Smart Agricultural Technology 9 (2024) 100627

Contents lists available at ScienceDirect

Smart Agricultural Technology

journal homepage: www.journals.elsevier.com/smart-agricultural-technology

Leveraging edge computing and deep learning for the real-time identification of bean plant pathologies

Andrew Katumba [a,*], Wayne Steven Okello [a], Sudi Murindanyi [b], Joyce Nakatumba-Nabende [b], Moses Bomera [a], Ben Wycliff Mugalu [b], Amos Acur [c]

[a] Department of Electrical and Computer Engineering, Makerere University, Kampala, P.O. Box 7062, Uganda
[b] Department of Computer Science, Makerere University, Kampala, P.O. Box 7062, Uganda
[c] National Agricultural Research Organization, Plot 11-13, Lugard Avenue, Entebbe, P.O. Box 295, Uganda

ARTICLE INFO

Dataset link: Bean Plant Pathologies Dataset for Deep Learning Tasks (Original data)

Keywords:
Bean disease detection
Deep learning
ALS
Bean rust

ABSTRACT

Beans are essential crops globally, standing out as one of the most consumed and nourishing legumes, thereby playing a significant role in human nutrition and food security. Their cultivation faces several challenges, such as pests, diseases, unpredictable weather patterns, and soil erosion. Of these challenges, diseases are recognized as a key challenge, resulting in a decline in both yield quality and quantity, and inflicting substantial financial losses on farmers.

This work proposes a deep learning-based approach for precise in-field identification of diseases in bean plants. We evaluate image classification and object detection models using state-of-the-art Convolutional Neural Network (CNN) architectures to identify Angular Leaf Spot (ALS) and bean rust diseases, key bean diseases in Uganda and the region in general, from smartphone images of bean leaves collected in various districts of Uganda.

The dataset employed to train these models is the Makerere University beans image dataset, comprising 15,335 images categorized into three (3) classes (ALS, bean rust, and healthy). To improve in-field performance, the dataset was expanded to include an additional class (unknown class) consisting of a diverse collection of 2,800 images to account for images unrelated to the three (3) predefined classes. Adversarial training was further employed to enhance model robustness in identifying the target classes. In addition, two (2) Out-of-Distribution (ODD) detection techniques, i.e., confidence thresholding and training with an auxiliary class (unknown class), were utilized to handle inputs unrelated to bean leaves.

Our results show that our custom CNN, BeanWatchNet, achieved 90% accuracy when tested on unseen data for the classification of the three (3) target classes, i.e., ALS, bean rust and healthy. EfficientNet v2 B0 and BeanWatchNet demonstrated superior performance for the four-class (with unknown class) image classification task, achieving 91% and 90% accuracy, respectively, when evaluated on the test dataset. YOLO v8 exhibited superior performance for the object detection models, attaining mAP@50 of 87.6. The custom CNN model and YOLO v8 model were quantized and deployed across two (2) edge platforms: a smartphone (through a mobile application) and a Raspberry Pi 4B to facilitate in-field disease detection. The benchmarking code and models are publicly available on GitHub.[1]

1. Introduction

Beans (Phaseolus vulgaris) are one of the world's most extensively grown and consumed crops, making them crucial in global agriculture. They are relatively easy to grow and can flourish in a wide range of climatic conditions, from humid tropical regions to semi-arid areas. Approximately 30% of bean yields across the world are produced by smallholder farmers in Africa and Latin America [1]. In Eastern, Central, and Southern Africa, beans are the most significant legume, with an extensive 6.3 million hectares of land devoted to their cultivation each year. Remarkably, Eastern Africa leads the world in per capita bean consumption with more than 30 kg per person per year [2].

Beans contribute significantly to food security in Africa by ensuring consistent access to an adequate food supply for diverse populations

* Corresponding author.
E-mail address: andrew.katumba@mak.ac.ug (A. Katumba).
[1] https://github.com/Marconi-Lab/Real-Time-Identification-of-Bean-Plant-Pathologies.

https://doi.org/10.1016/j.atech.2024.100627
Received 10 June 2024; Received in revised form 23 October 2024; Accepted 27 October 2024
Available online 9 November 2024

across the continent. They are a crucial component in the staple diet and are an inexpensive source of essential nutrients such as protein, minerals, vitamins, and fiber [3]. Furthermore, beans improve soil fertility by fixing nitrogen in the soil. This reduces the necessity for synthetic fertilizers, making beans excellent candidates for crop rotation.

Bean farmers in Africa experience several challenges throughout the plant lifecycle, such as unpredictable weather patterns, pests, diseases, restricted access to improved varieties, and soil erosion, with diseases identified as one of the major challenges [4]. This greatly leads to decreased farm yields and, consequently, financial losses to farmers. According to [5], ALS alone is responsible for a massive output loss of 384.2 tons per year across Sub-Saharan Africa. Furthermore, the majority of the bean production is carried out by smallholder farmers who lack adequate mechanisms and information to diagnose and combat these diseases promptly.

The conventional method for identifying plant diseases involves domain experts using their naked eyes to visually inspect disease symptoms such as lesions, wilting, and discoloration on plant parts such as leaves, fruits, and stems. This manual process is tedious, labor intensive, and requires domain experts at every stage of disease diagnosis [3], [4], [6], [7]. Domain experts in Africa are not only limited in number in comparison with the available fields but are also prone to human errors due to fatigue, human bias, and visual impairments, thereby reducing the accuracy of the diagnosis.

Over the years, advanced techniques such as computer vision and machine learning, particularly deep learning, have proved to be superior alternatives to the traditional approaches in providing efficient and timely disease diagnosis [8], [9], [10], [11], [12]. CNNs are particularly effective in computer vision tasks such as image classification and object detection as they automatically extract hierarchical features from images, eliminating the need for intermediate feature engineering [13], [14], [15], [16], [17], [18]. These cutting-edge technologies aid in the early and accurate detection of plant diseases and can contribute to minimizing crop yield and financial losses for farmers that might arise from late disease identification [19], [20], [21], [22].

Several researchers have explored the application of deep learning approaches for the identification of diseases in bean plants. Singh et al. [8] employed transfer learning to classify images as healthy or exhibiting ALS and bean rust diseases. EfficientNet B6 outperformed the other algorithms, achieving a validation loss of 0.2849 and a validation accuracy of 91.74%. Elfatimi et al. [9] presented a deep-learning approach for the classification of diseases in bean leaves. MobileNet and MobileNet v2 were employed in training the models, with the former demonstrating superior performance, achieving an accuracy of 92%. Abed et al. [3] proposed a deep learning framework in robot vision for the early detection of bean leaf diseases. The framework comprised two (2) main stages. Firstly, a U-Net architecture was employed to detect bean leaves in the input images. Subsequently, several deep-learning models were utilized for the diagnosis of diseases in the detected bean leaves. DenseNet-121 exhibited the best performance for multi-classification, achieving an accuracy of 91.01%. A publicly available dataset consisting of 1295 images categorized into three classes: bean rust, ALS, and healthy was employed in [3], [8] and [9]. Banerjee et al. [20] developed a hybrid model employing CNNs and Support Vector Machines (SVM) for the detection of six (6) bean plant diseases, i.e., Anthracnose, Rust, Bean's Common Mosaic Virus, Angular Leaf Spot, Bean's Yellow Mosaic Virus, and Bacterial Blight. CNNs were utilized to extract high-level features from input images, and the SVM was used to classify images into disease categories based on these CNN-derived features. The model achieved a weighted average F1 score of 87.44% across all six (6) classes.

The approaches in [3], [8], [9], and [20] not only utilized relatively small and less diverse datasets but were also restricted to image classification models. Moreover, these image classification models misclassify unrelated images, such as those depicting a dog, grass, or a person, etc, as healthy, ALS, or bean rust, thus reducing their reliability in practical environments. Furthermore, no deployment for the trained models was reported to facilitate in-field disease diagnosis in the existing work.

In this study, we propose a deep learning-based approach to enhance in-field identification of ALS and bean rust diseases in bean plants. Our system architecture is illustrated in Fig. 1. The primary contributions of this paper are as follows:

- Image classification models trained using state-of-the-art CNN architectures to detect ALS and bean rust diseases in bean leaves. Adversarial training was employed to enhance model robustness against perturbed inputs while confidence thresholding was applied to handle ODD images. Furthermore, the models were trained with an additional class (unknown class) [23]. This improves reliability in in-field disease detection as images unrelated to bean leaves, such as those depicting the sky, a person or a hoe, are classified as unknown rather than as healthy, ALS, or bean rust.
- Object detection models trained using state-of-the-art CNN architectures to localize different areas affected by bean rust or ALS in bean leaves. This provides a more thorough analysis compared to image classification models that only assign a predefined class to the entire image. Additionally, the object detection models offer more reliable in-field disease diagnosis due to their ability to handle background separation.
- The BeanWatchNet and YOLO v8 models were quantized and deployed on two (2) edge platforms, i.e., a smartphone (through a mobile application) and a Raspberry Pi 4B, to ease in-field disease detection.

The subsequent sections of this paper are organized as follows: Section 2 provides a comprehensive overview of the methodology applied, Section 3 discusses the study's empirical results and summarizes the findings, Section 4 provides a discussion of the work, and finally, Section 5 comprises the conclusion of the work.

2. Materials and methods

2.1. Dataset

The Makerere University beans image dataset [24] contains 15,335 images classified into three categories: ALS, bean rust, and healthy. The dataset was collected in two batches. The first batch collected data for six (6) days from April 22nd, 2021, to April 27th, 2021, and focused on collecting images of healthy beans in the early stages of crop development. The second batch collected data over seven (7) days from May 21st, 2021, to May 27th, 2021, and primarily captured diseased bean leaves. The image class distribution in the dataset is illustrated in Table 1. The dataset was collected and curated by a collaborative team from the Makerere Artificial Intelligence Lab (Makerere AI Lab), Marconi Research and Innovations Lab (Marconi Lab), and crop pathology experts from the National Crops Resources Research Institute (NaCRRI) in Uganda. The dataset is available on HARVARD Dataverse [2] and a second version of the data, used specifically for this project, is also published on HARVARD Dataverse.[3]

Four (4) experts from NaCRRI annotated the dataset using an internally customized version of the VGG Image Annotator (VIA) [25] as illustrated in Fig. 2. The customizations were tailored towards providing efficient data management during the annotation process. The dataset was split into three (3) portions, each assigned to a single expert. The fourth expert, senior to the others, reviewed all the labels and annotations, confirming and correcting any errors. For classification, each image was tagged with a label for the three classes: ALS, bean rust, and

[2] https://doi.org/10.7910/DVN/TCKVEW.
[3] https://doi.org/10.7910/DVN/WFSLBY.

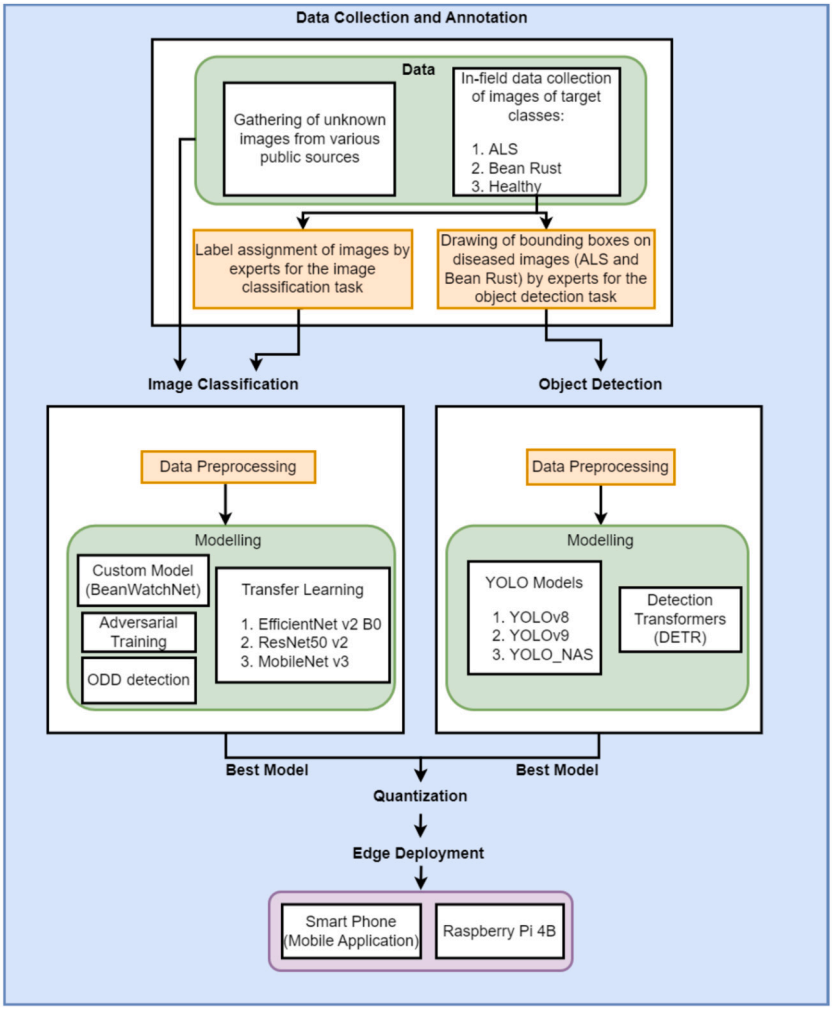

Fig. 1. System architecture showcasing the interrelations among workflow processes and tasks. It starts with data collection and annotation, and the annotated data is used for training image classification and object detection models, which are eventually quantized and deployed on a smartphone (through a mobile application) and a Raspberry Pi 4B.

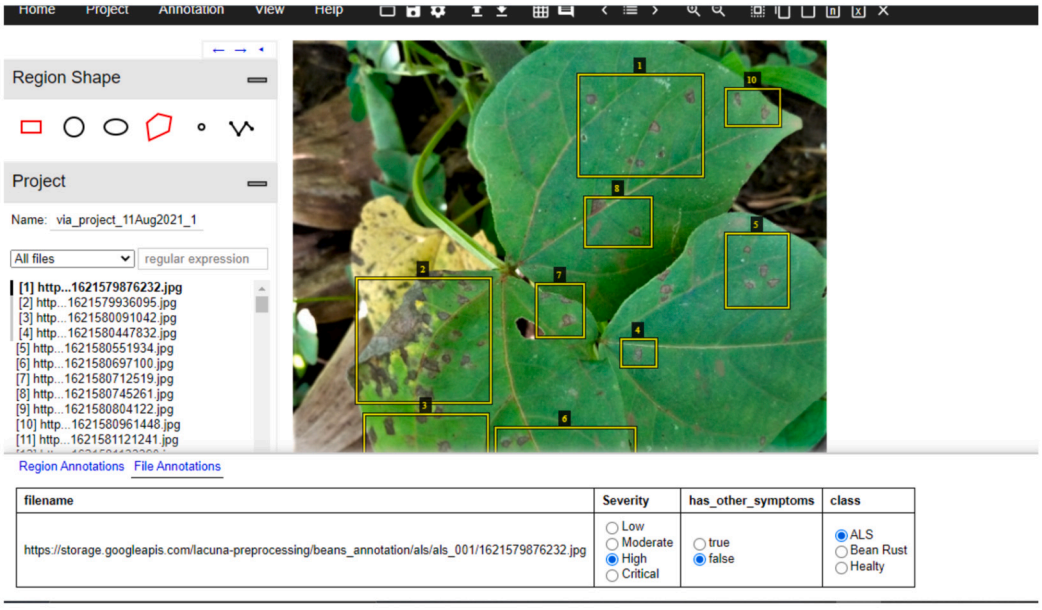

Fig. 2. A sample image of the tool used for the annotation process. The tool, a modified VIA, facilitated the annotators to draw bounding boxes around the disease-inflicted areas as well as capture image file attributes, i.e., severity, "other symptoms", and class.

Table 1
Class distribution of the dataset used for training, validation, and testing the ML models. This showcases the four (4) classes in the dataset utilized alongside their respective number of images.

Class	Number of Images
Healthy	5284
Bean Rust	5020
ALS	5031
Unknown	2800

Fig. 3. A sample image from the dataset representing healthy bean leaves.

healthy. The annotation for object detection involved drawing bounding boxes around the symptomatic areas on the leaflet. File attributes i.e., class, severity, and "other symptoms" were also captured for each image annotated. For severity, the annotators were required to make a judgment by inspecting how symptoms were spread across the lamina of the trifoliate. The "other symptoms" attribute was intended to indicate the existence of other symptoms on the leaf. This was due to the fact that during data collection, we often came across plants with other infections besides ALS and bean rust. The class attribute specifies the class of the image being annotated. This was intended to verify the class attributes that were captured during data collection.

Healthy bean leaves usually exhibit vibrant green coloration, well-defined veins and absence of spots or lesions as illustrated in Fig. 3. ALS manifests as angular or irregularly shaped lesions on the leaves, as demonstrated in Fig. 4, which may turn yellow and dry out. Bean rust is identifiable by the presence of yellow or orange pustules on the leaves, as illustrated in Fig. 5. Both the pustules and lesions exhibit variability in size and distribution across the leaf surface.

For the classification task, we further created an unknown class consisting of 2,800 images to account for ODD inputs i.e., images unrelated to the three (3) predefined classes as illustrated in Fig. 6. The unknown class contributes to the robustness of the trained models, enabling them to effectively handle the complexity of real-world scenarios in the identification of bean plant diseases. This dataset was created by compiling images from various public sources to capture a broad range of concepts unrelated to the task. The unknown class brought the number of classes for the classification task to 4.

ALS tends to be more prevalent in the latter stages of the growing season, whereas bean rust tends to affect bean plants throughout the entire season. This pattern is apparent in the age distribution of the dataset, as depicted in Fig. 7.

The dataset includes images of bean leaves from twelve (12) districts in Uganda, namely Bugiri, Hoima, Kayunga, Kiboga, Lyantonde, Mayuge, Mbale, Mubende, Ntungamo, Pallisa, Serere, and Sironko. These districts span three (3) regions, i.e., Eastern, Central and Western, as illustrated in Fig. 8. It also consists of bean leaves at various stages of development as illustrated in Fig. 7 and includes thirteen (13)

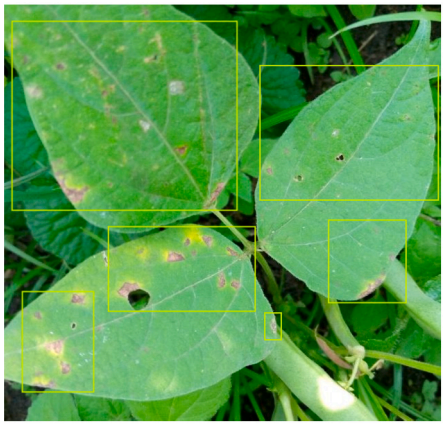

Fig. 4. A sample image of bean leaves affected by ALS disease. Bounding boxes are drawn around angular-shaped lesions on the leaves.

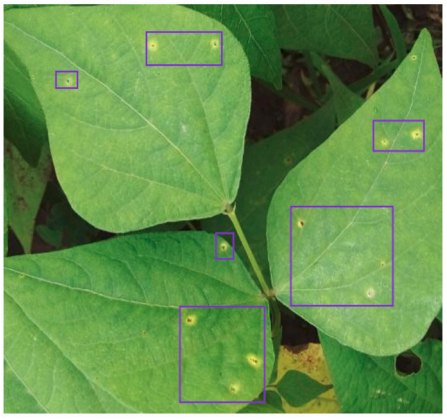

Fig. 5. A sample image of bean leaves affected by bean rust disease. Bounding boxes are drawn around yellow or orange pustules on the leaves.

different bean plant varieties i.e., Nambale, Nambale Short, Nambale Long, Saitoti, Kabonge, Kanyebwa, Masindi Yellow, Kamwanyi, NABE 2, Nabufumbo, Land Race, Kikoni and Kabulangiti.

2.2. Tools

The experiments in this research were primarily executed on an Alienware Aurora R12 featuring an i7-11700F @4.8 GHz CPU, 16 GB RAM, and an NVIDIA GeForce RTX 3060 Ti GPU. The Tensorflow framework was employed for the image classification models, while the Pytorch library was utilized for the object detection models.

2.3. Image classification

In image classification, the main objective was to swiftly and precisely assign a predefined label to the entire input image, ensuring the specific disease is identified with efficiency.

2.3.1. Data preprocessing

The images were resized to dimensions of 224x224. The TensorFlow ImageDataGenerator utility was utilized for pixel normalization and efficient loading of image data during training. The dataset was split into train, validation, and test sets in a ratio of 0.7:0.15:0.15.

2.3.2. Modeling

a. Custom Model (BeanWatchNet)

We developed a custom CNN model, BeanWatchNet, optimized for identifying bean rust and ALS diseases in bean leaves, healthy leaves,

Fig. 6. Sample images representing the unknown class. Besides the target classes of ALS, bean rust and healthy, images unrelated to bean leaves such as those depicting other crops, cars, the sky, houses, sports activities, etc. were added to the dataset to form the unknown class.

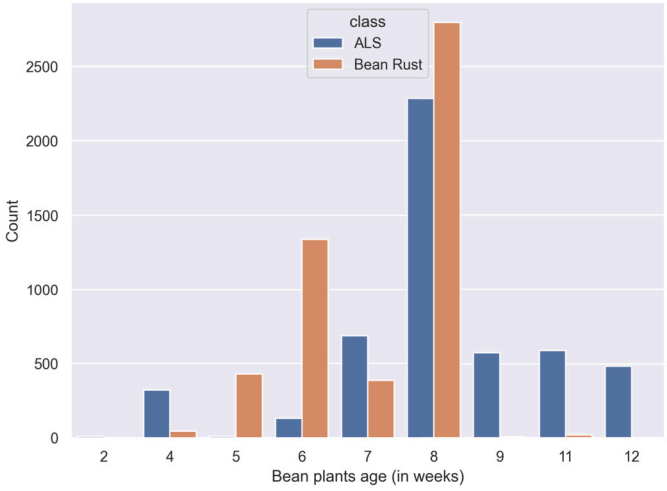

Fig. 7. Dataset distribution by bean plants age. The x-axis represents the age of the bean plants (in weeks) captured in the dataset images across the ALS and bean rust classes, while the y-axis represents the number of images.

and unknown images. It consists of three (3) distinct flows: an entry flow that prepares the feature maps, a middle flow that deeply processes these features, and an exit flow that aggregates and classifies the information. The model is illustrated in Fig. 9.

BeanWatchNet was inspired by the Xception model, which uses depthwise separable convolutions. This technique separates the convolution process into parts, making the model more efficient and able to learn from fewer data points. We adapted this approach better to recognize specific details in images of bean leaves, improving the model's ability to handle agricultural challenges. Additionally, using this architecture has several advantages. It reduces the complexity of the model and the number of parameters needed, which helps prevent overfitting, a frequent issue in plant disease detection where there is a slight variation in the data. This design makes our model better at identifying essential features in high-resolution images, leading to more accurate disease detection even in varying conditions. However, this method also comes with some challenges. Reducing the number of parameters can cause underfitting and therefore the model needs to be adjusted appropriately. We have carefully optimized and tested BeanWatchNet to ensure it consistently performs well, considering the specific settings needed for our convolutions.

The first convolutional layer uses 32 filters with a 3x3 kernel size and applies the same padding to maintain spatial dimensions. The ReLU activation is used for non-linearity, and batch normalization is applied to speed learning. The second convolutional layer follows a similar structure but utilizes 64 filters. After the initial layers, the network introduces a depthwise separable convolutional block that reduces the model's complexity and computational cost. This block consists of a DepthwiseConv2D layer with a 3x3 kernel and ReLU activation, followed by batch normalization, a pointwise Conv2D layer with 128 1x1 filters, and a MaxPooling layer with a 3x3 pool size and 2x2 strides for dimensionality reduction. The final segment of the model, or the exit flow, mirrors the middle flow's structure but increases the complexity using 256 filters in the pointwise Conv2D layer. The flattened output subsequently passes through a dense layer of 1024 neurons with ReLU activation, followed by a dropout layer with a 0.5 rate to prevent overfitting. The model uses a softmax classifier that outputs the probability distribution across the four (4) classes, i.e., ALS, bean rust, healthy and unknown. The hyperparameters used in training the BeanWatchNet model are illustrated in Table 3.

b. Transfer Learning

Transfer learning was employed, fine-tuning models pre-trained on the ImageNet dataset [26] for the classification of bean rust and ALS diseases in bean leaves. The image classification models were trained using three (3) selected CNN architectures: EfficientNet v2 B0, ResNet-50 v2, and MobileNet v3. EfficientNet is recognized for achieving high performance while maintaining efficiency in terms of both computational resources and model size [27]. The introduction of skip connections in ResNet enables the training of deep neural networks with improved convergence and generalization characteristics [28]. On the other hand, MobileNet is suitable for deployment on devices with limited resources due to its lightweight design, making it a great choice for various real-world applications in edge computing and mobile environments [29].

c. Hyperparameter Optimization

Optimal hyperparameters, i.e., number of units in the first dense layer, batch size, learning rate and dropout for each model, were obtained using the Keras tuner library's hyperband search algorithm [30]. Hyperband is a bandit-based technique that combines the concepts of random search as well as iterative halving to conduct adaptive and efficient hyperparameter optimization [31]. The search range for the hyperparameters is illustrated in Table 2, and the obtained optimal hyparemeters are indicated in Table 3.

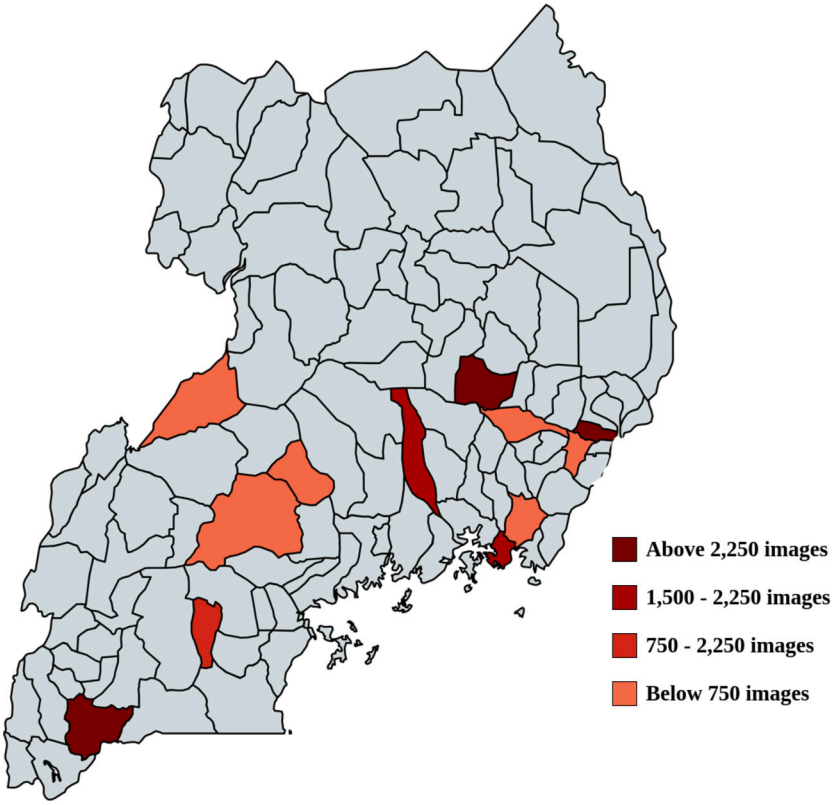

Fig. 8. Dataset distribution by location. The data was collected from twelve (12) districts across three (3) different regions in Uganda where beans cultivation is mostly practised. The experts from NaCRRI identified these districts; the red region in the map shows these areas.

Table 2
Hyperparameter search range used by the hyperband search algorithm.

Hyperparameter	Minimum value	Maximum value
Batch size	16	128
No of Units in 1^{st} Dense Layer	32	1120
Learning rate	0.00001	0.01
Droupout	0	0.5

Table 3
Hyperparameters used for training the image classification models.

Hyperparameter	BeanWatchNet	EfficientNet	MobileNet	ResNet-50
Batch Size	32	32	32	32
No of Units in 1^{st} Dense Layer	1024	160	256	64
Learning Rate	0.0001	0.0001	0.0001	0.0001
Epochs	100	100	100	100
Optimizer	Adam	Adam	Adam	Adam
Patience	10	10	10	10
Dropout	0.5	0.0	0.2	0.1

2.3.3. Model robustness

a. Adversarial Training

Adversarial training was undertaken to improve the robustness of the model predictions on the target classes (ALS, Bean rust, and Healthy) by enabling it to deal with deceptive inputs in addition to the expected valid inputs. Adversarial examples were created by introducing small perturbations of $\epsilon = 0.1$ to clean or valid input images using the Fast Gradient Sign Method (FGSM). FGSM generates adversarial instances by minimizing the maximum amount of perturbation introduced to any pixel in the image [32]. This approach is efficient and less computationally intensive compared to other methods such as Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS), Generative Adversarial Networks (GANs) and Jacobian-based Saliency Map Attack (JSMA). These adversarial images are corrupted versions of valid images that appear similar to the human eye but cause misclassifications by the trained model. Incorporating these images during training enables the model to generalize better and defend against adversarial attacks. Adversarial training is tailored to improve model robustness in predicting the target classes. However, it is not very effective in handling significantly dissimilar data, i.e., images unrelated to bean leaves, such as the sky, a car, maize leaves, etc.

b. Out-of-Distribution (OOD) Detection

ODD detection refers to techniques utilized in identifying and appropriately handling data substantially divergent from the target classes. This aims to eliminate the closed-world assumption usually made during training image classification models, thereby improving the model's robustness and reliability when deployed in the real-world environment. In this work, two (2) ODD approaches, i.e., confidence thresholding and training of the model with an auxiliary class (unknown class), were employed.

i. Confidence Thresholding

Confidence thresholding is an ODD technique that utilizes the confidence of the model's predictions to recognize data that deviates significantly from that of the target classes. It is based on the notion that a model should be relatively confident in its predictions when it comes across data similar to what it was trained on. A threshold for the model confidence score is set. The input data is considered ODD if the model predictions are below the threshold, otherwise it is within the target classes. For this work, a confidence score of 0.7 was set as the threshold. Images unrelated to bean leaves would ideally be classified with low confidence and therefore recognized as ODD. In addition to handling

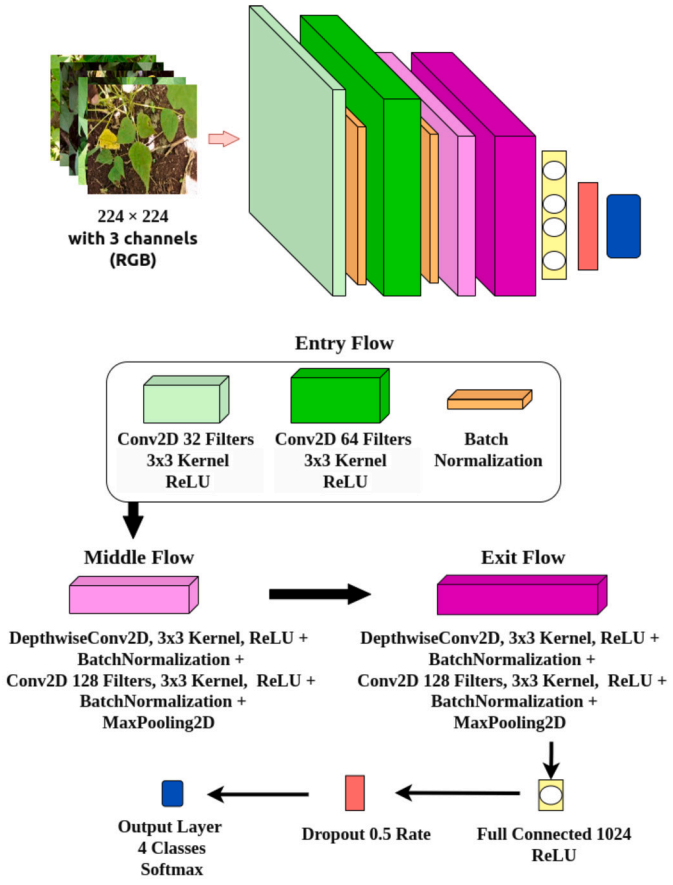

Fig. 9. BeanWatchNet model architecture. This CNN is divided into three flows: it employs multiple convolutional layers with increasing complexity, integrates depthwise separable convolutional, and ends with a dense layer and dropout to prevent overfitting. BeanWatchNet is inspired by Xception architecture.

ODD data, confidence thresholding ensures that predictions of the target classes are always of high certainty. This approach is easy to implement and is computationally efficient. Neural networks, however, tend to exhibit overconfidence even when they make wrong predictions, thereby leading to false acceptance of ODD data as part of the target classes.

ii. Unknown Class

This ODD approach involves training the classifier with an auxillary class, i.e., an unknown class in addition to the target classes. The unknown class includes examples of ODD data collected from various public sources as elaborated in subsection 2.1. This improves reliability in in-field disease detection as images unrelated to bean leaves, such as those depicting the sky, a person or a hoe, are classified as unknown rather than healthy, ALS, or bean rust. This methodology is more accurate and reliable than confidence thresholding as the model is trained with ODD data and therefore learns to detect unrelated inputs without affecting its performance on the target classes.

2.4. Object detection

Object detection, as opposed to image classification, goes a step further by not only recognizing but also localizing multiple disease-afflicted areas within an image, assigning predefined labels to each identified region.

2.4.1. Data preprocessing

The annotations were converted from the Pascal VOC XML format to the YOLO PyTorch TXT format. The dataset was split into train, validation, and test sets in a ratio of 0.7:0.15:0.15. The images were resized to dimensions of 640x640.

Table 4
Hyperparameters used for training of the object detection models.

Hyperparameter	Value
Batch Size	16
Learning Rate	0.001
Epochs	100
Optimizer	Adam
Patience	10
Momentum	0.9

2.4.2. Modeling

The object detection models for identifying bean rust and ALS diseases in bean leaves were trained using the YOLO v8, YOLO v9, YOLO-NAS and Detection Transformers (DETR) architectures, pre-trained on the COCO dataset [33]. The feature pyramid network architecture and dynamic prediction scheme of YOLO v8 enable adaptive predictions, allowing adjustments based on image complexity [34]. YOLO-NAS incorporates advanced training schemes and a novel quantization-friendly block, enhancing localization accuracy and improving the detection of small objects [35]. YOLO v9 introduces features such as Generalized Efficient Layer Aggregation Network (GELAN) and Programmable Gradient Information (PGI) that improve computational efficiency and minimize information loss [36]. DETR is a transformer-based architecture that provides a simplified approach to object detection by utilizing an encoder to process input images and generate a set of learnable object queries that inform the prediction of the final bounding boxes [37]. Table 4 indicates the hyperparameters used in the training process.

2.5. Deployment

The BeanWatchNet model and YOLO v8 model were deployed on two (2) edge platforms: a smartphone (through a mobile application) and a Raspberry Pi 4B. Before deployment, the models underwent post-training quantization to optimize inference speed and minimize their memory footprint. The TensorFlow Lite Converter was utilized to quantize the BeanWatchNet model, while the Ultralytics' Export mode was used to quantize the YOLOv8 model.

2.5.1. Mobile application

The mobile application was developed using Google's open-source Flutter development toolkit. Due to Flutter's single codebase feature, it is cross-platform, allowing installation on smartphones running either the Android or iOS operating systems.

2.5.2. Raspberry Pi 4B

The Raspberry Pi 4B is a 64-bit single-board micro-computer with a Quad-core Cortex-A72 processor and RAM options of 2 GB, 4 GB and 8 GB. Its enhanced processing power compared to its predecessors, i.e., Raspberry Pi 3 and Raspberry Pi 2, in addition to connectivity features such as the MIPI CSI camera port, were used alongside the Raspberry Pi camera module to implement computer vision tasks of object detection and image classification. A web application was also developed using the Python-based Flask framework and was hosted on the Raspberry Pi's web server. This allows for multiple devices to access the web application and perform disease diagnosis without necessarily having an Internet connection but rather being on the same local network as the Raspberry Pi.

Table 5
Comparison of the BeanWatchNet model performance on the target classes.

Architecture	Accuracy	Precision	Recall	F1 Score
BeanWatchNet	0.90	0.90	0.90	0.90
BeanWatchNet (with adversarial training)	0.91	0.91	0.91	0.91

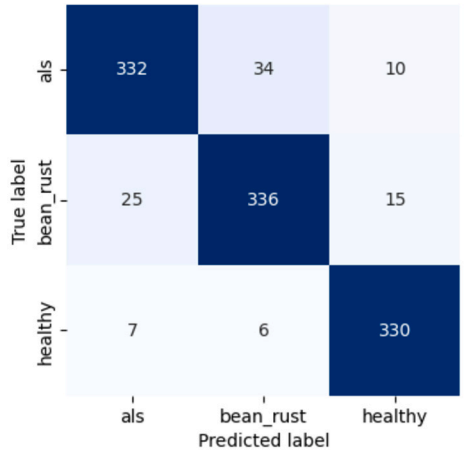

Fig. 10. BeanWatchNet confusion matrix on only target classes. The confusion matrix of BeanWatchNet shows good classification performance for the three (3) classes.

3. Results

3.1. Image classification

3.1.1. Three-class classification

The BeanWatchNet model was trained to classify bean leaves into three (3) target categories, i.e., ALS, bean rust and healthy. The model achieved 90% when evaluated on the test dataset as presented in Table 5. A confusion matrix showcasing the number of correct and incorrect predictions for each class is illustrated in Fig. 10. The model performs well in identifying images within the target classes; however, it misclassifies images unrelated to bean leaves as belonging to one of the target classes. This is demonstrated in the sample predictions in Fig. 11. This introduces the need to train the image classification models with ODD detection mechanisms.

The BeanWatchNet model was retrained with adversarial images to improve its robustness in classifying the target classes by enabling it to recognize deceptive images correctly. The retrained model achieved an improved accuracy of 91% on the test dataset, as shown in Table 5, and its confusion matrix is depicted in Fig. 12. Sample predictions of the BeanWatchNet model on both clean and perturbed images are illustrated in Fig. 13.

To enhance the model robustness to ODD data, i.e., images unrelated to bean leaves, confidence thresholding was employed with the threshold set at a confidence score of 0.7. This ensures that all predictions below the threshold are recognized as ODD data and guarantees that predictions of the target classes are always of high certainty, as demonstrated in the confusion matrix in Fig. 14.

3.1.2. Four-class classification

The tendency of neural networks to be overconfident, even when they make incorrect predictions, presents a challenge to the confidence thresholding ODD approach. To solve this, the BeanWatchNet was retrained with an auxiliary class (unknown class) in addition to the target classes of ALS, bean rust and healthy to explicitly handle images unrelated to bean leaves. Furthermore, transfer learning was utilized and the performance of the pre-trained models was compared to that of the BeanWatchNet.

Table 6
Comparison of the performance of the four-class image classification models.

Architecture	Accuracy	Precision	Recall	F1 Score
BeanWatchNet	0.90	0.90	0.90	0.90
EfficientNet v2 B0	0.91	0.91	0.91	0.91
ResNet-50 v2	0.87	0.86	0.86	0.86
MobileNet v3	0.87	0.89	0.88	0.88

Table 7
Comparison of the performance of the object detection models.

Architecture	mAP@50
YOLO v8	87.6
YOLO-NAS	73.5
YOLO v9	74.1
DETR	62.0

Early stopping was configured to prevent overfitting, allowing the models to converge effectively as illustrated in the loss curve in Fig. 15, the BeanWatchNet model converges last after 47 epochs. The EfficientNet v2 B0 model and the BeanWatchNet model performed best and achieved 91% and 90% accuracy, respectively, when evaluated on the test dataset, as detailed in Table 6. Their confusion matrices are illustrated in Figs. 16 and 17. Fig. 18 highlights BeanWatchNet's effectiveness in categorizing input images into the four (4) classes, i.e., ALS, bean rust, healthy and unknown. It illustrates sample images in the test set and their respective predictions. Images unrelated to bean leaves are classified as unknown rather than as ALS, bean rust or healthy.

3.2. Object detection

For training the object detection models, YOLO v8, YOLO-NAS, YOLO v9 and DETR architectures were employed. The YOLO v8 model exhibited the best performance when tested on unseen data, as indicated in Table 7. While YOLO v8 achieved an overall mAP@50 score of 87.6 for all classes, its precision and consistency in detecting ALS exceeded that for bean rust across all levels of recall, as evident in the PR curve illustrated in Fig. 20. Additionally, Fig. 21 showcases the precision of the YOLO v8 model in detecting bean rust and ALS diseases in bean foliage. It displays sample images from the test set, along with annotations from experts and their respective predictions.

3.3. Deployment

The BeanWatchNet model trained with an unknown class and YOLO v8 model were deployed on two (2) edge platforms i.e., a smartphone (through a mobile application), and Raspberry Pi 4B. Fig. 19 and Fig. 22 depict sample predictions on the edge devices. Prior to deployment, the models underwent post-training quantization to reduce their memory footprint and improve their inference time. The BeanWatchNet model was deployed instead of the Efficient v2 B0 model due to its significantly smaller model size and marginal difference in accuracy. The BeanWatchNet model is 94% smaller than the Efficient v2 B0 model, as illustrated in Table 8. The inference time and end-to-end time for the models deployed on edge devices, specifically the Raspberry Pi 4B and mobile phone, are evaluated in Table 10. BeanWatchNet, deployed on the Raspberry Pi 4B, demonstrated both the shortest inference and end-to-end time as 207 ms and 798 ms, respectively.

4. Discussion

The aim of this research was to develop a deep learning-based approach for the precise in-field identification of diseases in bean plants. CNN architectures were leveraged to train image classification and object detection models to detect ALS and bean rust diseases in bean leaves.

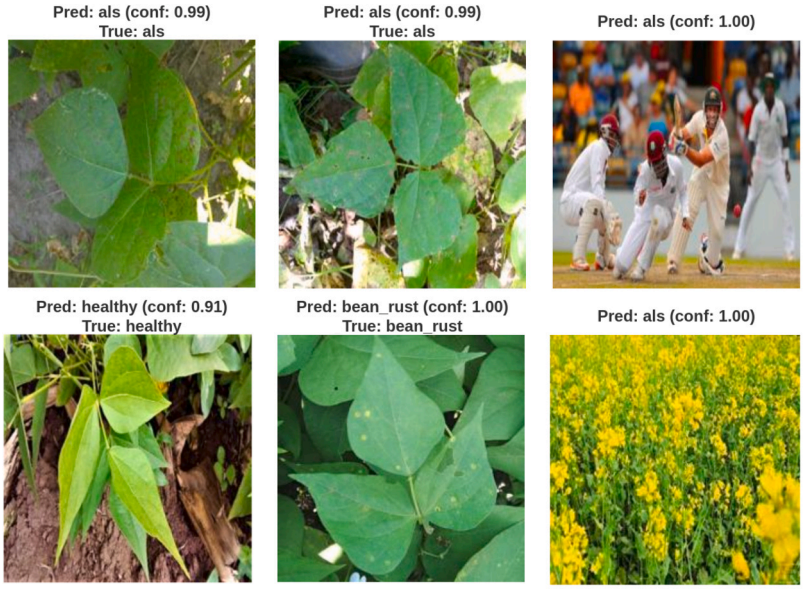

Fig. 11. Test data sample images processed using the BeanWatchNet model trained on target classes. The model correctly classified ALS in the first two (2) images (99% confidence) and a healthy in the bottom left (91% confidence). It also correctly identified Bean Rust (100% confidence). However, the model misclassified two unknown images (top right and bottom right) as ALS with 100% confidence.

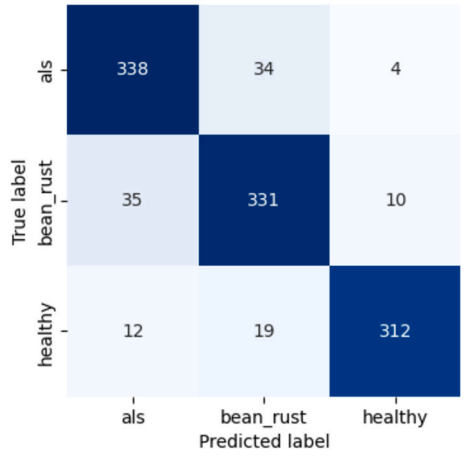

Fig. 12. Confusion matrix of the BeanWatchNet model with adversarial training classifying only the target classes. The matrix shows a great classification as well while inducing robustness.

Table 8
Comparison of the models sizes before and after quantization.

Architecture	Original model size (KB)	Quantitized model size (KB)
BeanWatchNet	4,009	353
EfficientNet v2 B0	72,314	6,644
YOLO v8	11,903	6,000

Table 9
Comparison of the model performance before and after quantization.

Architecture	Original model metric	Quantitized model metric
BeanWatchNet	90%	89%
YOLO v8	87.6 - mAP@50	86.8 - mAP@50

For the three-class image classification, the BeanWatchNet model exhibited good performance in identifying the target classes of ALS, bean rust and healthy, achieving a 90% accuracy when evaluated on unseen data. To enhance the robustness of the model, it was retrained with adversarial examples to enable it to correctly classify perturbed inputs and thereby strengthen its ability to generalize better in real-world scenarios. The retrained model achieved an accuracy of 91% on the test dataset, highlighting an improvement in recognizing the target classes. Adversarial training is, however, not effective in handling ODD inputs, and therefore, images unrelated to bean leaves are still misclassified as one of the target classes. Confidence thresholding was utilized to handle ODD inputs, i.e., images unrelated to the target classes. Input images with predictions below the threshold confidence score of 0.7 were regarded as ODD. This also ensures that predictions of the target classes are always of high certainty. This approach is based on the idea that a model ought to be relatively confident in its predictions when it encounters data comparable to what it was trained on. However, even in the case of incorrect predictions, neural networks tend to exhibit overconfidence, which can result in the mistaken acceptance of ODD data as belonging to the target classes. To alleviate the challenges of confidence thresholding, the model was retrained with an additional class, i.e., an unknown class, to explicitly deal with ODD images.

Table 10
Inference time and end-to-end time for edge deployment.

Architecture	Edge device	Inference time (ms)	End-to-end time (ms)
BeanWatchNet	Mobile phone	234	1236
BeanWatchNet	Raspberry Pi 4B	207	798
YOLO v8	Mobile phone	425	1430
YOLO v8	Raspberry Pi 4B	387	980

For the four-class image classification, the BeanWatchNet model was trained alongside three (3) pre-trained models, i.e., EfficientNet v2 B0, ResNet v2 and MobileNet v3. The EfficientNet v2 B0 model and the BeanWatchNet model demonstrated superior performance in categorizing input images into four (4) classes, i.e., ALS, bean rust, healthy and unknown. The unknown class was specifically introduced to handle images unrelated to bean leaves. Since the model is trained with ODD data, it learns to recognize unrelated images without affecting its performance on the target classes, making this technique more accurate and dependable than confidence thresholding. This enhances the model's reliability in real-world scenarios where they might be exposed to other images besides those of bean leaves. This is not the case with the existing approaches in [3], [8], [9], and [20] where images unrelated to bean leaves, e.g., those depicting a person, the sky or a dog are classi-

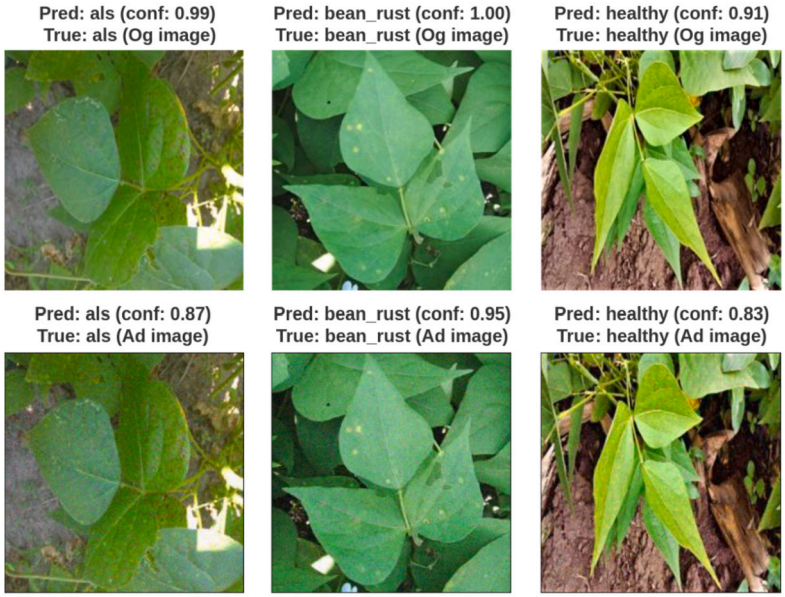

Fig. 13. BeanWatchNet model predictions on original (top) and adversarial (bottom) test images. Left to right: ALS, bean rust, and healthy bean leaves. The model correctly classified all images, demonstrating robustness to adversarial inputs.

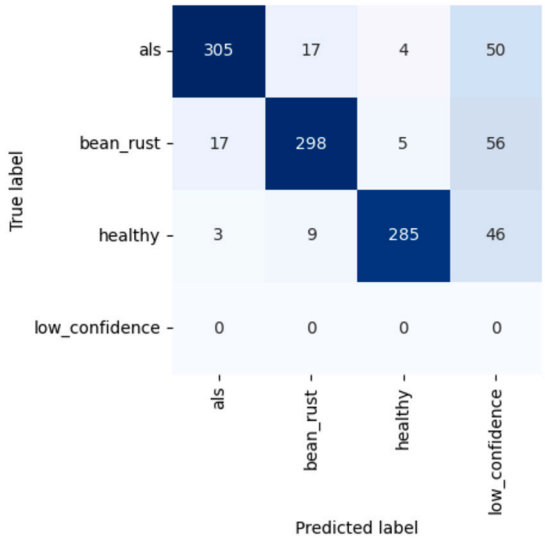

Fig. 14. Confusion matrix of the BeanWatchNet with confidence thresholding. The matrix shows that only the predictions above the threshold confidence score of 0.7 are classified into the target classes.

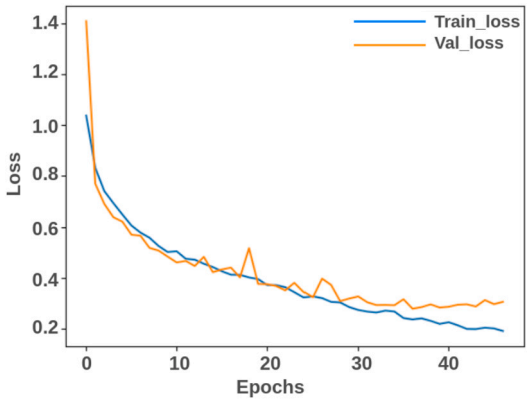

Fig. 15. The BeanWatchNet model's loss curve shows effective training, with both training and validation losses decreasing and around epoch 35, validation loss levels off, indicating potential overfitting. Early stopping was applied at epoch 38 to prevent this.

fied as ALS, bean rust or healthy. The EfficientNet v2 B0 model and the BeanWatchNet model achieved 91% and 90% accuracy, respectively, demonstrating their effectiveness in the identification of bean rust and ALS diseases in bean leaves. It should be noted that the introduction of the unknown class doesn't affect the performance of the model's predictions on the target classes and, therefore, is a favorable technique for handling ODD inputs.

Object detection models were trained to localise the disease-afflicted areas within an image as opposed to image classification that only assigns a predefined label to the entire image. The YOLO v8 model outperformed the other object detection models, achieving mAP@50 of 87.6. Farmers and agricultural experts are offered invaluable insights by precisely localizing the diseased areas in bean leaves, enabling focused intervention and disease management techniques.

The BeanWatchNet model and YOLO v8 model were deployed on two (2) edge platforms, i.e., smartphone (through a mobile application) and Raspberry Pi 4B, to ease in-field detection of bean rust and ALS diseases in bean leaves. Edge deployment provides offline functionality, enabling farmers to perform disease diagnosis with no internet connectivity. Quantization significantly reduced the memory footprint of the models and enhanced their inference time while maintaining accurate predictions as illustrated in Table 8, Table 9 and Table 10. This could, therefore, enable the models to achieve real-time performance on edge devices, thereby providing rapid in-field disease diagnosis. This could not only reduce the reliance on experts for disease diagnosis in bean leaves but also enable timely interventions to mitigate yield losses. The proposed approach can also be employed alongside other technologies, such as robotics and Unmanned Aerial Vehicles (UAVs), for large-scale disease surveillance in precision agriculture.

5. Conclusion

Diseases pose significant challenges to bean cultivation, causing substantial financial losses for farmers due to poor yields. In this paper, we have presented a deep learning-based approach to enhance the in-field identification of bean plant diseases, specifically ALS and bean rust. The proposed method improves precision in disease detection and facilitates early identification. Image classification and object detection

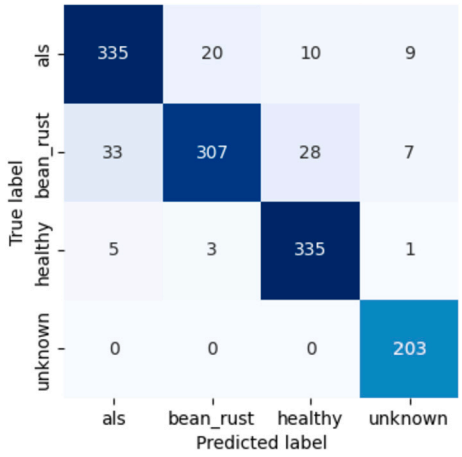

Fig. 16. EfficientNet v2 B0 confusion matrix. The confusion matrix of the EfficientNet B0 v2 model shows classification performance across four (4) classes. The model demonstrates high accuracy, with most true labels matching the predicted labels. A few misclassifications were observed between bean rust and ALS classes.

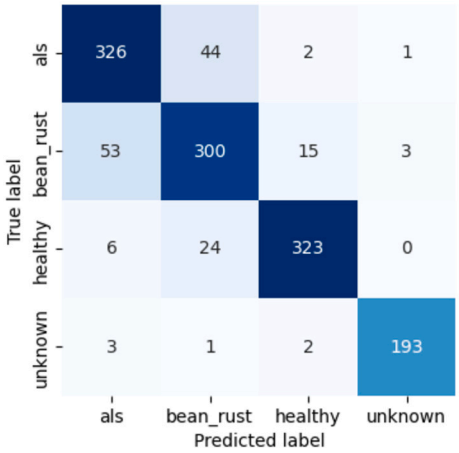

Fig. 17. BeanWatchNet confusion matrix. The confusion matrix of BeanWatchNet indicates good classification performance for the four (4) classes. The matrix shows a great classification rate for ALS and unknown classes while bean rust has some misclassifications with other classes.

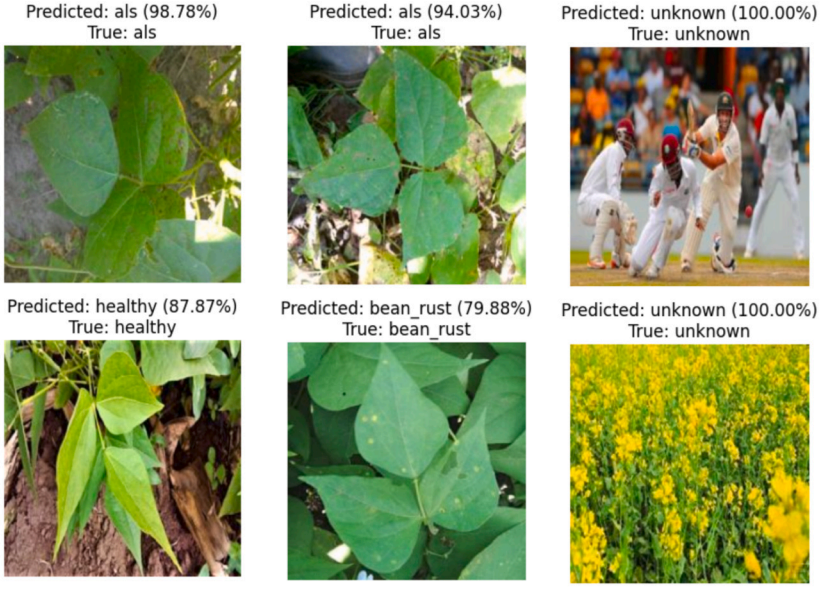

Fig. 18. Test data samples processed using the BeanWatchNet model. Top left: ALS (expert and model agree, 98.78% confidence). Second: ALS (agreement, 94.03% confidence). Third: Unknown (100% confidence). Bottom left: Healthy (agreement, 87.87% confidence). Next: Bean rust (agreement, 79.88% confidence). Last: Unknown (100% confidence).

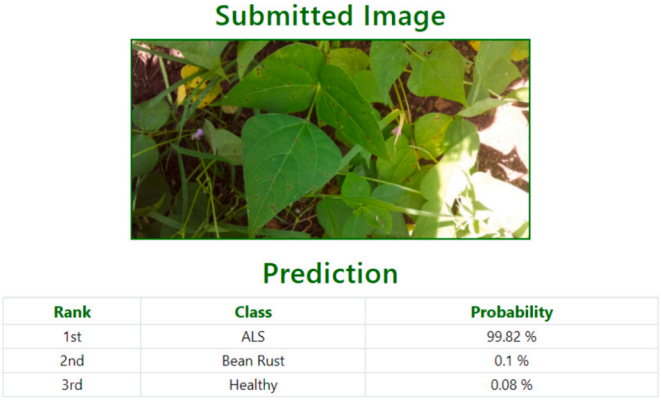

Fig. 19. A sample image of the image classification prediction from a web application hosted on the Raspberry Pi 4B's web server. The submitted image is classified, and the top three (3) associated classes are ranked with their respective probabilities.

models were trained using various CNN architectures. To enhance the robustness and reliability of the image classification models in practical environments, adversarial training was employed to recognize deceptive inputs. Furthermore, confidence thresholding was applied, and an additional class (unknown class) was introduced to account for images unrelated to healthy or diseased bean leaves. The developed models were evaluated on an unseen test dataset, with EfficientNet v2 B0 and BeanWatchNet outperforming other architectures for image classification, achieving an accuracy of 91% and 90%, respectively. At the same time, YOLO v8 performed best for object detection, achieving mAP@50 of 87.6. For practical and convenient use, the BeanWatchNet and YOLO v8 models were deployed on two (2) edge platforms: a smartphone (through a mobile application) and a Raspberry Pi 4B. Edge deployment facilitates convenient means of performing in-filed disease diagnosis in bean leaves.

The proposed work is limited to disease identification and doesn't provide any form of recommendation. In the future, we intend to combine computer vision techniques with the knowledge and reasoning capabilities of Large Language Models (LLMs) to provide highly

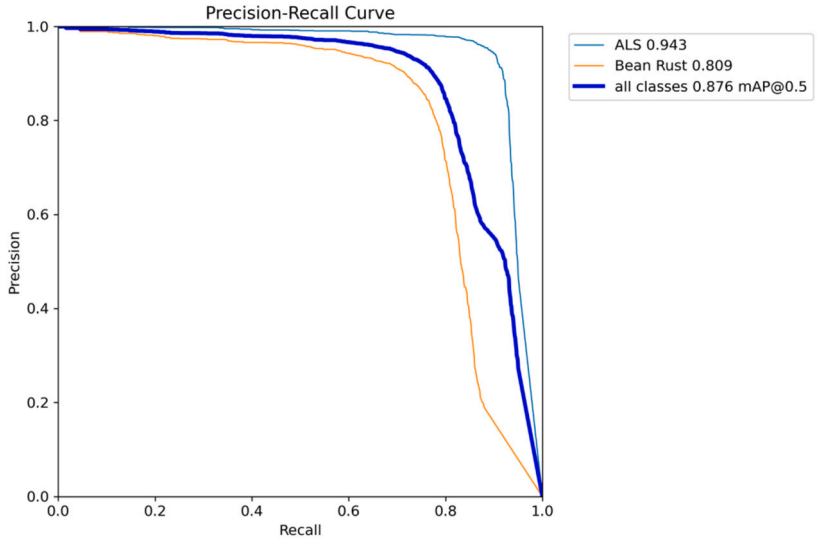

Fig. 20. YOLO v8 precision-recall curve. The model detection achieved mAP@0.5 of 0.876 for all classes. The curves demonstrate the model's ability to balance precision and recall effectively, with ALS being the most precisely detected.

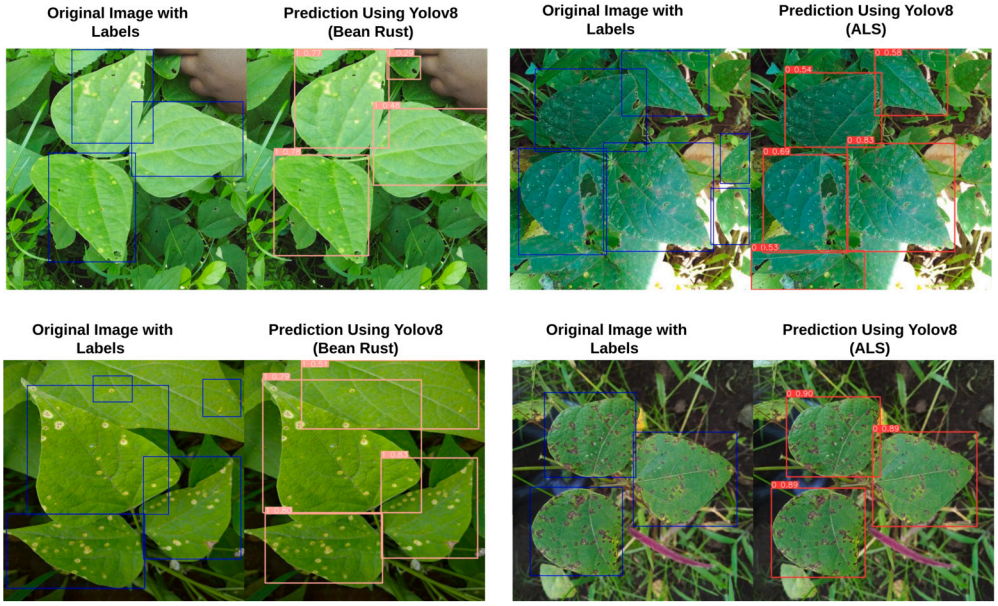

Fig. 21. Test image samples processed using YOLO v8 to detect bean diseases. The left column in each image displays the original images with blue bounding boxes indicating the areas labeled by experts for two (2) diseases: bean rust and ALS. The right column shows the corresponding predictions by YOLO v8, where the areas detected as diseased are marked with red bounding boxes, each with a confidence score demonstrating the model's certainty in the prediction. The scores range from 0.51 to 0.89 for bean rust and from 0.53 to 0.90 for ALS, indicating varying levels of detection confidence.

accurate, interactive and context-aware disease diagnosis and recommendations.

CRediT authorship contribution statement

Andrew Katumba: Writing – review & editing, Validation, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Wayne Steven Okello:** Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Formal analysis, Conceptualization. **Sudi Murindanyi:** Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Formal analysis, Conceptualization. **Joyce Nakatumba-Nabende:** Writing – review & editing, Supervision, Project administration, Investigation, Funding acquisition. **Moses Bomera:** Methodology, Data curation, Conceptualization. **Ben Wycliff Mugalu:** Data curation. **Amos Acur:** Data curation.

Declaration of competing interest

All authors declare that no conflicts of interest could have appeared to influence the work reported in this paper. The authors would like to acknowledge the financial support from Data Science Africa (DSA), Deep Learning Indaba (DLI), and the International Development Research Centre (IDRC), which facilitated this research. The Lacuna Fund provided additional funding for collecting the dataset utilized in this study. These funding sources had no role in the design, execution, interpretation, or writing of the study.

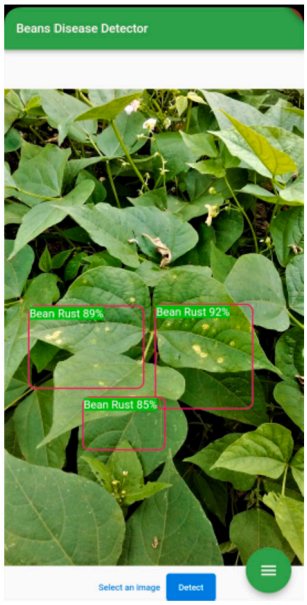

Fig. 22. A sample image of the object detection prediction using the mobile application on a smartphone. Disease-affected areas are localized using bounding boxes alongside the detected disease name tag and respective level of confidence.

Acknowledgement

The authors express gratitude to Data Science Africa (DSA), Deep Learning Indaba (DLI), and the International Development Research Centre (IDRC) for their financial support in facilitating this research. Additionally, appreciation is extended to the Lacuna Fund for providing funding that contributed to the collection of the dataset utilized in this study.

Data availability

The data used for the paper is available at https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/WFSLBY.

Bean Plant Pathologies Dataset for Deep Learning Tasks (Original data) (Dataverse)

References

[1] P. Pamela, D. Mawejje, M. Ugen, Severity of angular leaf spot and rust diseases on common beans in central Uganda, Uganda J. Agric. Sci. 15 (2014) 63–72.
[2] P.A.B.R.A. PABRA, PABRA and the Power of Beans in Africa: 25 Years of Transformation, International Center for Tropical Agriculture (CIAT), Nairobi, Kenya, 2022.
[3] S.H. Abed, A.S. Al-Waisy, H.J. Mohammed, S. Al-Fahdawi, A modern deep learning framework in robot vision for automated bean leaves diseases detection, Int. J. Intell. Robot. Appl. 5 (2021) 235–251.
[4] M.P.J. Mahenge, H. Mkwazu, R.R. Madege, B. Mwaipopo, C. Maro, Artificial intelligence and deep learning based technologies for emerging disease recognition and pest prediction in beans (Phaseolus vulgaris L.): a systematic review, 2023.
[5] R. Kijana, M. Abang, R. Edema, C. Mukankusi, R. Buruchara, Prevalence of angular leaf spot disease and sources of resistance in common bean in Eastern Democratic Republic of Congo, Afr. Crop Sci. J. 25 (2017) 109–122.
[6] P.I. Ritharson, K. Raimond, X.A. Mary, J.E. Robert, J. Andrew, Deeprice: a deep learning and deep feature based classification of rice leaf disease subtypes, Artif. Intell. Agric. 11 (2024) 34–49.
[7] M.E. Chowdhury, T. Rahman, A. Khandakar, M.A. Ayari, A.U. Khan, M.S. Khan, N. Al-Emadi, M.B.I. Reaz, M.T. Islam, S.H.M. Ali, Automatic and reliable leaf disease detection using deep learning techniques, AgriEngineering 3 (2021) 294–312.
[8] V. Singh, A. Chug, A.P. Singh, Classification of beans leaf diseases using fine tuned cnn model, Proc. Comput. Sci. 218 (2023) 348–356.
[9] E. Elfatimi, R. Eryigit, L. Elfatimi, Beans leaf diseases classification using mobilenet models, IEEE Access 10 (2022) 9471–9482.
[10] F. Mohameth, C. Bingcai, K.A. Sada, Plant disease detection with deep learning and feature extraction using plant village, J. Comput. Commun. 8 (2020) 10–22.
[11] A. Fuentes, S. Yoon, S.C. Kim, D.S. Park, A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition, Sensors 17 (2017) 2022.
[12] M.J.A. Soeb, M.F. Jubayer, T.A. Tarin, M.R. Al Mamun, F.M. Ruhad, A. Parven, N.M. Mubarak, S.L. Karri, I.M. Meftaul, Tea leaf disease detection and identification based on yolov7 (yolo-t), Sci. Rep. 13 (2023) 6078.
[13] M.H. Saleem, J. Potgieter, K.M. Arif, Plant disease detection and classification by deep learning, Plants 8 (2019) 468.
[14] L. Li, S. Zhang, B. Wang, Plant disease detection and classification by deep learning—a review, IEEE Access 9 (2021) 56683–56698.
[15] M. Fraiwan, E. Faouri, N. Khasawneh, Classification of corn diseases from leaf images using deep transfer learning, Plants 11 (2022) 2668.
[16] A. Picon, A. Alvarez-Gila, M. Seitz, A. Ortiz-Barredo, J. Echazarra, A. Johannes, Deep convolutional neural networks for mobile capture device-based crop disease classification in the wild, Comput. Electron. Agric. 161 (2019) 280–290.
[17] J. Ma, K. Du, F. Zheng, L. Zhang, Z. Gong, Z. Sun, A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network, Comput. Electron. Agric. 154 (2018) 18–24.
[18] C. DeChant, T. Wiesner-Hanks, S. Chen, E.L. Stewart, J. Yosinski, M.A. Gore, R.J. Nelson, H. Lipson, Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning, Phytopathology 107 (2017) 1426–1432.
[19] J. Eunice, D.E. Popescu, M.K. Chowdary, J. Hemanth, Deep learning-based leaf disease detection in crops using images for agricultural applications, Agronomy 12 (2022) 2395.
[20] D. Banerjee, V. Kukreja, R. Yadav, K. Joshi, A. Singh, Effective disease detection in bean leaves using deep cnn and svm ensemble, in: 2023 3rd Asian Conference on Innovation in Technology (ASIANCON), IEEE, 2023, pp. 1–6.
[21] Q. Wu, Y. Chen, J. Meng, Dcgan-based data augmentation for tomato leaf disease identification, IEEE Access 8 (2020) 98716–98728.
[22] P. Jiang, Y. Chen, B. Liu, D. He, C. Liang, Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks, IEEE Access 7 (2019) 59069–59080.
[23] TensorFlow, Cropnet for Cassava, https://www.tensorflow.org/hub/tutorials/cropnet_cassava, 2024. (Accessed 15 February 2024).
[24] B. Mugalu, J. Nakatumba-Nabende, A. Katumba, C. Babirye, F. Tusubira, C. Mutebi, S. Nsumba, G. Namanya, Makerere university beans image dataset, in: A. Katumba, J. Nakatumba-Nabende, W. Ssekandi (Eds.), Harvard Dataverse, 2022.
[25] A. Dutta, A. Zisserman, The via annotation software for images, audio and video, in: Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 2276–2279.
[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 248–255.
[27] M. Tan, Q. Le, Efficientnet: rethinking model scaling for convolutional neural networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 6105–6114.
[28] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[29] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: efficient convolutional neural networks for mobile vision applications, arXiv preprint, arXiv:1704.04861, 2017.
[30] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, et al., Kerastuner, https://github.com/keras-team/keras-tuner, 2019.
[31] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: a novel bandit-based approach to hyperparameter optimization, J. Mach. Learn. Res. 18 (2018) 1–52.
[32] J. Sen, A. Sen, A. Chatterjee, Adversarial attacks on image classification models: analysis and defense, arXiv preprint, arXiv:2312.16880, 2023.
[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: common objects in context, in: Computer Vision–ECCV 2014: 13th European Conference, Proceedings, Part V 13, Zurich, Switzerland, September 6-12, 2014, Springer, 2014, pp. 740–755.
[34] D. Reis, J. Kupec, J. Hong, A. Daoudi, Real-time flying object detection with yolov8, arXiv preprint, arXiv:2305.09972, 2023.
[35] J. Terven, D.-M. Córdova-Esparza, J.-A. Romero-González, A comprehensive review of yolo architectures in computer vision: from yolov1 to yolov8 and yolo-nas, Mach. Learn. Knowl. Extr. 5 (2023) 1680–1716.
[36] C.-Y. Wang, I.-H. Yeh, H.-Y.M. Liao, Yolov9: learning what you want to learn using programmable gradient information, arXiv preprint, arXiv:2402.13616, 2024.
[37] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-end object detection with transformers, in: European Conference on Computer Vision, Springer, 2020, pp. 213–229.