architectures for plant disease detection, with a focus on their deployment in browser- and mobile-based applications using TensorFlow.js. MobileNet is a lightweight convolutional neural network designed for efficient inference on resource-constrained devices, while TensorFlow.js enables real-time, on-device machine learning without reliance on cloud computing. The review analyzes MobileNet variants (V1, V2, and V3) in terms of accuracy, efficiency, and suitability for deployment. Comparative findings indicate that MobileNetV3 achieves the best balance between performance and computational cost, making it suitable for real-time agricultural applications. However, limitations such as dataset bias, lack of field validation, and efficiency–accuracy trade-offs remain significant challenges. This study provides a deployment-oriented synthesis and identifies key research gaps to support the development of scalable and practical AI-based plant disease detection systems.

1. Introduction

The rapid digital transformation of agriculture, coupled with the intensifying effects of climate change and rising global food demand, has significantly heightened the need for efficient, intelligent plant disease detection systems [1], [2]. Plant diseases remain a major threat to global food security, contributing to substantial yield losses and economic instability across agricultural sectors [3], [4]. Conventional diagnostic approaches, which rely on manual inspection and laboratory-based analysis, are increasingly inadequate due to their time-consuming

This is an open access article under the CC BY-NC-SA 4.0 license.

nature, dependency on expert knowledge, and susceptibility to human error [3], [4]. These limitations often lead to delayed detection and intervention, allowing diseases to spread rapidly across crops and ultimately reducing productivity [2]. Furthermore, traditional approaches lack scalability, making them unsuitable for large-scale or precision agriculture systems. As agriculture shifts toward automation and data-driven decision-making, there is a growing demand for intelligent, real-time, and scalable disease detection systems capable of operating across diverse environmental conditions.

In recent years, deep learning techniques, particularly convolutional neural networks (CNNs), have demonstrated significant success in image-based plant disease detection tasks [7]–[9]. CNN-based models can extract hierarchical and discriminative features from complex visual data, enabling highly accurate classification of plant diseases across multiple crop types. Studies have reported accuracy levels exceeding 95% using architectures such as ResNet, DenseNet, and InceptionV3 under controlled conditions [5], [8], [9]. Additionally, recent review studies highlight the robustness and adaptability of CNN-based approaches across diverse agricultural datasets and environments [6]. However, despite these advancements, a critical gap persists between high model performance in controlled laboratory settings and their practical applicability in real-world agricultural environments. Most high-performing models require substantial computational resources, including GPUs and cloud infrastructure, which are often unavailable in rural or resource-constrained settings [1], [2]. This limitation reduces their usability in field conditions and underscores the need for more efficient, deployable solutions.

To address these challenges, lightweight deep learning architectures such as MobileNet have emerged as viable alternatives, specifically designed for efficient operation on devices with limited computational capacity [10] – [12]. MobileNet models utilize depthwise separable convolutions to significantly reduce parameter count and computational cost while maintaining competitive accuracy. Empirical studies show that MobileNet-based approaches can achieve accuracy above 90% while requiring substantially less memory and processing power compared to conventional CNN architectures [7], [9]. Furthermore, optimized variants and hybrid lightweight models have demonstrated near state-of-the-art performance with minimal resource consumption, making them suitable for edge deployment in agricultural applications [13]. Comparative evaluations also confirm that MobileNet architectures provide lower latency and higher efficiency than heavier models such as ResNet and AlexNet when deployed on embedded devices [13]. These characteristics make MobileNet particularly attractive for real-time plant disease detection in mobile and resource-constrained environments.

In parallel, the emergence of browser-based machine learning frameworks such as TensorFlow.js has further expanded the feasibility of deploying deep learning models directly on edge devices [16], [17]. TensorFlow.js enables models to run within web browsers, supporting real-time inference, cross-platform accessibility, and offline functionality without reliance on cloud infrastructure. This capability is especially beneficial in rural agricultural regions where internet connectivity may be unstable or unavailable. Studies on edge-based AI systems indicate that on-device inference significantly reduces latency. It is also improving accessibility for end users such as farmers [16]. However, despite these advancements, most existing research still focuses primarily on improving model accuracy and architecture design, with limited attention given to deployment constraints such as memory efficiency, inference speed, and real-world usability.

Moreover, current literature lacks holistic evaluation frameworks that integrate model performance, computational efficiency, and deployment feasibility. Many studies rely heavily on controlled datasets such as PlantVillage, which do not adequately reflect real-world agricultural conditions. It can lead to an overestimation of model performance [8]. In addition, emerging research exploring hybrid deep learning models and modern architectures suggests the need for improved generalization in complex environments [5]. These limitations indicate that achieving high accuracy alone is insufficient; instead, practical deployment considerations must be incorporated into model design and evaluation.

Therefore, this paper presents a systematic and deployment-oriented review of MobileNet architectures for plant disease detection, with a particular focus on their integration with TensorFlow.js for edge-based applications. Unlike existing surveys, this review synthesizes findings across multiple studies to critically analyze trade-offs between accuracy, efficiency, and deployability. It further introduces a deployment-centric evaluation perspective. It emphasizes real-world applicability in resource-constrained agricultural environments. By identifying key research gaps, which include dataset bias, limited field validation, and insufficient user-centered design. This study aims to provide a comprehensive foundation for developing robust, scalable, and practical plant disease detection systems for precision agriculture.

The remainder of this paper is structured as follows. Section 1 describes the introduction of the study. Section 2 explains the deep learning, TensorFlow.js and MobileNet. Section 3 reviews the literature. Section 4 provides a synthesis and comparative analysis of existing studies. Section 5 gives the conclusion and future works.

2. Background Study

This section presents the theoretical and technical background of the study. It focuses on three related areas: deep learning for plant disease detection, TensorFlow.js for client-side deployment, and MobileNet as a lightweight convolutional neural network architecture. These areas provide the foundation for developing a plant disease detection system that is accurate, accessible, and suitable for devices with limited computational resources.

2.1 Deep Learning in Plant Disease Detection

Plant diseases can reduce crop quality and yield if they are not identified and treated at an early stage. Conventional diagnosis commonly depends on visual inspection by farmers or agricultural experts. However, manual inspection can be time-consuming, subjective, and difficult to conduct consistently across large farming areas. Disease symptoms may also resemble nutrient deficiencies, pest damage, or environmental stress. Therefore, automated image-based diagnosis has become an important area of precision agriculture.

Deep learning has significantly improved automated plant disease detection. Unlike traditional machine learning methods, which usually require manually designed features such as color, texture, shape, and lesion boundaries, deep learning models can learn relevant visual features directly from training images. Convolutional neural networks (CNNs) are particularly suitable for this task because they can capture spatial patterns and hierarchical features. Early convolutional layers normally identify simple features such as edges, colors, and textures. Deeper layers combine these features to represent more complex characteristics, including lesion shapes, discoloration patterns, spots, and damaged leaf regions.

A typical CNN-based plant disease detection process begins with image acquisition and preprocessing. The images are then resized and normalized. If necessary, the images require augmentation through rotation, flipping, cropping, zooming, or brightness adjustment. These operations increase the diversity of the training data and reduce overfitting. The processed images are passed through the CNN to extract relevant features, and the final classification layer then assigns each image to a healthy or diseased category. Depending on the application, the model may identify the plant species, distinguish several diseases, or estimate the severity of infection.

Deep architectures such as ResNet, DenseNet, and InceptionV3 have achieved strong classification performance on benchmark agricultural datasets. Many studies have reported accuracy above 90–95% when training and testing images were collected under controlled or relatively consistent conditions [9], [25], [28]. ResNet uses residual connections to support the training of deeper networks. DenseNet promotes feature reuse through connections between layers, while InceptionV3 processes visual information at different spatial scales. These characteristics allow the models to learn subtle differences between healthy tissue and disease symptoms.

However, high accuracy on a benchmark dataset does not necessarily indicate reliable performance in real agricultural settings. Public datasets often contain centered leaves, clean backgrounds, balanced lighting, and clearly visible symptoms. Images captured on farms may include complex backgrounds, shadows, uneven illumination, motion blur, overlapping leaves, different camera angles, and symptoms at different stages of development. Variations in plant varieties, climate, season, and image-capture devices can also introduce a distribution shift between training and operational data. Consequently, a model may perform well during laboratory evaluation but experience lower accuracy when used in the field.

Computational requirements present another challenge. Deep architectures generally contain many layers and parameters. Their training requires substantial processing power, memory, and time. Although training is usually performed on a workstation or cloud platform, inference can also be demanding when the trained model is deployed on a smartphone or low-cost device. Previous studies have noted that large models often depend on high-performance graphics processing units and substantial memory resources [12], [19]. This condition limits their practical use in farms, plantations, and rural areas where advanced computing infrastructure may not be available.

Cloud-based inference can reduce the computational workload on the user's device. In this approach, the captured image is uploaded to a remote server for classification. However, this process requires an internet connection and may introduce transmission delays, server dependencies, operating costs, and data privacy concerns. Large image uploads can also consume mobile data. These limitations are particularly relevant in rural locations where network coverage may be slow or unstable.

For these reasons, plant disease detection should not be evaluated solely according to classification accuracy. A practical system must also consider model size, inference time, memory consumption, network dependency, and compatibility with the target device. Lightweight models offer a suitable alternative because they reduce computational cost while retaining useful predictive capability [20], [29]. The main challenge is to identify a model that provides an appropriate balance between accuracy and efficiency under the intended deployment conditions.

2.2 TensorFlow.js for Edge Deployment

TensorFlow.js is an open-source machine learning library that enables the development and execution of machine learning models using JavaScript [16]. It supports model training and inference within web-based environments. It can also load models converted from compatible TensorFlow or Keras formats. This capability allows an existing trained model to be integrated into a web application without requiring the user to install specialized machine learning software.

In a conventional server-based system, a user captures an image and sends it to a remote server. The server processes the image and returns the prediction. In contrast, TensorFlow.js can execute the model on the client device. Once the web application and model files are available, image preprocessing, inference, and result presentation can occur locally. This reduces the need for repeated communication with a remote server and can improve response time, particularly when network connectivity is limited.

Client-side execution also offers potential privacy benefits because the captured leaf image does not need to be transmitted to an external server. This approach is useful when users are concerned about sharing farm data, location information, or images of commercial crops. It may also reduce server workload and operational costs because prediction is performed on the user's device.

TensorFlow.js supports several computational backends. Browser execution may use WebGL to access available graphics acceleration or WebAssembly for efficient processing across supported devices. A CPU-based backend may also be used, although its performance is generally more limited for complex models. The selected backend and its actual performance depend on the browser, operating system, device hardware, and support for acceleration technologies. Therefore, the same model may produce different inference times across devices.

Browser-based deployment is particularly relevant to agricultural applications because a web system can be accessed using common smartphones, tablets, or computers. Farmers do not necessarily need to install a platform-specific application. A responsive web interface can allow the user to capture or upload a leaf image, run the model, and view the predicted disease category. Previous studies have identified browser and edge-based approaches as useful in environments where continuous access to cloud services cannot be guaranteed [22], [29].

Nevertheless, offline operation is not automatic. The required web resources and model files must first be bundled with the application or downloaded and cached using an appropriate mechanism, such as a progressive web application. Once these resources are available locally, the system may make predictions without a continuous network connection. This design distinction is important because a system that only loads its model from a remote server may still fail when the user has no internet access.

TensorFlow.js deployment also introduces several technical constraints. Web browsers impose limitations on memory usage and resource access. Model loading can be slow when model files are large, particularly during the first use. Older or low-cost devices may experience longer inference times, application freezing, or browser termination when memory consumption is excessive. Differences in browser implementations and hardware acceleration can also lead to inconsistent performance [19], [20].

The model must therefore be prepared specifically for client-side deployment. Relevant measures include reducing the input image resolution, limiting model size, optimizing preprocessing operations, and selecting an efficient architecture. Model compression techniques such as quantization may further reduce storage and memory requirements. Although their effect on classification accuracy must be evaluated. Performance should be tested using measures that reflect actual deployment, including model loading time, average inference time, memory consumption, prediction accuracy, and compatibility across target devices.

TensorFlow.js consequently provides more than a method for displaying a machine learning model on a website. It offers a client-side inference environment that can reduce reliance on cloud infrastructure. However, its suitability depends on careful model selection and empirical testing. In this study, TensorFlow.js serves as the deployment platform, while the MobileNet architecture provides a lightweight image-classification model.

2.3 MobileNet Architectures

MobileNet is a family of CNN architectures developed for image-based applications on mobile, embedded, and resource-constrained devices. Unlike conventional CNNs that primarily aim to maximize predictive accuracy, MobileNet architectures explicitly consider computational cost, parameter count, memory use, and inference latency. These characteristics make them suitable for applications that require near-real-time prediction on smartphones or web browsers.

MobileNetV1 introduced depthwise separable convolution as its main efficiency mechanism [10], [14]. A standard convolution performs spatial filtering and channel combination in a single operation. This process becomes computationally expensive as the number of input and output channels and filters increases. Depthwise separable convolution divides the process into two stages. The depthwise convolution applies one spatial filter independently to each input channel. The pointwise convolution then uses a 1×1operation to combine information across channels.

This factorization substantially reduces the number of calculations and parameters compared with a standard convolution. MobileNetV1 also provides width and resolution multipliers that allow the model to be adjusted according to the available resources. A lower width multiplier reduces the number of channels, while a lower resolution multiplier reduces the input image size. Both options decrease computational demand, although aggressive reduction may also lower classification accuracy.

MobileNetV2 extends this design through inverted residual structures and linear bottlenecks [11], [32]. In a conventional residual block, a wide feature representation is usually compressed before being expanded again. MobileNetV2 uses the opposite arrangement. A narrow input is first expanded into a higher-dimensional representation. A depthwise convolution then processes the expanded features.

The inverted residual connection links the narrow input and output representations when their dimensions are compatible. This structure supports efficient information flow while limiting the computational cost of the spatial convolution. The linear bottleneck removes a nonlinear activation from the narrow projection layer because excessive nonlinearity in a low-dimensional space can cause useful information to be lost. Together, these mechanisms generally provide better feature representation than MobileNetV1 while maintaining a relatively small computational footprint.

MobileNetV3 further improves the balance between accuracy and latency through hardware-aware neural architecture search and network optimization [12], [15], [27]. Instead of manually selecting all architectural components, the design process evaluates candidate structures based on both predictive performance and efficiency on the target hardware. MobileNetV3 also incorporates efficient attention mechanisms, modified bottleneck blocks, and computationally economical activation functions. It is commonly available in large and small configurations to support different resource and accuracy requirements.

The small configuration is intended for highly constrained environments, while the large configuration generally provides stronger predictive performance at a higher computational cost. The most suitable variant, therefore, depends on the application's requirements. A low-end smartphone may benefit from a smaller model, whereas a modern device may support a larger variant with improved accuracy. The selection should be based on empirical comparison rather than architecture size alone.

MobileNet models are also well-suited to transfer learning. In this approach, a model pretrained on a large image dataset is adapted to a smaller dataset of plant images. The early layers provide general visual features, while the final classification layers are replaced or fine-tuned for the target disease categories. Transfer learning can reduce training time and improve performance when the available agricultural dataset is limited. However, fine-tuning must be performed carefully because plant disease symptoms may differ considerably from the objects represented in the original pretraining dataset.

Previous evaluations have shown that MobileNet can provide a strong balance between classification performance and computational efficiency [13], [34]. Nevertheless, the most accurate MobileNet version is not automatically the best model for every application. A slightly less accurate model may be preferable if it loads faster, uses less memory, and produces more stable predictions on low-cost devices. Conversely, reducing the model excessively may remove visual detail needed to distinguish diseases with similar symptoms.

Accordingly, a meaningful comparison of MobileNetV1, MobileNetV2, and MobileNetV3 should consider several dimensions. These include classification accuracy, precision, recall, F1-score, model size, parameter count, inference time, and browser compatibility. Testing should also include images captured under realistic conditions whenever possible. This evaluation will identify the architecture that offers the most appropriate trade-off for browser-based plant disease detection using TensorFlow.js.

3. Literature Study

This section presents a literature review on MobileNets variants.

3.1 MobileNet for Plant Disease Detection

Extensive research has explored the application of MobileNet architectures to plant disease detection. Studies consistently demonstrate that MobileNet models achieve competitive classification accuracy while significantly reducing computational complexity compared to deeper CNN architectures. For instance, MobileNetV2 has been reported to achieve accuracy levels exceeding 90% on the widely used PlantVillage dataset, making it suitable for practical agricultural deployment [11], [20], [34]. Similarly, MobileNetV3 improves upon previous versions by integrating optimized architectural components, resulting in higher classification accuracy and faster inference in certain scenarios [12], [27]. Comparative studies further confirm that lightweight architectures can achieve near state-of-the-art performance while maintaining significantly lower computational requirements, particularly when deployed in mobile or edge environments [16], [18], [21].

3.2 Comparison with Heavyweight Models

MobileNet architectures are frequently benchmarked against deeper CNN models such as ResNet, DenseNet, and InceptionV3. While these models typically achieve slightly higher accuracy due to their increased depth and representational capacity, they require substantially more computational resources, memory, and energy consumption [9], [25], [28]. This makes them less suitable for real-time or mobile deployment in agricultural environments.

Table 1 *Comparison of deep learning models for plant disease detection*

Model	Accuracy	Parameters	Deployment Suitability
ResNet50	~95%	High	Cloud-based
DenseNet121	~96%	High	Cloud-based
InceptionV3	~94%	High	Limited Mobile
MobileNetV2	~91%	Low	Mobile/Edge
MobileNetV3	~94%	Low	Mobile/Web

Table 1 summarizes the deep learning models for plant disease detection. Although heavyweight models slightly outperform MobileNet in accuracy, their deployment limitations significantly reduce their practicality in field-based agricultural applications. Consequently, MobileNet is often preferred for real-world implementation scenarios where computational efficiency is a priority.

3.3 TensorFlow.js-Based Deployment

Recent advancements in edge AI have led to increasing adoption of TensorFlow.js for deploying deep learning models in browser-based environments. TensorFlow.js enables machine learning models to run directly within web browsers, eliminating the need for dedicated servers and enabling real-time inference on client devices [16]. Studies have shown that MobileNet models integrated with TensorFlow.js can perform efficient plant disease classification without requiring server-side computation, thereby improving accessibility and responsiveness [22], [24].

This approach offers several important advantages for agricultural deployment. Reducing reliance on cloud infrastructure enables models to run directly on user devices without requiring continuous internet connectivity. This is particularly beneficial in rural or low-resource environments. In addition, it enables real-time inference capability. It enables immediate classification and feedback, critical for timely decision-making in plant disease management. Furthermore, its cross-platform compatibility across mobile and web environments ensures broader accessibility, enabling users to interact with the system seamlessly regardless of device type or operating system.

However, several limitations remain. Browser-based execution is constrained by memory limitations, device performance variability, and inconsistent hardware acceleration support. These factors can affect inference speed and model scalability, particularly for more complex architectures, and must be carefully considered during system design and deployment.

3.4 Performance vs Efficiency Trade-offs

A central theme in the literature is the inherent trade-off between model accuracy and computational efficiency. While deeper CNNs provide higher accuracy, they are often unsuitable for deployment in resource-constrained environments due to their computational overhead. In contrast, lightweight models such as MobileNet prioritize efficiency, enabling faster inference and reduced memory consumption at the cost of a marginal reduction in accuracy. Table 2 presents a performance comparison of MobileNet variants.

Table 2 *Performance comparison of MobileNet variants*

Model	Accuracy (%)	Model Size (MB)	Inference Time (ms)
MobileNetV1	88–90	~16	~200
MobileNetV2	90–92	~14	~180
MobileNetV3	92–94	~18	~220

Among these variants, MobileNetV3 generally offers the best balance between accuracy and efficiency. Though slight increases in inference latency may occur, depending on device constraints. This makes it particularly suitable for real-time agricultural applications requiring both responsiveness and reliability.

4. Comparative Analysis

A synthesis of existing studies on plant disease detection reveals consistent progress in applying deep learning techniques, particularly convolutional neural networks (CNNs) and lightweight architectures such as MobileNet. However, the literature also shows significant variation in dataset usage, model evaluation practices, and deployment considerations. To provide a structured comparison, Table 3 summarizes key studies in plant disease detection, emphasizing model types, datasets, accuracies, and limitations.

4.1 MobileNet-Based Plant Disease Detection Studies

Table 3 shows that most existing studies report high accuracy values, particularly those trained on the PlantVillage dataset. However, a recurring limitation is the reliance on controlled laboratory conditions. It does not reflect real-world agricultural environments. Studies that include field or mobile-captured images generally report lower but more realistic performance. And so, it highlights the importance of dataset diversity and environmental variability.

Table 3 *Summary of Mobile-Net-Based plant disease detection studies*

Study	Model	Dataset	Accuracy (%)	Key Contribution	Limitation
Mohanty et al. [7]	CNN baseline	PlantVillage	99.35	Early deep learning application in plant disease detection	Controlled dataset only
Ferentinos [8]	CNN	PlantVillage	99.53	High-accuracy plant disease classification	No field validation
Too et al. [9]	VGG, ResNet, MobileNet	PlantVillage	90–98	Transfer learning comparison	Limited real-world testing
Picon et al. [34]	MobileNet	Field + mobile images	~92	Mobile-based deployment feasibility	Lighting variability issues
Zhang et al. [31]	CNN variants	Maize dataset	~93	Crop-specific disease detection	Limited generalization
Liu & Wang [30]	Survey	Multiple datasets	N/A	Comprehensive deep learning review	No deployment focus

4.2 Deployment-Oriented Evaluation

While traditional studies focus primarily on accuracy, real-world agricultural applications require additional considerations such as model size, inference speed, and energy efficiency. To address this limitation, Table 4 provides a deployment-oriented comparison of commonly used deep learning models.

Table 4 *Deployment-oriented evaluation of deep learning models*

Model	Accuracy	Model Size	Inference Speed	Energy Efficiency	Edge Suitability
ResNet50	High (~95%)	Very High	Slow	Low	Not suitable
DenseNet121	Very High (~96%)	Very High	Slow	Low	Not suitable
InceptionV3	High (~94%)	High	Medium	Medium	Limited
MobileNetV1	Medium (~89%)	Low	Fast	High	Suitable
MobileNetV2	High (~91%)	Low	Fast	High	Highly suitable
MobileNetV3	Very High (~94%)	Low	Very Fast	Very High	Optimal

Table 4 emphasizes a key observation. Although heavyweight models such as ResNet and DenseNet achieve slightly higher accuracy, their computational cost makes them impractical for deployment in resource-constrained environments. In contrast, MobileNetV2 and MobileNetV3 provide an optimal balance between performance and efficiency. It makes them highly suitable for mobile and edge-based agricultural applications.

4.3 Dataset Bias and Generalization Issues

A critical issue identified across the literature is dataset bias, particularly the over-reliance on the PlantVillage dataset. Table 5 summarizes common datasets used in plant disease detection and their implications on model generalization. As shown in Table 5, models trained on controlled datasets often fail to generalize effectively to real-world agricultural environments. This indicates a significant gap between laboratory performance and field deployment, emphasizing the need for more diverse and realistic datasets in future research.

Table 5 *Common datasets in plant disease detection*

Dataset	Characteristics	Strength	Weakness	Impact on Model
PlantVillage	Controlled lab images	Clean and balanced data	No field variability	Overestimated accuracy
Real field datasets	Natural conditions	Realistic environment	Noisy data	Lower but realistic performance
Mobile capture datasets	Smartphone-based images	Practical deployment relevance	Inconsistent quality	Robustness testing
Synthetic datasets	Augmented images	Large dataset size	Unrealistic distribution	Risk of overfitting

Based on the synthesis presented in Tables 3–5, several important knowledge points emerge from the existing literature. First, there is a noticeable accuracy saturation trend, where many deep learning models already achieve high performance on benchmark datasets, and further improvements beyond architectures such as MobileNetV3 tend to be marginal rather than substantial. Second, the literature consistently highlights a clear efficiency advantage of lightweight architectures. Models like MobileNet significantly reduce computational cost, memory usage, and inference time while maintaining competitive accuracy. Third, a critical deployment gap exists between academic model evaluation and real-world agricultural applications. Here, many studies prioritize controlled experimental performance over practical usability under field conditions. Finally, dataset limitations remain a persistent issue. Most existing works rely heavily on controlled datasets such as PlantVillage. This thereby restricts model generalization and reduces robustness when applied to real-world, variable agricultural environments.

5. Discussion

This study synthesizes existing studies on MobileNet-based plant disease detection, with particular attention to predictive performance, computational efficiency, and practical applications. Overall, MobileNet provides a viable alternative to more sophisticated architectures such as ResNet, DenseNet, and InceptionV3. While these larger models may achieve slightly higher accuracy under controlled experimental conditions, they generally require more processing power, memory, and inference time. These requirements limit their suitability for mobile devices and agricultural environments with limited hardware or unreliable internet connectivity [10], [25], [28], [29].

A key finding is the trade-off between classification accuracy and deployment efficiency. Architectural improvements beyond MobileNetV3 may only provide marginal accuracy gains while significantly increasing computational overhead [12], [27]. In practical applications, small improvements in accuracy may not justify slower inference, higher memory usage, or reduced device compatibility. Therefore, model selection should consider more than accuracy alone. The evaluation should include model size, inference time, resource usage, responsiveness, and compatibility with the intended device. MobileNet remains attractive because it offers a reasonable balance between predictive performance and computational efficiency.

However, the performance of reported plant disease detection models is highly dependent on the dataset used. Many studies rely on benchmark datasets such as PlantVillage, which contain clear leaf images captured under controlled background and illumination. While these datasets are useful for model development and comparison, they do not adequately represent real-world agricultural conditions. Field images may contain uneven illumination, shadows, complex backgrounds, overlapping leaves, occlusions, different camera angles, and symptoms at various stages of development. As a result, models that achieve high accuracy on controlled datasets

may be less effective when tested in real-world environments [7], [23]. This discrepancy between laboratory and field performance remains a major barrier to practical adoption.

This review also identifies gaps between model development and system implementation. Many studies focus on improving classification accuracy but provide limited evaluation of usability, offline operation, device compatibility, and integration into accessible applications. These factors are particularly important since farmers may use low-cost smartphones and have limited technical expertise. TensorFlow.js offers practical deployment options by enabling inference in a web browser and reducing the reliance on persistent cloud services [16], [29]. However, browser-based systems remain vulnerable to device variability, memory limitations, model loading times, and differences in hardware acceleration. Therefore, performance should be tested on representative user devices rather than evaluated solely on development computers.

Future research should prioritize realistic and diverse datasets that capture variations in crops, terrain, lighting, backgrounds, disease stages, and environmental conditions. Such datasets will increase model robustness and reduce the performance gap between controlled experiments and real-world applications [23], [30]. Further work could also investigate improved lightweight or hybrid architectures. MobileNet can be combined with efficient attention mechanisms, EfficientNet components, or dense transformer-based methods. However, any improvements should be evaluated based on their additional computational requirements and not just classification accuracy [23].

Multimodal systems also provide valuable directions for future research. Leaf images can be integrated with weather records, soil conditions, crop information, location data, or Internet of Things sensor measurements. These additional inputs can help differentiate disease symptoms from nutrient deficiencies and environmental stresses, leading to more context-sensitive predictions [12], [33]. Model optimization methods such as pruning, quantization, and hardware-sensitive design should also be explored to reduce model size and inference time on resource-constrained devices. In conclusion, MobileNet is a promising architecture for browser-based plant disease detection because it enables efficient inference without requiring high-performance hardware. However, technical accuracy alone is not sufficient to create practical value.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors are responsible for the study conception, research design, data collection, data analysis, result interpretation and manuscript drafting.

References

[1] Jeger, M., Beresford, R., Bock, C., Brown, N., Fox, A., Newton, A., and Yuen, J. (2021). Global challenges facing plant pathology: multidisciplinary approaches to meet the food security and environmental challenges in the mid-twenty-first century. CABI Agriculture and Bioscience, 2(1), 1-18.

[2] Hussain, S. (2024). Advancing plant health management: challenges, strategies, and implications for global agriculture. International Journal of Agriculture and Sustainable Development, 6(2), 73-89.

[3] Strange, R. N., & Scott, P. R. (2005). Plant disease: A threat to global food security. Annual Review of Phytopathology, 43, 83–116.

[4] Savary, S., Willocquet, L., Pethybridge, S. J., Esker, P., McRoberts, N., & Nelson, A. (2019). The global burden of pathogens and pests on major food crops. Nature Ecology & Evolution, 3(3), 430–439.

[5] Upadhyay, A., Chandel, N. S., Singh, K. P., Chakraborty, S. K., Nandede, B. M., Kumar, M., ... & Elbeltagi, A. (2025). Deep learning and computer vision in plant disease detection: a comprehensive review of techniques, models, and trends in precision agriculture. Artificial Intelligence Review, 58(3), 92.

[6] Abuzar, M. (2024). Advance Plant Disease Detection Using VGGNet With Convolutional Neural Network (Doctoral dissertation, Sant Gadge Baba Amravati University, Amravati).

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

[8] Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. Computers and Electronics in Agriculture, 145, 311–318.

[9] Too, E. C., Yujian, L., Njuki, S., & Yingchun, L. (2019). A comparative study of fine-tuning deep learning models for plant disease identification. Computers and Electronics in Agriculture, 161, 272–279.

[10] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., … Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[11] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 4510–4520).

[12] Yan, J. (2023). Study for performance of mobilenetv1 and mobilenetv2 based on breast cancer. *arXiv preprint arXiv:2308.03076*.

[13] Srinivasagan, R., Mohammed, M., & Alzahrani, A. (2023). TinyML-sensor for shelf life estimation of fresh date fruits. Sensors, 23(16), 7081.

[14] Hafiz, A. M. (2023). A survey on light-weight convolutional neural networks: trends, issues and future scope. *Journal of Mobile Multimedia*, *19*(5), 1277-1297.

[15] Raj, R., Tiwari, P., Gourisaria, M. K., & Das, H. (2023, December). Efficient object detection and labeling in retail environments using MobileNetV2 with inverted residuals. In *2023 OITS International Conference on Information Technology (OCIT)* (pp. 634-639). IEEE.

[16] Google. (2023). TensorFlow.js: Machine learning for the web and beyond. https://www.tensorflow.org/js

[17] Pepe, F., Farkas, C., Nayebi, M., Antoniol, G., & Di Penta, M. (2025, April). How do papers make into machine learning frameworks: A preliminary study on tensorflow. In *2025 IEEE/ACM 33rd International Conference on Program Comprehension (ICPC)* (pp. 1-6). IEEE.

[18] Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., & Adam, H. (2019). Searching for mobilenetv3. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1314-1324).

[19] Schizas, N., Karras, A., Karras, C., & Sioutas, S. (2022). TinyML for ultra-low power AI and large scale IoT deployments: A systematic review. Future Internet, 14(12), 363.

[20] Deng, Y. (2019). Deep learning on mobile devices: a review. Mobile Multimedia/Image Processing, Security, and Applications 2019, 10993, 52-66.

[21] Halidou, A., Mohamadou, Y., Apen, P. T., Von Kallon, D. V., Baraza, W. J., Patrick, M. G., ... & Mardochée, R. Convolutional Neural Network-based Xception, MobileNetV2 and InceptionV3 Models for Plant Disease Identification in Sub-Saharan Africa. International Journal of Computer Applications, 975, 8887.

[22] Iftikhar, M., Kandhro, I. A., Kausar, N., Kehar, A., Uddin, M., & Dandoush, A. (2024). Plant disease management: a fine-tuned enhanced CNN approach with mobile app integration for early detection and classification. Artificial Intelligence Review, 57(7), 167.

[23] Barbedo, J. G. A. (2018). Impact of dataset size and variety on the effectiveness of deep learning for plant disease classification. Computers and Electronics in Agriculture, 153, 46–53.

[24] Challapalli, S. S. N., Mishra, G., Pachauri, Y., Mishra, A., Kumar, S. R., & Kumar, L. (2023, September). Comparing TensorFlow. js and TensorFlow in Python: An Accessibility and Usage Analysis. In *2023 6th International Conference on Contemporary Computing and Informatics (IC3I)* (Vol. 6, pp. 250-254). IEEE.

[25] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770–778).

[26] Kavyashree, P. S., & El-Sharkawy, M. (2021, January). Compressed mobilenet v3: a light weight variant for resource-constrained platforms. In *2021 IEEE 11th annual computing and communication workshop and conference (CCWC)* (pp. 0104-0107). IEEE.

[27] Zhao, L., & Wang, L. (2022). A new lightweight network based on MobileNetV3. *KSII Transactions on Internet & Information Systems*, *16*(1).

[28] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4700–4708).

[29] Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. Computers and Electronics in Agriculture, 147, 70–90.

[30] Liu, J., & Wang, X. (2021). Plant diseases and pests detection based on deep learning: A review. Plant Methods, 17(1), 22.

[31] Zhang, Y., Wang, S., Ji, G., & Phillips, P. (2018). "Maize leaf disease identification using deep convolutional neural networks," *Neural Computing and Applications*, vol. 30, no. 3, pp. 1–10, 2018.

[32] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4510–4520).

[33] Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning (pp. 6105–6114).

[34] S. Picon et al., "Deep convolutional neural networks for mobile capture device-based crop disease classification," Computers and Electronics in Agriculture, vol. 161, pp. 280–290, 2019.