International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Krishi Mitra: A Digital Farmer Assistant for Integrated Agricultural Advisory, Crop Health Support, and Farm Management

Dipanshu Suryawanshi[1], Himanshu Nagle[2], Jitendra Nagvanshi[3], Prof. Satish Chadokar[4]

Students[1,2,3], Assistant Professor[4]
[1,2,3,4]Department of Computer Science and Engineering, Shri Balaji Institute of Technology and Management, Betul, Madhya Pradesh, India

Abstract:
Crop diseases reduce yield and farmer income, a wrong crop choice adds to the risk, and most smallholder farmers cannot reach an agronomist in time. Smartphone assistants that combine leaf-image diagnosis, crop recommendation, treatment advice and market information can close part of this gap. This paper reviews recent research (2021 to 2026) that supports such an assistant. It covers deep-learning plant disease detection, lightweight and attention-enhanced networks for mobile deployment, machine-learning crop recommendation, pesticide recommendation, crop price forecasting and explainability. Four recent studies are examined in detail: a hybrid multimodal framework for disease detection, pesticide advice and LSTM price forecasting [1]; a MobileNetV2 network with one late CBAM attention block and measured TensorFlow Lite cost [2]; a MobileNetV2 network with a Modified Lite Attention module for chilli leaves [3]; and a cloud-connected Android crop recommendation system that compares 13 classifiers with SMOTE [43]. The studies are compared on datasets, architectures, reported accuracy and computational cost. Benchmark accuracy now exceeds 95% for many models, attention modules add small gains on controlled datasets, and perfect scores on small tabular datasets need careful interpretation. Deployment cost is reported far less often than accuracy, and field validation, multi-run statistics and integrated farmer-facing systems remain rare. These gaps define the design needs of integrated, multilingual assistants such as Krishi Mitra.

Keywords: Plant Disease Detection; Deep Learning; Lightweight Models; Crop Recommendation; Price Forecasting; Precision Agriculture.

1. Introduction

1.1 Background

Agriculture supports food security and rural livelihoods, especially in developing countries. Plant diseases caused by fungi, bacteria, viruses and pests lower both the quantity and quality of crops, and they cause large economic losses [1], [10]. Traditionally, farmers and agricultural experts identify diseases by looking at the plant. This is slow and depends on experience. Symptoms are often noticed only in the later stages

of the disease cycle, when treatment is less effective [5]. A survey of AI and IoT methods notes that manual identification is lengthy, inaccurate and practical only on small farms [10].

Deep learning has changed this situation. Convolutional neural networks (CNNs) learn colour, texture and shape features directly from leaf images, so they no longer depend on hand-designed features. Reviews published between 2022 and 2025 agree that CNN-based models now dominate image-based plant disease detection, while also warning about data diversity, generalisation and deployment limits [4], [5], [6], [7], [8], [9], [11], [12].

1.2 Importance of the Topic

A farmer needs more than a disease name. The useful questions are: what is wrong with this leaf, what should I spray and how much, will the weather allow it, and where and when should I sell the crop? Most research answers only the first question. Recent work has begun to join disease detection with treatment recommendation and price forecasting in one framework [1], and to shrink models so that they can run on phones and low-cost devices [2], [3], [13]. Reviewing this work is a necessary first step for designing a complete digital farmer assistant.

1.3 Objectives of the Review

- To summarise recent deep-learning research on plant disease detection, with attention to lightweight and attention-enhanced models.
- To review work on pesticide and treatment recommendation and on crop price forecasting that can be attached to a disease detector.
- To study in detail four recent papers [1], [2], [3], [43] and compare their datasets, methods, results and limits.
- To identify research gaps, challenges and future directions for integrated, mobile, multilingual farmer assistants.

1.4 Scope of the Review

Only studies published in 2021 or later are cited. The review is built around four recent studies [1], [2], [3], [43], and the wider literature they cite is used to place them in context. Where a statement about a secondary study is made, it follows the description given in the primary papers, and readers should consult the original article for full detail. Reported accuracies come from different datasets, splits and hardware, so they are compared with caution and are never treated as a single leaderboard.

2. Literature Review

2.1 Deep Learning for Plant Disease Detection: Survey Evidence

Several recent surveys describe how the field has developed. Upadhyay et al. review deep-learning computer vision for plant disease detection, covering techniques, models and trends in precision agriculture [4]. Ngugi et al. give a comprehensive review of computational deep learning for crop disease detection [5], and Dolatabadian et al. survey image-based crop disease detection with machine learning [6]. Zhao et al. [7], Nyawose et al. [8] and Sajitha et al. [9] review plant disease classification for industrial and real farming settings, and Jackulin and Murugavalli [11] and Omaye et al. [12] compare machine-learning approaches, including crop-specific comparisons for apple, cassava, cotton and potato.

These reviews report a consistent picture. CNN architectures such as AlexNet, VGG, ResNet, MobileNet and EfficientNet give accuracy above 90% on public datasets, and are reported to outperform traditional image-processing methods [1]. The same reviews list the same obstacles: illumination changes, similar symptoms across diseases, overfitting, dataset bias, weak generalisation and implementation limits [2], [7], [8], [9]. The AI and IoT survey by Orchi et al. adds that automated systems must be practical for small farms [10].

2.2 A Hybrid Multimodal Framework for Detection, Treatment and Price Forecasting

Sable et al. [1] propose a modular framework with three components: image-based disease classification, a knowledge-based pesticide recommendation module, and an LSTM model that forecasts crop prices. The aim is a single decision-support system for the farmer.

Data. Two public datasets are used. The cotton dataset holds 11,527 images in 14 categories, and the tomato dataset holds 37,363 images in 10 categories, one of which is healthy. Images are resized to 224 × 224 pixels and normalised. The data are split with stratified sampling: 20% for testing, and 20% of the remaining training data for validation. Augmentation includes rotation, zoom, shear, shifts and flips, and is applied only to training images.

Models. Four classifiers are compared: ResNet50 and EfficientNetB0 with ImageNet transfer learning, a custom three-block CNN with 32, 64 and 128 filters, and YOLOv8. The training settings are Adam, a batch size of 32 and at most 50 epochs, with early stopping, learning-rate reduction and checkpointing. The backend is built with FastAPI and the front end with React and Tailwind CSS.

Results. On tomato images EfficientNetB0 reaches 99.95% accuracy, ResNet50 99.21%, the custom CNN 97.66% and YOLOv8 95.27%. On cotton images ResNet50 leads with 98.87%, followed by the custom CNN (98.51%), YOLOv8 (98.41%) and EfficientNetB0 (97.04%). The authors note that YOLOv8 is designed for detection and localisation rather than whole-image classification, which may explain its lower tomato result.

Treatment and price modules. After classification, the system retrieves a structured report from a knowledge base: causes and symptoms, prevention methods, recommended pesticides with application rates, and eco-friendly alternatives. An LSTM trained on market prices of nine crops from January 2020 to September 2025 is used for price forecasting. The paper reports a mean absolute error of about 4.35% in its results and conclusion, while its abstract mentions 3.2% and 4.1%, so readers should treat the exact figure with care.

Explainability. SHAP is used to show which image regions drive a classification, and LIME is proposed for the recommendation and price models. The qualitative SHAP maps in the paper highlight lesion areas on cotton wilt, cotton boll rot, tomato early blight and tomato yellow leaf curl virus.

Stated limits. The experiments use benchmark datasets collected under fairly controlled conditions, no independent field data are used, and the multimodal fusion with environmental data and the price module are described more as a framework than as a fully evaluated pipeline. The comparison with earlier cotton and tomato studies in the paper (Table 2) mixes different protocols [1], [24], [25], [40].

2.3 Late Attention on a Compact Backbone: MobileNetV2 with CBAM

Udoh et al. [2] ask a focused question: can one attention block, placed near the classifier, improve MobileNetV2 without adding much cost? They insert a single Convolutional Block Attention Module (CBAM) between the last MobileNetV2 feature map and global average pooling. The authors state that

the idea of combining these two components is not new; the contribution is the late placement and a careful measurement of its cost.

Data and protocol. The study uses 54,306 PlantVillage images in 38 classes, split 70% / 15% / 15% into 37,998 training, 8,146 validation and 8,162 test images. A hash audit found 10 exact-duplicate groups that cross the splits, seven of which affect test images. The authors report results with and without these images and show that the effect is below 0.01 percentage points. The final 30 backbone layers are fine-tuned, training runs for up to 30 epochs with Adam at a learning rate of 1e-5, and the compared models are InceptionV3, MobileNetV2, MobileNetV2 with CBAM, and Xception.

Accuracy. In float form, MobileNetV2 scores 96.78%, MobileNetV2 with CBAM 97.17%, InceptionV3 96.89% and Xception 98.30%. After dynamic-range quantisation to TensorFlow Lite, the CBAM model keeps 96.70% accuracy. A paired McNemar test on the converted models gives a 0.64 percentage-point advantage for CBAM (95% confidence interval 0.31 to 0.96).

Cost. CBAM raises parameters from 3.61 M to 4.02 M, but raises computation by only about 0.3% (0.602 to 0.604 GFLOPs). The TensorFlow Lite file grows from 3.68 MiB to 4.07 MiB, and median CPU latency stays at about 45 ms for both on an Intel i7-11800H with XNNPACK. Xception is the most accurate but its TensorFlow Lite file is 22.57 MiB and its median latency is 200.15 ms.

Interpretation and limits. Grad-CAM examples show CBAM correcting a baseline error and focusing on lesions, but also a shared failure where both models predict the same wrong class. The authors are explicit about limits: controlled-background data, a single training run per model, no placement ablation, and no phone, Raspberry Pi or energy measurements. They describe their result as an accuracy-cost trade-off under measured conditions, not as proof of field readiness.

2.4 Multi-Level Lightweight Attention: MobileNetV2 with MLA

Rajendrakumar and Rajashekarappa [3] design a Modified Lite Attention (MLA) module and insert it at several depths of MobileNetV2 (after layers 2, 7 and 14). MLA has three branches. A channel gate uses global average pooling and a small two-layer network. A spatial gate uses a depthwise 3 × 3 convolution and a 1 × 1 convolution. An edge prior uses fixed Sobel and Laplacian filters to highlight lesion outlines and veins, and adds no learnable parameters. The three branches are fused with learnable weights and applied with residual scaling, $Y = X(1 + sM)$, where the scale s starts at zero so that the pretrained backbone is not disturbed early in training.

Data. The main experiments use 4,432 chilli leaf images in five classes: healthy, leaf spot, whitefly, yellowish and leaf curl. The split is 75:15:10 for training, validation and testing. The model is trained for 100 epochs with Adam in PyTorch, with dropout-regularised fully connected layers on top of the backbone.

Results. In the five-class test the proposed model reaches 96.24% accuracy and an F1-score of 0.9622, compared with 92.00% for plain MobileNetV2, 91.76% for SqueezeNet and EfficientNet, and 80.71% for ShuffleNet. Binary tasks give 99.29% for leaf curl versus healthy and 99.20% for whitefly versus healthy. The ablation study reports 97.84% accuracy for the full MLA module, against 93.12% for the baseline, with parameters rising from 3.47 M to 3.85 M and computation from 300 M to 322 M FLOPs. Removing the edge prior, the channel gate or the spatial gate lowers accuracy to between 95.02% and 96.21%. A comparison with other attention modules reports 91.68% for SE, 86.10% for CBAM, 82.62% for ECA and 96.91% for SDMNet, against 97.84% for MLA. On a PlantVillage subset of about 10,800 images in four classes, five-fold cross-validation gives an average of 98.4% and four-fold gives 98.69%.

Points to read carefully. The baseline MobileNetV2 accuracy appears as 92.00%, 93.12% and 95.91% in different tables, and the best multiclass figure is 96.24% in one section and 97.84% in another. The paper also states that up to 20 separate runs were performed and the best results were considered. These details limit how much weight a reader should place on the exact margins, although the direction of the improvement is consistent. The authors list mobile deployment, latency and power measurements as future work.

2.5 Lightweight and Edge-Oriented Models

The two MobileNetV2 papers sit inside an active family of efficient designs. Khan et al. study plant disease recognition for edge devices and stress optimisation and quantisation [13]. Duhan et al. introduce RTR_Lite_MobileNetV2 [14], and Li et al. combine MobileNetV2 with coordinate attention, multi-branch convolution and compression for tea leaf disease [15]. Subburaj et al. use a MobileNetV2 encoder in an attention-based model for multi-crop classification [16], and Wen et al. compare coordinate, ECA, SE and CBAM attention inside a lightweight tea-leaf model [17]. Pal et al. combine MobileNetV2 with residual learning (3.51 M parameters) and explain predictions with Grad-CAM, Grad-CAM++ and LIME, with 99.47% reported on PlantVillage under a different protocol [18].

Other compact designs include an improved MobileNetV2 with RepMLP and ECA that reaches 0.91 M parameters [19], an attention-based MobileNetV2 for plant disease classification [20], a lightweight model for plant disease and pest identification [27], and a MobileNet-based model for real-time guava leaf disease detection [23]. For chilli leaves, a multi-scale network [21] and a VGG16 plus EfficientNetB0 hybrid with attention [22] have been proposed, and Rajendrakumar and Rajashekarappa note that these are respectively heavier or lack attention [3]. Earlier squeeze-and-excitation and ECA variants have also been used with MobileNet and ResNet backbones [38], [39]. Federated learning has been explored for plant disease detection to protect data privacy [35], and real-time monitoring systems that join CNN classification with web or mobile platforms have been described [37].

2.6 Crop-Specific Studies: Cotton and Tomato

For cotton, Joshi et al. use a YOLOv8-based classifier with k-fold cross-validation and report 98.41% accuracy [24], and an ensemble CNN for cotton disease and pest recommendation reports 97.04% [40]. For tomato, a hybrid deep CNN reports 98.01% [25], and a transfer-learning study combines several CNN architectures with explainable AI [36]. Sujatha et al. combine machine learning and deep learning for plant leaf disease detection [26]. These studies are the direct benchmarks for the cotton and tomato results in [1].

2.7 Pesticide and Treatment Recommendation

Fewer studies address what to do after a disease is identified. Many recommendation systems are static and rule-based, and do not adapt to weather or disease severity [1]. Arakeri et al. build a pesticide recommendation system for cocoa using computer vision and deep learning [28]. Banothu et al. use DenseNet for disease identification and pesticide recommendation [29]. Kethineni et al. present a web-based system that recommends crops, fertilisers and pesticides using deep learning [30], and Kalaiselvi et al. use generative AI for fertiliser and pesticide advice from soil and pest inputs [31]. A survey of machine-learning pesticide recommendation models [41] and the cotton ensemble work [40] complete this line.

Remaining issues are data scarcity, scalability, interpretability and the risk of ignoring traditional knowledge [41].

2.8 Crop Price Forecasting and Market Information

Machine-learning price models are reported to be more reliable than classical statistical methods because markets are dynamic [34]. Bhardwaj et al. apply deep learning, including CNN and graph neural networks, and report accuracy up to 20% better than traditional approaches [33]. Sable et al. analyse real-time agricultural market data from Mumbai to suggest the best month for sale and harvest [32], and the hybrid framework in [1] uses an LSTM for the same purpose. Platform-level work includes AgriConnect, which connects farmers to consumers with timely price updates [42]. Persistent problems are data access, data quality and interpretability [1], [34].

2.9 Crop Recommendation with Machine Learning

Crop recommendation is usually treated as a supervised classification problem: given the location, season and sometimes soil or weather values, the model predicts the most suitable crop. Senapaty et al. [43] build a decision support system for this task. They start from a public historical dataset of 246,091 records covering 37 Indian crops, with state, district, year, season, crop, area and production fields, and reduce it to three districts of Odisha (Koraput, Gajapati and Rayagada), which leaves 1,480 records. They add data from agriculture offices and from interviews with experienced local farmers, store the combined data in Firebase, and split it 80% for training and 20% for testing.

Because the crop classes are imbalanced, the authors apply SMOTE (Synthetic Minority Oversampling Technique) and compare 13 classifiers: logistic regression, decision tree, KNN, SVC, random forest, gradient boosting, bagged tree, XGBoost, AdaBoost, CatBoost, histogram gradient boosting, SGDC and multinomial naive Bayes. Seven metrics are reported: accuracy, precision, recall, F1-score, ROC AUC, sensitivity and specificity. An Android application takes the farmer's location and the current season and displays the suggested crops, which makes this one of the few crop recommendation papers that connects a model to a cloud database and a mobile front end.

Results. Before SMOTE, SVC scores 1.00 accuracy, CatBoost 0.98, XGBoost and multinomial naive Bayes 0.97, and logistic regression only 0.76. After SMOTE, SGDC reaches 1.00 on accuracy, precision, recall, F1-score and ROC AUC, but with sensitivity of 0.91 and specificity of 0.54. SVC falls from 1.00 to 0.74 accuracy and its recall drops from 1.00 to 0.29, while XGBoost stays at 0.97. The authors conclude that SGDC is the best classifier once the data are balanced, and they list explainable AI and a larger dataset as future work.

Points to read carefully. First, the inputs are location, season, area and production values, not soil nutrients or weather, so the model learns past local cropping patterns rather than agronomic suitability. Second, only 1,480 public records plus the locally collected data remain after reduction. Third, a perfect SGDC score together with a specificity of 0.54, and a different best model before and after SMOTE, suggest that the ranking depends strongly on the split and may reflect overfitting. Fourth, the text describes balancing the dataset and then splitting it 80/20; if SMOTE was applied before the split, synthetic samples built from test-set neighbours can leak into training and inflate the scores. Finally, one specificity value in the paper's post-SMOTE table is printed as 8.45, which cannot be right for a ratio and is most likely a typing error.

Other crop recommendation work. Studies on soil and weather inputs are summarised in [43]. Thilakarathne et al. build a cloud-enabled recommendation platform and apply KNN, decision tree, random forest, XGBoost and SVC to a 2,200-record dataset of nitrogen, phosphorus, potassium, pH, temperature, humidity and rainfall values, with random forest reported at 97.18% [44]. Priyadharshini et al. compare linear regression, a neural network, KNN, naive Bayes and SVC, with the neural network best at 89.88% [48]. Pande et al. build a mobile application that uses GPS, and random forest is best at 95% [49]. Garg and Alam combine grid search with wrapper feature selection and report 99.31% [52], and Raju et al. propose a stacking ensemble with 97.1% accuracy [51]. Islam et al. join IoT soil sensors with machine learning, with CatBoost at 97.5% [46], Escorcia-Gutierrez et al. classify soil nutrients and pH with an ensemble of deep models (0.928 accuracy) [47], and Reyana et al. fuse multisensor data and find random forest best among the tree models they test [50]. Shams et al. add explainability with a LIME-based method called XAI-CROP [45]. For related tasks, tree-based models forecast vegetable prices better than the other methods in a Sri Lankan study [53], and LSTM-based networks are used for yield forecasting from weather data [54], [55]. Across these studies random forest and boosted trees appear most often, but the datasets differ and the scores cannot be ranked against each other.

2.10 Explainability, Trust and Privacy

Trust is a practical issue for farmers and agronomists. Grad-CAM is used in [2], [3] and [18] to show which leaf regions influence a prediction, and SHAP and LIME are used in [1] and [18]. These methods are mostly qualitative. Udoh et al. state that insertion, deletion, localisation and faithfulness metrics were not computed [2]. Explainable transfer learning for tomato disease [36] and federated learning for privacy [35] show two further directions that suit farmer-facing applications.

3. Review Methodology

3.1 Search Strategy and Sources

The review started from four core studies [1], [2], [3], [43] that match the functions of a mobile farmer assistant: disease detection, treatment advice, price information and crop recommendation. The wider literature was then collected around the topic terms plant disease detection, MobileNetV2, attention mechanism, edge and mobile deployment, pesticide recommendation, crop price prediction, crop recommendation and explainability, and by following the studies cited in the core papers. The 55 selected works appear in the journals and proceedings of IEEE, Springer, Elsevier, MDPI, Frontiers, Wiley, Nature Portfolio, Taylor & Francis and PLOS, among others.

3.2 Inclusion and Exclusion Criteria

A study was included when it (i) was published in 2021 or later, (ii) addressed plant disease detection, lightweight or edge models, pesticide or treatment recommendation, crop price forecasting, crop recommendation or explainability in agriculture, (iii) appeared in a journal or conference proceedings, and (iv) described its data and method and reported quantitative results, or was a review of the field. A study was excluded when it was published before 2021, gave no description of its data or method, or was not related to agricultural decision support.

3.3 Classification of the Selected Papers

The selected papers were sorted into seven groups according to the task they address. Table 1 lists the groups. The three image-based core studies [1], [2], [3] and the crop recommendation study [43] are read in full, and the remaining papers are used to place them in context. Each paper is compared on its dataset, method, reported accuracy or error and computational cost, as stated by its authors. No result was re-run, so all figures are reported values and not independent measurements.

Table 1. Classification of the 55 selected references

Group	References	Papers
Surveys and reviews of plant disease detection	[4]–[12]	9
Lightweight, edge and attention-based models	[2], [3], [13]–[23], [27], [38], [39]	16
Crop-specific, explainable, federated and monitoring studies	[24]–[26], [35]–[37]	6
Pesticide, treatment and integrated frameworks	[1], [28]–[31], [40], [41]	7
Price forecasting and market information	[32]–[34], [42], [53]	5
Crop recommendation	[43]–[52]	10
Yield forecasting	[54], [55]	2

4. Comparative Analysis

This section compares the three core studies and the wider literature on datasets, methods, results and cost. The studies differ in data, splits, hardware and metrics, so the tables show what each paper reports and are not a ranking.

4.1 Scope of the Three Core Studies

Table 2. Scope of the three core studies

Aspect	Sable et al. [1]	Udoh et al. [2]	Rajendrakumar and Rajashekarappa [3]
Year and venue	2026, Frontiers in Computer Science	2026, Electronics	2026, AgriEngineering
Main idea	Hybrid framework: detection, pesticide advice, price forecasting	One late CBAM block on MobileNetV2	Multi-level MLA module on MobileNetV2
Crops and data	Cotton (11,527 images, 14 classes); tomato (37,363 images, 10 classes)	PlantVillage (54,306 images, 38 classes)	Chilli (4,432 images, 5 classes); PlantVillage subset (about 10,800 images, 4 classes)

Aspect	Sable et al. [1]	Udoh et al. [2]	Rajendrakumar and Rajashekarappa [3]
Split	80% / 20% test, then 20% of training for validation	70% / 15% / 15%	75% / 15% / 10% (main); k-fold on PlantVillage subset
Models compared	ResNet50, EfficientNetB0, custom CNN, YOLOv8	MobileNetV2, MobileNetV2 + CBAM, InceptionV3, Xception	SqueezeNet, ShuffleNet, EfficientNet, MnasNet, MobileNetV2, proposed
Framework	TensorFlow / Keras; FastAPI backend	TensorFlow 2.13 / Keras; TensorFlow Lite 2.18	PyTorch 2.1
Beyond classification	Pesticide knowledge base, LSTM price model, SHAP and LIME	Quantisation, size, latency, McNemar test, Grad-CAM	Ablation, attention-module comparison, Grad-CAM
Field validation	No	No	No
Device measurement	No	Laptop CPU only	No (future work)

4.2 Reported Classification Results

Table 3. Reported classification results

Study	Model	Data	Reported accuracy
[1]	EfficientNetB0	Tomato	99.95%
[1]	ResNet50	Tomato / Cotton	99.21% / 98.87%
[1]	Custom CNN	Tomato / Cotton	97.66% / 98.51%
[1]	YOLOv8	Tomato / Cotton	95.27% / 98.41%
[2]	Xception	PlantVillage, 38 classes	98.30%
[2]	MobileNetV2 + CBAM	PlantVillage, 38 classes	97.17% (96.70% after TFLite)
[2]	InceptionV3	PlantVillage, 38 classes	96.89%
[2]	MobileNetV2	PlantVillage, 38 classes	96.78% (96.07% after TFLite)
[3]	MobileNetV2 + MLA	Chilli, 5 classes	96.24% test (97.84% in ablation)
[3]	MobileNetV2	Chilli, 5 classes	92.00% test (93.12% / 95.91% in other tables)
[3]	MobileNetV2 + MLA	PlantVillage subset, 5-fold / 4-fold	98.4% / 98.69%

Study	Model	Data	Reported accuracy
[24]	YOLOv8 with k-fold	Cotton	98.41%
[25]	Hybrid deep CNN	Tomato	98.01%
[18]	MobileNetV2 + residual learning	PlantVillage	99.47% (different protocol)

All values are close to or above 96%, and the differences between most models are within one or two percentage points. Such small gaps are easily changed by the split, augmentation, number of classes, training length and random seed. Only [2] reports a paired significance test, and none of the three core papers reports mean and standard deviation over several independently trained runs.

4.3 Accuracy versus Deployment Cost

Only the study by Udoh et al. [2] measures deployment cost in a reproducible way. Its figures are given below.

Table 4. Accuracy versus deployment cost [2]

Model	Parameters	GFLOPs	TFLite size (MiB)	TFLite accuracy	Median latency (ms)
MobileNetV2	3.61 M	0.602	3.68	96.07%	45.38
MobileNetV2 + CBAM	4.02 M	0.604	4.07	96.70%	45.42
InceptionV3	23.94 M	11.451	23.10	96.80%	10.71
Xception	23.00 M	9.115	22.57	98.22%	200.15

Two lessons follow. First, a small attention block adds about 11% to the parameter count but almost nothing to computation or latency. Second, operation counts do not predict speed: on the same CPU, InceptionV3 ran faster than MobileNetV2 although it needs about 19 times more GFLOPs. Latency depends on the hardware, the runtime delegate and the operators used. A lightweight model therefore has to be timed on the target phone, not only counted in FLOPs. The MLA paper [3] reports parameters and FLOPs (3.47 M to 3.85 M, and 300 M to 322 M) but no latency, and the framework paper [1] reports neither.

4.4 Attention Mechanisms Compared

Table 5. Attention mechanisms compared

Mechanism	What it does	Evidence in the reviewed papers
CBAM, late placement [2]	Channel then spatial attention, once, before pooling	+0.39 points over MobileNetV2 in float, +0.64 points on TFLite (paired test); about 0.3% extra computation
MLA, multi-level [3]	Channel gate, spatial gate and fixed edge prior, fused with learnable weights, at three depths	97.84% vs 93.12% baseline in ablation; each branch lowers accuracy when removed

Mechanism	What it does	Evidence in the reviewed papers
SE, ECA, CBAM, SDMNet baselines [3]	Channel-only, local channel, channel plus spatial, self-distillation	91.68%, 82.62%, 86.10% and 96.91% on the chilli data, below MLA
CA, ECA, SE, CBAM in tea model [17]	Attention ablation in a lightweight network	Described in [2] as an ablation that is still needed under one common protocol

The two attention studies draw different conclusions about CBAM. In [2] CBAM gives a small but significant gain on PlantVillage, and in [3] it performs worse than the baseline on the chilli data (86.10% against 95.91% in the same table). The difference may come from the dataset size, the way CBAM was inserted and trained, or the baseline used. It is a reminder that an attention module does not help in every setting, and that placement, data size and training protocol matter.

4.5 Isolated Models versus Integrated Systems

Table 6. Isolated models versus integrated systems

Approach	Examples	Strengths	Weaknesses
Classification only	[2], [3], [13], [14], [20]	Clear protocol, easy to compare, small models	Gives no treatment, price or weather advice
Detection plus recommendation	[1], [28], [29], [30], [40]	Closer to the farmer's real question	Recommendation quality is rarely tested with agronomists
Recommendation or price only	[31], [32], [33], [34]	Useful market and input guidance	Not linked to the disease result
Integrated multimodal framework	[1]	Joins detection, treatment and price in one flow	Fusion and price modules lightly evaluated; no field study

4.6 Observations

- Benchmark accuracy has saturated. Differences between good architectures are now smaller than the likely effect of the data split.
- Efficiency claims are common, but efficiency measurements on real mobile hardware are rare.
- Only a few papers report statistical tests, repeated runs, duplicate checks or explainability metrics, and the stronger papers say openly what they did not do [2].
- Integration is the least developed area. Detection models, recommenders and price models are mostly built and tested separately.

4.7 Crop Recommendation Studies Compared

The fourth core study [43] belongs to a different task from the three image-based papers, so it is compared with other crop recommendation work. Details of the other studies follow the summaries given in [43].

Table 7. Crop recommendation studies compared

Study	Inputs	Models	Best reported result	Remark
Senapaty et al. [43]	Location, season, area, production; Odisha, three districts plus local data	13 classifiers with SMOTE	SGDC 1.00 accuracy after SMOTE (specificity 0.54); SVC 1.00 before	Android app and Firebase; small reduced dataset
Thilakarathne et al. [44]	N, P, K, pH, temperature, humidity, rainfall; 2,200 records	KNN, DT, RF, XGBoost, SVC	RF 97.18%	Cloud-hosted platform
Priyadharshini et al. [48]	Soil, season, yield, price and rainfall data for 16 crops	LR, NN, KNN, NB, SVC	NN 89.88%	Considers profitability
Pande et al. [49]	Area, soil type, GPS location	SVC, ANN, RF, KNN, MLR	RF 95%	Mobile app; Maharashtra and Karnataka
Garg and Alam [52]	Soil features, humidity, rainfall; 2,200 instances	Grid search, wrapper selection, partial decision tree	99.31%	Shows the effect of feature selection
Raju et al. [51]	Agro-ecological zone data	Multilayer stacking ensemble	97.1% accuracy, F1 97.09%	Ensemble approach
Islam et al. [46]	IoT soil nutrient sensor data	Several ML models	CatBoost 97.5%	Needs sensor hardware
Escorcia-Gutierrez et al. [47]	Soil nutrient and pH values	Ensemble deep learning, voting	0.928 accuracy	Nutrient classification
Shams et al. [45]	Historical Indian crop, soil and weather data	XAI-CROP (LIME) against GB, DT, GNB, MNB	R-squared about 0.94	Adds explanations

Three points follow. First, most systems report 90% to 100% accuracy, but the datasets are small and the classes are well separated, so the scores say little about performance on a new farm. Second, the inputs decide what the system can promise: location and season data [43] are easy to collect, while soil values [44], [46] need tests or sensors that most farmers do not have. Third, only a few systems connect

recommendation to price information [48] or to explanations [45], and none of the compared work joins recommendation with disease detection in one application.

5. Research Gaps and Future Directions

5.1 Research Gaps

1. **Laboratory-to-field gap.** All three core studies use public benchmark images captured under fairly controlled conditions [1], [2], [3]. Clutter, shadows, partial leaves, overlapping symptoms and low-quality phone cameras are not tested. Reviews list the same weakness [7], [8], [9].
2. **Statistical rigour.** Single training runs, best-of-many reporting [3] and missing confidence intervals make small gains hard to trust. Only one core paper reports a paired significance test [2].
3. **Dataset quality.** Exact duplicates across splits were found in PlantVillage [2], and class imbalance and small datasets affect chilli and other crops [3]. Splits should be grouped by content before partitioning.
4. **Limited crop and region coverage.** The reviewed work concentrates on a few crops (tomato, cotton, chilli, PlantVillage species). Many Indian regional crops and diseases are not covered.
5. **Deployment evidence.** No core paper measures latency, memory or energy on a phone [1], [2], [3]. Reproducible laptop benchmarks are a start, but not a substitute.
6. **Weak evaluation of recommendations.** Pesticide recommendations come from knowledge bases, and the quality, safety and dose accuracy of the advice are not validated with experts [1], [41].
7. **Price forecasting validation.** The LSTM model in [1] is tested on market data from 2020 to 2025, but the paper itself calls for comparison with ARIMA, regression and other machine-learning baselines, and reports slightly different error figures in different places.
8. **Explainability is qualitative.** Grad-CAM, SHAP and LIME images are shown, but quantitative faithfulness and localisation scores are not computed [1], [2].
9. **Limited language and user studies.** The reviewed papers say little about local-language interfaces, voice input, low literacy, or tests with real farmers.
10. **Fragmented pipelines.** Disease detection, weather, market prices, government schemes and community learning are rarely studied as one system with measured usability.

Additional gaps from crop recommendation research [43] to [55].

- **Leakage-safe evaluation.** Balancing the data before the split, perfect scores with low specificity and unstable rankings before and after SMOTE show the need for grouped splits, cross-validation, per-class metrics and confidence intervals [43].
- **Small and local datasets.** Many recommenders rely on a few thousand records from one region, and the reduced dataset in [43] has about 1,480 public records.
- **Input limits.** Models that use only location, season and production do not capture soil, weather or water conditions, while soil-based models need data that farmers rarely have [43], [44], [46].
- **Missing integration.** Crop recommendation is rarely combined with weather, mandi price, disease detection and scheme information in one farmer-facing system.

5.2 Challenges

- **Data.** Collecting labelled field images across seasons, regions and phone cameras is costly, and datasets are often imbalanced [3], [8].

- **Generalisation.** Models that score above 97% on benchmarks can fail under new lighting, backgrounds or symptom stages [2], [7].
- **Resource limits.** Low-end phones and weak networks limit model size and server access, so offline or on-device inference is valuable [2], [13].
- **Interpretability and trust.** Farmers and experts need to know why a result was given and how confident the model is [1], [2].
- **Safety of advice.** A wrong pesticide or dose can harm crops, health and the environment, so recommendation modules need expert-verified content [28], [41].
- **Market volatility.** Price forecasts depend on data quality, seasonality and sudden events [1], [34].
- **Privacy and maintenance.** Location and farm data must be protected [35], and disease, scheme and price content must be kept up to date.

5.3 Future Scope

- **Field-validated models.** Train and test on real field images, with grouped splits, several random seeds and reported mean and standard deviation [2].
- **Attention placement studies.** Compare early, middle, multi-block and late attention, and different attention types, under one protocol [2], [3], [17].
- **On-device deployment.** Convert models to TensorFlow Lite, and measure latency, memory and energy on representative phones and edge boards [2], [3].
- **Confidence and fallback.** Show a confidence score and route uncertain cases to an expert or extension officer.
- **Verified treatment knowledge.** Build the recommendation base with agronomists, link each disease to prevention and eco-friendly options, and add weather-aware timing [1], [28], [29].
- **Better forecasting.** Compare LSTM with ARIMA, regression and tree ensembles, and add demand, weather and arrival data [1], [33].
- **Quantitative explainability.** Add insertion, deletion and localisation metrics to Grad-CAM, SHAP and LIME visuals [2].
- **Privacy-preserving learning.** Use federated learning to improve models from farmers' images without moving the images [35].
- **Inclusive interfaces.** Support regional languages, voice input and icon-based screens, and test usability with farmers.

Crop recommendation directions.

- **Leakage-safe training.** Apply SMOTE only inside training folds, report per-class metrics and use repeated cross-validation, so that a perfect score is not mistaken for a good model [43].
- **Profit-aware recommendation.** Combine location, season, soil values where available, weather forecasts and mandi price trends, so that the suggested crop is both suitable and profitable [32], [43], [48].
- **Explainable suggestions.** Show why a crop is suggested, for example with LIME or SHAP, so that farmers and officers can check the reasoning [45].

5.4 Implications for an Integrated Assistant such as Krishi Mitra

A mobile assistant such as Krishi Mitra combines a Flutter app, a FastAPI backend, Firebase, a disease classifier, mandi rates, weather, schemes, a farmer feed and multilingual screens. The reviewed literature suggests the following design choices.

Table 8. Design implications for an integrated assistant

Module	Supporting evidence	Design implication
Disease scanner	[2], [3], [13], [18]	Use a compact backbone such as MobileNetV2, convert it to TensorFlow Lite, and time it on a real phone
Model choice	[1], [2]	Compare at least two backbones and report accuracy, size and latency together
Field accuracy	[2], [7], [8]	Add real field photos to PlantVillage and state clearly that benchmark accuracy is not field accuracy
Treatment advice	[1], [28], [29], [41]	Link each predicted disease to a reviewed treatment record, with eco-friendly options
Price and mandi	[1], [32], [33], [34]	Show current prices first, and add forecasts only after comparison with simple baselines
Backend	[1]	A FastAPI service that hosts models and combines third-party data keeps keys out of the app
Trust	[1], [2], [18]	Show Grad-CAM style highlights and a confidence score with each diagnosis
Privacy	[35]	Keep location and farm data private, and consider federated learning later
Crop recommendation	[43], [44], [45], [52]	Start with location and season inputs and add soil values when the farmer has them; use random forest or boosting baselines; balance classes only inside training folds; explain each suggestion

5.5 Krishi Mitra: Technology and Methodology

Krishi Mitra is the mobile digital farmer assistant that motivates this review. It brings the functions found in the literature into one application: leaf-image disease diagnosis, treatment advice, mandi prices, weather, government schemes, a farmer community feed and multilingual screens. This section explains the technology used and the methodology followed to build it. Table 9 lists the technology used for each part of the system, and Fig. 1 shows how a request moves through it.

Table 9. Technology stack of Krishi Mitra

Layer	Technology	Role
Mobile application	Flutter	One application with multilingual screens for scanning, prices, weather, schemes and feed

Layer	Technology	Role
Backend service	FastAPI	Receives images and requests, runs the model, collects third-party data and keeps keys off the device
Cloud data	Firebase	Stores user data, the farmer feed and app content
AI module	Deep-learning disease classifier	Predicts the disease class and a confidence score from a leaf photograph
Market and weather	Mandi-rate and weather data services	Supply current crop prices and local weather
Advice content	Treatment records and scheme information	Link each disease to treatment advice and give scheme details

Fig. 1. Request flow of the Krishi Mitra system

Methodology

The development of Krishi Mitra follows a simple sequence: study the problem in the literature, design a modular client-server architecture, build each module, connect them through one backend, and plan an evaluation that reports the measures the reviewed papers often leave out. Each step is explained below.

Step 1: Requirement analysis. The first step is to decide what a farmer actually needs. The review shows that most research answers only one question, which disease is on the leaf, while a farmer also needs advice on treatment, current market prices, weather and scheme information, in a language he or she can read [1], [10]. These gaps fix the list of modules: disease scanner, treatment advice, mandi prices, weather, schemes, farmer feed and multilingual screens.

Step 2: System architecture. Krishi Mitra uses a three-part client-server design. The Flutter application is the client, the FastAPI service is the backend, and Firebase is the cloud data store. Flutter gives one code base for the whole interface, FastAPI is a light Python framework that can serve a trained model and other data through simple web endpoints, and Firebase is a managed service, so the team does not have to run its own database server. Keeping the model and the third-party data access in the backend means that the phone only sends and receives small requests, and that service keys are never stored inside the app. A similar split between a web or mobile front end and a model-serving backend is used in the framework of [1].

Step 3: Image capture and preparation. The farmer chooses a language and photographs the affected leaf. The application sends the photograph to the backend, where it is resized and normalised to the input size of the classifier before prediction. The reviewed studies follow the same preparation, for example 224 × 224 input with normalised pixel values [1].

Step 4: Disease classification. A deep-learning image classifier predicts the disease class and a confidence score. Convolutional networks are the standard choice for this task, and compact networks such as MobileNetV2 are preferred when the model has to be light enough for phones and low-cost devices [2], [3], [13]. The confidence score is returned with the disease name, so that the farmer can see how sure the model is, and an uncertain result can be sent to an expert instead of being trusted blindly. This follows the trust and fallback needs identified in Section 5.

Step 5: Treatment advice. A disease name alone does not help a farmer, so every disease class is linked to a treatment record. The record holds the symptoms, prevention methods, the recommended treatment

and eco-friendly alternatives. This is the same knowledge-base idea used for pesticide recommendation in [1], [28] and [29]. Because a wrong pesticide or dose can harm crops and health, the records have to be reviewed by agronomists before they are shown to farmers [41].

Step 6: Market, weather and scheme information. The backend collects mandi rates and weather data from third-party services and returns them to the application. Government scheme information and the farmer feed are served from the cloud data store, so content can be updated without releasing a new version of the app. Current prices are shown first, and forecasting is added only after it has been compared with simple baselines, as the price-forecasting literature advises [1], [33], [34].

Step 7: Local-language presentation. All results, advice and screens are shown in the language chosen by the farmer. The reviewed papers say little about local-language interfaces, voice input or tests with real farmers, so this step addresses one of the gaps found in Section 5.

Step 8: Evaluation plan. The evaluation follows the gaps identified in this review. Accuracy is reported together with model size and latency measured on a real phone [2], the classifier is tested on field photographs as well as benchmark images [7], [8], treatment records are checked by agronomists [1], [41], and usability is tested with farmers. Benchmark accuracy is not presented as field accuracy.

6. Conclusion

This review examined recent work that supports a mobile digital farmer assistant, with three 2026 papers as the core. The hybrid multimodal framework [1] shows how disease detection, pesticide recommendation and price forecasting can be joined in one pipeline, with strong benchmark accuracy on tomato and cotton images. The late-attention MobileNetV2 study [2] shows that a small attention block can give a modest, statistically supported gain at almost no extra computation, and it sets a good example by reporting size, latency, duplicates and limits. The multi-level attention study [3] shows that a lightweight module combining channel, spatial and edge cues can raise accuracy on a small chilli dataset, although its inconsistent baseline figures call for careful reading. The crop recommendation study [43] adds a cloud-connected Android application and a comparison of 13 classifiers with SMOTE, but its perfect SGDC score, paired with a specificity of 0.54, should be read with caution.

Across the literature, benchmark accuracy is no longer the main problem, and perfect scores on small datasets can hide overfitting or data leakage. The open problems are field validation, repeated-run statistics, real-device measurements, verified treatment knowledge, quantitative explainability, regional-language access and the integration of separate modules into one usable system. Work that closes these gaps will move plant disease detection from research benchmarks to tools that farmers can trust in the field.

REFERENCES:

[1] N. P. Sable, V. K. Shukla, P. N. Mahalle and V. Khedkar, “Hybrid multimodal learning framework for crop disease detection, adaptive treatment, and price forecasting,” *Front. Comput. Sci.*, vol. 8, art. 1855851, 2026. doi: 10.3389/fcomp.2026.1855851.

[2] E. Udoh, M. A. Alaoui Mhamdi and M. Allili, “An efficient attention-enhanced MobileNetV2 framework for plant disease detection on resource-constrained devices,” *Electronics*, vol. 15, no. 15, art. 3343, 2026. doi: 10.3390/electronics15153343.

[3] S. Rajendrakumar and Rajashekarappa, "Improved MobileNetV2 architecture with modified lite attention model for detection of plant leaf disease," *AgriEngineering*, vol. 8, no. 6, art. 248, 2026. doi: 10.3390/agriengineering8060248.

[4] A. Upadhyay, N. S. Chandel, K. P. Singh, S. K. Chakraborty, B. M. Nandede and M. Kumar, "Deep learning computer vision in plant disease detection: a comprehensive review of techniques, models, trends in precision agriculture," *Artif. Intell. Rev.*, vol. 58, art. 92, 2025. doi: 10.1007/s10462-024-11100-x.

[5] H. N. Ngugi, A. E. Ezugwu, A. A. Akinyelu and L. Abualigah, "Revolutionizing crop disease detection with computational deep learning: a comprehensive review," *Environ. Monit. Assess.*, vol. 196, art. 302, 2024. doi: 10.1007/s10661-024-12454-z.

[6] A. Dolatabadian, T. X. Neik, M. F. Danilevicz, S. R. Upadhyaya, J. Batley and D. Edwards, "Image-based crop disease detection using machine learning," *Plant Pathol.*, vol. 74, pp. 18–38, 2025. doi: 10.1111/ppa.14006.

[7] J. Zhao, L. Xu, Z. Ma, J. Li, X. Wang, Y. Liu and X. Du, "A review of plant leaf disease identification by deep learning algorithms," *Front. Plant Sci.*, vol. 16, art. 1637241, 2025.

[8] T. Nyawose, R. C. Maswanganyi and P. Khumalo, "A review on the detection of plant disease using machine learning and deep learning approaches," *J. Imaging*, vol. 11, art. 326, 2025.

[9] P. Sajitha, A. D. Andrushia, N. Anand and M. Z. Naser, "A review on machine learning and deep learning image-based plant disease classification for industrial farming systems," *J. Ind. Inf. Integr.*, vol. 38, art. 100572, 2024.

[10] H. Orchi, M. Sadik and M. Khaldoun, "On using artificial intelligence and the Internet of Things for crop disease detection: a contemporary survey," *Agriculture*, vol. 12, no. 1, art. 9, 2022. doi: 10.3390/agriculture12010009.

[11] C. Jackulin and S. Murugavalli, "A comprehensive review on detection of plant disease using machine learning and deep learning approaches," *Meas. Sens.*, vol. 24, art. 100441, 2022.

[12] J. D. Omaye, E. Ogbuju, G. Ataguba, O. Jaiyeoba, J. Aneke and F. Oladipo, "Cross-comparative review of machine learning for plant disease detection: apple, cassava, cotton and potato plants," *Artif. Intell. Agric.*, vol. 12, pp. 127–151, 2024.

[13] A. T. Khan, S. M. Jensen, A. R. Khan and S. Li, "Plant disease detection model for edge computing devices," *Front. Plant Sci.*, vol. 14, art. 1308528, 2023.

[14] S. Duhan, P. Gulia, N. S. Gill and E. Narwal, "RTR_Lite_MobileNetV2: a lightweight and efficient model for plant disease detection and classification," *Curr. Plant Biol.*, vol. 42, art. 100459, 2025.

[15] Z. Li, Y. Li, C. Yan, P. Yan, X. Li, M. Yu, T. Wen and B. Xie, "Enhancing tea leaf disease identification with lightweight MobileNetV2," *Comput. Mater. Contin.*, vol. 80, pp. 679–694, 2024.

[16] B. Subburaj, M. Rohan, S. Ananthanarayanan and D. Won, "Plantention: a general-purpose, lightweight and attention-based model for multi-crop leaf disease classification," *Results Eng.*, vol. 26, art. 105075, 2025.

[17] X. Wen, Q. Liu, X. Tang, F. Yu and J. Chen, "A lightweight convolutional neural network for tea leaf disease and pest recognition," *Plant Methods*, vol. 21, art. 129, 2025.

[18] C. Pal, S. Karmakar, I. Mukherjee and P. P. Chakrabarti, "A lightweight and explainable CNN model for empowering plant disease diagnosis," *Sci. Rep.*, vol. 15, art. 30720, 2025.

[19] J. Lu, X. Liu, X. Ma, J. Tong and J. Peng, "Improved MobileNetV2 crop disease identification model for intelligent agriculture," *PeerJ Comput. Sci.*, vol. 9, art. e1595, 2023.
[20] H. Wang, S. Qiu, H. Ye and X. Liao, "A plant disease classification algorithm based on Attention MobileNet V2," *Algorithms*, vol. 16, art. 442, 2023.
[21] D. Li, C. Zhang, J. Li, M. Li, M. Huang and Y. Tang, "MCCM: multi-scale feature extraction network for disease classification and recognition of chili leaves," *Front. Plant Sci.*, vol. 15, art. 1367738, 2024.
[22] R. Rani, S. Bharany, D. H. Elkamchouchi, A. Ur Rehman, R. Singh and S. Hussen, "VGG-EffAnNet: hybrid deep learning model for automated chili plant disease classification using VGG16 and EfficientNetB0 with attention mechanism," *Food Sci. Nutr.*, vol. 13, art. e70653, 2025.
[23] M. M. U. Nobi, M. Rifat, M. F. Mridha, S. Alfarhood, M. Safran and D. Che, "GLD-Det: guava leaf disease detection in real-time using a lightweight deep learning approach based on MobileNet," *Agronomy*, vol. 13, art. 2240, 2023.
[24] K. Joshi, Y. Yadav, S. Hooda, R. Nandal, B. Singh, K. Singh et al., "Classification of cotton leaf disease using YOLOv8 based k-fold cross validation deep learning method for precision agriculture," *Sci. Rep.*, vol. 15, art. 35602, 2025. doi: 10.1038/s41598-025-13147-4.
[25] T. Anandhakrishnan and S. M. Jaisakthi, "Deep convolutional neural networks for image based tomato leaf disease detection," *Sustain. Chem. Pharm.*, vol. 30, art. 100793, 2022. doi: 10.1016/j.scp.2022.100793.
[26] R. Sujatha, S. Krishnan, J. M. Chatterjee and A. H. Gandomi, "Advancing plant leaf disease detection integrating machine learning and deep learning," *Sci. Rep.*, vol. 15, art. 11552, 2025. doi: 10.1038/s41598-024-72197-2.
[27] H. Guan, C. Fu, G. Zhang, K. Li, P. Wang, Z. Zhu et al., "A lightweight model for efficient identification of plant diseases and pests based on deep learning," *Front. Plant Sci.*, vol. 14, art. 1227011, 2023. doi: 10.3389/fpls.2023.1227011.
[28] M. M. P. D. Arakeri, A. V. Kavan, K. S. Murthy, N. L. Nishitha and N. Lakshmi, "Intelligent pesticide recommendation system for cocoa plant using computer vision and deep learning techniques," *Environ. Res. Commun.*, vol. 6, art. 075003, 2024. doi: 10.1088/2515-7620/ad58ae.
[29] S. Banothu, K. Madhavi, M. V. M. K. Kumar, R. Gajula, C. Mallikarjuna Rao, S. Dixit et al., "Plant disease identification and pesticides recommendation using DenseNet," *Cogent Eng.*, vol. 11, art. 2353080, 2024. doi: 10.1080/23311916.2024.2353080.
[30] K. Kethineni, S. H. Mekala, M. Kodali, V. V. Kota and J. P. Jampani, "A web-based agriculture recommendation system using deep learning for crops, fertilizers, and pesticides," in *Proc. Int. Conf. Comput. Intell. Green Sustain. Technol. (ICCIGST)*, Vijayawada, IEEE, 2024. doi: 10.1109/ICCIGST60741.2024.10717535.
[31] G. Kalaiselvi, M. Nisha, M. Rudhra and V. Shrivaishnavi, "Generative AI based fertilizer and pesticide recommendation