A Convolutional Neural Networks Approach to Detecting Foliar Diseases in Zea Mays

***[1]Adejimi Alaba O., [2]Falana Olorunjube J., [3]Olowofeso Elizabeth O., [4]Alabi Orobosade A.**

[1,3,4]Department of Computer Science, College of Computing Sciences, Federal University of Agriculture, Abeokuta, Ogun State, Nigeria.

[2]Department of Cyber Security, College of Computing Sciences, Federal University of Agriculture, Abeokuta, Ogun State, Nigeria.

DOI: https://doi.org/10.51244/IJRSI.2026.1304000261

Received: 08 December 2025; Accepted: 14 December 2025; Published: 20 May 2026

ABSTRACT

The presence of diseases in Zea Mays (Maize) pose a big threat to food security. This could cause big losses in output to farmers, hence subjecting them to economic burdens. Traditional methods for identifying maize diseases depend on skilled visual identification and expertise, which are not very effective and can lead to wrong diagnosis. This study hereby proposes an effective deep learning approach for the automated identification of foliar diseases in maize plants. The study specifically utilized Convolutional Neural Networks (CNN), incorporating two pre-trained models: EfficientNet and VGG16. Varied dataset comprising images of maize leaves afflicted with several foliar diseases were ensembled to train and test the CNN models. EfficientNet and VGG16 were pretrained models that were fine-tuned to work better for the specific goal of finding the diseases in maize. The experimental findings showed that both VGG16 and EfficientNet demonstrated promising performance, while VGG16 remains a robust baseline with 91% accuracy, the structural efficiency of EfficientNet (96% accuracy) provides a more viable solution for deployment in mobile devices for real-time agricultural diagnosis. The higher accuracy of EfficientNet suggests its suitability for this specific agricultural application and indicates the effectiveness of CNN models in accurately identifying foliar diseases in maize.

Keywords: Maize, EfficientNet, VGG16, Disease, Sustainability, Streamlit

INTRODUCTION

Corn or maize (Zea Mays), is a versatile food crops used in many parts of the world. It is a major source of food, feed, and raw materials. However, the sustainable production of maize is at risk from a number of diseases which includes common rust and other leaf spots. These diseases can cause big losses in output and subject farmers to economic burdens. Billions of dollars were lost each year globally due to the reduction in crop yields arising from corn diseases. These losses have far-reaching consequences, affecting not only the livelihoods of farmers but also food security and agricultural sustainability on a global scale (Askale et al., 2025).

Diseases are the principal biotic variables that limit maize output and productivity. These diseases are a big danger to food security and the long-term health of agriculture around the world. The International Maize and Wheat Improvement Center (CIMMYT) says that illnesses cause about 30% of maize production to be lost each year, and in some areas, the losses are considerably higher (Askale et al., 2025). A variety of pathogens, such as fungi, bacteria, viruses, and nematodes, can cause these diseases. They can infect different sections of the plant with each illness with its own set of symptoms. If these infections are found and treated properly, they can have a huge impact on crop yields (Liu et al., 2024).

Trained agronomists used to look at plants to find diseases, however this method can take a long time, be subjective, and make mistakes (Yuan et al., 2022). Experts' visual inspections are subjective, depend on human knowledge, and can be affected by things like lighting and how each person sees things. Also, the need for

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

more trained personnel in plant pathology limits the widespread availability of accurate disease diagnosis, especially in remote or underprivileged agricultural areas. These constraints show how important it is to have an automated system that can quickly, accurately, and consistently identify and diagnose diseases in maize crops.

Recent progress in deep learning, especially in computer vision, has improved on the identification and diagnosis of diseases in crops through automation. Deep learning algorithms may learn to distinguish tiny visual patterns linked to different diseases by being trained on vast sets of images of healthy and sick maize plants which makes it possible to quickly and accurately identify diseases (Chibesa et al. 2021).

Previous studies have explored various Convolutional Neural Networks (CNNs) architectures for crop disease detection, the comparative effectiveness of older "heavyweight" models like VGG16 versus "optimized" models like EfficientNet specifically for Zea mays foliar diseases is an area of ongoing investigation. This study addresses this gap by specifically utilizing two pre-trained Convolutional Neural Networks (CNN) models: EfficientNet and VGG16, for the automated identification of foliar diseases in maize plants and provide a more viable solution for deployment in mobile or edge-computing devices for real-time agricultural diagnosis.

A deep learning detective system for maize diseases will address the limitations of traditional methods and provide farmers and agricultural stakeholders with a powerful tool for early disease detection and give farmers and other people in agriculture a powerful tool for early detection of Maize diseases.

The arrangement of the remaining sections is hereby outlined: the second section gives the literature review, third section discusses the methodology involved in the research, while the fourth section presented the implementation and results. The conclusion and recommendation were discussed in the fifth section.

LITERATURE REVIEW

Foliar diseases are diseases that primarily affect the leaves of plants and are caused by viruses, bacteria, or fungi. They leave spots, blotches, wilting, lesions, discoloration, and uneven development on leaves, which is bad for the health and yield of crops because leaves are very important for photosynthesis and nutrient absorption. Common Rust, Northern Leaf Blight, and Grey Leaf Spot are some of the most frequent diseases that affect maize leaves.

Common Rust

Common Rust is a foliar disease caused by the fungus Puccinia sorghi. It appears as small, round to oval, reddish-brown pustules on the leaves as shown in Figure 1. Common rust can weaken plants and reduce production, but it is generally thought to be less economically harmful than other maize diseases.

Figure 1: Maize Common Rust

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

The disease is widespread in regions with warm temperatures and high humidity, where it thrives and spreads rapidly. Although yield losses due to common rust are relatively lower, they can still range from 5% to 20% in severe cases.

Northern Leaf Blight

Northern Corn Leaf Blight (NCLB) is caused by the fungus Exserohilum turcicum and has devastating effects on maize crops worldwide. NCLB manifests as long, elliptical lesions with parallel sides on the leaves, often appearing tan or greyish-brown as shown. It is particularly prevalent in regions with warm and humid conditions, where it can cause significant yield losses in Figure 2. Severe infections can lead to premature leaf death, reducing the plant's photosynthetic capacity and overall productivity (American Phytopathological Society, 2023).

Figure 2: Maize Northern Leaf Blight

According to the American Phytopathological Society (APS), NCLB can cause yield reductions of up to 20-50% in susceptible maize varieties.

Grey Leaf Spot

Gray Leaf Spot (GLS) is caused by the fungus Cercospora zeae-maydis. It appears as rectangular to blocky lesions with grey centres and dark brown to purple margins on the leaves as shown in Figure 3. GLS is prevalent in regions with warm temperatures and high humidity, where it thrives and spreads rapidly.

Figure 3: Maize Grey Leaf Spot

Yield losses due to GLS can vary depending on environmental conditions and the susceptibility of maize varieties, ranging from 10% to 50% or more in severe outbreaks. Severe infections can lead to significant defoliation, compromising photosynthetic efficiency and reducing maize yields.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

Related Work

Lui and Wang (2021) performed an extensive evaluation of detection techniques with deep learning algorithms. The research evaluated the effectiveness of deep learning algorithms against conventional image-processing methods, emphasizing the extensive opportunities and capabilities of deep learning approaches. Convolutional Neural Networks was found to provide a more efficient and precise methodology, executing end-to-end feature extraction, streamlining the detection process and presenting extensive opportunities for future development.

Hassan et al. (2021) suggested employing deep convolutional neural network (DCNN) models to analyze leaf morphology and detect illnesses in plants. The suggested DCNN models were trained on a series of plant photos, with a focus on the architecture of the leaves. The models were able to correctly sort the pictures and find diseases in the plants. The work showed that DCNN models with depth separable convolution work well for finding plant diseases which shows a better and more practical way of doing it than typical CNN models.

Jana et al. (2020) put forward a framework with Deep Belief Network (DBN) that could sort diverse illnesses of pepper plant. The framework has four main parts: getting the image, pre-processing it, extracting features, and classifying it. The suggested framework was meant to automate the process of classifying diseases in pepper plants, making it easier and more reliable to find damaged leaves. The model was able to learn and sort patterns linked to different diseases well with Deep Belief Network.

Guo et al. (2020) put forward a recognition model for identifying diseases in plants. The work employed three different algorithms: the region proposal network (RPN) algorithm, the Chan-Vese (CV) algorithm, and the transfer learning (TL) algorithm. The model was created to help identify diseases in complicated settings. The suggested model's combination of the RPN, CV, and TL algorithms can accurately identify diseases in plants, even in complicated situations, with an accuracy rate of 83.57%.

Atila et al. (2020) presented an EfficientNet deep learning framework for finding diseases in plants. The work utilized transfer learning, enhancing the pre-trained EfficientNet framework and additional deep learning models on the plant village dataset. Their results shown a 99% accuracy and 98% precision results, demonstrating its efficacy in precisely diagnosing and classifying plant diseases.

Bedi and Gole (2020) employed a mixed approach for automatically finding plant diseases. Plant village dataset were employed to find Bacterial Spot illness in peach plants. Convolutional Auto Encoder (CAE) and a Convolutional Neural Network (CNN) were ensembled. An accuracy of 99.35% and 98.38% were gotten on both the training and testing set. The model only used 9,914 training parameters, which shows how efficient and successful it is in finding diseases.

RESEARCH METHODOLOGY

This section describes the detection of foliar diseases in maize. Traditional techniques employed in disease detection have proved to be inefficient since they could not bring out features from images and choose the best ones. This study then uses sophisticated deep learning techniques to analyze the images of leaves, identify them, and classify the occurrence of foliar disease.

Framework of the Proposed Model

All the phases involved in the model creation and deployment are shown in Figure 4. These phases include data collection and preprocessing, foliar detection and classification.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

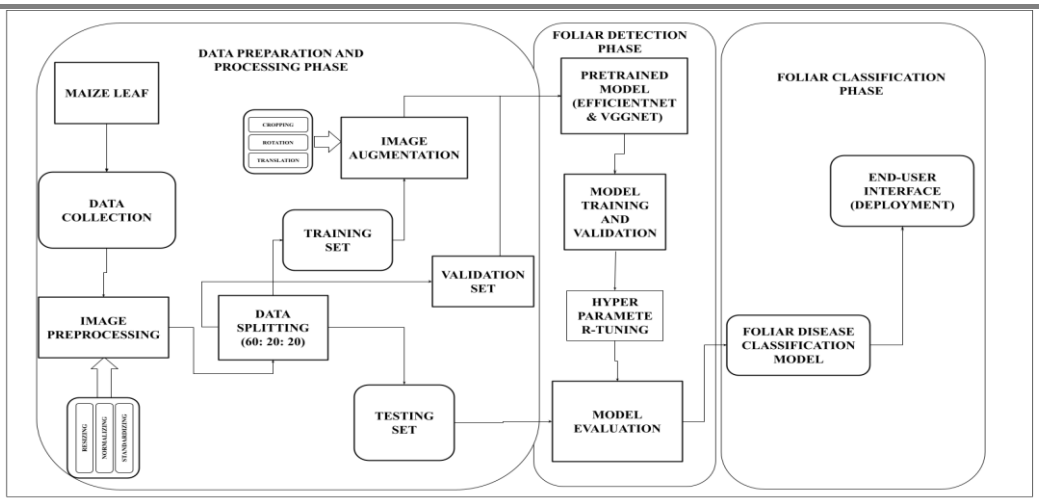

Figure 4: Framework for the Proposed Model

MODEL DESCRIPTION

All the processes involved in the infection detection are hereby described accordingly.

Data Collection and Preprocessing

The dataset for this work was downloaded from the Kaggle website. This data is called the "Corn or Maize Leaf Disease Dataset" and it contains different classes of maize leaf images. It includes four categories: Blight, Common Rust, Gray Leaf Spot, and Healthy Maize Leaves. Samples of the images contained in the dataset are shown in Figures 5 to 8 respectively.

Figure 5: Maize Leaf with Leaf Blight

Figure 6: Maize Leaf with Common Rust

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

Figure 7: Maize Leaf with Gray Leaf Spot

Figure 8: Healthy Maize Leaf

The data contains image features necessary to classify the image into the right classes. These features are areas or spots in the image that represent the image's visual information. Some of these features are explained below:

i. **Edges:** Edges are points in an image where there is a boundary between two different regions of an image. The edges detect the boundaries of the affected area.

ii. **Corners:** This is a point where two or more boundaries meet. They are used to distinguish the classes of foliar diseases. The different classes of foliar diseases can be identified by how close the spots are to each other.

iii. **Blobs:** This is a region in the image where the property of the image (intensity, brightness, colour) differs from that of its environment. In the case of infected leaves, spots, blotches, etc., can be used to identify the class of foliar disease.

Image Preprocessing

This is the process of manipulating and analyzing the dataset images. The steps involved are:
i. Image Resizing: This step involves changing the dimension of the images without affecting the aspect ratio. The images' size must be unified to a standard pixel before the model training so as to reduce the computational stress thereof.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

ii. Image Normalization: This is a step that is used to rescale the values of an image to a specific range of values. The pixel values are resized to between 0 and 1. The formula behind the normalization of an image to the range of 0 and 1 is stated below:

Normalized x = $x - \min(x)/\max(x) - \min(x)$

iii. Image Standardization: This process normalizes pixel values so that they have a consistent scale and distribution across the dataset. This helps the algorithms converge faster and ensures that the model treats each pixel equally.

Augmentation

A random transformation approach is employed in this study to help balance the dataset by making images more realistic and restrain overfitting during the training process. The technique involves randomly transforming the image data to generate more of the training data. The methods used in this work are explained below:

i. Random Rotation: The images are rotated at different angles. This allows the model to identify unseen images when they are rotated at any angle.

ii. Random cropping: This involves cropping different parts or pieces of the image. The model can learn imperfect pieces of data improving its opportunity to fit into real-world scenarios.

iii. Random Flipping: This involves randomly turning the image, horizontally or vertically, This way, the model learns from the image from different perspectives.

iv. Random Zooming: The image is zoomed in or zoomed out at random ranges for the model to identify the image when in different sizes.

Data Splitting

The dataset is separated into training, validation and testing. The ratio of the splitting is 60:20:20, which implies 60% for training, 20% for validation, and 20% for testing. This will determine the performance of the model.

Foliar Detection Phase

This involves the processes involved in the detection of foliar diseases in maize leaf. The section discusses the construction of models with classifiers.

Convolutional Neural Network (CNNs)

Convolutional neural network (CNN) is a class of artificial neural networks with feed-forward back propagation, which has already successfully been applied to solving problems of computer vision and machine learning. It comprises of many interconnected layers, which includes convolution, pooling, and fully-connected layers. The convolution and pooling layers extract features from the images and the fully connected layers classify the extracted features into the right class.

Convolutional Layers

Convolutional Neural Networks (CNNs) utilize a convolution layer to capture spatial patterns and hierarchical representations in the image. This layer applies a series of convolution operations to the input image and helps to identify patterns and features in the image. The first few layers of a CNN typically extract low-level features, such as edges and corners, while the deeper layers find more complex patterns, such as objects and faces.

Pooling Layer

The pooling layer aims to reduce the resolution of the features to make the extracted features robust against noise and distortion. This layer changes the height and width of the feature maps but does not change their

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

depth. The pooling layer is responsible for downsampling or subsampling in the network. This reduces the number of parameters and calculations, improving efficiency and preventing overfitting.

Flatten Layer

The Flatten layer converts a multidimensional array (the extracted feature map) into a one-dimensional array. The Flatten layer takes the high-dimensional feature maps from the previous layer and "flattens" them into a single 1D array of values, which is fed into the Dense Layer for classification and prediction.

Activation Function

The Rectified Linear Unit and Softmax Activation function are utilized for this study

1. Rectified Linear Unit (ReLU): This is the most common activation function. It displays the input when it is positive (>0) and outputs zero when negative (<0). The function works based on the equation below:

$$f(x) = \max(0, x)$$

2. Softmax function: The softmax function is used for a multiclass classification problem, it is a combination of several sigmoids. It returns the probability for an input to belong to each class. It is represented by the mathematical expression below:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

CNN Algorithm

A CNN Classifier follows an algorithm to get an accurate class for any input. This section provides the general algorithm which the CNN works upon.

Step 1: Get the input matrix to the network with f(x) = max(0,x)

Step 2: Conduct convolution operation on the input matrix.

Step 3: Filter input matrix to produces a feature map.

Step 4: Pass the feature map through ReLU to introduce non-linearity

Step 5: Apply pooling operation to the feature map

Step 6: Step 2 - 5 repeated for all iterations of convolutions and pooling operation.

Step 6: Extract feature map from step 6 above

Step 7: Pass feature map for classification.

Step 8: classifying the output into the class it belongs to using softmax activation function

Step 8: Compute the loss function using the class output from the final layer.

Step 9: Carry out backpropagation to reduce error and update training parameters

Step 10: After the backpropagation, a forward pass is performed.

Step 11: Repeat Steps 2 to 10 until the network converges.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

IMPLEMENTATION, RESULTS AND DISCUSSIONS

This section explains the approach used while researching and building the work. The approach used involves training the dataset on pre-trained models which include VGG16 and EfficientNet. The process of implementation involves the collection of the dataset, processing and augmenting the dataset, fine-tuning the pre-trained model to fit the data, selecting the right evaluation metric and deploying the model to a simple user interface.

Data Collection

The dataset was collated from the popular PlantVillage and PlantDoc library and is available on Kaggle website for download. The dataset contains four classes of the maize leaf. Each class discussed in section 3 is adequately represented in the dataset. Table 1 shows the classes of the maize leaf and the number of images in the data.

Table 1: Dataset Description

S/N	DATA CLASS	NUMBER OF IMAGES
1	Leaf Blight	1146
2	Grey Leaf Spot	574
3	Common Rust	1308
4	Healthy	1162

Data Processing

This section discusses the techniques used in preprocessing and preparing the data for the training process. Each of the techniques is shown and explained below:

Data Splitting

Splitfolder, a package present in the Python3 version, is used in splitting the dataset into the train, validation and test sets with a proportion of 60:20:20% respectively. The output train and test sets are saved into different folders with “corn/train”, “corn/val” and “corn/test” as their names. The train and validate data contains the images to be sent into the Convolutional Neural Network.

Data Augmentation

Random Augmentation on the training set was done using the ImageDataGenerator class in the Tensorflow library. The images are rescaled such that the pixels are reduced from 0-255 to 0-1. The augmentation process was carried out at random. Some of the images augmented can be seen in Figure 10.

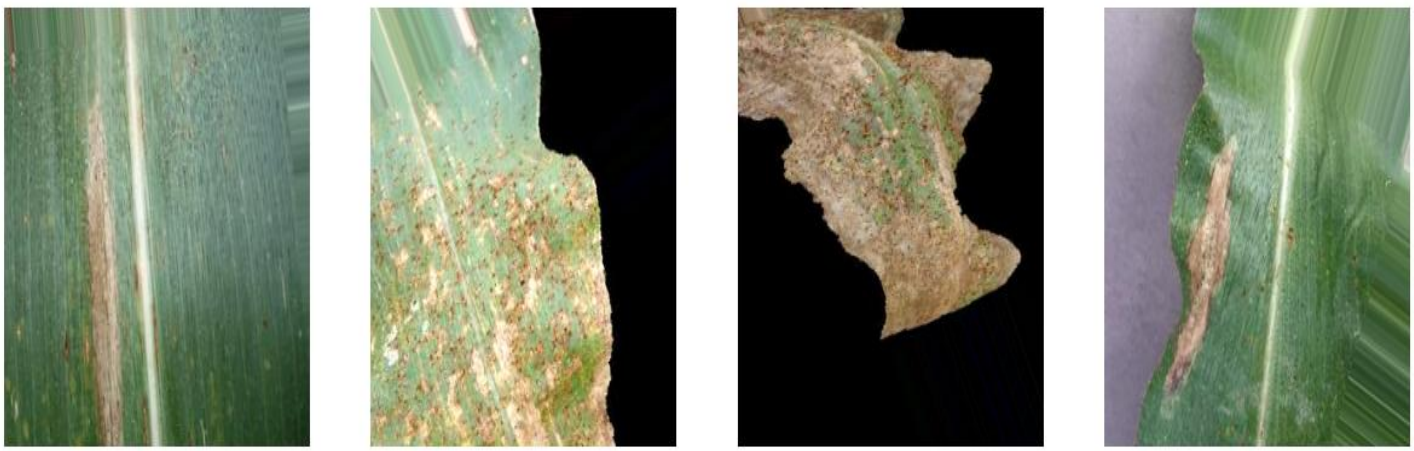

Figure 10: Random Images from the Augmentation Process.

Images from the augmentation of the train set are fed into the pre-trained convolutional neural networks.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

MODEL BUILDING AND TRAINING

After pre-processing the data, different pre-trained models are used to train the training set. The models are fine-tuned to give out the best performance and are then compared with each other. VGG16 and EfficientNet are selected as the base models for the project. The summary for the two models used is given below:

VGG16

VGG16 model is a convolutional neural network model which consists of 16 layers and is trained on the ImageNet dataset which is a publicly available dataset that consist of millions of images classified into various categories. The VGG16 model is modified to classify the foliar diseases in maize crops.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 global_average_pooling2d_3  (None, 512)               0
  (GlobalAveragePooling2D)

 dense_10 (Dense)            (None, 1024)              525312

 batch_normalization_98 (Ba  (None, 1024)              4096
 tchNormalization)

 dropout_6 (Dropout)         (None, 1024)              0

 dense_11 (Dense)            (None, 512)               524800

 batch_normalization_99 (Ba  (None, 512)               2048
 tchNormalization)

 dropout_7 (Dropout)         (None, 512)               0

 dense_12 (Dense)            (None, 256)               131328

 batch_normalization_100 (B  (None, 256)               1024
 atchNormalization)

 dropout_8 (Dropout)         (None, 256)               0

 dense_13 (Dense)            (None, 128)               32896

 batch_normalization_101 (B  (None, 128)               512
 atchNormalization)

 dropout_9 (Dropout)         (None, 128)               0

 dense_14 (Dense)            (None, 4)                 516

=================================================================
Total params: 15937220 (60.80 MB)
Trainable params: 14787972 (56.41 MB)
Non-trainable params: 1149248 (4.38 MB)
_________________________________________________________________
```

Figure 11: Summary of the VGG16 model after fine-tuning.

The last block of the VGG16 model is unfrozen and is set to trainable i.e. the weights of the layers are trainable. The outputs from the unfrozen layers are passed through a set of fully connected layers which contain the BatchNormalization layer for reducing the generalization error, the Dropout layer to avoid overfitting and the Dense layers to carry out the final classification.

Efficientnet

EfficientNet is a family of convolutional neural networks which was introduced to optimize both efficiency and accuracy in a Convolutional neural network. The EfficientNet is built upon the concept known as “compound scaling”. Compound scaling is a balanced way of scaling the dimensions of the neural network which include the width, depth and resolution. The compound scaling is guided by the compound coefficient denoted by phi which uniformly scales the dimensions.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

```
final_model.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================
input_4 (InputLayer)         [(None, 260, 260, 3)]     0         []

stem_conv (Conv2D)           (None, 130, 130, 32)      864       ['input_4[0][0]']

stem_bn (BatchNormalizatio   (None, 130, 130, 32)      128       ['stem_conv[0][0]']
n)

stem_activation (Activatio   (None, 130, 130, 32)      0         ['stem_bn[0][0]']
n)

block1a_dwconv (DepthwiseC   (None, 130, 130, 32)      288       ['stem_activation[0][0]']
onv2D)

block1a_bn (BatchNormaliza   (None, 130, 130, 32)      128       ['block1a_dwconv[0][0]']
tion)

block1a_activation (Activa   (None, 130, 130, 32)      0         ['block1a_bn[0][0]']
tion)

block1a_se_squeeze (Global   (None, 32)                0         ['block1a_activation[0][0]']
AveragePooling2D)

block1a_se_reshape (Reshap   (None, 1, 1, 32)          0         ['block1a_se_squeeze[0][0]']
e)

block1a_se_reduce (Conv2D)   (None, 1, 1, 8)           264       ['block1a_se_reshape[0][0]']

block1a_se_expand (Conv2D)   (None, 1, 1, 32)          288       ['block1a_se_reduce[0][0]']

block1a_se_excite (Multipl   (None, 130, 130, 32)      0         ['block1a_activation[0][0]',
y)                                                               'block1a_se_expand[0][0]']

block1a_project_conv (Conv   (None, 130, 130, 16)      512       ['block1a_se_excite[0][0]']
2D)

block1a_project_bn (BatchN   (None, 130, 130, 16)      64        ['block1a_project_conv[0][0]']
ormalization)
```

```
AveragePooling2D)

block7b_se_reshape (Reshap   (None, 1, 1, 2112)        0         ['block7b_se_squeeze[0][0]']
e)

block7b_se_reduce (Conv2D)   (None, 1, 1, 88)          185944    ['block7b_se_reshape[0][0]']

block7b_se_expand (Conv2D)   (None, 1, 1, 2112)        187968    ['block7b_se_reduce[0][0]']

block7b_se_excite (Multipl   (None, 9, 9, 2112)        0         ['block7b_activation[0][0]',
y)                                                               'block7b_se_expand[0][0]']

block7b_project_conv (Conv   (None, 9, 9, 352)         743424    ['block7b_se_excite[0][0]']
2D)

block7b_project_bn (BatchN   (None, 9, 9, 352)         1408      ['block7b_project_conv[0][0]']
ormalization)

block7b_drop (FixedDropout   (None, 9, 9, 352)         0         ['block7b_project_bn[0][0]']
)

block7b_add (Add)            (None, 9, 9, 352)         0         ['block7b_drop[0][0]',
                                                                 'block7a_project_bn[0][0]']

top_conv (Conv2D)            (None, 9, 9, 1408)        495616    ['block7b_add[0][0]']

top_bn (BatchNormalization   (None, 9, 9, 1408)        5632      ['top_conv[0][0]']
)

top_activation (Activation   (None, 9, 9, 1408)        0         ['top_bn[0][0]']
)

average_pooling2d_1 (Avera   (None, 1, 1, 1408)        0         ['top_activation[0][0]']
gePooling2D)

flatten_1 (Flatten)          (None, 1408)              0         ['average_pooling2d_1[0][0]']

Predictions (Dense)          (None, 4)                 5636      ['flatten_1[0][0]']

=================================================================
Total params: 7774198 (29.66 MB)
Trainable params: 7706630 (29.40 MB)
Non-trainable params: 67568 (263.94 KB)
_________________________________________________________________
```

Figure 12: Summary of EfficientNet Model.

The EfficientNet model is pretrained on the Imagenet dataset for an image classification task. For this project, the last layer of the EfficientNet is allowed to train while the top layers are left frozen. For the model to carry out accurate prediction on the data, a fully connected layer is added to the base model of the EfficientNet. The output layer of the fully connected layer gives out four classes. It uses the softmax activation function as its activation.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

MODEL TRAINING

The pre-trained models were allowed to train for 50 epochs with a batch size of 64. The models adopted Adam as the optimizer used while training. Categorical cross-entropy and Accuracy were used as the loss function and accuracy metric for evaluating the training and validation set.

```
]:
  hist=final_model.fit(train_data,epochs=30,
                       validation_data=(valid_data),
                       callbacks=callbacks,class_weight=class_weights)

Epoch 1/30
2023-11-10 03:06:42.985870: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:954] layout failed: INVALID_ARGUMENT: Siz
e of values 0 does not match size of permutation 4 @ fanin shape inmodel_1/block1b_drop/dropout/SelectV2-2-TransposeNHWCToNCH
W-LayoutOptimizer
79/79 [==============================] - ETA: 0s - loss: 0.6817 - accuracy: 0.8120
Epoch 1: val_loss improved from 1.49405 to 0.72187, saving model to my_model.h5
79/79 [==============================] - 93s 523ms/step - loss: 0.6817 - accuracy: 0.8120 - val_loss: 0.7219 - val_accuracy:
0.7093 - lr: 1.0000e-04
Epoch 2/30
79/79 [==============================] - ETA: 0s - loss: 0.2456 - accuracy: 0.9295
Epoch 2: val_loss improved from 0.72187 to 0.37149, saving model to my_model.h5
79/79 [==============================] - 42s 531ms/step - loss: 0.2456 - accuracy: 0.9295 - val_loss: 0.3715 - val_accuracy:
0.8660 - lr: 1.0000e-04
Epoch 3/30
79/79 [==============================] - ETA: 0s - loss: 0.1495 - accuracy: 0.9614
Epoch 3: val_loss improved from 0.37149 to 0.22056, saving model to my_model.h5
79/79 [==============================] - 42s 522ms/step - loss: 0.1495 - accuracy: 0.9614 - val_loss: 0.2206 - val_accuracy:
0.9151 - lr: 1.0000e-04
Epoch 4/30
79/79 [==============================] - ETA: 0s - loss: 0.1283 - accuracy: 0.9669
Epoch 4: val_loss improved from 0.22056 to 0.15827, saving model to my_model.h5
79/79 [==============================] - 42s 527ms/step - loss: 0.1283 - accuracy: 0.9669 - val_loss: 0.1583 - val_accuracy:
0.9474 - lr: 9.0484e-05
Epoch 5/30
79/79 [==============================] - ETA: 0s - loss: 0.1125 - accuracy: 0.9709
```

Figure 13: Training Loop

RESULTS AND EVALUATION

The effectiveness and efficiency of the models are evaluated with accuracy and F1-score metrics. Other metrics such as precision, recall, and Confusion matrix scores will also be considered.

Accuracy: Accuracy measures the percentage of correctly classified samples compared to the total number of samples in the dataset.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

F1-score: The F1 score is calculated as the harmonic mean of precision and recall. It balances precision and recall, providing a single score that summarizes the overall performance of a classifier. A higher F1 score indicates better performance.

$$\text{F1 score} = \frac{2*\text{Precision}*\text{Recall}}{(\text{Precision}+\text{Recall})}$$

Both models were trained for 50 epochs. The train set contained 2511 images, a validation set of 836 images, and a test set of 841 images. The VGG16 model had an accuracy of 91% and an F1 score of 90%, while the EfficientNet model had an accuracy of 96% and an F1 Score of 95% after evaluating the test data. the system result is shown in figure 14 and 15 respectively.

```
]:
test_loss, test_acc = model.evaluate(test_generator)
f1_score= f1_score(y_real, all_pred, average="macro")
print('Test Loss:', test_loss)
print('Test Accuracy:', test_acc)
print("F1_Score:", f1_score)

27/27 [==============================] - 4s 126ms/step - loss: 0.3486 - accuracy: 0.9191
Test Loss: 0.34856116771698
Test Accuracy: 0.9191438555717468
F1_Score: 0.9020035297525548
```

Figure 14: VGG16 Model Evaluation on the Test Set

```
:
test_loss, test_acc = final_model.evaluate(test_data)
f1_score= f1_score(y_real, all_pred, average="macro")
print('Test Loss:', test_loss)
print('Test Accuracy:', test_acc)
print("F1_Score:", f1_score)

27/27 [==============================] - 2s 86ms/step - loss: 0.1161 - accuracy: 0.9620
Test Loss: 0.11612752079963684
Test Accuracy: 0.9619500637054443
F1_Score: 0.9540211884732086
```

Figure 15: EfficientNet Model Evaluation on the Test Set

The visual representation of the VGG16 Model Accuracy and Loss is depicted in figure 16, which shows the model's performance per training epoch.

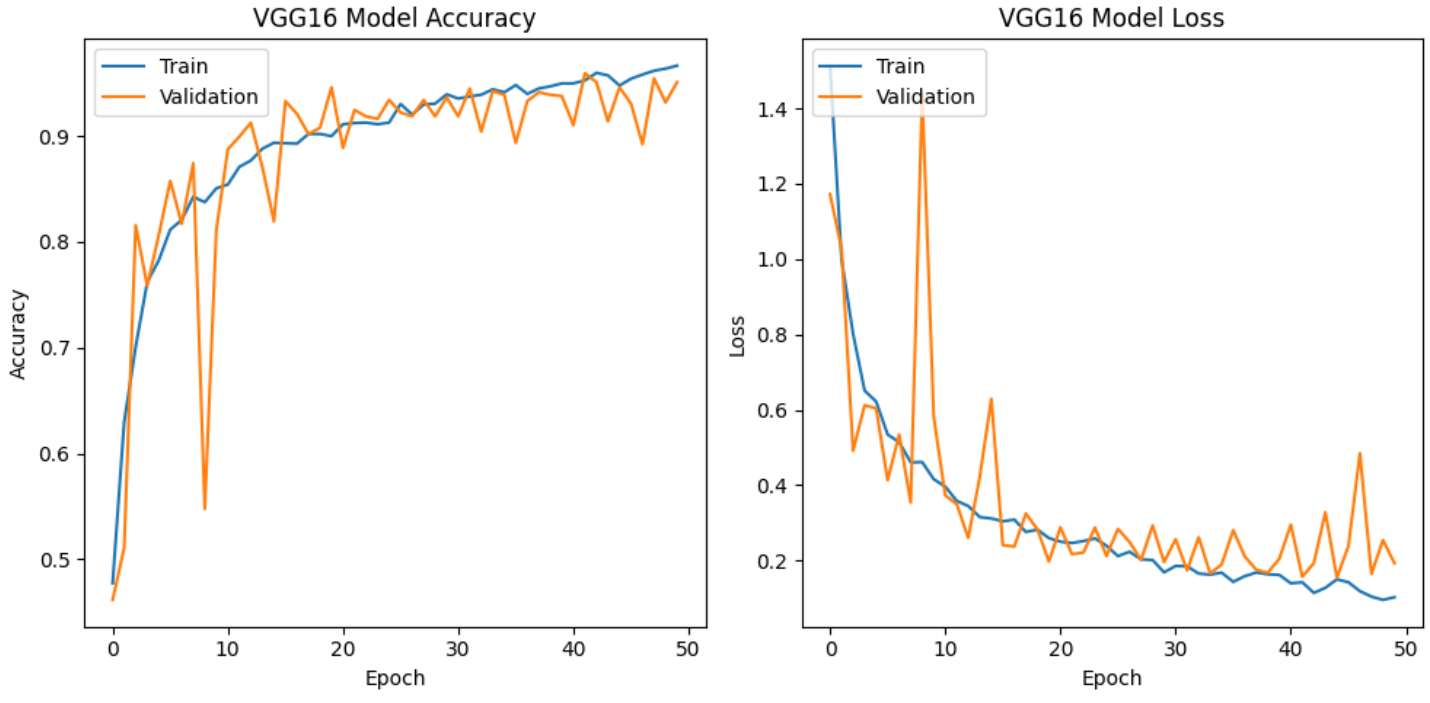

Figure 16: VGG16 Model Accuracy and Loss.

The accuracy curve of the training data also shows that the model fits well on the training data but the fluctuation in the curve of the validation data shows that the model does not perform well on underrepresented classes.

Confusion Matrix

The confusion matrix for the VGG16 model is shown in figure 17. It could be observed that the model performs well on the set and classifies the represented classes well. It also shows that the model did not fit well on the underrepresented class which is the Gray Leaf Spot. The model correctly classified "Blight" 222 times, but confused it with "Common_Rust" 13 times, "Gray_Leaf_Spot" 26 times, and misidentified "Healthy" as "Blight" 4 times.

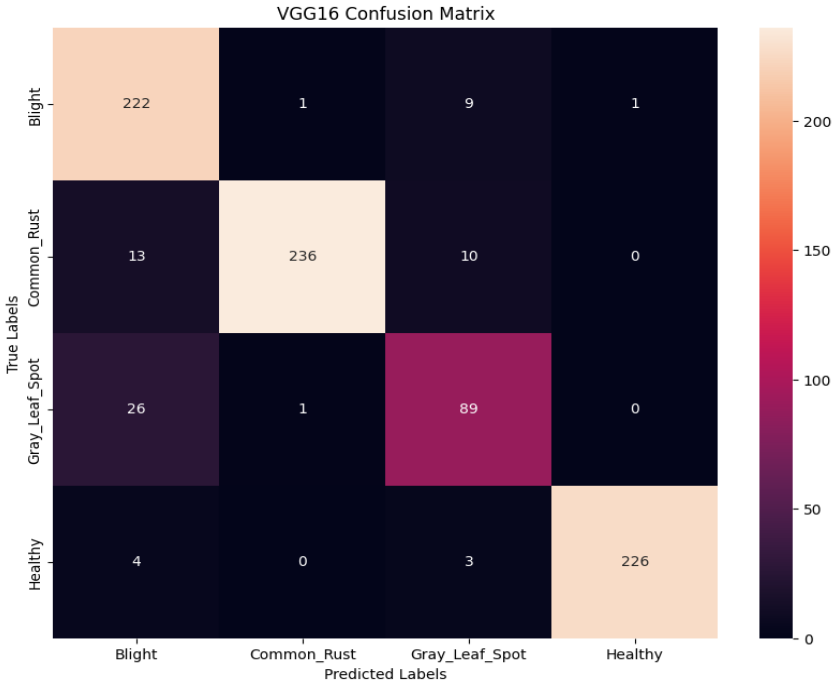

Figure 17: Confusion Matrix for VGG16 Model

The visual representation of the EfficientNet Model Accuracy and Loss is also depicted in figure 18. The plots correlate with the accuracy and F1 score which proves that the training data and validation data fit well with the model. The model loss plot also shows to be reducing over time, meaning the model learns the training data well and also performs well on the validation data.

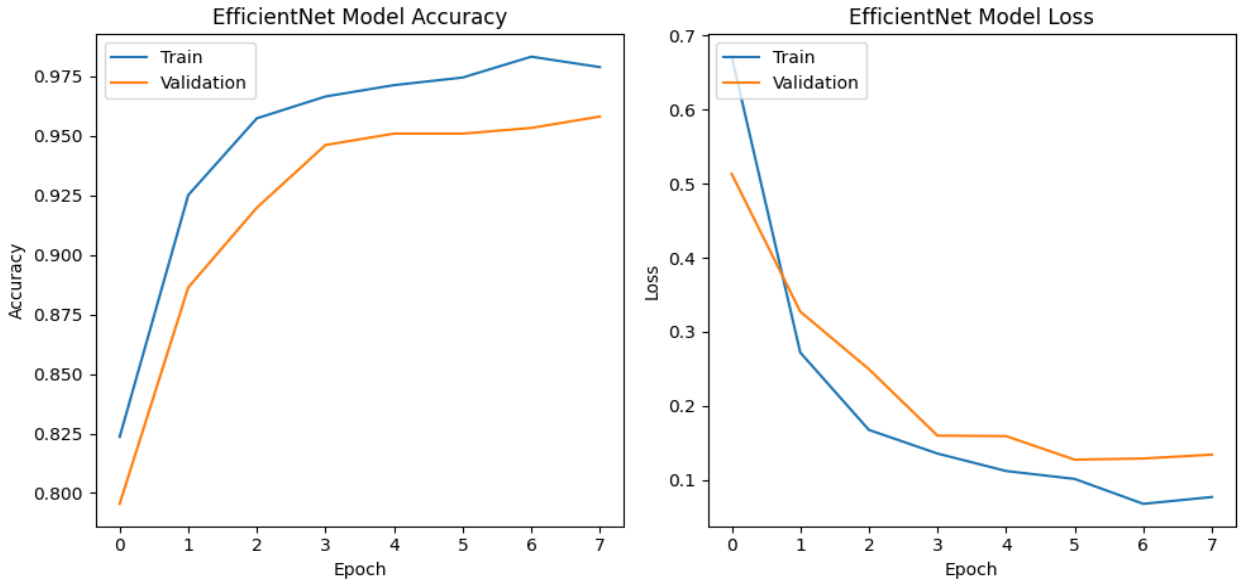

Figure 18: EfficientNet Model Accuracy and Loss.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

Figure 19 shows the Confusion Matrix for the EfficientNet. The model fits well on the test data and classifies it into the right classes although it misclassifies in some cases. It also shows that the model generalizes well on each of the classes present in the data, including the underrepresented data (Gray Leaf Spot).

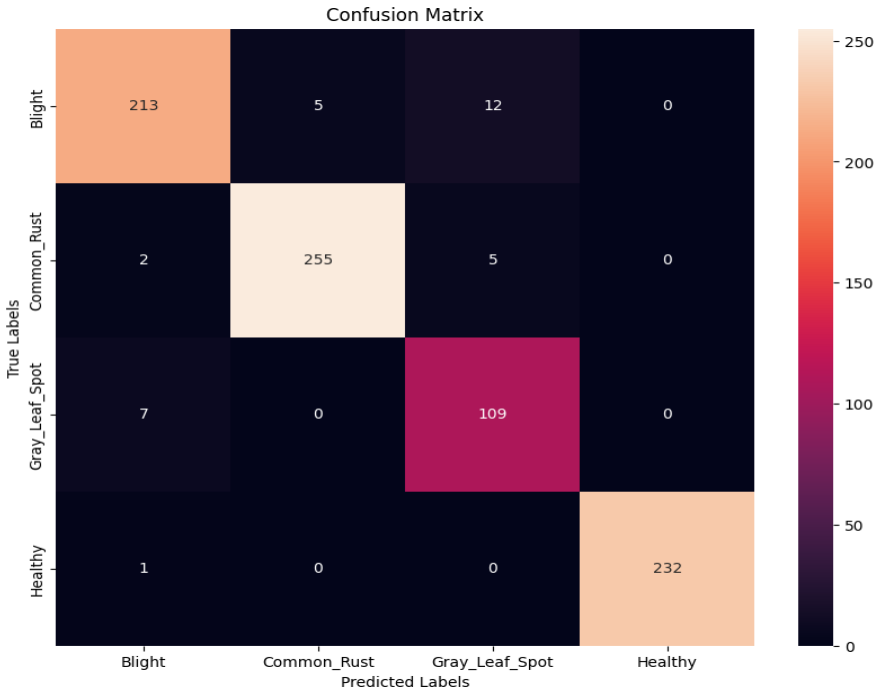

Figure 19: Confusion Matrix for the EfficientNet

Deployment

Upon concluding the model training and evaluation stage, the EfficientNet model was selected since it generalizes better on the data and exhibits optimal performance on unseen test data.

Streamlit User Interface and Experience

Streamlit is an open-source app framework used in the deployment of data science and machine learning projects. Streamlit was used to craft a user interface that is accessible to individuals without technical experience. The interface includes an image upload feature, a brief introduction of the app, real-time model inference and the diagnosis displayed in a clear and concise format.

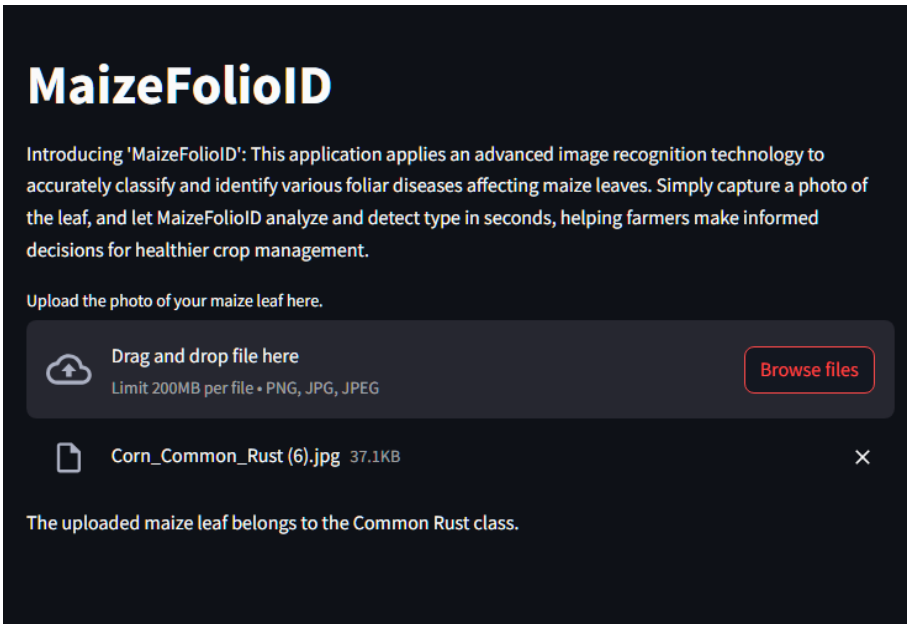

Figure 20: User Interface for the deployed model.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue IV April 2026

CONCLUSION AND RECOMMENDATION

This study examined the effectiveness of deep learning techniques in automatically detecting foliar diseases in maize. The pre-trained models of Convolutional Neural Networks (CNNs) - EfficientNet and VGG16 were employed to accurately classify and differentiate between healthy maize leaves and those affected by common diseases like Leaf Blight, Common Rust, and Gray Leaf Spot. The results suggested that with the right training data and model design, the methods were found to be scalable and sustainable compared to traditional disease detection methods, which are often subjective, labor-intensive and susceptible to human error. An appropriate and easy method of deploying the model using Streamlit has resulted in a scalable, robust and accessible tool for farmers and agricultural experts. This tool allows users to leverage cutting-edge deep learning solutions in maize management.

To enhance the system's efficiency and impact, it is recommended that more data should be collated for the underrepresented classes so as make the model generalize better with data. Also, partnerships should be created with agricultural organisations and technology companies to help improve the availability of resources and expertise.

REFERENCES

1. Askale, G. T., Yibel, A. B., Taye, B. M., & Wubneh, G. D. (2025). Mobile based deep CNN model for maize leaf disease detection and classification. Plant Methods, 21(1), 72.
2. Atila, Ü., Polat, H., & Özkan, M. (2020) "EfficientNet-based deep learning architecture for plant disease detection." Computers and Electronics in Agriculture, 178, 105739. doi:10.1016/j.compag.2020.105739.
3. Bedi, P., & Gole, A. (2020) "Hybrid model based on Convolutional Auto Encoder and Convolutional Neural Network for automatic plant disease detection." In 2020 International Conference on Smart Electronics and Communication (ICOSEC) (pp. 1-5). IEEE. doi:10.1109/ICOSEC49019.2020.9266972.
4. Chibesa, M., et al. (2021) "Machine Learning Techniques for Crop Disease Detection and Classification: A Review." Computers and Electronics in Agriculture, 191, 106264.
5. Guo, Y., Zhang, Y., Zhang, C., & Jiang, Y. (2020) "A deep learning-based mathematical model for plant disease detection and recognition." International Journal of Advanced Technology and Engineering Exploration, 7(1), 1-6. doi:10.19101/ijatee.2022.10100136.
6. Hassan, M., Elhoseny, M., Hassanien, A. E., & Muhammad, K. (2021) "Deep learning-based plant disease detection using convolutional neural networks." Computers, Materials & Continua, 7(2), 2017-2032. doi:10.32604/cmc.2021.014764.
7. Jana, S., Begum, A. R., Selvaganesan, S. (2020) "Design and Analysis of Pepper Leaf Disease Detection Using Deep Belief Network." European Journal of Molecular & Clinical Medicine, Volume 7, pp. 1724-1731.
8. Liu J, He C, Jiang Y, Wang M, Ye Z, & He M. (2024) A high-precision identification method for maize leaf diseases and pests based on LFMNet under complex backgrounds. Plants. 2024. https://doi.org/10.3390/plants13131827.
9. Liu, J. and Wang, X. (2021) "Plant diseases and pests detection based on Deep Learning: a review," Plant Methods, pp.17, 22.
10. Wang, N., Sundin, G. W., Fuente, L. D. L., Cubero, J., Tatineni, S., Brewer, M. T., ... & Munkvold, G. (2024). Key challenges in plant pathology in the next decade. Phytopathology®, 114(5), 837-842.
11. Yuan, Y., Chen, L., Wu, H., & Li, L. (2022). Advanced agricultural disease image recognition technologies: A review. Information Processing in Agriculture, 9(1), 48-59.