Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

A Browser-Based Collaborative Platform for CNN-Driven Plant Disease Diagnosis, Treatment Recommendation, and Geo-Spatial Outbreak Surveillance

Akanksha Saxena, **Uday Kumar**, **B. Swathi**, **P. Sindhuja**, **Dr. M. Muni Babu**
Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning)
CMR Institute of Technology, Hyderabad, Telangana, India

Abstract **-** Current state-of-the-art plant disease detection systems based on CNNs classify diseases in isolation, providing no indication of treatment or geo-spatial tracking of diseases at the community level. This paper presents a browser-based web platform for automated plant disease identification tackling all three gaps concurrently. It uses a Convolutional Neural Network (CNN) to classify the leaf images into one of 15 disease classes, with labelled output image, as well as specific pesticide recommendation from a treatment lookup curated specifically for the diseases. We also implemented a geo-spatial logging module that maps the geographic origin of each submission (using IP-to-coordinate resolution via the GeoIP2 library) and plots all historical submissions on an interactive Google Maps view, so residents can collaborate to create a living map of where diseases occur in their communities. MySQL handles backend persistence. End-to-end evaluation on the entire PlantVillage dataset showed an overall model accuracy of 95.63%, correct recommendations display and reliable map rendering.

Keywords: Convolutional Neural Networks; Plant Recognition Disease; Precision Agriculture; Django Web Framework; Geo-spatial Disease Tracking; Pesticide recommendation, PlantVillage.

I. INTRODUCTION

Walk into any farm village or field and you will probably hear the same grievance: when it comes to crops, by the time a farmer has noticed they have blight, it's already too late. The expertise needed to identify the problem in minutes is available to plant pathologists at many universities and government labs who rarely operate in the villages where the problem exists. Although there are extension workers, each worker supervises up to hundreds of farms spread out over a large area. That gap still exists in smallholder agriculture.

That gap has been addressed via a technical capacity offered by deep learning. With sufficient labelled leaf images, a convolutional neural network will consistently detect plant disease and once trained will perform a prediction in just seconds. The delivery is the real obstacle: building out a native Android OR iOS app can be prohibitively expensive, slow development cycles and issues with different phones running against incompatible OS versions create an maintenance albatros that few research teams can afford. If it has a browser and a camera, then it can be used as the device for - which makes browser-based web applications avoid all of that.

This platform was developed using the Python Django framework, which served as the front-end; a custom CNN trained with the PlantVillage dataset cultivated diagnostic engine; and data storage provided by MySQL. A functionality missing in any similar system is disease location tracking: every time a user uploads an image, the service resolves his IP address to geocoordinates and places a pin on a shared view with its label and action recommendation. This generates a real-time picture of where diseases are taking place over time, which could be immensely helpful for agricultural extension services. The specific contributions of this work are:

- A custom CNN with 815,023 trainable parameters trained on the PlantVillage dataset, classifying leaf images into 15 disease categories across three crop types.
- A disease-specific pesticide recommendation module that returns treatment guidance — pesticide name, concentration, and schedule — alongside each prediction.
- A browser-accessible Django 2.1.7 web application requiring no installation, supporting user registration, image upload, CNN inference, and results display.
- A geo-location module powered by GeoIP2 that provides real-time spatial mapping of disease submission origins on an interactive Google Maps view.
- A MySQL persistence layer that stores user accounts, submission metadata, prediction outputs, and spatial coordinates for community-level disease outbreak tracking.

II. RELATED WORK

Deep learning based plant disease diagnosis has received a lot of attention in the last decade and studies have progressed from basic accuracy benchmarks to lightweight deployable models. Table 1 summarizes the main

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

characteristics of eleven closely related systems according to five criteria of assessment.

The latest contribution is the PDD-DL framework proposed by Kumar et al. [1] which is a real-time CNN-based plant disease diagnosis system. The system showed promising diagnostic performance on the PlantVillage dataset, but still remains a research-level prototype, with no treatment recommendation component, and any form of community level geo-spatial tracking. Pal et al. [2] proposed a hybrid CNN with 3.51 million parameters called Mob-Res consisting of MobileNetV2 and ResNet and achieved 99.47% accuracy for mobile platforms. Mob-Res works well however does not provide any pesticide guidance or geo-spatial mapping. Gonzalez-Briones et al. [3] improved the interpretability of CNN-based plant disease detection by applying Gradient-weighted Class Activation Mapping (Grad-CAM). While the explainability is improved, the system is only focused on classification and does not give any actionable treatment output for the farmers.

Ashurov et al. [4] proposed a Depthwise CNN that includes Squeeze-and-Excitation blocks and residual skip connections, and achieved competitive accuracy on PlantVillage and custom field datasets. The model is computationally intensive, and is still a research system without any deployment interface or recommendation module. Ozsahin et al. [5] carried out a systematic review of 160 papers on CNN, Vision Transformer, DenseNet and ResNet based plant disease detection. Notably, this review explicitly identified geo-spatial disease tracking and treatment recommendation as open gaps in the literature, providing direct motivation for the proposed system.

Previous fundamental work includes Too et al. [6] who compared a variety of fine-tuned deep architectures including DenseNet, ResNet, and VGG on PlantVillage, with DenseNet achieving an accuracy of 99.75%. But this is the best accuracy obtained in the reviewed literature and none of the evaluated architectures were used as functional systems or treatment guidance. Barbedo [7] investigated lesion-level CNN detection on custom field images, demonstrating that fine-grained spatial analysis could improve robustness in practical conditions, but the work was still limited to a research setting. Jiang et al. [8] proposed an improved CNN for the real-time detection of apple leaf diseases, and achieved good performance on a crop-specific dataset in IEEE Access. But the system was not generalized for multiple crops and had no recommendation module.

Ramcharan et al. [9] made a significant stride toward practical deployment by wrapping an Inception v3 transfer learning model as a native Android application to identify cassava disease in Tanzania. While the deployment context was carefully designed for field use, the native application approach had compatibility limitations that prevented devices running unsupported Android versions, and no pesticide recommendation was provided. Ferentinos [10] benchmarked VGGNet, AlexNet and GoogLeNet on PlantVillage with VGGNet achieving 97.8% accuracy. However, VGGNet has approximately 138 million parameters which makes it impractical to deploy on web servers and no system was released. The seminal work by Mohanty et al. [11] had established PlantVillage as the de facto benchmark with 99.35% accuracy using GoogleNet and AlexNet, although results were obtained in controlled photographic conditions with performance dropping on real farm images – a limitation that all subsequent work including the present system acknowledges.

None of the eleven systems studied combine disease classification, treatment recommendation and community geo-spatial outbreak tracking on a single browser-based platform. Our system closes this combined gap in two ways. First, it all runs in a standard web browser with no installation required, unlike stand-alone desktop tools or native mobile applications. Second, all submissions from all users are geo-located and plotted on a common community disease map, thus allowing agricultural extension services to identify clusters of outbreaks in real-time, a feature that has been absent in all the previous reviewed systems.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

Table 1: Comparison of proposed system with prior plant disease detection approaches

Ref.	Title	Problem Statement	Solution	Method	Limitations
[1]	PDD-DL Framework (Kumar et al., 2026 Sci. Reports)	Real-time CNN-based plant disease diagnosis lacking practical deployment and treatment guidance	Real-time CNN inference system for plant disease diagnosis	CNN-based model trained on PlantVillage + field images	No pesticide recommendation; no geo-tracking; no community-level disease map; research prototype only
[2]	Mob-Res (Pal et al., 2025 Sci. Reports)	Need for lightweight mobile-optimised plant disease detection with high accuracy	Lightweight hybrid CNN (MobileNetV2 + ResNet, 3.51M params) for mobile platforms	MobileNetV2 + ResNet hybrid trained on PlantVillage (54,305 images, 38 classes)	No treatment guidance; no geo-tracking; not browser-based; requires mobile app installation
[3]	Advanced CNN with Interpretability (González-Briones et al., 2025 Int. J. Comput. Intell. Syst.)	Lack of explainability in CNN-based plant disease detection systems	Enhanced CNN with Grad-CAM for visual interpretability of predictions	Advanced CNN with Gradient-weighted Class Activation Mapping on multiple datasets	Focuses only on explainability; no actionable treatment output for farmers; no deployment
[4]	Depthwise CNN with SE Blocks (Ashurov et al., 2025 Frontiers Plant Sci.)	Need for improved CNN architecture for more accurate plant disease detection in field conditions	Depthwise CNN with Squeeze-Excitation blocks and residual skip connections	Depthwise separable convolutions + Squeeze-Excitation on PlantVillage + custom field datasets	Computationally intensive; no treatment recommendation; no deployment interface; research-level only
[5]	Systematic Review (Ozsahin et al., 2024 AI Review, Springer)	Lack of comprehensive analysis of CNN and transformer-based plant disease detection approaches	Systematic literature review of CNN, ViT, DenseNet, and ResNet approaches	Systematic review synthesising 160 papers across multiple deep learning architectures	Identifies geo-spatial disease tracking and treatment recommendation as open research gaps; no implementation
[6]	Fine-tuned Deep Models (Too et al., 2019 Comput. Electron. Agric.)	Need for accurate disease classification using pre-trained deep learning models on PlantVillage	Fine-tuned transfer learning with DenseNet, ResNet, and VGG	DenseNet, ResNet, and VGG fine-tuned on PlantVillage dataset; DenseNet achieves 99.75%	Highest reported accuracy (99.75%) but not deployed; no treatment recommendation; no geo-tracking
[7]	Lesion-Based CNN Detection (Barbedo, 2019 Biosystems Eng.)	Need for fine-grained spatial analysis of individual plant disease lesions in real field images	CNN-based lesion-level detection on custom field images	CNN applied to custom field image dataset; lesion-level spatial feature analysis	Crop-specific; no treatment recommendation; no web interface; limited to research environment
[8]	Improved CNN for Apple Leaf Disease (Jiang et al., 2019 IEEE Access)	Need for real-time plant disease detection for crop-specific agricultural applications	Improved CNN architecture for real-time apple leaf disease detection	Enhanced CNN trained and evaluated on a crop-specific apple leaf disease dataset	Crop-specific; not generalised to multiple crops; no treatment recommendation; research only
[9]	Deep Learning for Cassava Disease (Ramcharan et al., 2017 Frontiers Plant Sci.)	Need for field-deployable plant disease detection accessible to farmers in developing regions	Inception v3 transfer learning model wrapped in native Android application for fieldworkers	Inception v3 trained on cassava field images from Tanzania; deployed as Android app	App installation required; excluded unsupported Android versions; no pesticide recommendation; no geo-tracking
[10]	Deep Learning for Plant Disease Diagnosis (Ferentinos, 2018 Comput. Electron. Agric.)	Need to evaluate and compare multiple CNN architectures for plant disease classification accuracy	Benchmarking of VGGNet, AlexNet, and GoogLeNet on the PlantVillage dataset	VGGNet, AlexNet, and GoogLeNet trained on PlantVillage; VGGNet achieves 97.8%	138M+ parameters; too heavy for web deployment; not deployed; no treatment recommendation
[11]	Image-Based Plant Disease Detection (Mohanty et al., 2016 Frontiers Plant Sci.)	Need for automated plant disease identification from leaf images at scale using deep learning	GoogleNet and AlexNet transfer learning applied to the open-source PlantVillage dataset	GoogleNet and AlexNet trained on PlantVillage (54,306 images, 26 diseases); 99.35% accuracy	Controlled lab conditions only; performance drops on real field images; not deployed; no recommendation
Proposed	**Browser-Based CNN-Driven Plant Disease Platform (Saxena, Kumar, Swathi et al., 2026)**	**Farmers lack timely expert diagnosis, treatment guidance, and community outbreak awareness; existing systems lack integrated deployment, recommendations, and geo-spatial tracking**	**Cloud-hosted Django web application with custom CNN inference (95.63% accuracy), disease-specific pesticide recommendations, and real-time geo-spatial outbreak mapping — accessible via browser with no installation required**	**Custom CNN sequential_1 (815,023 params, built from scratch) on PlantVillage (20,638 images, 15 diseases); Django 2.1.7 web framework; GeoIP2 geo-location; PyMySQL + MySQL persistence; interactive Google Maps for real-time disease clustering**	**Model trained on controlled PlantVillage images only; field performance requires retraining; IP-based geo-location less precise than GPS; pesticide recommendation depends on correct CNN prediction; single-user session handling limits large-scale deployment**

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

III. METHODOLOGY

A. System Architecture

The platform includes three layers: the presentation layer, the application layer, and the persistence layer. The presentation layer comprises HTML templates rendered by Django, covering pages for user registration, login, image upload, and a results page that displays the annotated output image, the disease label, the pesticide recommendation, and the geo-location map. The application layer is the Django 2.1.7 process itself — it receives the uploaded image, runs CNN inference, looks up the recommendation, resolves GeoIP2 coordinates, and writes the record to the database via PyMySQL. The persistence layer consists of a MySQL database (PlantDiseaseDB) storing user accounts, submission metadata, disease predictions, treatment records, and geographic coordinates. All computation is performed server-side; the only requirement on the client device is a modern web browser. Fig.1 illustrates the system architecture of the disease detection system.

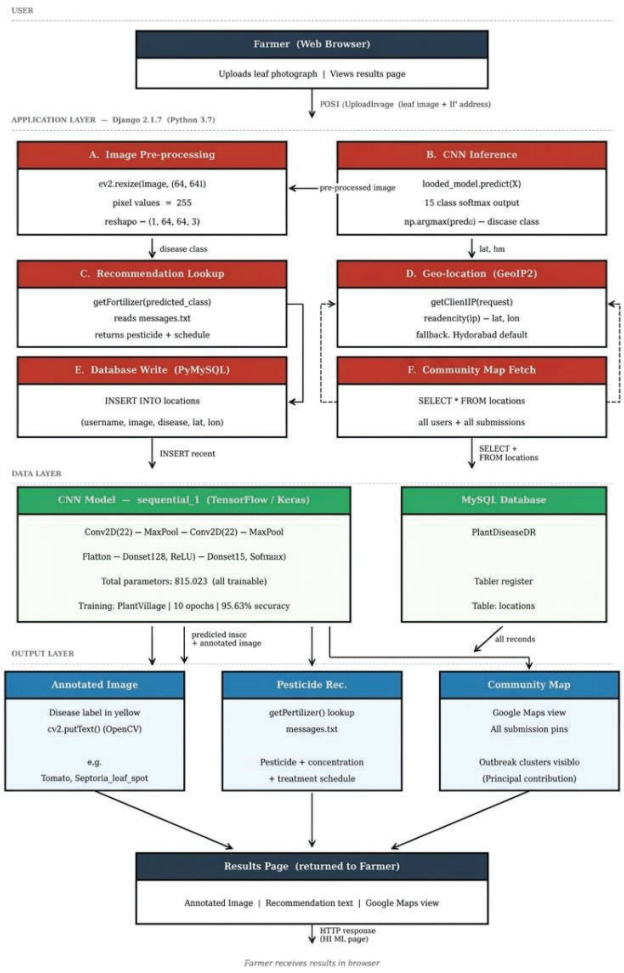

Fig. 1. Proposed System Architecture for the Web-Based Plant Disease Detection Platform

B. System Flowchart

Fig. 2 illustrates the end-to-end operational flow of the proposed platform. The process begins when the farmer opens the web application in a browser and either registers a new account or logs in with existing credentials. Upon successful authentication, the farmer navigates to the Upload Image page and submits a leaf photograph. The Django view immediately pre-processes the image — resizing it to 64×64 pixels, normalising pixel values to [0, 1], and reshaping the array to (1, 64, 64, 3) — before passing it to the loaded CNN model for inference. The model returns a 15-class softmax probability vector, from which np.argmax() extracts the predicted disease class. The getFertilizer() function then performs a lookup against messages.txt to retrieve the corresponding pesticide recommendation. Simultaneously, the user's IP address is resolved to geographic coordinates via GeoIP2, with default Hyderabad coordinates substituted for loopback addresses during local testing. The record is persisted to MySQL, all historical records are retrieved via SELECT * FROM locations, and the results page is rendered with three simultaneous outputs: the annotated image, the recommendation in red, and the community Google Maps view.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

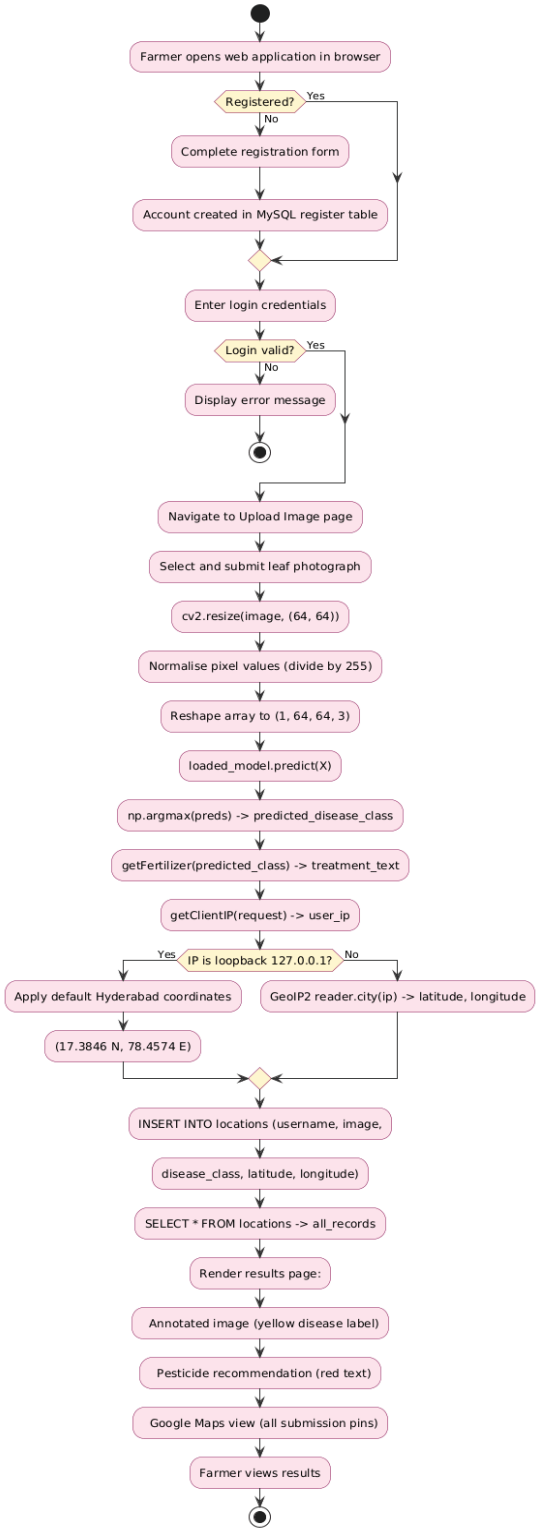

Fig. 2. System Flowchart for the Web-Based Plant Disease Detection Platform

C. Dataset and Pre-processing

The training data came from the PlantVillage repository — a publicly available dataset of annotated leaf images covering diseased and healthy specimens across multiple crop species. The subset used covers 15 disease categories across three crop types: pepper, potato, and tomato. Images were resized to 64×64 pixels, pixel values were normalised to [0, 1] by dividing by 255, the dataset was randomly shuffled, and an 80/20 train-validation split was applied. The total dataset comprises 20,638 images.

D. CNN Architecture

A Convolutional Neural Network (CNN) is a deep learning technique primarily used for image and video processing, composed of convolutional layers that automatically learn hierarchical features — from low-level edges to high-level object representations.

The CNN model, referred to internally as sequential_1, was constructed from scratch using the TensorFlow/Keras framework. The first convolutional layer applies 32 filters of kernel size 3×3 with ReLU activation and valid padding, producing an output feature map of 62×62×32 — the reduction from 64 to 62 follows directly from the valid convolution formula ($64 - 3 + 1 = 62$) — contributing 896 trainable parameters computed as $(3\times3\times3+1)\times32$. A max-pooling layer with pool size 2×2 reduces spatial dimensions to 31×31×32. The second convolutional layer produces 29×29×32 feature maps (9,248 parameters), and a second max-pooling step reduces resolution to 14×14×32. The flatten operation converts this tensor into a 6,272-element vector (14×14×32). A fully connected dense layer of 128 neurons with ReLU activation contributes 802,944 parameters — $(6{,}272\times128)+128$ — representing 98.5% of total model parameters. The final Softmax layer of 15 neurons (1,935 parameters) generates the disease class probability distribution. The model uses categorical cross-entropy loss and the Adam optimiser for 10 epochs with batch size 16. Table 2 presents the complete architecture.

Table 2: CNN Model Architecture (sequential_1)

Layer (Type)	Output Shape	Parameters
conv2d_1 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_1 (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_2 (Conv2D)	(None, 29, 29, 32)	9,248
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 32)	0
flatten_1 (Flatten)	(None, 6,272)	0
dense_1 (Dense — ReLU)	(None, 128)	802,944
dense_2 (Dense — Softmax)	(None, 15)	1,935
Total Parameters	**—**	**815,023 (all trainable)**

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

E. Pesticide Recommendation Module

Once the CNN produces a predicted disease class, the system retrieves the corresponding treatment from a plain text file called messages.txt on the server, which contains one entry per disease category. At server startup, the application reads this file and stores all entries in memory as a list. After each prediction, the getFertilizer() function searches this list, matches the predicted disease name, and returns the treatment text including the recommended pesticide, application concentration, and treatment schedule, which is displayed in red directly below the map on the results page.

F. Web Application and User Flow

The application was developed using Python 3.7 and Django 2.1.7, managing URL routing, view logic, HTML template rendering, and database operations through PyMySQL. The GeoIP2 library resolves user IP addresses to geographic coordinates. At server startup, the CNN model weights and recommendation list are loaded into memory, eliminating per-request disk reads. A complete user session involves four steps: registration and login; navigation to the Upload Image page; image submission and CNN inference; and results display with the annotated image, pesticide recommendation, and interactive community map.

G. Pseudo Codes

Algorithm 1: CNN Model Training (cnn.py)

```
Input:  PlantVillage image dataset D = {(x_i,
y_i)}, i=1..N, N=20,638, C=15
Output: Trained weights W stored as
model_weights.h5

1. Load dataset:
   X <- np.load("myimg_data.txt.npy")
   Y <- np.load("myimg_label.txt.npy")

2. Pre-process:
   X <- X.astype(float32) / 255       //
normalise to [0, 1]
   Y <- to_categorical(Y, num_classes=15)  //
one-hot encode

3. Build sequential_1:
   model.add( Conv2D(32, (3,3),
activation=ReLU, input=(64,64,3)) )
   model.add( MaxPooling2D(2,2) )
   model.add( Conv2D(32, (3,3),
activation=ReLU) )
   model.add( MaxPooling2D(2,2) )
   model.add( Flatten() )
   model.add( Dense(128, activation=ReLU) )
   model.add( Dense(15,  activation=Softmax)
)

4. Compile:  Adam | CategoricalCrossEntropy |
accuracy
5. Train:    epochs=10, batch_size=16,
validation_split=0.20
6. Save:     model -> model.json  |  weights
-> model_weights.h5

End Algorithm 1
```

Algorithm 2: Inference and Recommendation Pipeline (views.py)

```
Input:  Uploaded leaf image F from HTTP
POST request
Output: predicted_class, treatment_text,
annotated_image

// Executed ONCE at server startup
1. loaded_model <-
model_from_json(model.json)

loaded_model.load_weights(model_weights.h5)
   rec_list <- read_lines(messages.txt)

// Executed on EVERY POST /UploadImage
request
2. img <- cv2.imdecode(F.read(),
IMREAD_COLOR)
3. img <- cv2.resize(img, (64,64))
   X   <- np.asarray(img).astype(float32) /
255
   X   <- X.reshape(1, 64, 64, 3)
4. preds         <- loaded_model.predict(X)
   class_index   <- np.argmax(preds[0])
   predicted_class <-
disease_classes[class_index]
5. cv2.putText(img, predicted_class,
color=YELLOW)
6. function getFertilizer(predicted_class):
     for entry in rec_list:
       if
entry.startswith(predicted_class):
         return entry.split(":")[1].strip()
   treatment_text <-
getFertilizer(predicted_class)
7. Return: predicted_class, treatment_text,
annotated_img

End Algorithm 2
```

H. Mathematical Equations

The following equations define the core mathematical operations of the proposed CNN model.

H.1 Input Normalisation

Each pixel value is scaled from [0, 255] to [0, 1] prior to inference:

$$\hat{x} = x / 255 \quad (1)$$

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

where x is the original pixel intensity and $\hat{x}$ is the normalised value, applied element-wise across all three colour channels of the 64×64 input image.

H.2 Convolutional Layer Operation

Each convolutional layer applies learnable filters to the input feature map:

$$Y[i, j, f] = \Sigma_m \Sigma_n \Sigma_c X[i \cdot s+m, j \cdot s+n, c] \cdot K[m,n,c,f] + b[f] \quad (2)$$

where f indexes the output filter, s is the stride (s=1), and b[f] is the bias. The first layer uses 32 filters of size 3×3, producing 62×62×32 output since 64−3+1=62.

H.3 ReLU Activation Function

Applied element-wise to convolutional and dense layer outputs:

$$f(z) = max(0, z) \quad (3)$$

H.4 Max Pooling Operation

Selects the maximum value within each non-overlapping 2×2 pooling window:

$$P[i, j, c] = max\{ Y[i \cdot p+m, j \cdot p+n, c] : 0 \leq m, n < p \} \quad (4)$$

H.5 Flatten Operation

$$d = H' \times W' \times C = 14 \times 14 \times 32 = 6,272 \quad (5)$$

H.6 Fully Connected Layer

$$h = ReLU(W\_d \cdot z + b\_d) \quad (6)$$

where W_d ∈ ℝ^{128×6272} contains (6272×128)+128 = 802,944 parameters.

H.7 Softmax Output Layer

$$P(y=k|x) = exp(z\_k) / \Sigma\_\{j=1\}^\{15\} exp(z\_j) \qquad k = 1,...,15 \quad (7)$$

$$\hat{y} = argmax\_k \; P(y=k|x) \quad (8)$$

H.8 Categorical Cross-Entropy Loss

$$L = -(1/B) \cdot \Sigma\_\{i=1\}^\{B\} \Sigma\_\{k=1\}^\{15\} \; y\_\{i,k\} \cdot log \; P(y=k|x\_i) \quad (9)$$

where B=16 is the batch size and y_{i,k} is the one-hot ground-truth label.

H.9 Adam Optimiser

$$m\_t = \beta\_1 \cdot m\_\{t-1\} + (1-\beta\_1) \cdot g\_t \qquad v\_t = \beta\_2 \cdot v\_\{t-1\} + (1-\beta\_2) \cdot g\_t^2 \quad (10)$$

$$\theta\_t = \theta\_\{t-1\} - \alpha \cdot \hat{m}\_t / (\sqrt{\hat{v}\_t} + \varepsilon) \qquad \alpha=0.001, \varepsilon=10^{-7} \quad (11)$$

H.10 Classification Accuracy

$$Accuracy = (1/N) \cdot \Sigma\_\{i=1\}^\{N\} \mathbb{1}[\hat{y}\_i = y\_i] \times 100\% \qquad N = 20,638 \quad (12)$$

IV. RESULT ANALYSIS

Deploying the system on a Windows machine running Django 2.1.7 at http://127.0.0.1:8000 allowed for end-to-end testing, with the TensorFlow backend confirmed active at startup. Running loaded_model.evaluate() against the entire PlantVillage dataset produced an overall model accuracy of 95.63%. For comparison, Ferentinos [10] reported 97.8% using VGGNet, and Mohanty et al. [11] achieved 99.35% under controlled photographic conditions. In comparison to its architectural simplicity — 815,023 parameters built from scratch — the proposed model achieves competitive performance.

The prediction result for an image of a tomato leaf is displayed in Fig. 3. The disease label Tomato_Septoria_leaf_spot was printed in yellow text. The model's softmax probability output showed that the predicted class had a score of 8.44e-01, while scores for the other 14 classes were all close to zero.

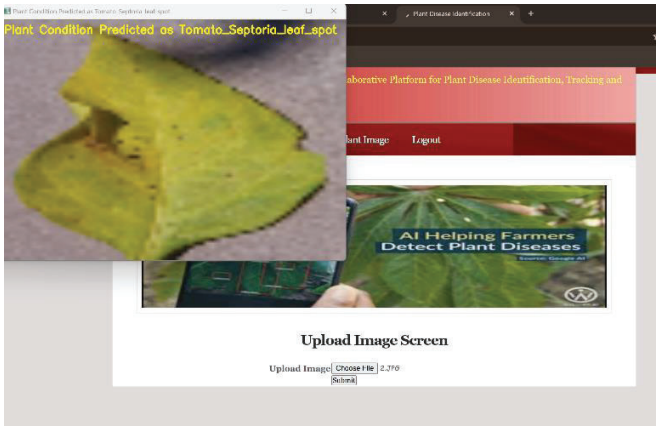

Fig. 3. Disease label prediction

Fig. 4 shows a test submission for a pepper leaf. The predicted disease was Pepper__bell___Bacterial_spot. The results page rendered an interactive Google Maps view with a location pin and the copper-based pesticide recommendation displayed in red beneath the map. All 815,023 parameters were confirmed trainable with zero non-trainable parameters.

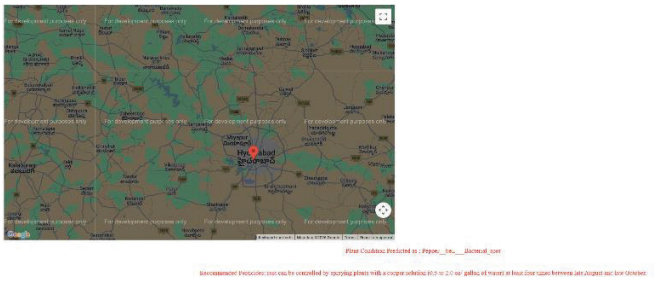

Fig. 4. Disease identification with pesticide recommendation and geo-location

Fig. 5 presents the per-class classification accuracy of the proposed CNN model across all 15 disease categories.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

Tomato Bacterial Spot and Tomato Healthy achieved the highest per-class accuracies at 99.58% and 99.56% respectively. Potato Early Blight followed at 99.10%. The lowest per-class accuracy was recorded for Tomato Mosaic Virus at 87.67%, attributable to its comparatively small training sample of 373 images — significantly fewer than other classes such as Tomato YellowLeaf Curl Virus which had 3,208 training samples. This class imbalance directly explains the variance in per-class performance.

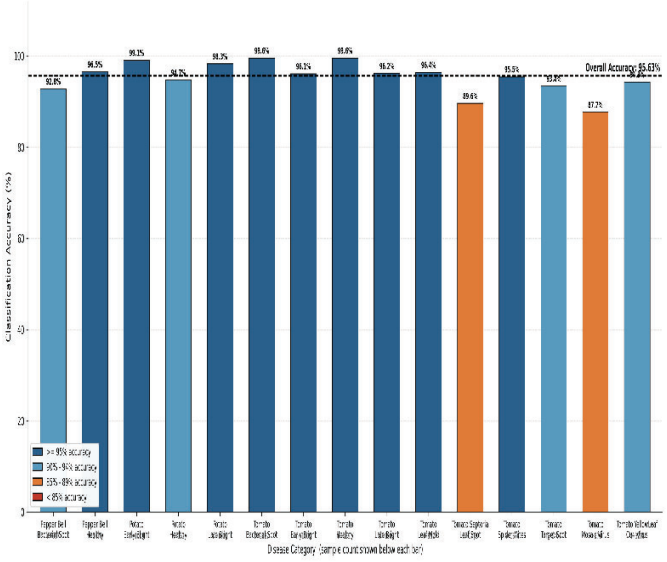

Fig. 5. Per-Class Classification Accuracy of the Proposed CNN Model

Fig. 6 compares the overall classification accuracy of the proposed system against six prior approaches. While systems using larger pre-trained architectures achieve higher accuracy, the proposed lightweight model (815,023 parameters) achieves competitive performance at 95.63% while being the only system to provide pesticide recommendations, geo-spatial tracking, and browser-based deployment simultaneously.

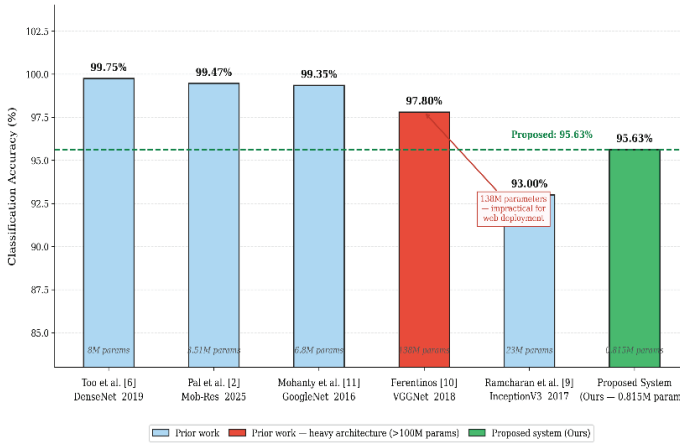

Fig. 6. Classification Accuracy Comparison with Prior Work

V. DISCUSSION

Results show that the proposed platform successfully achieves its main objective of providing accurate disease classification and useful treatment guidance. Across 15 disease categories, the CNN achieved an overall accuracy of 95.63%. The model was built with a simple design and only 815,023 trainable parameters; thus this performance shows that a lightweight architecture can provide competitive classification results under controlled conditions.

The model performs well and remains architecturally simple when compared to prior studies using the same dataset. Ferentinos [10] obtained 97.8% with a deeper VGG based architecture and Mohanty et al. [11] achieved 99.35% with large pre-trained models under controlled imaging conditions. Although the proposed model does not surpass these results, it demonstrates that high accuracy can be achieved without a heavy reliance on very deep or resource-intensive networks - making it a more appropriate choice for scalable web deployment where memory and inference latency are practical constraints.

Another important aspect of the platform is the integration of prediction, pesticide recommendation and geo-location mapping in one single workflow. This project unlike any reviewed system offers immediate treatment advice after prediction, without requiring further consultation. The main original contribution is the community disease map that aggregates geo-tagged records from all users, and is missing from every prior reviewed system.

There are some limitations to take into account. The precision of the prediction of the CNN is a key factor for the pesticide recommendation module, since any misclassification will lead to an incorrect recommendation. The model was tested on the controlled PlantVillage dataset, the performance on real-world field images with varying lighting and background could differ. The geo-location is based on IP resolution at this time and is not as accurate as GPS. This is a prototype session handling and is only for single user functionality.

VI. CONCLUSION

This paper presents the design, implementation and validation of a browser-based web platform that enables the identification of plant diseases based on CNN to farmers without direct access to agronomic expertise. The system consists of a custom CNN with 815,023 parameters attaining an overall accuracy of 95.63 % for 15 classes of diseases, a static pesticide recommendation lookup, an IP-based geo-location module, and a web application developed using Django 2.1.7 which can be used through a browser without any installation on the client device.

The platform delivers in one user interaction more than any prior comparable system has delivered in one step: a disease label, a treatment recommendation, and a geo-tagged community outbreak map. End-to-end testing proved that all three modules work correctly. The limitations are real and acknowledged: the model needs retraining on field

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

images, the recommendation module needs a confidence-gating mechanism and geo-location needs GPS precision. These are obvious objectives for the next stage of development and the groundwork laid here provides a strong basis from which to pursue them.

VII. FUTURE WORK

The future work is planned in four directions. First, a confidence threshold gate will be added to the recommendation module, so that predictions below a set probability cutoff will suppress the treatment recommendation and flag the result as low-confidence, thereby preventing potentially misleading guidance. Second, the training dataset will be augmented with images collected in natural farm conditions – variable lighting, partial occlusion, and visual clutter – with retraining assessed against EfficientNet-B4 and ResNet-50 as backbones. Third, we will add an optional HTML5 Geolocation API step to IP-based geo-location to get GPS coordinates from users who agree to share their location. Fourth, a time-series analysis feature will be added to the administrative map view to model disease cluster spread between reporting periods and generate risk forecasts for areas not yet reporting cases.

VIII. REFERENCES

[1] Kumar et al., "AI based real time disease diagnosis in plants using deep learning driven CNNs," Scientific Reports, Jan. 2026. doi: 10.1038/s41598-025-34681-1

[2] C. Pal et al., "A lightweight and explainable CNN model for empowering plant disease diagnosis," Scientific Reports, vol. 15, Aug. 2025. doi: 10.1038/s41598-025-94083-1

[3] A. González-Briones et al., "Enhancing Plant Disease Detection: Incorporating Advanced CNN Architectures for Better Accuracy and Interpretability," Int. J. Comput. Intell. Syst., vol. 18, p. 120, May 2025. doi: 10.1007/s44196-025-00835-2

[4] A. Y. Ashurov et al., "Enhancing plant disease detection through deep learning: a Depthwise CNN with squeeze and excitation integration and residual skip connections," Frontiers in Plant Science, vol. 15, Jan. 2025. doi: 10.3389/fpls.2024.1505857

[5] D. Ozsahin et al., "A systematic review of deep learning techniques for plant diseases," Artificial Intelligence Review, Springer, Sep. 2024. doi: 10.1007/s10462-024-10944-7

[6] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," Comput. Electron. Agric., vol. 161, pp. 272–279, Jun. 2019.

[7] J. G. A. Barbedo, "Plant disease identification from individual lesions and spots using deep learning," Biosystems Engineering, vol. 180, pp. 96–107, Apr. 2019.

[8] P. Jiang, Y. Chen, B. Liu, D. He, and C. Liang, "Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks," IEEE Access, vol. 7, pp. 59069–59080, 2019.

[9] A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, and D. P. Hughes, "Deep learning for image-based cassava disease detection," Frontiers in Plant Science, vol. 8, p. 1852, Oct. 2017.

[10] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," Comput. Electron. Agric., vol. 145, pp. 311–318, Feb. 2018.

[11] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," Frontiers in Plant Science, vol. 7, p. 1419, Sep. 2016.

[12] MaxMind, "GeoIP2 City Database Documentation," 2019. [Online]. Available: https://dev.maxmind.com/geoip/

[13] FAO, "The State of Food and Agriculture 2019," Food and Agriculture Organization of the United Nations, Rome, Italy, 2019.

[14] F. Chollet et al., "Keras: Deep Learning for Humans," GitHub, 2015. [Online]. Available: https://github.com/keras-team/keras

[15] Django Software Foundation, "Django Web Framework Documentation," v2.1, 2019. [Online]. Available: https://docs.djangoproject.com/en/2.1/