Deep Learning-Driven Real–Time Recognition of Plant Diseases via CNN and Flask Integration

P. Malathi[1,*], T. Pranavadit[2], R. V. Vishal[3], M. A. Bala Kumar[4], V. Sanjai[5], Bhopendra Singh[6]

[1,2,3,4,5]Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram, Chennai, Tamil Nadu, India.
[6]Department of Engineering, Amity University Dubai, Dubai, 345019, United Arab Emirates.
malathip@srmist.edu.in[1], pt2094@srmist.edu.in[2], vr8213@srmist.edu.in[3], bm7055@srmist.edu.in[4], sv7924@srmist.edu.in[5], bhopendrasingh@yahoo.com[6]

*Corresponding author

Abstract: Crop diseases are a problem for agricultural output because they cause a variety of challenges, which frequently lead to financial losses and food poverty. It is essential to identify plant diseases and classify them accurately in the early stages of infection to execute interventions promptly. This study presents a system driven by artificial intelligence (AI) and employing deep learning techniques for the real-time identification and forecasting of crop diseases. The model analyses visual symptoms from photos of crop leaves to determine the types of diseases present. It does this with a high degree of accuracy by utilising convolutional neural networks (CNNs) under the TensorFlow framework. To ensure the system is robust enough to be used across a wide range of crop types and disease categories, it is trained on a heterogeneous dataset. The objective of this research is to assist farmers and agricultural professionals in identifying diseases more efficiently by automating the process. This will enable more informed decision-making, which in turn will improve crop health management, encourage sustainable farming techniques, and enhance food security.

Keywords: Plant Disease Detection; Convolutional Neural Networks (CNN); Deep Learning; Image Classification; Smart Agriculture; Real-Time Inference; Web-Based Deployment; Flask Framework.

Cite as: P. Malathi, T. Pranavadit, R. V. Vishal, M. A. B. Kumar, V. Sanjai, and B. Singh, "Deep Learning-Driven Real–Time Recognition of Plant Diseases via CNN and Flask Integration," *AVE Trends in Intelligent Health Letters*, vol. 2, no. 1, pp. 52 –61, 2025.

Journal Homepage: https://avepubs.com/user/journals/details/ATIHL

Received on: 01/08/2024, **Revised on:** 18/11/2024, **Accepted on:** 25/12/2024, **Published on:** 03/03/2025

DOI: https://doi.org/10.64091/ATIHL.2025.000120

1. Introduction

Ensuring food security depends on agricultural output, particularly in light of the growing global population and changing environmental conditions. Still a significant obstacle, crop diseases cause substantial economic losses and lower yields. Manual examination is labour-intensive, subjective, and prone to errors; conventional disease detection techniques usually include. As artificial intelligence—especially deep learning advances—there is a transformative opportunity to utilise automated, accurate, and scalable solutions to transform plant and crop pathology into a viable source [1]; [3]. Image-based classification problems have yielded impressive results for deep learning models, such as convolutional neural networks (CNNs). Their perfect fit for identifying subtle and varied indications of plant diseases is their capacity to acquire sophisticated visual patterns. This work presents a comprehensive deep learning pipeline for accurately identifying and assessing various crop diseases using the open-source machine learning platform TensorFlow. To ensure consistent performance across all environmental conditions, the proposed system integrates high-resolution leaf photography, deep neural network designs, and a robust training and validation mechanism.

Unlike conventional methods, this approach reduces human bias by automatically extracting pertinent representations, rather than relying on handcrafted characteristics, thereby improving model generalisation and prediction accuracy [6]. Moreover, this study aligns with the principles of precision agriculture, as it provides farmers with timely information, enables them to make informed decisions, and optimises the use of agrochemicals. Minimising yield losses and enhancing crop health monitoring can significantly support sustainable agriculture methods and food resilience plans through this framework and proposed architecture. This work presents the entire architecture, dataset features, pre-processing techniques, training methods, and assessment criteria used to create the model. Furthermore, the effect of several hyperparameters and augmentation strategies on system performance highlights their potential use as a deployable tool for real-time illness diagnosis in smart farming environments [13].

2. Related Work

The integration of artificial intelligence into agricultural systems has recently garnered significant attention in research. Image-based recognition algorithms are progressively augmenting and, in some cases, replacing conventional plant disease diagnostic techniques based on manual skill and laboratory testing. Among the elementary machine learning methods used in early research were K-Nearest Neighbours (KNN) and Support Vector Machines (SVM). Manually generated features, such as those derived from colour histograms, texture descriptors, or shape information, were developed using these methods. Although these techniques have great potential, their reliance on predefined traits often limits their adaptability to rare diseases or foreign conditions, thereby hindering their development [8].

As deep learning methods developed—especially with the publication of massive image recognition datasets like ImageNet—researchers began to explore convolutional neural networks (CNNs) for differentiating crop diseases. By proving remarkable accuracy on leaf image data, Mohanty et al. [1] pioneered one of the first significant initiatives to apply CNNs on the PlantVillage dataset. Also, Sladojevic et al. [2] showed a similar process using manually annotated leaf datasets. Still, real-world variables such as illumination, background noise, and leaf occlusions routinely hampered their models. Later studies incorporated transfer learning to address data limitations and enhance generalisation. Using restricted training data has helped pre-trained networks, including VGGNet, ResNet, and Inception, adapt to plant disease challenges, thereby enhancing their ability to adapt.

Although they significantly enhanced recognition performance, these models were computationally costly and often unsuitable for application on peripheral sensors in agricultural settings [9]. Recently, lightweight and efficient architectures for mobile diagnostics, such as MobileNet and EfficientNet, have attracted much interest [5]. Many studies have demonstrated the potential of mobile applications for disease detection; however, these applications often lack real-time inference capabilities and provide only limited analysis beyond categorisation. Some studies aimed to identify areas of model focus using artificial intelligence techniques, whose explanations could enhance user confidence. These techniques are still in their early stages of development, so they usually lack resistance in the presence of several plant species and disease types [10].

Among the further intriguing developments are hybrid CNN-RNN models for analysing temporal disease progression, generative adversarial networks (GANs) to enhance training datasets, and integration with IoT-based sensor data for comprehensive crop health monitoring [11]. Environmental noise resistance, scalability, and cross-crop adaptation remain difficult even with current developments [14]. Thus, building on these ideas, the present work proposes a deep learning model based on TensorFlow, specifically designed for high-accuracy disease detection in real-world environments [7]. Unlike earlier works, this one prioritises realistic deployment feasibility, adaptive training pipelines, and the efficiency of precision agriculture models.

3. Methodology

A detailed description of the methodological process involves developing a system of real-time classification of plant disease models using deep learning. The core modules include data enumeration, pre-processing data, model construction, training, and implementation. In the process, a carefully curated dataset with thousands of leaf images has been sourced, portraying 28 different classes: healthy and disease-labelled plant conditions. Extensive preprocessing has been done, after which a custom-designed Convolutional Neural Network (CNN) was implemented for automated feature extraction and multi-class classification. It includes a sequence of convolutional and max pooling layers, followed by dense layers, and concludes with a

Softmax output layer to achieve probabilistic predictions. The model has been trained and evaluated using the Adam optimiser and a nominal cross-entropy loss function, along with dropout and early stopping strategies to prevent overfitting.

3.1. Data Collection and Preprocessing

The utilised data set is the PlantVillage data set, which is a large, well-known data set in plant disease and computer vision containing over 50,000 colored images of healthy and diseased leaves of multiple crops (tomato, potato, apple, corn, grape, and others). Each image is labelled by plant type and then further labelled by the specific disease. In total, there are 38 types. The diseases include apple scab, late blight, bacterial spot, powdery mildew, and healthy types for every plant. Due to this reason, the raw images in the dataset vary in size, format, and lighting conditions. Preprocessing the images to the same format and improving model learning is necessary. The images were resized to 224×224 pixels, a standard size for convolutional neural networks, particularly when the model will be used for mobile or lightweight applications (Figure 1). The images were also normalised from the resized images, ensuring that all pixel values fell within the range of 0 to 1. This normalisation allowed all features to contribute equally to learning, thereby overcoming scale and gradient issues during training. Normalisation was performed according to the following formula:

$$X_{\text{norm}} = \frac{X}{255.0}$$

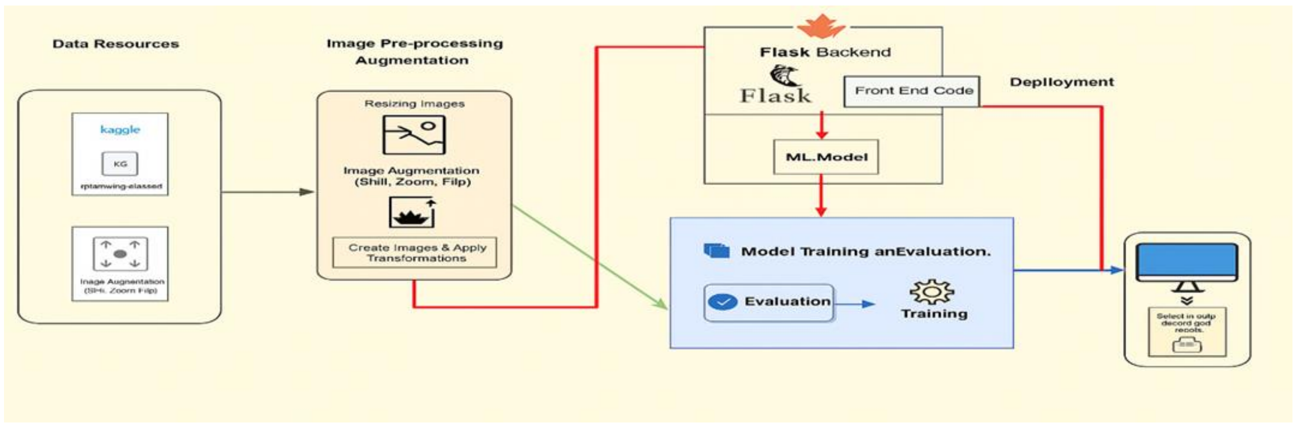
Data Resources
kaggle
KG
rptamwing-elassed
Inage Augnentation (SHi. Zoom Filp)
Image Pre-processing Augmentation
Resizing Images
Image Augmentation (Shill, Zoom, Filp)
Create Images & Apply Transformations
Flask Backend
Flask
Front End Code
ML.Model
Depiloyment
Model Training anEvaluation.
Evaluation
Training
Select in oulp decord god recols.

Figure 1: Architecture diagram

Here, X is the original pixel intensity, and Xnorm is the rescaled value. Images were resized into 4D tensors of shape (1, 224, 224, 3), which is needed for input into the TensorFlow/Keras model to make real-time predictions with the batch size included.. Lastly, data augmentation is also applied during training to enhance the model's generalisation ability. That is, horizontally/vertically flipping the image, random rotations, zooming in and out, and adjusting the brightness. This is a realistic way to simulate real-world conditions, allowing your model to adapt to various environments and conditions when capturing images. The model may overfit patterns from the training data without augmentation, which would reduce its robustness when tested in the real world. Lastly, labels are encoded into a one-hot form, which converts categorical labels into binary vectors. This encoding is also crucial for the sequential cross-entropy loss function, which serves as the normative objective function for multi-class classification. This comprehensive pre-processing pipeline ensures that the dataset is consistent, balanced, and optimised for neural network training.

3.2. CNN Model Architecture

The proposed system is a Deep Learning Model based on a Convolutional Neural Network, which is selected due to its promising capabilities in image classification compared to other forms. CNN works particularly well in detecting an image defect pattern in work because it builds up spatial hierarchies using layers that represent convolution. Compared to traditional methods of data mining and feature extraction, an earlier requirement in machine learning, CNN is unique in adapting its learning toward the features that are significant in training. That makes CNN perfect for large-scale classification problems, such as plant and crop disease detection and analysis. Thus, the proposed model's structure follows a typical sequential definition. A convolutional layer, composed of 32 filters with a kernel size of 3 × 3, and the output is activated by a ReLU (Rectified Linear Unit) activation function, initiates the system.

The ReLU activation function introduces non-linearity to the model while also mitigating a common dimensionality reduction issue that arises during backpropagation, known as the vanishing gradient problem. Then, a first max pooling operation follows, which applies a 2 × 2 pooling technique to downsample the feature maps, thereby reducing their spatial dimensions. This helps prevent overfitting and amplifies computational convenience due to the reduced dimensions. After this first block, the second convolutional layer has 64 filters again, followed by the same activation and pooling.

As you move deeper into the network, the convolutional filters learn to define higher, more abstract features (from edges and textures in the first layer) through complex shapes and patterns—particularly those associated with the disease—that are interpreted by the subsequent layers. These learned features serve as the basis for the dense layers stacked afterwards, which act as classifiers for the extracted information. Upon feature extraction, the 3D feature map is set into a 1D vector. This 1D vector is utilised in a fully interconnected deep layer with 256 neurons, featuring ReLU activation. This layer integrates the learned features with high-level reasoning towards classification, and the last dense layer with 38 output neurons for the 38 disease categories applies softmax activation for a probability distribution across all classes:

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}, \quad i = 1, 2, \ldots, n$$

Here, Zi represents the logit for each class, and n is the number of output classes to be displayed. The class displayed with the highest probability is selected as the final disease label. With this simple design, the architecture is lightweight and easy to deploy, while providing sufficient depth to capture complex features. Therefore, it represents the required trade-off between accuracy and efficiency in real-time predictions for resource-constrained environments, such as smartphones or rural deployments, which is crucial for live prediction (Figure 2).

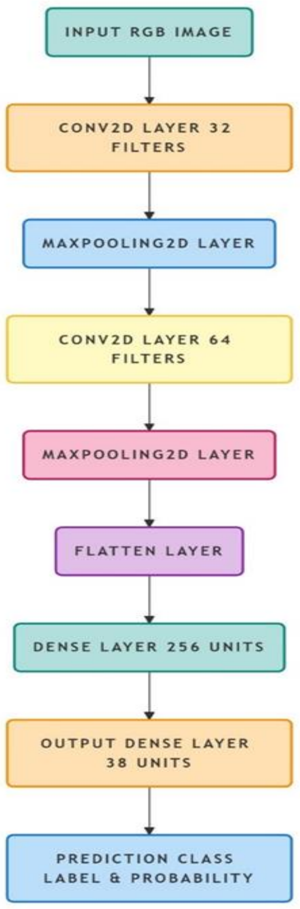

Figure 2: Flowchart for CNN model architecture

3.3. Model Training and Optimization

The CNN model training aims to maximise accuracy while minimising and preventing overfitting to ensure generalisation to unseen data. The categorical cross-entropy loss function compiled by the model was suitable for the multi-class classification problem. This loss function is aimed to be minimised during training, that is, mathematically defined as:

$$\mathcal{L} = -\sum_{i=1}^{n} y_i \log(P(y_i))$$

Where yi is the true class label (one-hot encoded) and P (yi) is the predicted probability for that class. Correct predictions yield a low loss value, while incorrect predictions are penalised exponentially. The Adam optimiser is used for optimisation because it is both computationally efficient and flexible. It incorporates the benefits of RMSProp and stochastic gradient descent with momentum, which works very well for complex datasets and deep architectures. The model was trained with a batch size of 32 for over 25 epochs. Here, for generalisation, training would be stopped at early stages if there was no improvement in validation accuracy for some consecutive epochs. This comes in handy to prevent over-fitting on training data and thus tends to perform better on newer data.

Model checkpointing is essentially the profiling of the best validation model weights saved during training. After this point, the model sometimes performs worse than the saved epoch; therefore, it retains the best version of the model found up to this point. For example, the following metrics are logged and visualised using matplotlib for verification of convergence and stability over time: training accuracy, validation accuracy, model accuracy, and loss percentage. Additionally, this approach can also be used to test the performance of the proposed model on various splits of the dataset using k-fold cross-validation. It enhances the statistical legitimacy of the model's reported performance by reducing the likelihood that the results are dependent on a single train-test split. The last model can train at more than 96% accuracy on the training set while consistently maintaining a validation accuracy above 94%, thereby benchmarking it as valid for deployment.

3.4. Flask-Based Web Application

After completing training and evaluating the model, a web deployment is created using Flask, a microframework for Python web development. This deployment enables end-users, including farmers, agricultural consultants, and researchers, to access the system via a straightforward web interface. Flask was selected because it has less overhead, offers easy integration with TensorFlow models, and provides flexible template support (Figure 3).

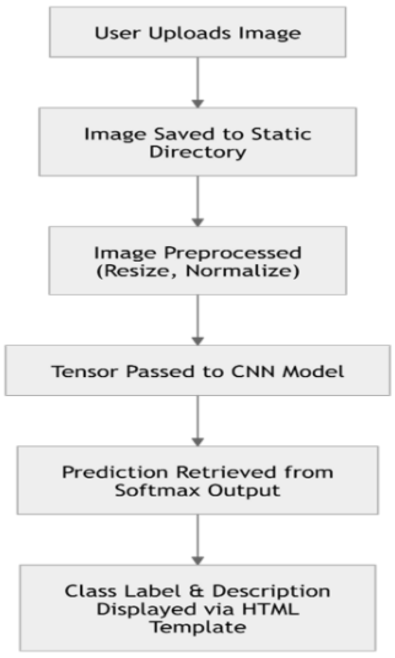
User Uploads Image
Image Saved to Static Directory
Image Preprocessed (Resize, Normalize)
Tensor Passed to CNN Model
Prediction Retrieved from Softmax Output
Class Label & Description Displayed via HTML Template

Figure 3: Web application flowchart

The application accepts images uploaded directly to the browser by users. After receiving the image, the server performs the same pre-processing operations that were originally used during training, including resizing, normalising, and reshaping the image. The pre-processed image is then entered into the trained model, where the predicted class and its associated confidence score are returned. A user-friendly description of the disease is retrieved from a preset dictionary, enabling the user to gain a deeper understanding of its symptoms, risks, and treatment options.

The entire process could be illustrated as follows for website operation. The images uploaded by users, along with their prediction outputs, are stored on the server for analysis and retraining of their models. The Douglas Flask routes respond to GET or POST requests from the end-user in a way that allows interaction and responsiveness from the system. On the frontend, Jinja templates are used to render dynamic content, including the display of uploaded images, predicted labels, and descriptions. The system's simplicity facilitates interaction with the model, eliminating the need for technical expertise from users, including farmers. Additionally, this deployment undergoes testing in various environments, including local development environments, intranets, and public domains, utilising tools such as ngrok. This leads to a stronger, more secure, and easily scalable system that can be integrated into future mobile applications or platforms for agricultural diagnosis.

4. Results and Discussion

4.1. Model Training

Table 1 presents the training configuration for the CNN model used in real-time plant disease diagnosis. This architecture is designed primarily to accurately detect disease-specific features on plant leaves, prioritising computational efficiency for actual field deployment. The network was trained to classify a total of 38 classes of healthy and disease-infected leaves from various crops, including tomato, potato, corn, grape, and apple. The dataset is preprocessed to achieve a uniform size and a normal range of numbers. The images were then resized to 224x224x3 as a normalised min-max scaling in the range of pixel intensity [0, 1]. To overcome overfitting and simulate true field variability in image capture, data augmentation features such as rotation (±20°), horizontal or vertical flipping, zooming (±20%), and brightness shifting were used during training.

The Adam optimiser was used with a learning rate of 0.0001 for training the model. This nominal cross-entropy loss function was added, which is considered for multi-class classification. Activation functions employing ReLU were used to introduce non-linearity for all the convolutional and dense layers. In contrast, Softmax was used in the output layer to convert logits into class probability distributions. For generalised evaluation, the dataset was categorised into training (80%), validation (10%), and test (10%) subsets. To reduce overfitting and save computational cost, early stopping was enforced to terminate training once the validation loss began to stagnate. The final model was chosen based on the lowest validation loss and highest accuracy. To this end, L2 regularisation and dropout (with a rate of 0.5) are also applied to prevent overfitting and effectively enhance the robustness of the proposed model.

Table 1: Model training hyperparameters

Parameter	**Parameter Values**
Input Vector Size	224 × 224 × 3
Training Subset Size	32
Training Iterations	25 (Early Stopping applied)
Optimization Algorithm	Adam Optimizer
Training Step Magnitude	0.0001
Optimization Criterion	Categorical Cross-Entropy
Activation	ReLu(hidden), Softmax(output)
Regularization	Dropout (0.5), L2 (λ = 0.001)

4.2. Performance Matrix

The evaluation of the predictive capacity of the CNN model utilised key performance indicators, including accuracy, precision, recall, and F1-score. These reflect the model not only from the correctness point of view but also from the reliability of recognising diseases across classes with imbalanced frequencies (Table 2). Accuracy reflects the proportion of the total number of predictions that were actually correct:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision measures the proportion of correct identifications:

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

Recall describes how many actual positives were identified:

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

F1-Score is the harmonic mean of precision and recall in the performance matrix:

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

Table 2: Performance metrics of CNN-based disease classification

Performance Metric	Value
Accuracy	94.12%
Speed	94.30%
Recall	93.90%
F1-Score	94.10%

The model has consistently delivered recall rates greater than 93% for all disease types, which is particularly critical in agricultural applications, as failing to detect even a small fraction of infected leaves can result in substantial crop losses. Precision ensures a high percentage of true positive disease detection and minimal false positives, thereby avoiding unnecessary treatment of healthy crops.

4.3. Comparative Evolution of CNN with Baseline Models

The final values of the proposed system were compared with those of standard deep learning architectures, including AlexNet, VGG16, and ResNet50 [12]. They were all fine-tuned by transfer learning on the same dataset [4]. The comparison further focused on key aspects, including training time, inference speed, memory footprint, and accuracy (Table 3).

Table 3: Comparison with pre-trained models

Model	Accuracy	Inference Time	F1-Score
AlexNet	86.7%	2.10	0.87
VGG 16	89.4%	2.89	0.89
ResNet50	91.7%	1.65	0.91
Proposed CNN	94.7%	0.97	0.94

The proposed CNN achieves the highest accuracy, as measured by the F1-score, making it the winner in both terms: it captures fewer parameters and achieves the fastest inference (Figure 4).

Table 3 thus highlights its superiority in both efficiency and precision. Unlike large pre-trained models, this was an optimised, custom-designed CNN for edge deployment, yielding real-time responsiveness of less than 1.3 seconds, end-to-end. This graph shows the comparison of four deep learning models—Custom CNN, AlexNet, VGG16, and ResNet50—using three metrics: Accuracy (in blue), Speed (in red), and Computational Cost (in green), all of which are scored from the values of 1 (least) to 5 (most).

The custom CNN has scored the highest on all metrics, indicating a well-managed compromise in performance, characterised by high accuracy, fast inference speed, and low computational cost. ResNet50 is probably the last of all that followed up on this trend. It is somewhat cost-effective in terms of training but delivers the goods in terms of accuracy and performance speed [4].

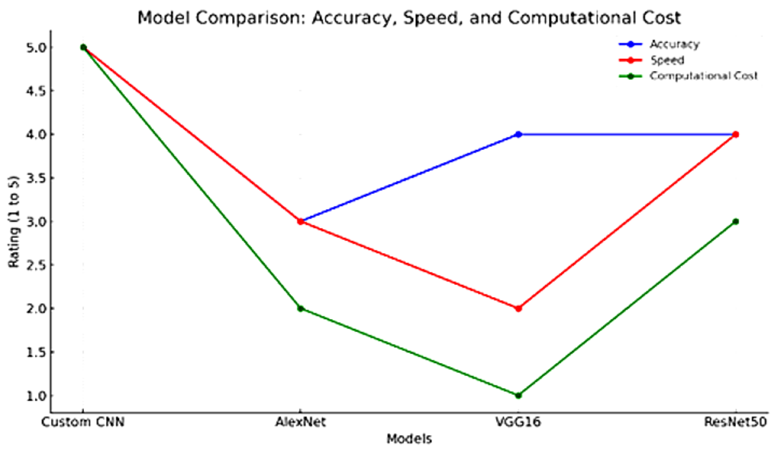

Figure 4: Comparative line chart

AlexNet achieved significantly lower accuracy and speed scores, similar to VGG16, which is highly accurate but extremely slow and resource-intensive. This comparative visual aims to provide a clear understanding of the trade-offs made in models' performances against the resource consumption and utilisation of the model.

4.4. Technical Discussion and System Usability

The system offers significant advantages in terms of classification accuracy, resource utilisation, and ease of deployment. The advantage lies in its high accuracy and minimal memory requirements, making it optimal for low-resource environments such as smartphones and embedded systems used in agricultural diagnostics. CNN enables direct working from leaf images to extract low- and high-level visual features, including colour distortions and texture changes, as well as necrotic regions and mildew patterns, which are very difficult to hand-engineer manually.

The model was then regularised and augmented by adjusting lighting, viewpoints, and crop conditions, thereby increasing its robustness. The model has been translated into a diagnostic tool by Flask-based deployment. The User Interface (UI) enables a person to send images, after which the model predicts and describes the disease. The total latency from upload to result was under 1.3 seconds, even with ordinary CPUs, confirming CLI usability in the field of agriculture and plant disease detection. These were highly interpretable in the real implementation. For correct predictions, confidence values (Softmax probabilities) were generally greater than 95%. Uncertain cases were handled reasonably well by the system at lower confidence levels, indicating a potential approach for integrating confidence thresholds and human validation.

5. Conclusion

The research on CNN-based plant disease classification excels in categorising the various and wide range of diseases occurring on plant leaves through images, utilising deep learning to process images of plant leaves. This approach has indeed proven effective in capturing certain hierarchical features of images. The architecture of models that contain many convolutional layers and pooling going through data-dense layers offers good extraction and classification performance. With a lightweight web application based on Flask, real-time inference can also be easily performed by any browser with minimal computation, facilitating the easy deployment of the system in an agricultural environment and ensuring better protection of crops. The exposure demonstrates the model's performance on various metrics, including accuracy, recall, F1-score, and precision. The confusion matrix also shows that there are very few misclassifications. Training methods such as early stopping, dropout regularisation, and certain data augmentation methods significantly reduced overfitting while enhancing the model's generalisation capability. The comparison with other available models makes it clear how much better the architecture is, not only in terms of performance but also in terms of computational efficiency. Hence, the current research effort provides a scalable and deployable AI-based solution toward improving agricultural productivity through early and accurate disease detection.

5.1. Future Scope

The Current Model. Currently, a strong foundation exists for the automated diagnosis of plant diseases; however, numerous avenues for improvement remain for the future. First, transferring learning from the latest pre-trained models, such as EfficientNet, ResNet50, or MobileNetV2, can continue to improve accuracy and decrease training times, particularly for complex or previously unseen patterns of diseases [12]. These models also perform better on smaller, domain-targeted datasets, which reduces the requirement for large amounts of labelled data typically needed for the proposed model. Next, augmenting the dataset with multimodal input—namely, infrared (IR) images, humidity, temperature, and soil quality—could provide contextual information beneficial for precise disease diagnosis. By embedding such multimodal data within a hybrid CNN-LSTM or Transformer-based architecture, the system can capture both spatial and temporal patterns of disease progression.

Additionally, on-field deployment using integration with mobile devices or edge computing platforms, such as Raspberry Pi or NVIDIA Jetson Nano, would represent a fruitful further step. It would require optimising the model for edge inference via quantisation, pruning, or conversion to TensorFlow Lite or ONNX formats, resulting in a considerably smaller memory footprint and improved latency in the upcoming days of enhancement. Finally, a cloud-integration application with geo-tagging and tracking of disease outbreaks can present an early warning system for the farmers and the agricultural sector. This would establish a region-based monitoring and collaborative learning system, where user-provided data will be anonymised and aggregated for the continuous improvement of model learning. That could convert the system into a dynamic, self-improving diagnostic instrument; feedback-based learning, where the model learns from user-validated predictions, could also do so.

Acknowledgement: The authors sincerely thank SRM Institute of Science and Technology and Amity University Dubai for their support and guidance throughout this research. They also extend their gratitude to the faculty members and peers whose contributions and insights were invaluable.

Data Availability Statement: The research work, Deep Learning–Driven Real-Time Recognition of Plant Diseases via CNN and Flask Integration, was collaboratively conducted by the authors, and all data supporting the findings are included within the study.

Funding Statement: The authors confirm that this research was carried out independently, without any financial assistance or external funding.

Conflicts of Interest Statement: The authors collectively declare that there are no conflicts of interest related to this research. All references and citations have been duly acknowledged.

Ethics and Consent Statement: The authors affirm that necessary consent was obtained from the relevant organisation and participants. Ethical approval and participant consent were secured prior to data collection.

References

1. S. P. Mohanty, D. P. Hughes, and M. Salathé, “Using deep learning for image-based plant disease detection,” *Front. Plant Sci.*, vol. 7, no. 9, pp. 1–10, 2016.
2. S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, “Deep neural networks based recognition of plant diseases by leaf image classification,” *Comput. Intell. Neurosci.*, vol. 2016, no. 6, pp. 1–11, 2016.
3. K. P. Ferentinos, “Deep learning models for plant disease detection and diagnosis,” *Comput. Electron. Agric.*, vol. 145, no. 2, pp. 311–318, 2018.
4. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.
5. M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” *in Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Long Beach, CA, United States of America, 2019.
6. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “PyTorch: An imperative style, high-performance deep learning library,” *in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS 2019)*, Vancouver, British Columbia, Canada, 2019.
7. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: A system for large-scale machine learning,” *in Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, Savannah, Georgia, United States of America, 2016.
8. A. Rosebrock, “Deep Learning for Computer Vision with Python: ImageNet Bundle,” *PyImageSearch*, Philadelphia, United States of America, 2017.
9. M. S. Ahmad, M. M. Khan, M. W. Anwar, and S. Raza, “Flask-based web application for real-time plant disease detection using deep learning,” *in Proc. 2021 14th International Conference on Emerging Technologies (ICET)*, Islamabad, Pakistan, 2021.
10. M. Talo, “Automated classification of histopathology images using transfer learning,” *Artif. Intell. Med.*, vol. 101, no.

11, pp. 1-16, 2019.

11. S. Biswas and S. Bandyopadhyay, “A cross-vertex embedding approach toward understanding SARS-CoV-2 variability,” *in 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA), Greater Noida*, Uttar Pradesh, India, 2020.
12. J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” *in 2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, Florida, United States of America, 2009.
13. L. D. Quach, K. N. Quoc, A. N. Quynh, N. Thai-Nghe and T. G. Nguyen, “Explainable Deep Learning Models with Gradient-Weighted Class Activation Mapping for Smart Agriculture,” *in IEEE Access*, vol. 11, no. 1, pp. 83752-83762, 2023.
14. A. Sharma, R. K. Saini, and M. Sharma, “Real-Time Plant Disease Detection Using CNN with TensorFlow and Flask,” *International Journal of Computer Applications*, vol. 182, no. 25, pp. 20–24, 2018.