Natural Resources for Human Health
eISSN: 2583-1194

Original Research

Received 20 May 2026
Revised 30 June 2026
Accepted 14 July 2026

Natural Resources for Human Health 2026, Vol. 6(7s): 1049–1061

KEYWORDS:
Plant disease detection, CNN, deep learning, care suggestion, personalized guidance.

AgriVisio: An Intelligent Deep Learning Framework for Plant Disease Detection and Personalized Care Guidance

Hemlata Goyal[1], Vishakha Singhal[2], Renu Kumawat[3], Neha Janu[4*]

[1,2,3,4] School of Computer Science and Engineering, Manipal University Jaipur, Jaipur
[1]hemlata.goyal@jaipur.manipal.edu
[2]vishakhasinghal30@gmail.com
[3]renu.kumawat@jaipur.manipal.edu
[4]neha.janu@jaipur.manipal.edu

ABSTRACT: Plant diseases significantly influence crop production and agricultural sustainability. Early and appropriate user-friendly disease identification for personalized use for farmers is essential for effective crop management. Now-days advancement in deep learning and artificial intelligence afford promising results for automated plant disease detection in leaf images. In this study, we propose an integrated AgriVisio model, that combines deep learning techniques for plant disease detection with real-time plant identification and customized care recommendations. We developed a custom Sequential Convolutional Neural Network (CNN) model trained on the opensource Plant Village Dataset, available on Kaggle having fourteen types of plants (apple, blueberry, cheery, corn, grape, orange, peach, pepper, potato, raspberry, soyabean, squash, strawberry, tomato) suffering from either bacterial, viral or fungal infection of 38 type of diseases. Proposed AgriVisio model achieved an accuracy of 99.23%, significantly outperforming to identify plant disease and suggest care guidance, over pre-trained models. PlantID API is used to identify the species of the plant and OpenWeather API fetches real time weather data to provide location specific care guides and has potential improvement for agricultural practices from early disease detection and for directly providing personalized guidance.

1. INTRODUCTION

Agriculture is an important part of the global economies and crop health is vital to ensure food security. However, there is a serious threat to agricultural productivity from plant diseases that cause heavy financial loss. It is important to detect plant diseases at early stages to minimize these losses and improve crop yields. Generally, plant disease detection has been conducted manually, and such a method is very time consuming and requires expert knowledge [1, 2].

Nowadays, deep learning, considered one of the most advanced branches of artificial intelligence (AI), has presented very promising results for automating plant disease detection. Specifically, when it comes to plant disease detection, Convolutional Neural Networks (CNNs) have been very successful, being able to distinguish diseases on plant leaves based on visual attributes. Custom models for some datasets are used to outperform previous generalized pre-trained models by a large margin. In addition, contextual information such as the weather condition can provide a more holistic perspective about the plant health monitoring.

Plant identification and personalized guidance on how to take care of them are also equally important for plant health optimization besides disease detection. There are many plant identification systems, but most do not provide plant specific care recommendations designed to meet a plant's needs and the user's local climate. This research presents a novel system that

combining disease detection using CNN, real time identification of plant using the PlantID API, and location based care recommendations using OpenWeather API. The end-user system has a user-friendly web interface in ReactJS for front-end and Flask for dealing with image processing, model inference and API requests on the backend. It provides early disease detecting, species identifying and care guidance systems to the farmers, gardeners and agriculture enthusiasts.

A comprehensive analysis of recent papers reveals some common methodologies in plant disease detection. Mohanty et al. (2016), achieved over 99.0% accuracy using a sequential CNN on the PlantVillage dataset for 38 crop disease classes, demonstrating the power of structured models without complex architectures [3]. Sladojevic et al. (2016), created a basic sequential CNN for 13 plant diseases, establishing that straightforward architectures can provide good performance with nothing more than convolutional, pooling, and fully connected layers [4]. Kumar and Jain (2021), developed a sequential multi-layer network to detect powdery mildew in maize, accomplishing an accuracy of 85.9%. This points to the effectiveness of using stacked architectures for agricultural applications [5]. Prajapati et al. (2017), used a simple CNN for identifying diseases in rice leaves, in which, the model worked quite well on some kinds of disease, especially when the disease was in an early state, with a limited amount of training data, it was also surprisingly effective [6], Amara et al. (2017), concentrated on classifying diseases of the banana plant by using CNNs with grayscale images.

Table 1. Description of the experimental data (plant type and diseases)

Plant Type	**Diseases Classes**
1. Apple	AppleScab, AppleBlackRot, AppleCedarRust, AppleHealthy
2. Blueberry	BlueberryHealthy
3. Cherry	CherryPowderyMildew, CherryHealthy
4. Corn	CornGrayLeafSpot, CornCommonRust, CornNorthernLeafBlight, CornHealthy
5. Grape	GrapeBlackRot, GrapeBlackMeasles, GrapeLeafBlight, GrapeHealthy
6. Orange	OrangeHuanglongbing
7. Peach	PeachBacterialSpot, PeachHealthy
8. Pepper	PepperBellBacterialSpot, PepperBellHealthy
9. Potato	PotatoEarlyBlight, PotatoHealthy, PotatoLateBlight
10. Raspberry	RaspberryHealthy
11. Soyabean	SoybeanHealthy
12. Squash	SquashPowderyMildew
13. Strawberry	StrawberryHealthy, StrawberryLeafScorch
14. Tomato	TomatoBacterialSpot, TomatoEarlyBlight, TomatoHealthy, TomatoLateBlight, TomatoLeafMold, TomatoSeptoriaLeafSpot, TomatoSpiderMites, TomatoTargetSpot, TomatoMosaicVirus, TomatoYellowLeafCurlVirus

This research shows that even very basic inputs can result in effective classification, which is extremely beneficial to field settings that lack resources [7], Zhang et al. (2018), in a sequential CNN for apple leaf diseases, this work emphasized the potential benefit to be gained not from adding model complexity but from optimizing hyperparameters such that the model is more attuned to the specific dataset at hand [8]. Jha et al. (2020), they put forward a hybrid model but observed that sequential CNNs are quite capable by themselves and that there is room for improvement with such models if they are given environmental data as inputs. They suggest that this could be a way forward for enhancing sequential models [9]. Patel et al. (2019), reviewed the shortcomings of plant disease datasets and pointed out the efficacy of simple sequential models by transfer learning from larger

datasets to improve generalization [10]. To perform multi-class classification of plant diseases [11], a series of convolutional neural network (CNN) architectures were explored and evaluated.

Existing approaches have confirmed promising performance score with the help of deep learning and computer vision methods for plant disease identification and contribute limited support for disease management and care recommendations. Although there is much more scope to address these limitations. This research proposes comprehensive support for plant disease detection and care guidance, enabling the system to move beyond disease classification toward actionable guidance for plant health management.

2. MATERIALS AND METHODS

2.1 Dataset

Proposed model experimentation is leveraged on the PlantVillage-Dataset downloaded from Kaggle open source data repository. The dataset contains approximately 54,306 high-resolution RGB images of 14 distributed plants having 38 disease classes. Each class have either a healthy leaf or to one of the common and rare foliar diseases affecting ten major crop species including corn, tomato, grape and apple. Each image was resized into 224×224 pixels for standardization inputs into proposed model. Detailed description of the plant type, diseases classes from the dataset used (see Table1) and visualization of each plant type and disease (see Figure1) is given in this section.

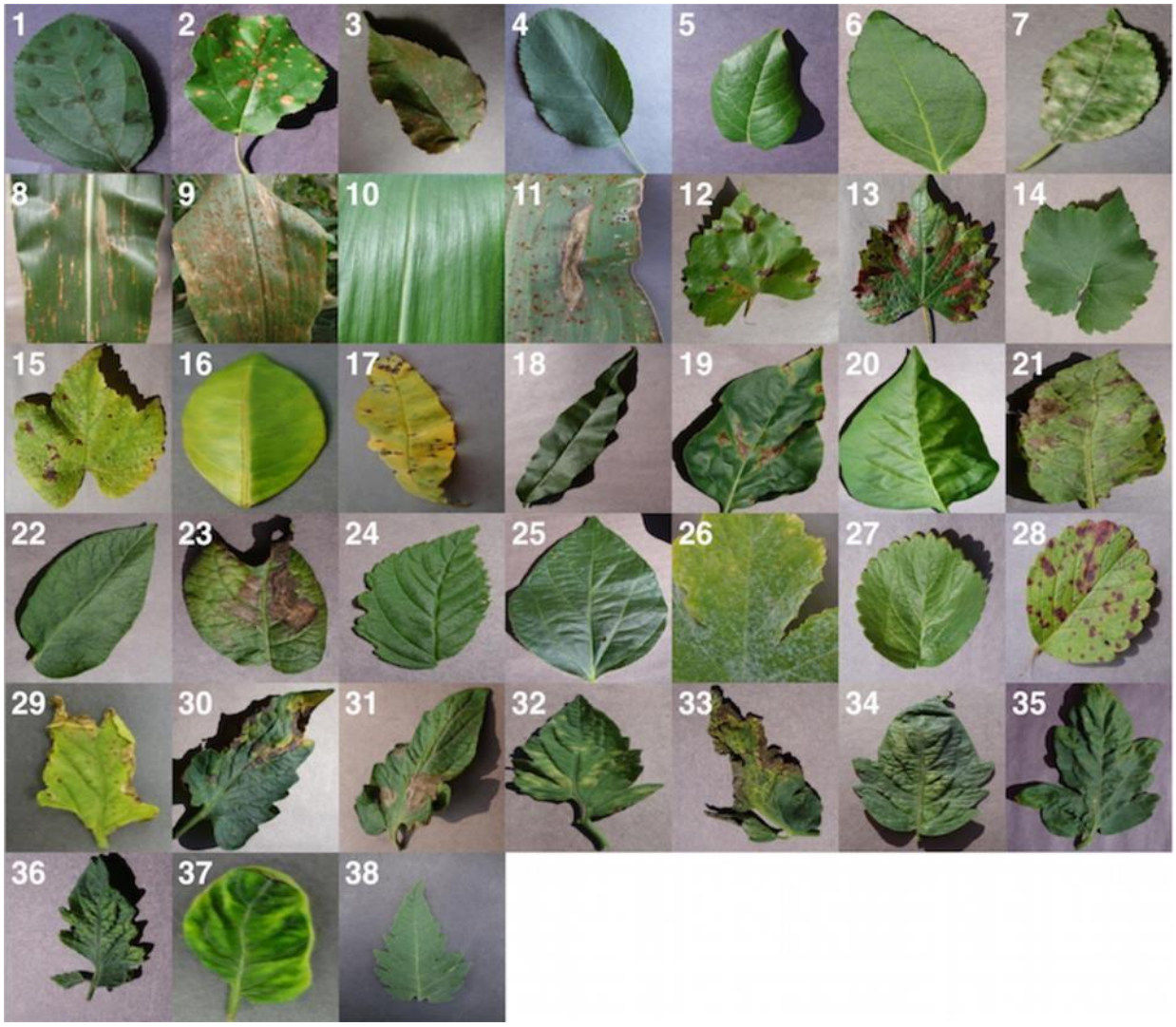

Fig. 1. Plant Village Dataset [17]: Visualization of Plant Type and Disease

2.2 Dataset Preprocessing

Each image of 224×224 pixels, were scaled to the [0, 1] range by multiplying by 1/255, thereby normalizing across different lighting conditions and device sensors. Keras's ImageDataGenerator is used for data augmentation to reduce overfitting and replicate the diversity of real-world agricultural imagery. Instead of creating and storing augmented images on storage,, transformations were applied dynamically in each batch, significantly expanding the effective dataset without inflating storage requirements. Augmentations included random horizontal and vertical flips to emulate varied leaf orientations; rotations up to ±25° to account for camera tilt and perspective changes; zoom operations within ±20 % to simulate differences in distance; and brightness shifts of ±15 % to mimic fluctuations in ambient light. This regime of stochastic, batch-level augmentation both enriches the model's exposure to realistic distortions and enhances its ability to generalize to unseen field-acquired images.

2.3 Disease Detection Model Architecture

We have explored four pre-trained CNN model architecture of VGG16, InceptionV4, ResNet50 and DenseNet121 to compare the proposed architecture. VGG-16 is a deep convolutional network with a simple 3×3 filters and sequentially architecture of convolutional and max-pooling layers, which can be easily implemented but computationally heavy and negatively affected by vanishing gradients.

On the other hand, InceptionV4 designs its blocks to incorporate multiple filter sizes and owns auxiliary classifiers so as to facilitate gradient flow in training. ResNet-50 proposes to compensate the vanishing gradient problem with residual connections and make possible deeper architectures using bottleneck convolution blocks. DenseNet-121 further improves features reuse and gradient propagation by densely connecting each layer to all previous layers in the same block, controlling model complexity through transition layers to preserve and try to reuse the model at the same level of simplicity achieving better learning efficiency through lower number of parameters.

Table 2. Pre-trained network architecture model

Network Model	Total Layers	Max Pool Layers	Dense Layers	Drop-out Layers	Flatten Layers	Filter Size	Stride	Trainable Parameters
VGG-16	16	5	3	2	1	3 × 3	2 × 2	41.2 M
Inception V4	22	5	-	-	-	1 × 1, 3 × 3, 5 × 5	2 × 2	119.6 M
ResNet-50	50	1	3	2	1	3 × 3	2 × 2	23.6 M
DenseNet-121	121	4	4	-	-	3 × 3, 1 × 1	2 × 2	7.05 M

2.4 Proposed Model Architecture

The proposed customized sequential model is built using a Convolutional Neural Network (CNN) architecture, designed explicitly for image classification such as identifying plant diseases (see Figure2). The architecture follows a sequential structure, where layers are stacked in a linear manner to progressively extract and learn features from the input images.

We conducted all experiments using the recently published "PlantVillage-Dataset" dataset on Kaggle, which comprises 54,306 high-resolution images spanning 38 different disease categories across ten common crop species. To ensure reproducibility, grouped all images by both crop type and disease label before shuffling and splitting into 80% training, 20% validation, and test partitions.

Each image was resized to 256×256 pixels, and we applied real-time data augmentation—random rotations up to 25°, horizontal and vertical flips, brightness and contrast jitter, and random zoom—to mimic field conditions and increase robustness. Prior to training, pixel values were scaled to the [0,1] range and standardized using channel-wise mean and standard deviation computed from the training set.

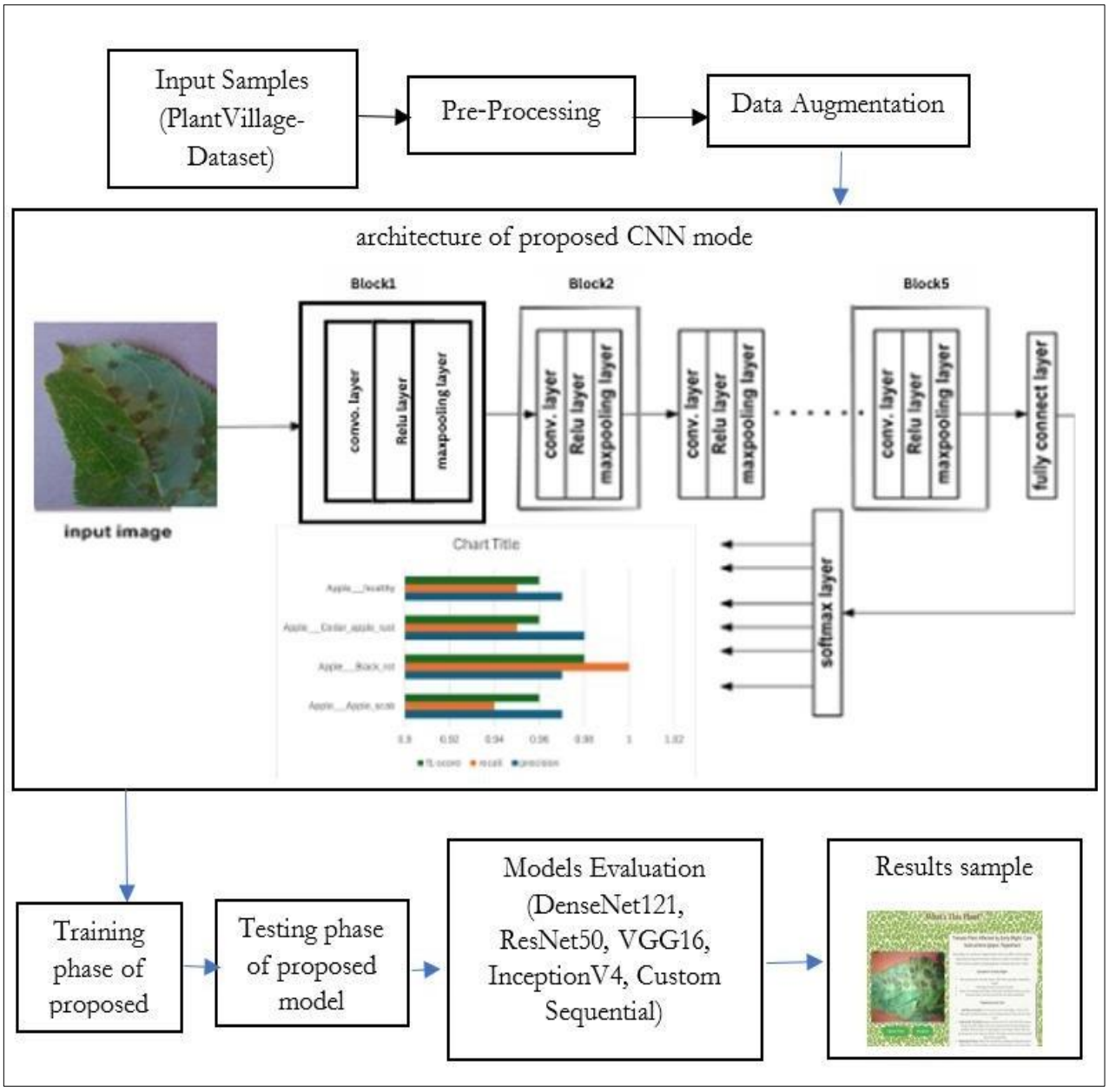

Fig. 2. Methodology of the proposed AgriVisio model framework

The model begins with three convolutional blocks. Each block consists of a convolutional layer followed by a max-pooling layer [12][13]. These convolutional layers are responsible for detecting various visual features, starting with basic patterns like edges and textures in the initial layers and gradually identifying more complex and abstract features in the deeper layers. Max-pooling layers help in reducing the spatial dimensions of the feature maps, which not only decreases computational load but also helps in retaining the most significant features.

The multi-dimensional data is extracted to feature and then model flattens the feature into one dimensional vector before classifying. This is followed by a dense (fully connected) layer that interprets the extracted features and learns their relationships. A dropout layer is included to reduce the risk of overfitting by randomly disabling a portion of the neurons during training.

Finally, the output layer uses a SoftMax activation function to produce a probability distribution over three possible classes, representing the model's prediction of the plant's condition [14]. Model training proceeded on this augmented dataset with proposed custom Sequential CNN, which contains just under five million parameters but achieves state-of-the-art performance

for this collection of plant diseases. We used the Adam optimizer with an initial learning rate of 1e-3, decayed by a factor of 0.5 every ten epochs, and monitored validation loss to trigger early stopping. Dropout layers (rate 0.3) and batch normalization further prevented overfitting.

The overall test accuracy of 97.23%, with per-class F1 scores above 0.95 for the ten largest disease groups. Confusion-matrix analysis revealed only minor class confusion between closely related leaf-spot diseases, suggesting that a few targeted samples or class-weighted loss adjustments could push performance even higher [15][16] to distinguish between healthy and diseased plants.

Table 3. Custom sequential network architecture model

Network Model	**Custom CNN (Sequential)**
Total layers	10
Conv2D layers	3
Max pool layers	2
Dense layers	2
Drop-out layers	1
Flatten layers	1
Filter size	3 × 3
Stride	Default (1 for Conv, 2 × 2 for Pooling)
Trainable parameters	~2.6 M

3. RESULTS AND DISCUSSION

The proposed model was examined using various evaluation metrics and visualization techniques given in subsequent sections.

3.1 Classification Performance

The model scored the classification performance score for training, test, validation accuracy of 99.0%, 99.23% and 96.7% respectively. The detailed classification report reveals consistently high values of precision, recall, and F1-score across all 38-plant disease and health classes. Several classes, such as Cherry (including sour) as healthy, Grape as Leaf blight (Isariopsis Leaf Spot), and Corn (maize) as healthy, attained perfect scores, indicating highly reliable predictions.

Macro-average precision, recall, and F1-score: 0.99, reflecting uniform model performance across all classes regardless of class distribution. Weighted average scores: 0.97, suggesting the model performed well even in the presence of minor class imbalance. These results demonstrate the model's effectiveness in multi-class classification tasks involving fine-grained image distinctions (Figure3).

3.2 Training and Validation Accuracy

The training score (see Figure4) shows a consistently increasing as epochs increasing. The training accuracy increased notably during the initial epochs and become stable at approximately 99.0% by the tenth epoch. Similarly, the validation accuracy demonstrated a steady trend, reaching approximately 96.7%, without any sign of overfitting and reveal towards generalization for unseen data.

Natural Resources for Human Health
eISSN: 2583-1194

Fig. 3. Disease detection result for Tomato

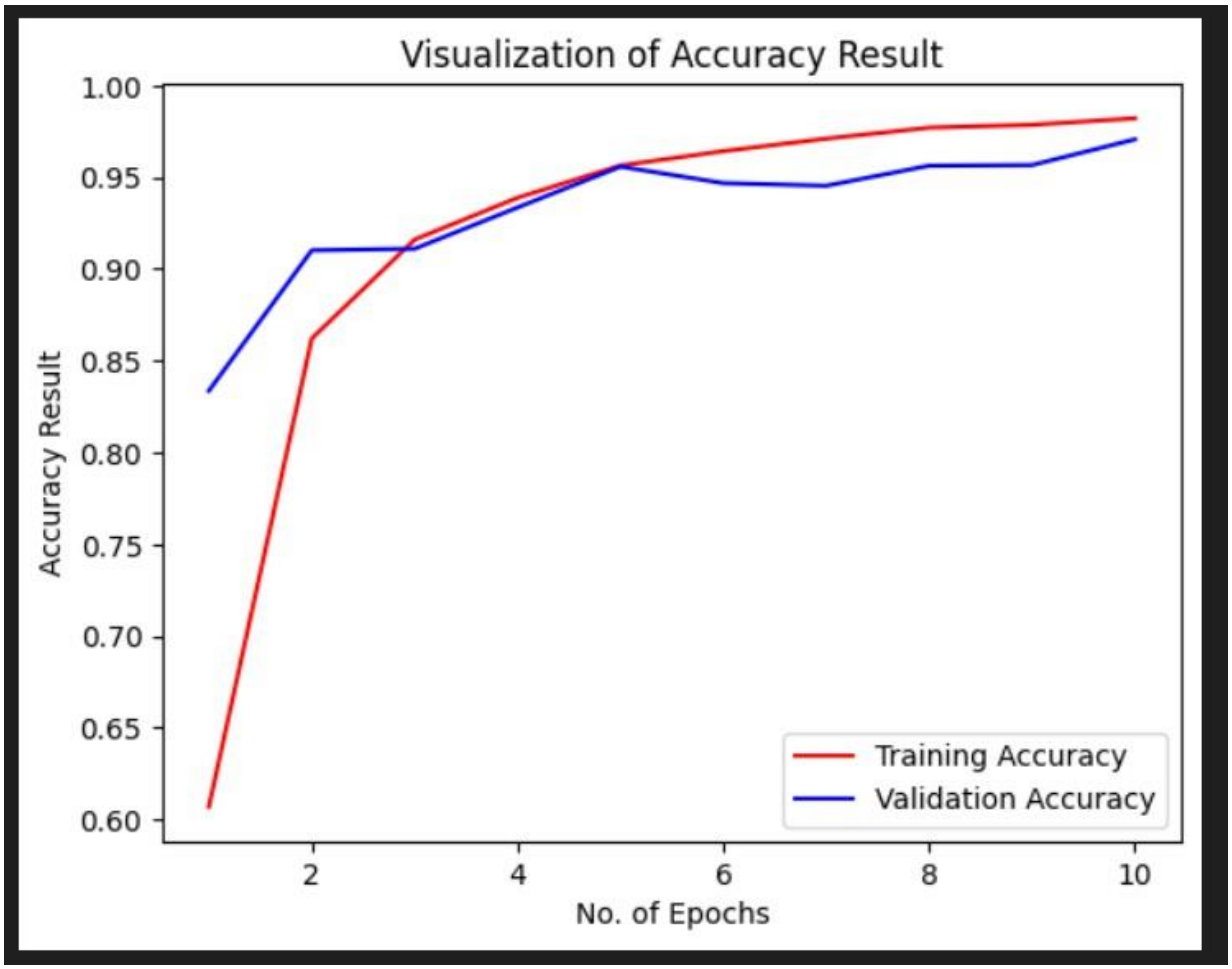

Fig. 4. Disease detection accuracy results for Tomato

3.3 Model Accuracy and Comparison

The model accuracy of each model is given in Table4. Following the evaluation of pre-trained models, a Custom Sequential CNN architecture was also developed and trained from scratch. While the pre-trained models are significantly deeper and contain tens of millions of parameters (e.g., VGG16 has ~138M), the custom model was designed with a relatively shallow architecture and fewer trainable parameters (~2.6M).

Despite its smaller size, the custom model achieved high performance on the same dataset. This improvement is attributed to its domain-specific design, efficient architecture, and use of strong data augmentation strategies, which allowed it to generalize effectively without overfitting.

VGG16, generally lower performance compared to DenseNet121, sensitive to data augmentation while ResNet50 have good precision, but lower recall; prone to false negatives in some diseases. InceptionV3 scored high recall but slightly lower precision; performs well on complex disease types.

DenseNet121 scored best performance in accuracy and F1 score; well-suited for fine-grained classification. Existing state-of-the-art models compared with proposed Custom Sequential model, scored extremely high recall and precision; excels in identifying both common and complex disease types with minimal false classifications. The visualization of performance measures can be interpreted in Figure 5.

Table 4. Comparative accuracy of state-of-the-art models and proposed model

Model	Accuracy	Precision	Recall	F1 Score	Training Time (hrs.)
VGG16	85.12%	84.00%	83.50%	83.75%	6
ResNet50	87.41%	88.00%	85.00%	86.47%	5
InceptionV3	88.99%	89.50%	87.00%	88.25%	5
DenseNet121	91.23%	92.50%	90.00%	91.24%	4
Proposed Custom Sequential	99.23%	97.60%	97.8%	92%	2

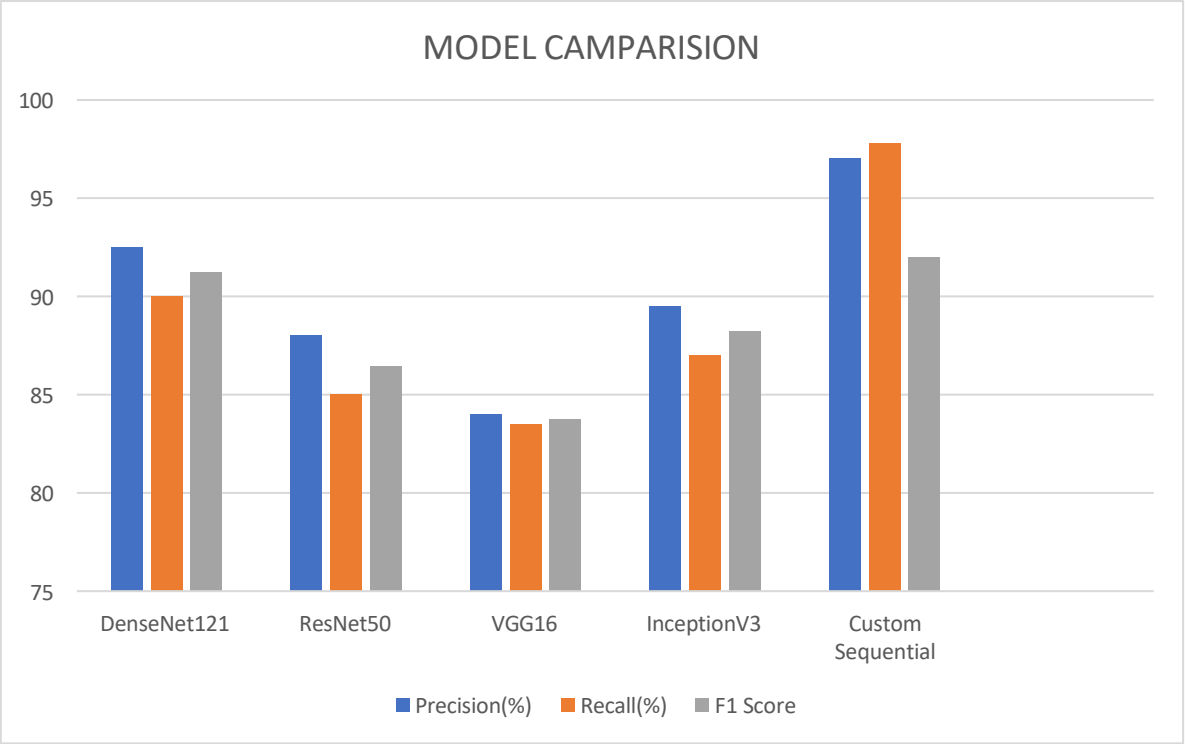

Fig. 5. Comparative disease detection accuracy results for Tomato

Combined with careful tuning, the Sequential CNN model has the highest accuracy (99.23%) in all models owing to its architecture with a balanced depth and efficiency due to its generalization. The model takes advantage of integrating dropout and batch normalization and thus avoids overfitting by ensuring stable learning. This architecture provides excellent performance across plant species and disease types due to its ability to capture both subtle as well as the most prominent disease features, which makes it very reliable.

3.4 Deep Learning Models and Weather Data Integration:

Model performance can be significantly enhanced with the integration of weather data through the OpenWeather API [5]. To introduce environmental layer to model's prediction layer, it made its prediction much more context sensitive and adaptive. Additionally, integration with the Gemini API can be beneficial to users by providing region specific watering schedules, fertilization routines, and disease prevention methods to follow [10].

3.5 Plant Identification

The end-user system starts at the browser to upload the leaf image of plant for recognition, (built in HTML, CSS and JavaScript in a minimalist interface). After selection, a JavaScript procedure gets the file input and decode the image into a Base64 encoded string for easy transmission of binary data to a digital request without needing multipart form handling. A Flask endpoint is called asynchronously sending this request with the payload as JSON object (see Figure 6).

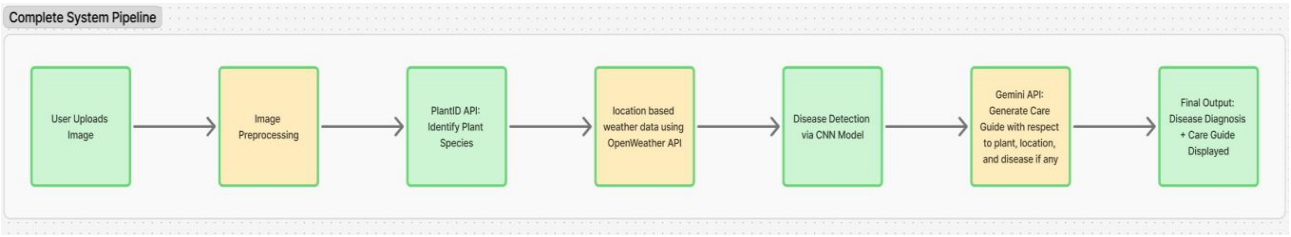

Fig. 6. End-user System Pipeline

Inside the Flask application, the Base64 string is once again decoded back to image buffer match for OpenCV that can be processed in the usual way—e.g., resizing, normalization, converting to another color space, etc.—before being forwarded. The application then constructs an HTTP POST request to the PlantID API, including the image data and authentication credentials in the headers. PlantID's backend, analyses the image and returns a JSON response containing the most probable scientific and common names along with associated confidence scores. The server captures this response, applies simple validation (for example, discarding results with confidence below a set threshold), and formats the remaining information into a clear JSON structure.

Back in the browser, the JavaScript success callback parses the server's JSON and dynamically updates the page to display the plant's name, a thumbnail of the uploaded image, and the confidence percentage. If multiple candidates are returned, the interface can list them in descending order of confidence, enabling users to explore alternatives. Error handling is built in at every stage: network failures trigger retry logic, low-confidence identifications prompt a user-facing suggestion to try a different image angle, and any unexpected API errors result in a friendly notification rather than a crash. By offloading recognition to the PlantID API and handling encoding, transmission, and response parsing on the client and server, this module delivers rapid, accurate, and scalable plant identification without the overhead of local model storage or GPU infrastructure.

3.6 Care Guide System

Once a plant has been identified, the system enriches this result with customized care instructions by incorporating real-time environmental data. Upon receiving the plant's common name and the user's location, the frontend transmits these details to the Flask backend, which immediately queries the OpenWeather API for local meteorological measurements—primarily temperature and humidity, and where available, light index or UV index. These conditions are crucial for plant health, as the y directly influence watering needs, susceptibility to disease, and overall growth habits (see Figure7).

Armed with both the species identity and live weather metrics, the backend constructs a single descriptive prompt that succinctly captures the plant's requirements under today's conditions (for example: "Generate care advice for a Ficus lyrata in 22 °C, 60 % humidity, with moderate sunlight"). This prompt is sent to the Gemini conversational AI service, which synthesizes the data into a structured care guide. The response typically includes recommended watering intervals, ideal light exposure, temperature range guidance, fertilization schedules, and preventative measures against moisture-related diseases. By leveraging a powerful language model, the system transforms raw data into clear, actionable steps without requiring extensive rule-based coding for each plant species.

Natural Resources for Human Health
eISSN: 2583-1194

Fig. 7. Care Guide View of Apple Plant

The final care guide is returned to the frontend as a JSON object, which the browser's JavaScript then parses and renders into a user-friendly layout. Users see distinct sections—such as watering, lighting, and fertilization—each accompanied by concise explanations. Error handling and fallback strategies ensure robustness: if the weather service is unavailable, cached or default regional values are used; if the language API fails to respond, a simplified generic guide is delivered. This seamless integration of species identification, live environmental data, and AI-driven content generation provides growers with precise, context-aware recommendations tailored to their specific locale and plant needs.

6. CHALLENGES AND FUTURE DIRECTIONS

One of the main challenges was ensuring that the model did not overfit to specific leaf backgrounds or lighting artifacts. We addressed this by applying aggressive data augmentation and validating on unseen image sets. Another challenge was processing latency when running the system on lower-end devices. To avoid this, we minimized the model size and considered TensorFlow Lite conversion for mobile deployment.

Future work can also be done to further enhance the utility and impact of this work. The first would be to take pest detection, though combined with disease identification, as this was an area in which they often present together or amplify plant diseases and having a model that could call out on both vectored stressors would yield more accurate diagnosis and directed intervention. Second, though, a care guidance framework would be enriched with soil-type recommendations (pH ranges, texture classifications, nutrient profiles) to facilitate caring with soil-type recommendations so users can tailor treatment plan and cultivation practice according to their environment context. Third, putting the system into a mobile application also with on-device inference and offline support would greatly expand the reach of the system, enabling farmers in remote or connectivity-challenged areas to undertake real time, many fields' observations and receive on the spot recommendations. At last, marrying complementary agricultural data entities (including in situ soil-cues, local weather arbets, and pest trap counts) would transform the system into a full-blown decision support system, and with biotic and abiotic cue parity. These enhancements will together drive a much better, user friendly solution to precision agriculture.

7. CONCLUSION

Our custom CNN model demonstrated exceptional performance with training accuracy (99.0%), validation accuracy (96.7%). Its simple yet efficient architecture made it fast to train and easy to deploy. Unlike DenseNet121, which had a 91.23% accuracy and heavier computational demands, our model is more suitable for real-time applications and edge deployment. Performance was also consistent across crops and lighting conditions, which can often skew classification results. Data augmentation played a crucial role in achieving generalizability. The integration of APIs like PlantID and OpenWeather added intelligent contextualization to disease predictions, enabling user-friendly and region-specific care recommendations. We also observed that combining visual and environmental data can reduce misclassifications that arise in ambiguous leaf images.

This research developed an advanced and efficient plant disease detection with real-time plant identification and customized care recommendations with plant health monitoring system by leveraging a custom-built deep learning model and intelligent API integration. The high accuracy, low complexity, and practical deployment options make our system ideal for scalable agricultural solutions. The use of a comprehensive dataset and a tailor-made CNN architecture demonstrates a meaningful improvement over traditional pre-trained model. Our system bridges the gap between AI-based disease detection and actionable plant care, enabling farmers and gardeners to act quickly and with confidence.

AUTHOR CONTRIBUTIONS

Hemlata Goyal: Conceptualization, Supervision, Methodology, Analysis and Writing. **Vishakha Singhal:** Data Curation, Laboratory Analysis. **Renu Kumawat:** Revision and Editing. **Neha Janu:** Correspondence, Revised & Editing.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

ETHICS STATEMENT

NA

DATA AVAILABILITY STATEMENT

The dataset [17] is taken from kaggle repository https://www.kaggle.com/datasets/emmarex/plantdisease

REFERENCES

[1] Sankaran, S., Mishra, A., Ehsani, R., & Davis, C. (2010). A review of advanced techniques for detecting plant diseases. Computers and electronics in agriculture, 72(1), 1-13.

[2] Khakimov, A., Salakhutdinov, I., Omolikov, A., & Utaganov, S. (2022). Traditional and current-prospective methods of agricultural plant diseases detection: A review. In IOP Conference series: earth and environmental science (Vol. 951, No. 1, p. 012002). IOP Publishing.

[3] Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in plant science, 7, 215232.

[4] Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016). Deep neural networks based recognition of plant diseases by leaf image classification. Computational intelligence and neuroscience, 2016(1), 3289801.

[5] Kumar, D., & Jain, S. (2021). Maize crop disease classification using deep learning. Materials Today: Proceedings, 47, 599–606

[6] Prajapati, H. B., Shah, J. P., & Dabhi, V. K. (2017). Detection and classification of rice plant diseases. Intelligent Decision Technologies, 11(3), 357–367.

[7] Amara, J., Bouaziz, B., & Algergawy, A. (2017). A deep learning-based approach for banana leaf diseases classification. In BTW Workshop (pp. 79–88). CEUR Workshop Proceedings.

[8] Zhang, S., Wu, X., & You, Z. (2018). Leaf image based cucumber disease recognition using sparse representation classification. Computers and Electronics in Agriculture, 134, 135–141.

[9] Jha, S., Doshi, A., Patel, N., & Shah, M. (2020). Precision agriculture using IoT and deep learning techniques for early detection of crop diseases. Procedia Computer Science, 167, 515–524.

[10] Patel, K., Sharma, Y., & Shah, M. (2019). Transfer learning in crop disease detection: Survey and performance evaluation. International Journal of Scientific Research in Computer Science and Engineering, 7(5), 1–6.

[11] Negi, A., Kumar, K., & Chauhan, P. (2021). Deep neural network-based multi-class image classification for plant diseases. Agricultural informatics: automation using the IoT and machine learning, 117-129.
[12] Bhojani, S. H., & Bhatt, N. (2020). Wheat crop yield prediction using new activation functions in neural network. Neural Computing and Applications, 32(17), 13941-13951.
[13] Sewada, R., & Goyal, H. (2025). A Novel VGG-16 Adaptation for Multi-band Satellite Image Classification: Optimized Preprocessing and Class-specific Augmentation. Journal of Computational and Cognitive Engineering.
[14] Eunice, J., Popescu, D. E., Chowdary, M. K., & Hemanth, J. (2022). Deep learning-based leaf disease detection in crops using images for agricultural applications. Agronomy, 12(10), 2395.ResearchGate Link
[15] Reyad, M., Sarhan, A. M., & Arafa, M. (2023). A modified Adam algorithm for deep neural network optimization. Neural Computing and Applications, 35(23), 17095-17112.
[16] Zafar, A., Aamir, M., Mohd Nawi, N., Arshad, A., Riaz, S., Alruban, A., et al. (2022). A comparison of pooling methods for convolutional neural networks. Applied Sciences, 12(17), 8643.
[17] PlantVillage Dataset (Kaggle): https://www.kaggle.com/datasets/emmarex/plantdisease