A TinyML Approach for the Classification of Bean Crop Diseases

1st Mir Hassan
Department of Information Engineering and Computer Science
University of Trento, Italy
mir.hassan@unitn.it

2nd Wamiq Raza
Deep Matrix AI, Italy
wamiq.raza@deepmatrixai.com

3rd Varvara Fadeeva
Deep Matrix AI, Italy
varvara.fadeeva@studenti.unitn.it

4th Leonardo Lucio Custode
Department of Information Engineering and Computer Science
University of Trento, Italy
leonardocustode@gmail.com

5th Giovanni Iacca
Department of Information Engineering and Computer Science
University of Trento, Italy
giovanni.iacca@unitn.it

***Abstract*—Modern agriculture has enabled food production for nearly eight billion people, yet plant diseases and climate change continue to threaten food security. Beans, a key nutritional crop worldwide, are vulnerable to diseases such as bean rust and angular leaf spot, which significantly reduce yield. Early detection is critical for effective treatment. While existing methods leverage cloud-based Deep Learning (DL) models for plant disease classification, they often require substantial computational resources. In this work, we propose an edge-based solution by deploying a quantized MobileNetV2 model on a resource-constrained embedded device. We compare the performance of the full-precision (Float 32-bit) model and its quantized (Int 8-bit) counterpart deployed on a microcontroller unit (MCU), targeting Internet of Things (IoT) scenarios. Our study analyzes model accuracy, inference speed, and memory utilization, focusing on peak RAM and flash memory requirements. The results show that quantization significantly reduces memory footprint and inference time while maintaining competitive classification accuracy. These findings highlight the potential of quantization to enable efficient, sustainable, and deployable TinyML models for plant disease detection at the edge.**

***Index Terms*—Agriculture, Classification, Deep Learning, Internet of Things, Microcontrollers, Plant Disease, TinyML.**

Crop disease diagnosis traditionally relies on complex visual inspections of plant leaves. Incorrect or delayed diagnosis can lead to excessive or insufficient pesticide use, raising production costs and causing environmental and health hazards [1]. Precise and timely diagnosis is therefore essential for sustainable farming and to minimize financial losses.

Manual examination is expensive and time-consuming, particularly as the number and dispersion of plants increase [2]. Global warming further exacerbates the issue by introducing new diseases and increasing species diversity, complicating timely diagnosis [3]. Automated plant disease detection offers a promising solution to enhance accuracy, efficiency, and timeliness. Recent studies have shown the effectiveness of Machine Learning (ML), particularly Deep Learning (DL), models for disease classification based on plant images [4]. However, most DL-based approaches rely on centralized, cloud-based systems, requiring costly infrastructure. Internet of Things (IoT) and edge computing technologies offer an alternative by enabling local, energy-efficient inference, reducing communication overhead and latency [5].

Despite these advantages, many existing models are too large for resource-constrained edge devices. Techniques such as quantization [6], pruning [7], and distillation [8] have been developed to compress models for edge deployment. Recent advances in Tiny Machine Learning (TinyML) [9] enable DL inference on ultra-low-power microcontrollers (MCUs) [10]. MCUs operate on coin-sized batteries, process data locally, and eliminate the need for energy-intensive data transmission. However, only limited research has explored TinyML solutions for plant disease classification.

In this paper, we propose a TinyML-based approach for automatic classification of bean leaf diseases, enabling real-time, energy-efficient disease detection directly on MCUs. Building upon prior work in bean disease classification [11], we adapt and optimize lightweight DL architectures for edge deployment. Our solution incorporates model optimization and quantization techniques to meet the strict memory and computation constraints of MCUs, while maintaining competitive classification accuracy.

To the best of our knowledge, this is the first work proposing an effective and deployable TinyML solution for bean crop disease classification. Our findings demonstrate the feasibility of combining lightweight DL models with quantization to enable practical, low-power plant health monitoring at the edge.

To summarize, our research contributions are as follows:

1) In our proposed approach, we achieved higher accuracy for MobileNetV2 to classify the bean leaf diseases by increasing the input image size from 128×128 to 224×224, due to the effectiveness of a higher resolution image for the leaf classification task.

2) Our approach incorporates multiple image augmentation techniques to enhance the diversity and quality of training data, resulting in better model generalization and performance.
3) We experimented with MobileNetV2 and EfficientNetV2 models and conducted a comprehensive grid search to optimize key architecture parameters to boost the model's accuracy and robustness further.
4) We selected the best-performing full precision and quantized it before deploying it on an MCU using TensorFlow Light for Microcontrollers (TFLM). By quantizing the model and leveraging a TinyML approach, such as offline inference and achieving low latency, we ensure efficient on-device processing.

The rest of the paper is organized as follows. In Section I, we discuss the related work. The methods are presented in Section II. We then present the results in Section III, and finally, we provide the conclusions in Section IV.

I. Related Works

Traditional ML and DL techniques have been widely applied to plant leaf disease detection. Early works primarily used CNNs and SVMs but faced challenges related to deployment, scalability, and data limitations.

For rice disease detection, VGG16 [12] showed strong performance but lacked integration of environmental factors. A lightweight CNN [13] was proposed but remained difficult to deploy on hardware-constrained devices. ReliefF-based feature selection for Alfalfa leaves [14] incurred high computational costs. SVM and ResNet34-based classifiers for apple diseases [15] faced limitations due to small datasets and high memory requirements. Sugar beet diseases were classified using SVM and spectral vegetation indices [16], but early-stage specificity and computational overhead were concerns. For grape leaf diseases, transfer learning combining VGG16 and Random Forest [17] achieved good accuracy but increased model complexity. Tomato diseases were addressed using CNNs and CapsNet architectures [18]. However, dataset diversity, computational resource needs, and real-world deployment remained challenges. Wheat disease detection with ResNet50 [19] showed high sensitivity to data quality, while maize crop classification was limited to Northern Corn Leaf Blight. MobileNet and MobileNetV2 [11] were applied to bean leaf disease detection but remained restricted to specific disease types. A semi-automatic CNN pipeline [20] was introduced for multi-dataset disease classification but relied heavily on manual lesion cropping.

Recent works on EfficientNet architectures have addressed various plant diseases. Sahu et al. [21] employed GoogleNet and VGG16 but faced high computational demands. Singh et al. [22] applied EfficientNetB6, encountering deployment variability. EfficientNetV2-based methods [23] introduced preprocessing complexity and showed weaker transfer learning capabilities on binary-class datasets.

Overall, while ML/DL-based approaches achieve high detection accuracy, they often suffer from high computational complexity, memory demands, and limited scalability. Several methods [14], [17], [20] also rely on expert-driven pre- and post-processing steps, reducing their generalizability across different crops and conditions. Moreover, most existing solutions focus on maximizing accuracy without considering deployment on resource-constrained devices. This limitation is critical for practical IoT-based plant disease monitoring. Models like ResNet50 and MobileNetV2 exhibit computational and memory overheads that make their deployment on microcontroller units (MCUs) infeasible.

Recently, *quantization* [6] has emerged as an effective technique to adapt DL models for edge deployment. Reducing numerical precision from 32 to 8 bits significantly improves inference speed, reduces memory footprint, lowers energy consumption, and enables deployment on low-power devices with minimal loss of accuracy.

II. Methods

In the following, we provide detailed insights about the dataset used in our experiments, the models and input enhancements, the model training process, the rationale for choosing the model for quantization, the quantization technique employed to reduce the size of the model while maintaining its performance, and, finally, the MCU deployment platform.

A. Dataset

We use the iBean dataset, collected by the Makerere AI Lab in collaboration with the National Crops Resources Research Institute (NaCRRI) in Uganda [24]. It contains 1296 bean leaf images of size 224x224, captured using cell phones under real field conditions. The dataset is divided into training (80%), validation (10%), and testing (10%) subsets. It includes three classes: Angular Leaf Spot (432 images), Bean Rust (436 images), and Healthy leaves (428 images). Angular Leaf Spot results from a bacterial infection causing water-soaked lesions confined by leaf veins, while Bean Rust is caused by the fungus *Uromyces appendiculatus* affecting aerial parts of the plant. Due to the relatively small dataset size, we applied data augmentation to improve model generalization. While techniques like rotation, shearing, and brightness adjustment showed limited benefits, rescaling, random flipping, and random zooming were found to enhance training performance. These augmentations increased data diversity without altering the essential features of the leaves.

B. Models

We selected two efficient CNN architectures for our study: MobileNetV2 and EfficientNetV2. MobileNetV2 [25] is a lightweight architecture designed for mobile vision applications. It introduces an inverted residual structure with lightweight depthwise convolutions, significantly reducing model size and computational cost while maintaining high classification accuracy. Linear bottlenecks and skip connections further enhance feature propagation and minimize information loss, making MobileNetV2 well-suited for real-time classification on resource-constrained devices.

EfficientNetV2 [26] focuses on optimizing training speed and parameter efficiency. It employs progressive learning, where smaller models are trained first and then scaled up, combined with an improved compound scaling method. These strategies allow EfficientNetV2 to achieve high accuracy with fewer parameters and reduced computational complexity. While both models are efficient and suitable for edge deployment, MobileNetV2 was selected for quantization due to its superior performance during grid search on full-precision models. Its architecture offers a balanced trade-off between accuracy and computational efficiency, which is crucial for minimizing performance degradation during the quantization process.

C. Enhancing Models' Inputs

In our study, we aim to advance the state-of-the-art results in bean disease classification by building on the foundational work that used a MobileNetV2 model [11]. Our methodology began with an attempt at reproducing the results in order to establish a baseline. We then attempted to improve the baseline performance through two key adjustments: increasing the input image resolution to 224x224 pixels for capturing more detailed features, and applying image augmentation techniques to improve model generalization. The results of these experiments are reported in Table I. Furthermore, we extended our experimental setup to incorporate another model, EfficientNetV2, and then performed a comprehensive grid search on the parameters of both MobileNetV2 and EfficientNetV2, to optimize the architectural parameters while also reducing the number of epochs. Finally, we quantized the best-performing full-precision model to evaluate its performance in hardware-constrained situations, with the objective of bridging the gap between ML models and their real-world implementation in IoT scenarios for plant disease classification.

D. Model Training and Hyperparameter Tuning

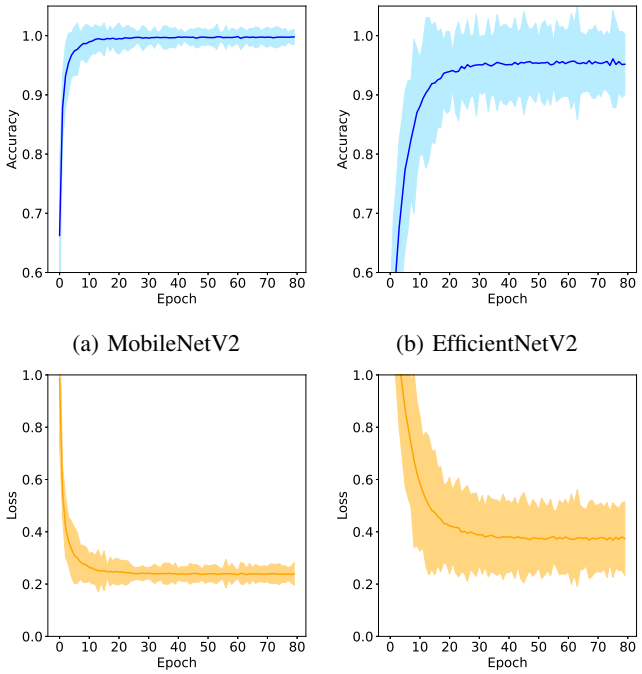
In a preliminary phase, we conducted training and evaluation on the two selected CNN models. In selecting the models for our study, we aimed to strike a balance between two critical factors: classification accuracy and resource efficiency. Achieving high accuracy is essential for reliable classification tasks, while resource efficiency measured in terms of computational demand, memory usage, and power consumption is key for practical deployment on edge devices.

We trained both models by using Stochastic Gradient Descent (SGD) using the same settings as in [11], namely a learning rate of 0.01, L2 regularization with weight 0.001, and a batch size of either 32 or 64 (see Table I). Differently from [11], we used a number of training epochs set to 80 (instead of 100), to reduce the training time. As for the number of dense layers, number of neurons per layer, and dropout rate, these values were deemed to be more impactful on the performance, hence we set them through the grid search shown in Table II.

E. Model Selection for Quantization

In evaluating model performance, we observed a significant difference in classification accuracy between MobileNetV2 and EfficientNetV2, as shown in Table II. MobileNetV2 consistently outperformed EfficientNetV2 across all settings, achieving a training accuracy of 98.81% and a test accuracy of 93.53% with the optimal parametrization (last row of Table II). In contrast, EfficientNetV2 reached a lower training accuracy of 92.67% and a test accuracy of 85.48% under the same configuration.

To further illustrate training dynamics and model robustness, Figure 1 shows the training accuracy and loss curves averaged over 10 runs for both models. Based on these results, MobileNetV2 was selected for quantization due to its superior classification performance and efficiency.



(a) MobileNetV2 (b) EfficientNetV2

(c) MobileNetV2 (d) EfficientNetV2

Fig. 1: Training accuracy (top) and loss (bottom), expressed as mean $\pm$ std. dev. on 10 runs.

F. Quantization

To deploy the selected MobileNetV2 model on a microcontroller unit (MCU), we started from its TensorFlow (TF) implementation and converted it into TensorFlow Lite (TFL), which offers a lightweight format for mobile and embedded applications. The TFL model was then quantized following the process described in [27]. Specifically, each fully connected layer was processed through a feed-forward pass defined as:

$$H_{l+1} = Q[f(W_l H_l + b_l)]$$

where W_l, H_l, and b_l represent the weights, feature maps, and biases of the l-th layer, respectively, and $Q(\cdot)$ denotes the quantization function.

The quantization step reads the full-precision checkpoint of the pre-trained model and generates a quantized version. We applied 32-bit quantization for the final output layer

and 8-bit quantization for all other layers, including weights, biases, and activations. During inference, the input data were treated as Int8, while the output remained in Float32 format, preserving prediction precision while reducing memory and computational requirements.

G. Deployment

After quantization, the quantization model was converted into a program that includes the proper header files to retain its weights, bias, and hyperparameters. The program also encapsulates special requirements, such as the deployment's configuration files. The quantized model was deployed on an MCU supporting the TFLM framework. In our experiments, we employed a Sony Spresense MCU as our TinyML platform. Sony Spresense is equipped with a 6-core Cortex-M4F @156 MHz processor, 1.5MB SRAM, and 8MB Flash.

III. Results and Discussion

This study evaluated the deployment of DL models on edge devices for plant leaf disease classification. Achieving effective deployment on microcontrollers (MCUs) requires balancing model complexity with accuracy under hardware constraints. We first attempted to replicate the results from [11] using a fixed parameter MobileNetV2. Initial results showed inconsistencies, prompting us to increase the input resolution from 128x128 to 224x224 pixels, which significantly improved performance. We further applied data augmentation (rescaling, flipping, and zooming), leading to better generalization. Results in Table I confirm consistent accuracy improvements across training, validation, and test sets for both batch sizes (32 and 64), with each configuration averaged over 10 training runs. The best setup augmented 224x224 input and batch size 32 was selected for further experiments.

Using this configuration, we performed a grid search over dense layer counts, neuron sizes, and dropout rates for both MobileNetV2 and EfficientNetV2 (full-precision). Each setup was trained 10 times with different seeds. Table II shows that simpler architectures outperformed complex ones. For instance, one dense layer with 128 neurons and 0.2 dropout yielded the highest test accuracy (93.53% for MobileNetV2), outperforming even deeper configurations. These findings highlight the sensitivity of DL models to architectural choices, emphasizing the need for careful hyperparameter tuning. We selected the best-performing setup (MobileNetV2 with 128 neurons and 0.2 dropout) for subsequent quantization. Finally, we compared our results with existing works (Table III). While [11] reported 92.00% with MobileNetV2 and [22] achieved 91.74% with EfficientNetB6, deeper models like VGG16 reached 93.75% [21]. Although VGG16 offers slightly better accuracy, its large size and high computational demands make it impractical for edge deployment. In contrast, MobileNetV2 provides a strong trade-off between accuracy and efficiency, making it a more suitable choice for resource-constrained environments.

TABLE I: Effect of data augmentation for different batch sizes on the full-precision (Float 32-bit) MobileNetV2 model (mean ± std. dev. on 10 runs).

	Batch size	Without Augmentation		With Augmentation	
		128x128	224x224	128x128	224x224
Training	32	97.20	100.00	98.13	98.81
Validation	32	95.83	96.99	97.34	98.50
Test	32	89.93	90.53	90.81	**93.53**
Training	64	96.90	100.00	97.68	98.55
Validation	64	94.74	94.74	95.49	95.49
Test	64	89.41	90.15	90.37	91.66

TABLE II: Model settings and training/test accuracy (%) for the full-precision (Float 32-bit) MobileNetV2 and EfficientNetV2 (mean ± std. dev. on 10 runs).

Dense layers	Neurons	Dropout rate	MobileNetV2		EfficientNetV2	
			Training	Test	Training	Test
2	512	0.5	92.99 ± 0.04	88.59 ± 0.01	73.02 ± 0.01	70.23 ± 0.01
2	512	0.3	95.42 ± 0.04	91.21 ± 0.02	86.40 ± 0.01	82.89 ± 0.01
2	512	0.2	97.49 ± 0.05	90.82 ± 0.01	88.24 ± 0.01	82.25 ± 0.01
2	256	0.5	95.93 ± 0.05	88.96 ± 0.02	86.94 ± 0.01	80.27 ± 0.01
2	256	0.3	96.51 ± 0.05	91.02 ± 0.04	80.16 ± 0.01	77.65 ± 0.01
2	256	0.2	95.80 ± 0.05	88.67 ± 0.01	85.73 ± 0.01	79.80 ± 0.01
2	128	0.5	94.36 ± 0.05	89.20 ± 0.05	85.12 ± 0.01	80.18 ± 0.01
2	128	0.3	96.21 ± 0.05	91.41 ± 0.07	82.27 ± 0.01	82.36 ± 0.01
2	128	0.2	97.43 ± 0.05	91.86 ± 0.02	83.67 ± 0.01	82.84 ± 0.01
1	512	0.5	94.58 ± 0.05	89.07 ± 0.02	85.12 ± 0.01	80.06 ± 0.01
1	512	0.3	96.27 ± 0.04	90.08 ± 0.01	87.23 ± 0.01	81.07 ± 0.01
1	512	0.2	97.13 ± 0.04	89.53 ± 0.02	88.42 ± 0.01	80.58 ± 0.01
1	256	0.5	92.58 ± 0.04	90.47 ± 0.02	83.41 ± 0.01	81.42 ± 0.01
1	256	0.3	96.01 ± 0.04	90.94 ± 0.01	87.05 ± 0.01	82.89 ± 0.01
1	256	0.2	97.05 ± 0.04	91.25 ± 0.01	88.20 ± 0.01	82.13 ± 0.01
1	128	0.5	92.06 ± 0.04	90.55 ± 0.01	81.86 ± 0.01	81.50 ± 0.01
1	128	0.3	95.38 ± 0.04	89.30 ± 0.01	86.43 ± 0.01	80.37 ± 0.01
1	128	0.2	**98.81 ± 0.04**	**93.53 ± 0.02**	92.67 ± 0.02	85.48 ± 0.01

TABLE III: Comparison of the full-precision and quantized models considered in this paper against models from the literature (results taken from original papers).

Ref.	Model	Training Acc. (%)	Test Acc. (%)	No. Parameters (approximated)
[11]	MobileNetV2	97.00	92.00	2.26 million
[22]	EfficientNetB6	96.62	91.74	5.92 million
[21]	GoogleNet	n/a	95.31	5.97 million
[21]	VGG16	n/a	**93.75**	138.00 million
Ours	MobileNetV2 (full-precision)	**98.81**	93.53	2.26 million
Ours	MobileNetV2 (quantized)	93.75	88.72	2.26 million

A. Full-precision vs. Quantized MobileNetV2 Models

The full-precision (Float 32-bit) MobileNetV2 model selected in Section II-E served as the baseline for exploring model quantization. The confusion matrix for the full-precision model is shown in Figure 2 (left). The model achieved class-wise accuracies of 93.1% (Angular Leaf Spot), 90.5% (Bean Rust), and 96.2% (Healthy). The overall test accuracy was 93.53%. The model demonstrated an inference time of 243 ms, requiring 947.8 kB of RAM and 1.6 MB of flash memory, as measured using Edge Impulse. To further reduce resource requirements, we quantized the model. The confusion matrix for the quantized version is reported in Figure 2 (right). It achieved class-wise accuracies of 88.21%, 86.49%, and 91.45% for Angular Leaf Spot, Bean Rust, and Healthy leaves, respectively. The overall accuracy was 88.72%, which remains acceptable for practical deployment. Quantization significantly

improved resource efficiency. The inference time was reduced to 109 ms, RAM usage dropped to 333.8 kB, and flash memory consumption to 569.6 kB, making the model suitable for resource-constrained embedded systems.

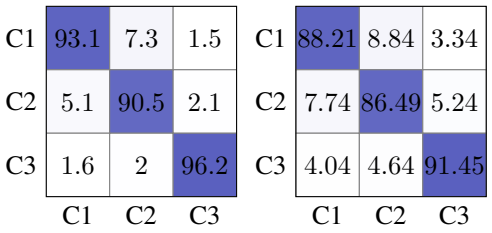


Fig. 2: Confusion matrices (over 10 runs) for MobileNetV2. x-axis: Predicted; y-axis: Actual. Left: Float 32-bit model. Right: Int 8-bit model derived from the Float 32-bit model. C1: Angular Leaf Spot; C2: Bean Rust; C3: Healthy.

IV. Conclusions

In this study, we proposed a TinyML solution for the automatic classification of bean leaf diseases and optimized it for real-time execution on edge devices through model quantization. The resulting quantized (Int8) model achieved competitive accuracy while significantly reducing memory consumption and inference time. Our results demonstrate the feasibility of deploying DL models under strict memory, energy, and computational constraints. The quantized model exhibited notable improvements in inference speed and memory efficiency, highlighting its suitability for hardware-constrained environments. Future work could explore the integration of supplementary sensors and multimodal data to further enhance disease detection performance. Additionally, Neural Architecture Search (NAS) may help identify architectures specifically tailored for resource-limited leaf disease classification tasks.

References

[1] L. Rani, K. Thapa, N. Kanojia, N. Sharma, S. Singh, A. S. Grewal, A. Srivastav, and J. Kaushal, "An extensive review on the consequences of chemical pesticides on human health and environment," *Journal of Cleaner Production*, p. 124657, 2020.

[2] Ghosh, Shyamasree and Dasgupta, Rathi, *Machine Learning in Plant Disease Research*. Singapore: Springer Nature, 2022, pp. 299–311.

[3] Sladojevic, Srdjan and Arsenovic, Marko and Anderla, Andras and Culibrk, Dubravko and Stefanovic, Darko and others, "Deep neural networks based recognition of plant diseases by leaf image classification," *Computational intelligence and neuroscience*, vol. 2016, 2016.

[4] A. Ahmad, D. Saraswat, and A. El Gamal, "A survey on using deep learning techniques for plant disease diagnosis and recommendations for development of appropriate tools," *Smart Agricultural Technology*, vol. 3, p. 100083, 2023.

[5] M. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Computing Surveys*, vol. 54, no. 8, pp. 1–37, 2021.

[6] Gholami, Amir and Kim, Sehoon and Dong, Zhen and Yao, Zhewei and Mahoney, Michael W and Keutzer, Kurt, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*. Chapman and Hall/CRC, 2022, pp. 291–326.

[7] Vadera, Sunil and Ameen, Salem, "Methods for pruning deep neural networks," *IEEE Access*, vol. 10, pp. 63 280–63 300, 2022.

[8] Larasati, Harashta Tatimma and Prihatno, Aji Teguh and Kim, Howon and others, "A Review of Dataset Distillation for Deep Learning," in *International Conference on Platform Technology and Service*. IEEE, 2022, pp. 34–37.

[9] Banbury, Colby R and Reddi, Vijay Janapa and Lam, Max and Fu, William and Fazel, Amin and Holleman, Jeremy and Huang, Xinyuan and Hurtado, Robert and Kanter, David and Lokhmotov, Anton and others, "Benchmarking tinyml systems: Challenges and direction," *arXiv preprint arXiv:2003.04821*, 2020.

[10] Osman, Anas and Abid, Usman and Gemma, Luca and Perotto, Matteo and Brunelli, Davide, "Tinyml platforms benchmarking," in *International Conference on Applications in Electronics Pervading Industry, Environment and Society*. Springer, 2021, pp. 139–148.

[11] Elfatimi, Elhoucine and Eryigit, Recep and Elfatimi, Lahcen, "Beans Leaf Diseases Classification Using MobileNet Models," *IEEE Access*, vol. 10, pp. 9471–9482, 2022.

[12] C. R. Rahman, P. S. Arko, M. E. Ali, M. A. I. Khan, S. H. Apon, F. Nowrin, and A. Wasif, "Identification and recognition of rice diseases and pests using convolutional neural networks," *Biosystems Engineering*, vol. 194, pp. 112–120, 2020.

[13] Lu, Yang and Yi, Shujuan and Zeng, Nianyin and Liu, Yurong and Zhang, Yong, "Identification of rice diseases using deep convolutional neural networks," *Neurocomputing*, vol. 267, pp. 378–384, 2017.

[14] Qin, Feng and Liu, Dongxia and Sun, Bingda and Ruan, Liu and Ma, Zhanhong and Wang, Haiguang, "Identification of alfalfa leaf diseases using image recognition technology," *PloS ONE*, vol. 11, no. 12, p. e0168274, 2016.

[15] Chuanlei, Zhang and Shanwen, Zhang and Jucheng, Yang and Yancui, Shi and Jia, Chen, "Apple leaf disease identification using genetic algorithm and correlation based feature selection method," *International Journal of Agricultural and Biological Engineering*, vol. 10, no. 2, pp. 74–83, 2017.

[16] Rumpf, T and Mahlein, A-K and Steiner, U and Oerke, E-C and Dehne, H-W and Plümer, Lutz, "Early detection and classification of plant diseases with support vector machines based on hyperspectral reflectance," *Computers and electronics in agriculture*, vol. 74, no. 1, pp. 91–99, 2010.

[17] Aravind, KR and Raja, P and Aniirudh, R and Mukesh, KV and Ashiwin, R and Vikas, G, "Grape crop disease classification using transfer learning approach," in *International Conference on IoT, Social, Mobile, Analytics and Cloud in Computational Vision and Bio-Engineering*. Springer, 2019, pp. 1623–1633.

[18] Shijie, Jia and Peiyi, Jia and Siping, Hu and others, "Automatic detection of tomato diseases and pests based on leaf images," in *Chinese Automation Congress*. IEEE, 2017, pp. 2537–2510.

[19] Picon, Artzai and Alvarez-Gila, Aitor and Seitz, Maximiliam and Ortiz-Barredo, Amaia and Echazarra, Jone and Johannes, Alexander, "Deep convolutional neural networks for mobile capture device-based crop disease classification in the wild," *Computers and Electronics in Agriculture*, vol. 161, pp. 280–290, 2019.

[20] L. C. Ngugi, M. Abdelwahab, and M. Abo-Zahhad, "A new approach to learning and recognizing leaf diseases from individual lesions using convolutional neural networks," *Information Processing in Agriculture*, vol. 10, no. 1, pp. 11–27, 2023.

[21] Sahu, Priyanka and Chug, Anuradha and Singh, Amit Prakash and Singh, Dinesh and Singh, Ravinder Pal, "Deep learning models for beans crop diseases: Classification and visualization techniques," *International Journal of Modern Agriculture*, vol. 10, no. 1, pp. 796–812, 2021.

[22] Singh, Vimal and Chug, Anuradha and Singh, Amit Prakash, "Classification of Beans Leaf Diseases using Fine Tuned CNN Model," *Procedia Computer Science*, vol. 218, pp. 348–356, 2023.

[23] C. K., Sunil and C. D., Jaidhar and Patil, Nagamma, "Cardamom Plant Disease Detection Approach Using EfficientNetV2," *IEEE Access*, vol. 10, pp. 789–804, 2022.

[24] AIR Lab Makerere University, "iBean: A Dataset for Bean Disease Classification," 2020.

[25] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[26] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 096–10 106.

[27] Krishnamoorthi, Raghuraman, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, 2018.