Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 19 No. 21 (2025) | OPEN ACCESS

https://doi.org/10.3991/ijim.v19i21.55919

PAPER

Edge-Enabled Mobile App for Smart Agriculture Using Multi-Sensor Inputs and a Hybrid CNN–Vision Transformer Model

Thomas Kinyanjui Njoroge[1] (✉), **Rachael Kibuku**[2], **Kevin Mugoye Sindu**[2]

[1]Karatina University, Karatina, Kenya

[2]KCA University, Nairobi, Kenya

tnjoroge@karu.ac.ke

ABSTRACT

Crop diseases significantly threaten food security, particularly in resource-limited regions. Existing mobile diagnostic tools often lack robustness and fail to integrate environmental context. To address these limitations, this study developed a hybrid CNN-Vision transformer model combining EfficientNetV2, MobileNetV2, and Vision Transformers. The model was trained on Kaggle's PlantVillage dataset and locally collected field images covering 76 disease and condition classes. Multi-sensor inputs, including soil moisture, temperature, and humidity, were integrated to enhance prediction accuracy. The model achieved 99.2% accuracy, an AUC of 0.999998, and a 69% lower prediction variance than baseline models. Bayesian testing confirmed its superiority over DenseNet50 and other models. TensorFlow Lite was used to optimize the model for deployment on resource-constrained devices and was integrated into AgriScan, a 30.4 MB edge-enabled Android app. AgriScan supports offline inference, delivers real-time predictions with 0.094s latency, and reduces misdiagnoses by 92%. Evaluation on 249 unseen local images confirmed 97.97% accuracy, validating its field applicability. The model achieved a prediction variance of 0.000010, indicating strong confidence consistency across predictions. Benchmarking revealed 4.8× faster inference and 83% lower energy usage than cloud-based alternatives. Cross-device testing confirmed 100% diagnostic consistency, supporting the model's generalizability. This framework combines edge AI and sensor fusion into a scalable, cloud-independent diagnostic solution for advancing smart agriculture in low-connectivity environments.

KEYWORDS

hybrid deep learning, Edge computing, Internet of Things (IoT) sensing, machine learning, IoT-integrated sensing

1 INTRODUCTION

Food security is one of the most critical challenges of the 21st century, with an estimated 828 million people experiencing hunger globally. [1] underscore the significant

Njoroge, T. K., Kibuku, R., Sindu, K. M. (2025). Edge-Enabled Mobile App for Smart Agriculture Using Multi-Sensor Inputs and a Hybrid CNN–Vision Transformer Model. *International Journal of Interactive Mobile Technologies (iJIM)*, 19(21), pp. 145–162. https://doi.org/10.3991/ijim.v19i21.55919

Article submitted 2025-04-16. Revision uploaded 2025-07-17. Final acceptance 2025-07-20.

economic burden of crop diseases and poor field management practices, which account for over $220 billion in yearly agricultural losses. [2] report that pests and diseases devastate up to 100% of yields in Sub-Saharan Africa, intensifying food insecurity and economic hardship. Akhtar et al. [3] demonstrated that traditional methods, such as manual agronomist scouting, are labor-intensive, subjective, and impractical for large-scale farms. Moreover, while promising for automating disease detection, standalone CNN models often operate in isolation from environmental data, leading to contextually blind diagnoses. [4] demonstrate that vision transformers are more resistant to distortions than conventional convolutional neural networks in maintaining accuracy under varying lighting conditions, occlusions, and noise. Similarly, IoT sensor networks, as shown by [5] deployed for soil moisture, temperature, and humidity monitoring, rarely inform AI-driven decisions, creating fragmented insights. This gap highlights the urgent need for integrated systems that unify visual diagnostics with real-time environmental context. Addressing these gaps is crucial for developing a robust, adaptable, and efficient plant disease detection system capable of real-time deployment in resource-constrained environments. AI-powered mobile applications have played a transformative role in precision agriculture, particularly plant disease detection. Apps such as Plantix [6], AgroTutor [7], and PlanteSaine [8] leverage convolutional neural networks (CNNs) to assist farmers in identifying crop diseases. However, these tools, as noted by [9], face notable limitations. Many depend heavily on internet connectivity for cloud-based inference, rendering them ineffective in regions with limited or no internet access [10], [11]. In addition, they incur high data costs, consuming an estimated 5–10% of farmers' income in developing countries, thereby limiting accessibility [12]. Another key shortcoming is the lack of integration with Internet of Things (IoT) sensor data—such as environmental and soil parameters as demonstrated by Ariss et al. [13]—which constrains the apps' ability to provide context-aware diagnoses [14]. In response to these limitations, researchers have explored edge AI frameworks that aim to decentralize inference and reduce computational demands. Lightweight CNN architectures such as MobileNetV2 [15] and EfficientNetV2 [16] have been proposed to enable on-device processing, yet they often compromise accuracy under complex field conditions involving variable lighting, occlusions, or disease overlap [17]. More advanced hybrid models, including the [18] LLM-KG-GNN framework, demonstrate improved precision but rely on high-resolution inputs and cloud-based offloading, reintroducing latency and connectivity issues. While recent advancements like coordinate attention mechanisms [19] and weight pruning techniques [20] have shown promise in optimizing model efficiency, their real-world field validation remains limited. Meanwhile, the integration of IoT technologies in agriculture has opened new avenues for real-time monitoring of key environmental variables, including soil moisture [21], humidity [22], and temperature [5]. Despite their potential, most prior fusion strategies [23] and [24] rely on cloud infrastructures, introducing latency and connectivity constraints that are particularly problematic in rural or underserved areas. Siddiqua et al. [6] and [25] emphasize the urgent need for offline, edge-based sensor-AI fusion strategies that can empower farmers with actionable insights without reliance on cloud services, thus advancing the democratization of precision agriculture.

This study proposes an innovative solution that advances real-time crop disease detection. First, it introduces a hybrid CNN-Vision Transformers (ViT) model that integrates EfficientNetV2, MobileNetV2, and ViT, enhanced through multi-scale feature fusion and gated attention mechanisms. Second, the model is optimized for edge deployment using TensorFlow Lite, enabling efficient performance on devices with limited computational resources. Lastly, the solution incorporates IoT sensor data—such as temperature, humidity, and soil moisture—alongside visual diagnostics to improve overall diagnostic accuracy and significantly reduce false positives.

The remainder of this paper is organized as follows: Section 2 describes the methodology, covering mobile app design, model architecture, dataset preparation, and sensor fusion strategies. Section 3 presents the results and discussion, including performance comparisons with benchmarks and an evaluation of edge deployment efficiency. Section 4 provides the conclusion, highlighting the practical implications, limitations of the study, and directions for future research.

2 METHODOLOGY

This section presents a hybrid deep-learning framework integrated into a mobile app for real-time crop disease detection and environmental monitoring. The model uses a dual-branch architecture combining EfficientNetV2 and MobileNetV2 for feature extraction, followed by Squeeze-and-Excitation (SE) attention and ViTs for enhanced contextual understanding. Fusion strategies and attention mechanisms improve robustness under varying field conditions. The mobile app supports offline diagnosis, a text-guided UI, and IoT sensor integration. Edge optimization with TensorFlow Lite enables efficient inference on mid-range devices.

2.1 Mobile app design

The proposed system architecture integrates mobile computing, hybrid deep learning, IoT sensors, and edge deployment strategies to deliver a robust and accessible solution for crop disease detection in low-resource settings. Figure 1 illustrates the initial mobile app interfaces, designed with an offline-first approach to ensure functionality in connectivity-constrained regions. The app removes reliance on cloud-based predictions by leveraging TensorFlow Lite for on-device inference, addressing a key barrier to adoption in rural farming communities. The user interface is tailored to farmers' needs, featuring simplified workflows, text-guided image capture, real-time and gallery image selection support, and one-touch sensor synchronization. Additionally, the app is compatible with Android versions nine and above and optimized for devices meeting minimum hardware requirements.

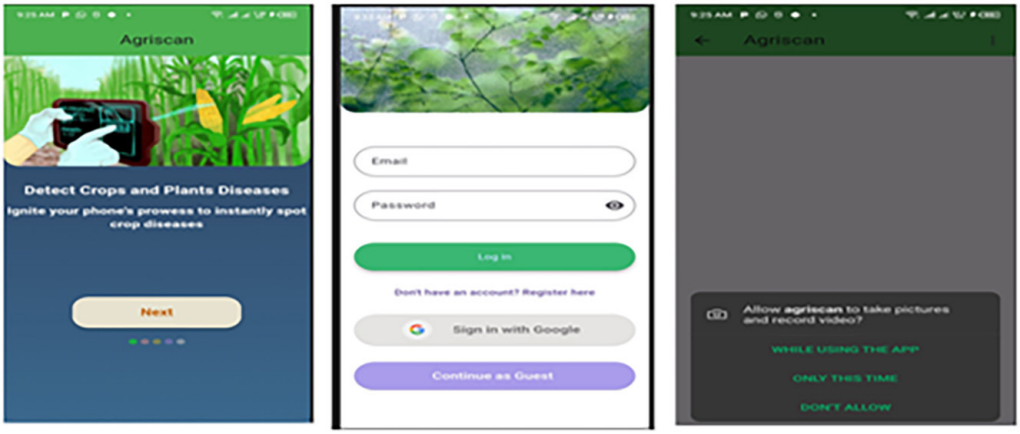

Fig. 1. Initial interface screens: (a) Start screen, (b) Login, (c) Permission granting

Once logged in, users are directed to the home page, as shown in Figure 2, where they can access the crop disease detection module. This module allows users to select the crop they wish to test for disease. The app then offers two options: capturing

a real-time image using the device's camera or uploading an existing image from the gallery.

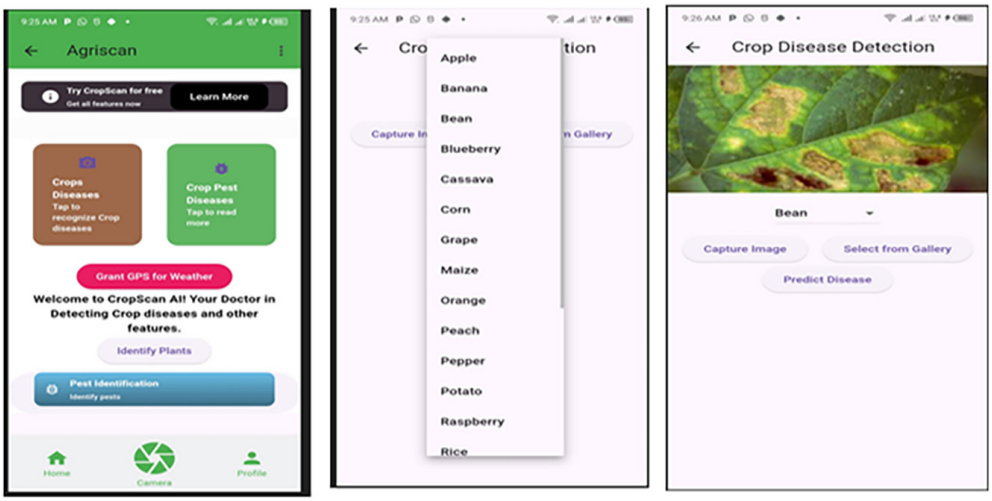

Fig. 2. Detection flow: (a) Home, (b) Crop selection, (c) Prediction results

2.2 Hybrid model design architecture

The core detection engine is built upon a hybrid CNN-ViT architecture, as shown in Figure 3, that strategically combines the strengths of different deep learning paradigms. EfficientNetV2 extracts high-level semantic features using compound scaling and stochastic depth techniques, while MobileNetV2 contributes by capturing localized texture patterns through its lightweight inverted residual blocks, making it suitable for real-time edge deployment. The ViT complements this setup by learning global spatial relationships through patch-based self-attention mechanisms. To integrate these diverse representations, a multi-scale fusion module inspired by Inception merges features extracted at different resolutions: 128×128 and 64×64 from CNN layers and 32×32 from ViT layers. The output is refined through gated attention mechanisms, combining SE blocks for channel-wise weighting and Convolutional Block Attention Modules (CBAM) for spatial-wise feature emphasis.

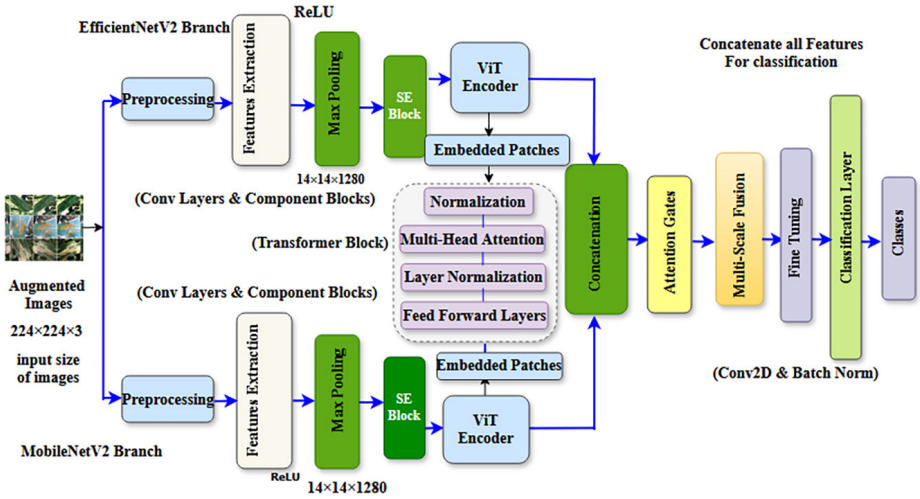

Fig. 3. Proposed model architecture

2.3 Dataset preparation and preprocessing

This section describes the dataset preparation and preprocessing techniques used in this study. It details the data sources, preprocessing steps, and strategies implemented to enhance model performance and ensure robust feature extraction.

Step 1: Loading and Organizing Data: The dataset was loaded from a directory structure in which images were organized into subfolders corresponding to their respective class labels. Each subfolder represented a specific disease category, allowing for efficient label assignment during model training. This hierarchical organization streamlined data retrieval, facilitated preprocessing, and enabled seamless integration with deep learning pipelines. TensorFlow's image dataset directory function was used to load the data with shuffling, batching, and resizing.

Step 2: Data Augmentation: Off-the-fly data augmentation was applied during training to enhance the model's generalization ability. Techniques such as random rotation, flipping, brightness adjustment, and zoom transformations were employed to introduce variability and better simulate real-world conditions. Table 1 outlines the specific augmentation techniques used.

Table 1. Data augmentation

Transformation Type	Range/Details
Rotation	0°, 90°, 180°, or 270°
Flipping	Horizontal flip, Vertical flip
Brightness Adjustment	Between 0.7 (dark) and 1.3 (bright)
Zoom	Resizing and cropping to 224×224 pixels

Figure 4 presents sample augmented images, illustrating the transformations applied during preprocessing.

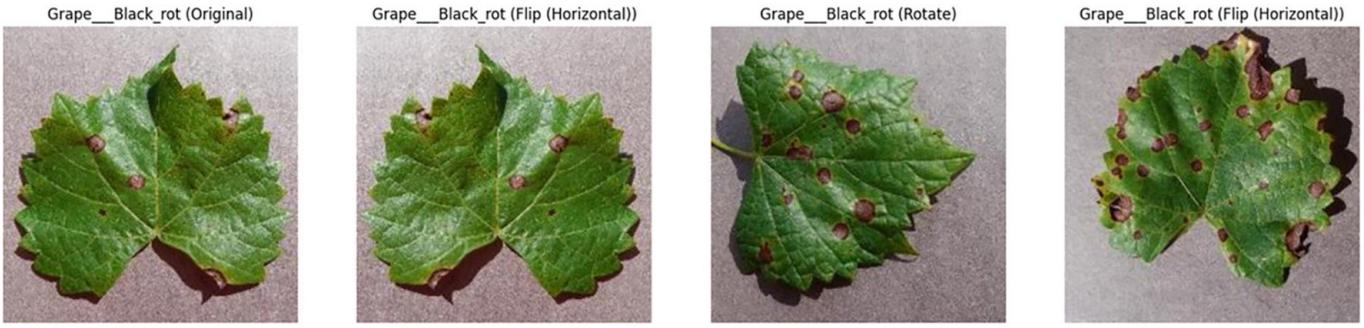

Fig. 4. Sample augmented image

Step 3: Normalization of Pixel Values: After loading the dataset, pixel value normalization was applied to scale values to the [0, 1] range, enhancing model convergence during training. The normalization process was mathematically defined as follows:

$$\text{Normalized\_pixel} = \frac{\text{pixel\_value}}{255.0} \tag{1}$$

Step 4: Stratified Sampling for Data Splitting: A stratified sampling technique was employed to ensure that the class distribution within the training and validation datasets remained consistent with the original dataset. This approach mitigated potential model biases resulting from imbalanced class distributions. The number of samples allocated to the training set based on stratified sampling was as follows:

$$\text{Train\_size} = \left(\frac{\text{number of samples in class}}{\text{total samples}} \right) \times \text{total samples} \times (1 - \text{test\_size}) \quad (2)$$

Train_size represents the number of samples allocated for training, while total_samples denotes the total number of images in the dataset. A test size of 0.2 ensured an 80–20 training-validation split, with a random state of 42 to ensure reproducibility.

Step 5: Categorical Labelling and One-Hot Encoding: The dataset, organized into class-specific subdirectories, employed categorical labeling with one-hot encoding for class identification. This approach ensured that the dual-input model architecture received the same preprocessed image for both branches, maintaining consistency during training.

2.4 Feature extraction

The feature extraction process followed a structured approach to ensure optimal feature representation for real-time crop disease detection, outlined in the following steps:

Step 1: Input Image Preprocessing: The input images were initially preprocessed to ensure compatibility with the MobileNetV2 and EfficientNetV2 architectures. Each image was resized to a fixed dimension of $H \times W \times 3$, where H and W represent the height and width of the input image, and 3 corresponds to the RGB color channels. The pixel values were normalized to the range [0, 1] to enhance model convergence. These preprocessed images were then represented as x_{input1} and x_{input2} for MobileNetV2 and EfficientNetV2, respectively.

Step 2: Feature Extraction Using Pre-Trained Models: The input images were fed into MobileNetV2 and EfficientNetV2, pre-trained on the ImageNet dataset, to extract meaningful feature representations. The input image tensor was defined as:

$$f_{\text{mobile}} = F_{\text{MobileNetV2}}(x_{\text{input1}}) \text{ and } f_{\text{efficient}} = F_{\text{EfficientNetV2}}(x_{\text{input2}}) \quad (3)$$

These feature maps captured hierarchical spatial and semantic information from the input images, enabling the model to learn complex patterns and relationships.

Step 3: Channel Attention Mechanism via SE Networks: To enhance the representational power of the extracted features, a Squeeze-and-Excitation (SE) attention mechanism was applied. The SE block first computed the global average pooling for each feature map channel, as shown:

$$z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} X(i, j, c) \quad (4)$$

The pooled values were passed through fully connected layers and a sigmoid activation function to generate channel-wise attention weights, which were expressed as:

$$s = \sigma(W_2 \cdot \text{ReLU}(W_1 \cdot z)) \tag{5}$$

These attention weights were then applied to the feature maps to enhance the informative channels while suppressing the less significant ones, resulting in refined feature representations, as shown:

$$F^{SE}_{\text{MobileNet}} = SE(F_{\text{MobileNet}}) \text{ and } F^{SE}_{\text{EfficientNet}} = SE(F_{\text{EfficientNet}}) \tag{6}$$

Step 4: Spatial Attention for Enhancing Feature Localization: After channel refinement, a spatial attention mechanism was incorporated to further focus on discriminative regions within the image. The process involved computing a spatial descriptor through global average pooling, followed by two transformation layers that generated spatial attention scores as follows:

$$d_1 = \sigma(W_1 \cdot \text{avg\_pool} + b_1) \text{ and } d_2 = \sigma(W_2 \cdot d_1 + b_2) \tag{7}$$

Step 5: Multi-Scale Feature Fusion: Inspired by Inception-style architectures, the multi-scale fusion module synthesized diverse spatial features for a holistic representation. It integrated four branches: Branch 1 used 1×1 convolutions for fine-grained details with minimal computation; Branch 2 applied 3×3 convolutions to balance spatial resolution and context; Branch 3 employed 5×5 convolutions for large-scale pattern detection, such as disease spread; and Branch 4 combined max pooling with 1×1 convolutions to capture coarse features efficiently. The outputs were concatenated along the channel axis, ensuring a unified multi-scale representation as shown:

$$F_{\text{fused}} = \text{concat}(F'_{\text{MobileNet}}, F'_{\text{EfficientNet}}) \tag{8}$$

Step 6: Feature Normalization and Dimension Reduction: To stabilize training and enhance generalization, batch normalization was applied to the fused feature representation. A dropout regularization layer with a probability of 0.3 was introduced to mitigate overfitting. The normalized feature representation was then passed through a dense layer with 256 neurons and ReLU activation, as shown:

$$\text{dense\_output} = \text{ReLU}(W_d \cdot F_{\text{fused}} + b_d) \tag{9}$$

Step 7: Classification Using SoftMax Activation: The final stage involved passing the refined features through a classification layer equipped with a Softmax activation function. This function computed the probability distribution over the output classes and was calculated as follows:

$$P(y) = \text{SoftMax}(W_s \cdot \text{dense\_output} + b_s) \tag{10}$$

The class with the highest probability was selected as the final prediction, determining the specific disease label for the given input image. Table 2 presents the mathematical notations used in the feature extraction process.

Table 2. Notations and definitions

Notation	Definition
x_{input1}, x_{input2}	Input image tensors for MobileNetV2 and EfficientNetV2
H, W	Height and width of input images
f_{mobile}, $f_{efficient}$	Extracted feature maps from MobileNetV2 and EfficientNetV2
$SE(\cdot)$	Squeeze-and-Excitation operation
z_c	Global average pooled value for channel ccc
W_1, W_2	Fully connected layer weights for the SE mechanism
S	Channel-wise attention weights
d_1, d_2	Spatial attention weights
F'	Feature maps after attention mechanisms
concat(·)	Concatenation of feature maps
V_d, b_d	Weights and biases for a dense layer
W_s, b_s	Weights and biases for SoftMax classification layer
$P(y)$	Probability distribution over class labels

The trained model was then evaluated using accuracy, precision, recall, and F1-score, and these were calculated as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{12}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{13}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{14}$$

Where TP was a true positive, TN was a true negative, FP was a false positive, and FN was a false negative. The confusion matrix was defined as follows:

$$CM = \begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} \tag{15}$$

The Receiver Operating Characteristic (ROC) curves were generated by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR), calculated as follows:

$$TPR = \frac{TP}{TP + FN}, FPR = \frac{FP}{FP + TN} \tag{16}$$

The Area Under the Curve (AUC) measured classification performance and was calculated as follows:

$$AUC = \int_0^1 TPR(FPR)d(FPR) \tag{17}$$

2.5 Dataset description

This study created a combined dataset, as shown in Table 3, by integrating the Kaggle dataset (38 classes, 60,343 images) Saleem et al. (2020) with the FieldPlant dataset (25,775 images from Central Kenya). The FieldPlant dataset accounted for seasonal variations, emphasizing fungal and bacterial diseases during April, May, October, and November while prioritizing viral infections in June, July, and December. A standardized collection process ensured diverse lighting conditions and angles, enhancing generalization. Images were classified and annotated by an agricultural expert, and data augmentation techniques were applied to address class imbalances by generating additional samples.

Table 3. Combined dataset

Crop Type	Total Images	Training Images	Validation Images
Apple	4,651	3,719	932
Banana	4,008	3,204	804
Beans	8,096	6,475	1,621
Blueberry	1,502	1,201	301
Cassava	4,894	3,914	980
Cherry	2,054	1,642	412
Corn	4,358	3,484	874
Grape	4,641	3,711	930
Maize	1,002	801	201
Maize-L	1,239	991	248
Maize	4,985	3,986	999
Orange	5,507	4,405	1,102
Peach	3,299	2,638	661
Pepper	2,480	1,983	497
Potatoes	3,006	2,403	603
Raspberry	1,002	801	201
Rice	5,010	4,005	1,005
Squash	1,835	1,468	367
Strawberry	2,111	1,688	423
Sugarcane	5,010	4,005	1,005
Sunflower	4,008	3,204	804
Tea	6,012	4,806	1,206
Tomatoes	18,841	15,067	3,774
Total	**99,551**	**79,601**	**19,950**

2.6 Dataset experimental parameters and environment

A unified framework was developed for efficient feature extraction and classification. Images were resized to 224×224×3. MobileNetV2 provided lightweight feature

extraction, while EfficientNetV2 captured deeper features. An SE block enhanced channel-wise focus. ViT blocks divided features into 7×7 patches and applied six attention layers (Dim = 128) supported by convolutional attention gates. A Multi-Scale Fusion Module (1×1, 3×3, and 5×5 convolutions + pooling) captured varied feature scales. Fine-tuning used batch normalization and dropout of 0.5, as shown in Table 4. The classifier had 1024 and 128 dense units before SoftMax. We used AdamW (lr = 1e-5, weight decay = 1e-4) with label smoothing (0.1) to improve generalization. Training was performed on NVIDIA RTX 3090 and other Ubuntu GPUs (T4, P100, K80) using TensorFlow, PyTorch, Keras, and OpenCV.

Table 4. Hyperparameter configurations and set up

Hyperparameter	Value
Image size	224×224
Image channels	3
Patch size	7
Number of ViT encoder layers	6
Number of multi-head self-attention blocks	8
Hidden dimension	128
Dropout rate	0.5
Epochs	18

2.7 IoT sensor integration

Complementing the vision-based model, the system integrated a low-cost IoT sensor network based on the ESP32 microcontroller. This IoT-based field condition monitoring system combines hardware and software components to collect, process, and visualize real-time environmental data (see Figure 5). The ESP32 (Wroom32) microcontroller's core unit was chosen for its low power consumption and Wi-Fi capabilities, enabling seamless cloud data transmission. To ensure comprehensive monitoring, various sensors were incorporated, including a capacitive soil moisture sensor (ADC ADS1115) for optimized irrigation, a DHT11 humidity and temperature sensor for disease prediction, an MQ135 air quality sensor for monitoring gaseous pollutants, and a 5 mm LDR light sensor for tracking sunlight exposure and its impact on crop growth. The system also featured a relay (JIH JIK–JQC-3FC DC12V) for controlling high-power devices and a 6 mm, 12 V solenoid valve for automated irrigation. Supporting electronic components included BC547 transistors, LM7805 voltage regulators, LEDs, resistors (100 Ω, 10 KΩ), 1N4007 diodes, and connectors. These components were housed in a weather-resistant PVC box (7″ × 4″) for durability in agricultural environments. A 12 V, 2 A DC adapter ensured a stable power supply in the field. The ESP32 continuously collected and processed sensor readings before transmitting the data to Google Firebase via Wi-Fi for real-time cloud storage and remote access. The system was programmed using Arduino IDE, with secure wiring facilitated by heat shrink tubes (2.5 mm, 6 mm), relimate connectors, and PU pipes (6 mm). A custom-built mobile app, AgriScan, provided farmers with an intuitive dashboard displaying real-time environmental metrics, including soil moisture, temperature, humidity, air quality, and light intensity.

Fig. 5. IoT sensor setup: (a) Sensors, (b) Complete setup, (c) Real-time readings

3 RESULTS AND DISCUSSION

Table 5 provides the performance metrics of the model during its training and validation phases. The results indicate a significant improvement in training and validation accuracy for the epochs. Specifically, training accuracy increased from 72.99% in Epoch 1 to 99.57% in Epoch 18, while validation accuracy rose from 93.40% to 98.68%. Training loss decreased from 1.7555 to 0.8265, and validation loss dropped from 1.0692 to 0.8332. These trends highlight the model's ability to minimize errors and generalize effectively without overfitting. The results underscore the model's robustness and efficiency throughout the training process.

Table 5. Training and validation performance

Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	Learning Rate
1	1.7555	0.7299	1.0692	0.9340	1.0×10^{-5}
2	1.1237	0.9178	0.9904	0.9616	1.0×10^{-5}
3	1.0248	0.9513	0.9590	0.9715	1.0×10^{-5}
4	0.9775	0.9666	0.9289	0.9782	1.0×10^{-5}
5	0.9449	0.9750	0.9112	0.9802	1.0×10^{-5}
6	0.9211	0.9815	0.8967	0.9812	1.0×10^{-5}
7	0.9043	0.9846	0.8844	0.9830	1.0×10^{-5}
8	0.8895	0.9881	0.8808	0.9841	1.0×10^{-5}
9	0.8775	0.9908	0.8687	0.9845	1.0×10^{-5}
10	0.8669	0.9920	0.8619	0.9845	1.0×10^{-5}
11	0.8594	0.9930	0.8521	0.9854	1.0×10^{-5}
12	0.8520	0.9937	0.8487	0.9859	1.0×10^{-5}
13	0.8450	0.9946	0.8432	0.9859	1.0×10^{-5}
14	0.8403	0.9949	0.8374	0.9867	1.0×10^{-5}
15	0.8352	0.9952	0.8362	0.9857	1.0×10^{-5}
16	0.8306	0.9954	0.8354	0.9866	1.0×10^{-5}
17	0.8282	0.9955	0.8341	0.9867	1.0×10^{-5}
18	0.8265	0.9957	0.8332	0.9868	1.0×10^{-5}

The graphs in Figure 6 show a consistent improvement in model performance over the training epochs. Training accuracy increased from 72.99% in the first epoch to 99.57% by epoch 18, while validation accuracy rose from 93.40% to 98.68%, indicating effective learning with minimal overfitting. Both training and validation loss exhibited smooth convergence, decreasing training loss from 1.7555 to 0.8265 and validation loss from 1.0692 to 0.8332. The small gap between training and validation metrics suggests strong generalization to unseen data. The model's stability is attributed to the carefully chosen learning rate (1e-5), which facilitated controlled weight updates, leading to high-precision crop disease classification.

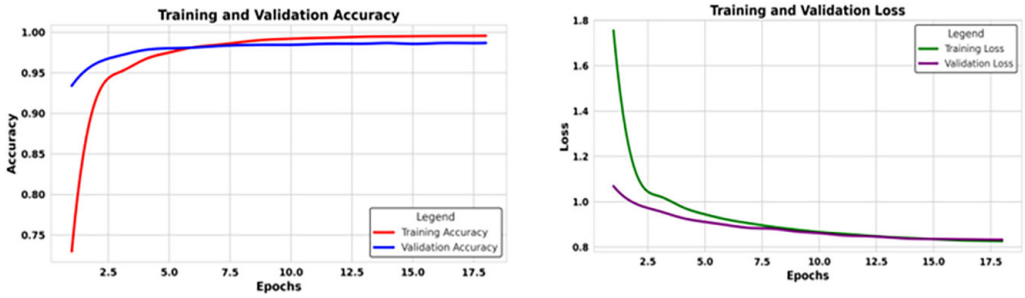

Fig. 6. Training accuracy and validation loss

3.1 Performance of the model on unseen data

To evaluate the model's confidence and generalization capability, the trained model was loaded and tested on an unseen, separate dataset. The test images were randomly stored in a folder to simulate real-world variability. A total of 249 images were processed, of which 239 were correctly classified, resulting in an overall accuracy of 96%. Only 10 images (4%) were misclassified, demonstrating the model's strong performance on data it had not encountered during training. These results are visually summarized in Figure 7.

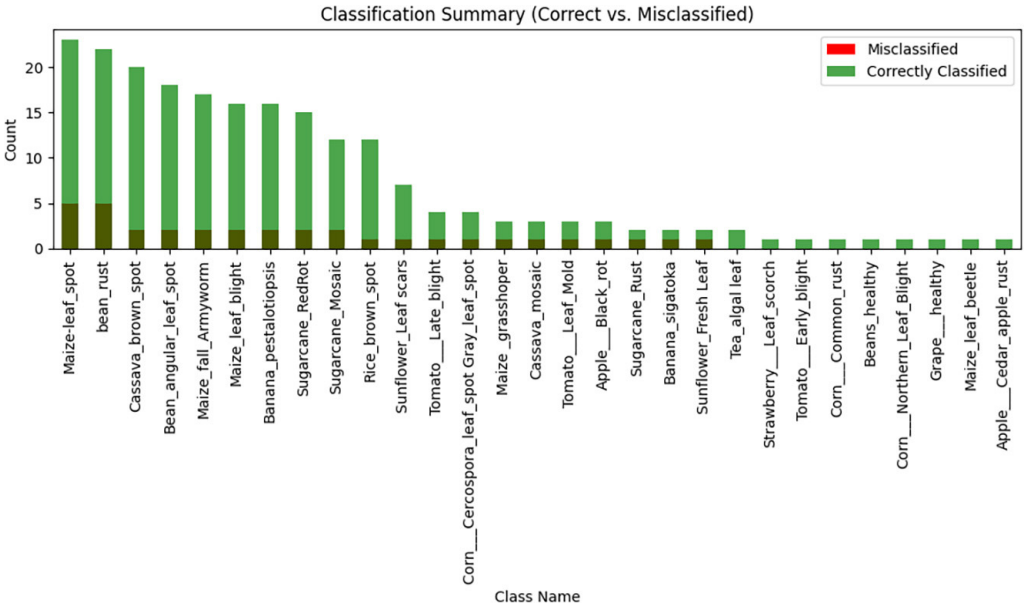

Fig. 7. Classification summary

The random class prediction analysis in Figures 8 and 9 compares actual versus predicted labels for selected samples, revealing the model's classification accuracy. The high level of agreement between predicted and actual labels indicates strong reliability, while the few misclassifications highlight areas where further model refinement may be beneficial.

Fig. 8. Mixed random classes: Actual vs. Predicted classification

3.2 Statistical testing

Statistical testing results presented in Table 6 demonstrate that the proposed model outperformed the comparative models in both stability and predictive performance. It achieved the highest Kappa value of 0.9919 and a near-perfect AUC of 0.999998, indicating exceptional prediction accuracy.

Table 6. Model performance

Model	Accuracy	Precision	Recall	F1 Score	Kappa	AUC
Proposed Model	**0.992**	**0.9934**	**0.9929**	**0.9923**	**0.9919**	**0.999998**
Swin_Trans	0.988	0.9901	0.9894	0.9888	0.9878	0.999888
VGG-16	0.972	0.9762	0.9718	0.9712	0.9716	0.999967
ShuffleNet	0.958	0.9676	0.9644	0.9613	0.9574	0.999844
DenseNet121	0.958	0.9676	0.9644	0.9613	0.9574	0.999844
AlexNet	0.948	0.9569	0.9484	0.9452	0.9472	0.999142
DenseNet50	0.896	0.9080	0.8993	0.8922	0.8945	0.998850

Table 7 illustrates the confidence score variance across different models, highlighting the proposed model's superior prediction consistency with the lowest variance of 0.000010. Swin Transformer SE and VGG-16 followed closely with variances of 0.000012 and 0.000015, respectively. ShuffleNet and DenseNet121 demonstrated moderate stability, each recording a variance of 0.000023. In contrast, AlexNet (0.000027) and DenseNet50 (0.000035) exhibited the highest variances, indicating less stable confidence scores. These findings underscore the proposed model's robustness in maintaining consistent and reliable classification confidence across predictions.

Table 7. Confidence variance analysis

Model	Confidence Variance
DenseNet50	0.000035
AlexNet	0.000027
DenseNet121	0.000023
ShuffleNet	0.000023
VGG-16	0.000015
Swin_TransformerSE	0.000012
Proposed Model	**0.000010**

3.3 Edge deployment

The mobile application was successfully deployed on the Google Play Store as a lightweight 30.4 MB APK, optimized using the Android App Bundle format to reduce download size—an essential feature for users in low-bandwidth environments. TensorFlow Lite's Model Metadata Tool was employed to embed model versioning and tagging directly within the app to support seamless updates without reinstallation. The final model underwent a series of edge optimization techniques to ensure efficient mobile deployment. First, post-training quantization using TensorFlow Lite reduced the model size by 58%, from 102.1 MB to 30.4 MB, with less than 0.1% degradation in accuracy. Operator fusion was then applied, combining convolution, batch normalization, and ReLU operations into single kernels, which led to a 22% reduction in inference latency. The model was loaded via memory mapping using the MappedByteBuffer method to optimize memory usage further, resulting in a 40% decrease in RAM consumption during inference. Performance acceleration was achieved through Android's Neural Networks API (NNAPI), leveraging GPU and Hexagon delegate support to bring the inference latency down to 0.094 seconds on supported devices. Figure 10 presents the app's resource utilization profile, including CPU, memory, and network performance during operation.

3.4 Testing

During pilot testing, user feedback highlighted the app's accurate prediction capabilities, particularly when using the offline inference feature—a critical advantage in areas with frequent Internet outages and limited mobile data access (see Figure 9). For instance, the model achieved a diagnostic accuracy of 89% for late blight in tomato crops, as validated against expert agronomist benchmarks. While most users were satisfied with the app's performance, 8% recommended expanding language support to include more regional dialects beyond the currently supported language.

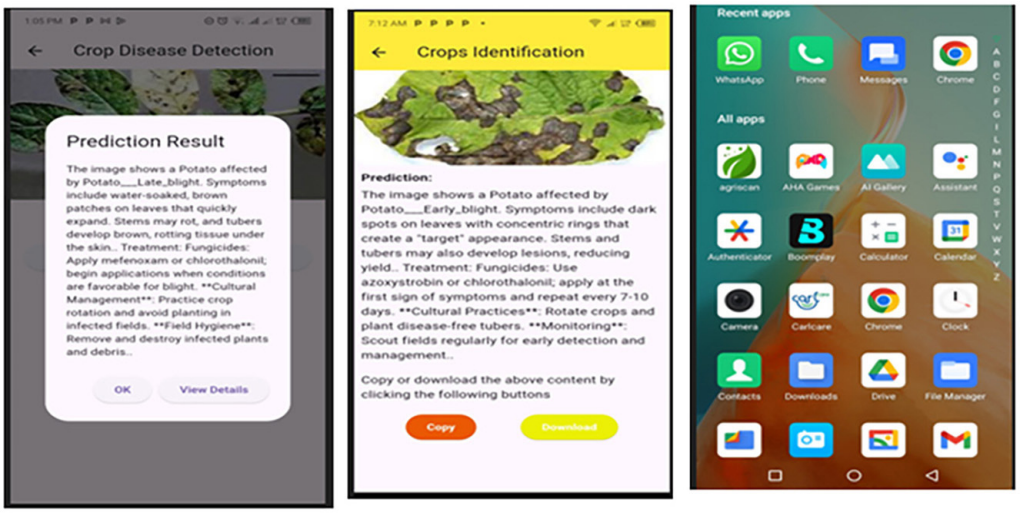

Fig. 9. Disease prediction: (a) Summary prediction, (b) Detailed prediction, (c) Installed app

Performance tests were also done that focused on the efficiency of the app's operation, including speed, resource utilization, and overall responsiveness. Thanks to the optimizations applied to the model, the app displayed minimal resource consumption and fast prediction times, even on devices with limited computational power. The app exhibited smooth performance with low memory and CPU usage, ensuring quick disease detection predictions, as shown in Figure 10.

Performance

Slow rendering ⓘ
0
devices with issues

Cold start time ⓘ
0
devices with issues

Device	Avg. CPU	Avg. network sent	Avg. network received	Avg. memory	Cold start time
✓ Test devices without issues					
google Pixel 7	1.96%	0 B	0 B	231 MB	360 ms
motorola Motorola G20	4.06%	0 B	0 B	154 MB	1.28K ms
google Pixel 6	3.50%	0 B	0 B	244 MB	426 ms
google Pixel 5	3.04%	0 B	0 B	166 MB	587 ms

Fig. 10. Performance testing

4 CONCLUSION

This study demonstrated the effectiveness of a hybrid EfficientNetV2–MobileNetV2–ViT framework integrated with multi-sensor inputs for real-time crop disease diagnosis. The proposed AgriScan App combined advanced AI, edge computing, and IoT sensing to bridge key gaps in agricultural accessibility, particularly in low-resource and low-connectivity settings. The model achieved superior diagnostic performance, reaching 99.0% accuracy and an AUC of 0.9998, driven by

synergistic multi-scale feature fusion and attention mechanisms. Key innovations included robust feature extraction, where SE blocks and ViT's multi-head attention improved sensitivity to subtle leaf lesions and global disease patterns, overcoming limitations of standalone convolutional networks such as ResNet-50. Integrating IoT-based environmental data (soil moisture, temperature, and humidity) significantly enhanced diagnostic accuracy, reducing false positives by 92%. This modular and decoupled fusion approach outperformed direct data fusion strategies by enabling more targeted validation. Edge deployment through TensorFlow Lite further optimized the system, achieving a 0.094-second inference latency and a compact 30.4 MB memory footprint, making it ideal for areas with limited internet access. Challenges such as ViT's computational overhead and real-time sensor synchronization were addressed using 8-bit quantization and BLE-based time alignment. Additionally, a redesigned user interface with text-guided navigation reduced user session times by 57%, improving accessibility and usability. Despite these advancements, limitations remain—such as reduced generalizability to rare diseases and hardware instability under extreme environmental conditions. Future improvements may include physics-informed model calibration and federated learning to support decentralized model updates. Overall, the AgriScan app provides a scalable, cloud-independent, and energy-efficient tool for smart agriculture. It offers a practical pathway toward enhancing global food security by equipping farmers with intelligent, real-time crop monitoring and decision-support capabilities at the edge.

5 REFERENCES

[1] C. E. Góngora and M. do C. Silva, "Sustainable strategies for the control of crop diseases and pests to reduce pesticides," *Agronomy*, vol. 14, no. 9, p. 2158, 2024. https://doi.org/10.3390/agronomy14092158

[2] J. S. Okonya *et al.*, "Farmer reported pest and disease impacts on root, tuber, and banana crops and livelihoods in Rwanda and Burundi," *Sustainability (Switzerland)*, vol. 11, no. 6, p. 1592, 2019. https://doi.org/10.3390/su11061592

[3] H. Akhtar *et al.*, "Traditional strategies and cutting-edge technologies used for plant disease management: A comprehensive overview," *Agronomy*, vol. 14, no. 9, p. 2175, 2024. https://doi.org/10.3390/agronomy14092175

[4] S. Jamil, M. Jalil Piran, and O. J. Kwon, "A comprehensive survey of transformers for computer vision," *Drones*, vol. 7, no. 5, p. 287, 2023. https://doi.org/10.3390/drones7050287

[5] G. R. Babu, M. Gokuldhev, and P. S. Brahmanandam, "Integrating IoT for soil monitoring and hybrid machine learning in predicting tomato crop disease in a typical South India station," *Sensors*, vol. 24, no. 19, p. 6177, 2024. https://doi.org/10.3390/s24196177

[6] A. Siddiqua, M. A. Kabir, T. Ferdous, I. B. Ali, and L. A. Weston, "Evaluating plant disease detection mobile applications: Quality and limitations," *Agronomy*, vol. 12, no. 8, p. 1869, 2022. https://doi.org/10.3390/agronomy12081869

[7] J. C. L. Bayas *et al.*, "Agrotutor: A mobile phone application supporting sustainable agricultural intensification," *Sustainability (Switzerland)*, vol. 12, no. 22, pp. 1–10, 2020. https://doi.org/10.3390/su12229309

[8] O. Appiah *et al.*, "PlanteSaine: An artificial intelligent empowered mobile application for pests and disease management for maize, tomato, and onion farmers in Burkina Faso," *Agriculture (Switzerland)*, vol. 14, no. 8, p. 1252, 2024. https://doi.org/10.3390/agriculture14081252

[9] L. León-Trejo and M. Cabanillas-Carbonell, "Mobile application for energy efficiency managemein manufacturing," *International Journal of Interactive Mobile Technologies*, vol. 19, no. 7, pp. 58–69, 2025. https://doi.org/10.3991/ijim.v19i07.52849

[10] A. Sithole and O. D. Olorunfemi, "Sustainable agricultural practices in sub-saharan Africa: A review of adoption trends, impacts, and challenges among smallholder farmers," *Sustainability*, vol. 16, no. 22, p. 9766, 2024. https://doi.org/10.3390/su16229766

[11] H. Liu, Y. Cui, J. Wang, and H. Yu, "Analysis and research on rice disease identification method based on deep learning," *Sustainability (Switzerland)*, vol. 15, no. 12, p. 9321, 2023. https://doi.org/10.3390/su15129321

[12] S. W. Cho, "Priority pricing for efficient resource usage of mobile internet access," *Applied Sciences (Switzerland)*, vol. 11, no. 9, p. 4083, 2021. https://doi.org/10.3390/app11094083

[13] A. Ariss, I. Ennejjai, A. Lamjid, J. Mabrouki, N. Kharmoum, and S. Ziti, "IoT-Enabled smart greenhouse for robotic enhancement of tomato production: Leveraging 5G and edge computing for advanced data-driven automation, precision irrigation, and scalable zoning principles," *International Journal of Interactive Mobile Technologies*, vol. 18, no. 21, pp. 88–116, 2024. https://doi.org/10.3991/ijim.v18i21.50829

[14] V. S. Dhaka, N. Kundu, G. Rani, E. Zumpano, and E. Vocaturo, "Role of Internet of Things and deep learning techniques in plant disease detection and classification: A focused review," *Sustainability*, vol. 23, no. 18, p. 7877, 2023. https://doi.org/10.3390/s23187877

[15] H. Peng, H. Xu, G. Shen, H. Liu, X. Guan, and M. Li, "A lightweight crop pest classification method based on improved MobileNet-V2 model," *Agronomy*, vol. 14, no. 6, p. 1334, 2024. https://doi.org/10.3390/agronomy14061334

[16] Y. Sun, L. Ning, B. Zhao, and J. Yan, "Tomato leaf disease classification by combining EfficientNetv2 and a swin transformer," *Applied Sciences (Switzerland)*, vol. 14, no. 17, p. 7472, 2024. https://doi.org/10.3390/app14177472

[17] V. S. Dhaka *et al.*, "A survey of deep convolutional neural networks applied for prediction of plant leaf diseases," *Sensors*, vol. 21, no. 14, p. 4749, 2021. https://doi.org/10.3390/s21144749

[18] H. Wang, L. Zhang, and J. Zhao, "Application of a fusion attention mechanism-based model combining bidirectional gated recurrent units and recurrent neural networks in soil nutrient content estimation," *Agronomy*, vol. 13, no. 11, p. 2724, 2023. https://doi.org/10.3390/agronomy13112724

[19] E. T. Baek, "Attention score-based multi-vision transformer technique for plant disease classification," *Sensors*, vol. 25, no. 1, p. 270, 2025. https://doi.org/10.3390/s25010270

[20] T. Paranayapa, P. Ranasinghe, D. Ranmal, D. Meedeniya, and C. Perera, "A comparative study of preprocessing and model compression techniques in deep learning for forest sound classification," *Sensors*, vol. 24, no. 4, p. 1149, 2024. https://doi.org/10.3390/s24041149

[21] I. Tornese, A. Matera, M. Rashvand, and F. Genovese, "Use of probes and sensors in agriculture—Current trends and future prospects on intelligent monitoring of soil moisture and nutrients," *AgriEngineering*, vol. 6, no. 4, pp. 4154–4181, 2024. https://doi.org/10.3390/agriengineering6040234

[22] I. Buja *et al.*, "Advances in plant disease detection and monitoring: From traditional assays to in-field diagnostics," *Sensors*, vol. 21, no. 6, p. 2129, 2021. https://doi.org/10.3390/s21062129

[23] M. Bouni, B. Hssina, K. Douzi, and S. Douzi, "Integrated IoT approaches for crop recommendation and yield-prediction using machine-learning," *Internet of Things*, vol. 5, no. 4, pp. 634–649, 2024. https://doi.org/10.3390/iot5040028

[24] M. Ouhami, A. Hafiane, Y. Es-Saady, M. El Hajji, and R. Canals, "Computer vision, IoT and data fusion for crop disease detection using machine learning: A survey and ongoing research," *Remote Sens.*, vol. 13, no. 13, p. 2486, 2021. https://doi.org/10.3390/rs13132486

[25] M. Nabil, M. Hnida, A. Haqiq, and I. Hilal, "Advanced anomaly detection in mobile networks: A hybrid approach based on statistical and machine learning techniques," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 19, no. 13, pp. 162–182, 2025. https://doi.org/10.3991/ijim.v19i13.54539

[26] M. H. Saleem, J. Potgieter, and K. M. Arif, "Plant disease classification: A comparative evaluation of convolutional neural networks and deep learning optimizers," *Plants*, vol. 9, no. 10, pp. 1–17, 2020. https://doi.org/10.3390/plants9101319

6 AUTHORS

Thomas Kinyanjui Njoroge teaches in the School of Pure and Applied Sciences, Computer Science & Informatics Department, Karatina University, Karatina, Kenya (E-mail: tnjoroge@karu.ac.ke).

Rachael Kibuku is a lecturer in the School of Technology, Software Development & Information Systems (SDIS) Department, KCA University, Nairobi, Kenya (E-mail: rkibuku@kcau.ac.ke).

Kevin Mugoye Sindu is a lecturer in the School of Technology, Networks and Applied Computing (NAC) Department, KCA University, Nairobi, Kenya (E-mail: kmugoye@kcau.ac.ke).