



## RESEARCH ARTICLE

## OPEN ACCESS

## HYBRID DEEP LEARNING MODEL FOR AUTOMATED PLANT DISEASE CLASSIFICATION

Hayat Meddeber<sup>1\*</sup> and Meriem Meddeber<sup>2</sup><sup>1,2</sup>Department of Computer Science, LABTEC-AI Laboratory of Mascara, University of Mustapha Stambouli, 29000 Mascara, Algeria.<sup>1</sup><https://orcid.org/0009-0006-9229-7383>, <sup>2</sup><https://orcid.org/0000-0001-5276-1328>Email: \*[hayet.meddeber@univ-mascara.dz](mailto:hayet.meddeber@univ-mascara.dz), [meriem.meddeber@univ-mascara.dz](mailto:meriem.meddeber@univ-mascara.dz)

## ARTICLE INFO

**Article History**

Received: February 25, 2026

Reviewed: March 20, 2026

Accepted: May 10, 2026

Published: June 30, 2026

**Keywords:**Plant Disease Detection,  
Plant Village,  
Deep Learning,  
Hybrid Learning Model,  
AgriNet.

## ABSTRACT

Plant disease detection is essential to ensure the quality of agricultural production. However, manual identification of disease symptoms is often tedious and susceptible to human error. Deep learning techniques have been widely integrated into agricultural systems, providing innovative solutions thanks to their robustness. Most of these studies are essentially based on Convolutional Neural Networks (CNNs) architectures. This paper presents a lightweight and accurate model for the classification of 29 plant leaf classes using the PlantVillage dataset. We propose a hybrid learning model called Agriculture Network (AgriNet). AgriNet consists of only six layers, and integrates the strengths of three CNN models : Residual Neural Networks-50 (ResNet-50), Visual Geometry Group-16 (VGG-16) and AlexNet. ResNet-50 is a very deep and powerful network, whereas VGG-16, despite having a large number of parameters, achieves good accuracy using (3×3) filters. In contrast, AlexNet presents a relatively simple architecture compared to the other two deep models. An extensive set of experiments was conducted to evaluate the performance of the proposed model for plant disease detection. The analysis, based on assessment measures, including accuracy, precision, recall, and F1-score, demonstrates that our model achieves highly reliable results, with an accuracy of 97.13%. Furthermore, AgriNet maintains low complexity due to its lightweight hybrid architecture and reduced parameters count.



Copyright ©2026 by authors and Galileo Institute of Technology and Education of the Amazon (ITEGAM). This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

## I. INTRODUCTION

Agriculture can be defined as the process of raising animals, cultivating crops, and producing food. It also includes preparing plants to improve the quality of production. Plant disease is a natural issues that affect plant development and, in severe cases lead to plant death. It can cause significant damage to agricultural crops [1]. Therefore, plant disease detection is fundamental to preventing reductions in food production. The use of computer systems has driven a significant transformation in agriculture. This convergence, referred to as Digital Agriculture, relies on technologies including artificial intelligence (AI) to improve and optimize agricultural yields. Earlier studies have employed machine learning (ML) and deep learning (DL) techniques for the detection and classification of plant diseases. Several ML algorithms have been developed, such as K-means clustering [2], Naïve Bayes [3], support vector machine[4], and Random Forest [5], [6].

Deep learning technique is a specialized and swiftly advancing domain of machine learning. Convolutional neural networks (CNNs) are commonly integrated into this methodology to handle image classification tasks. In [7], the authors have shown that deep learning outperforms traditional machine learning techniques in identifying diseases in plant leaves. The main advantage of deep learning is the direct exploitation of image without any intervention from a human expert. In plant disease detection, Convolutional Neural Networks analyze leaf images by learning discriminative features, including patterns linked to shape, texture, and color and then classify them into categories that correspond to specific plant diseases. The quantity of features increases as the total number of the layers increases. Several CNN architectures have been developed for disease classification, such as AlexNet, VGG-16, and ResNet-50.

The selection of these three models was motivated by their complementary capabilities: while some excel in feature extraction, others provide greater network depth or improved training stability, making them well-suited for integration in our approach. AlexNet was developed in 2012 [8]. The AlexNet architecture was relatively straightforward in design, consisted of 8 layers : five convolutional layers and three dense layers. It utilized ReLU activation functions and dropout for regularization. The first layers of AlexNet used larger kernel sizes (11x11 and 5x5). This structure allowed the model to capture low- and mid-level patterns. Deeper layers progressively learned more abstract and hierarchical representations, which were crucial for achieving high performance in image classification tasks [9], [10]. In 2014, the Visual Geometry Group-16 (VGG-16) was proposed [11]. This model employed 13 convolutional layers with (3x3) filters, and 3 dense layers. The model demonstrate a strong ability to extract rich and discriminative features from images, which were essential for accurate classification.

This capability made it particularly well-suited for agricultural applications. VGG-16 is known for its high number of parameters [12], [13]. Residual Neural Network-50 (ResNet-50) was introduced in 2016 [14]. It featured 50 layers, and has been recognized for its remarkable depth and performance in plant disease detection. ResNet-50 incorporates residual blocks, which enable it to capture finer details that may be missed by other models. It has proven effective when applied to datasets with complex patterns and multiple categories [15], [16]. In this work, we propose AgriNet (Agriculture Network), a novel model that detects plant diseases by fusing the three prominent deep learning architectures: AlexNet, VGG-16, and ResNet-50. We have two main goals: The first is to detect and categorize 29 types of plant diseases with the highest possible accuracy. The second is to develop a lightweight model, allowing AgriNet to be used on mobile or IoT devices, even in environments with limited computational resources.

To achieve these goals, we have defined the following secondary objectives:

- To thoroughly analyze and effectively leverage the strengths of AlexNet, VGG-16, and ResNet-50, enabling AgriNet to combine their advantages for higher accuracy and better generalization in plant disease classification.
- To design a novel, lightweight architecture that is computationally efficient and adept at capturing plant diseases.
- To improve classification accuracy while minimizing error rates.

The rest of this paper is structured as follows : Section 2 provides an overview of related studies. Section 3 describes the dataset employed and introduces the principles and foundations of the proposed deep learning model for plant disease classification. Section 4 presents the experimental results obtained from the experiments and includes comparisons. Finally, Section 5 reports the conclusion and perspectives for future research.

## II. RELATED WORK

This section is organized into two main subsections. Section 2.1 reviews notable studies that have used AlexNet, VGG-16, and ResNet as baseline models to enhance plant-leaf disease recognition. Section 2.2 presents studies aimed at developing lightweight and less complex deep learning architectures.

### II.1 BASELINE MODELS: ALEXNET, VGG-16, AND RESNET

To ensure accurate plant disease detection, numerous methods have been proposed. Notably, in [17] the authors developed a hybrid CNN model to classify diseases in tomato leaf images. The model was evaluated using two datasets: the Taiwan dataset and the Tomato Leaf Disease Detection dataset. The authors used AlexNet, ResNet-50, and VGG-16 to extract features. Their last fully connected layers produced feature vectors. Then, the MRMR algorithm optimized these features by removing redundancy and keeping only the most relevant ones. The refined features were combined into one set and sent to standard machine learning classifiers. This approach reached about 96.3% on the Taiwan dataset and 98.3% accuracy on the tomato disease dataset.

Farhat and Chokka [18] proposed a convolutional neural network that combines VGG-16 and ResNet50. Important features are first extracted from the image using VGG-16, and then more intricate and abstract details are captured using ResNet-50. The outputs from both models are fused to yield a richer and more complete representation of the image. The performance of the model was assessed on several datasets : PlantVillage, PlantDoc, FieldPlant, and a rice leaf. The proposed hybrid approach reached about 99% accuracy.

In [19], a fusion at the feature level is used to ensemble the CNNs: features from VGG-16 and ResNet-50 are concatenated before being fed into a unified model. Based on experiments conducted using a publicly available dataset of 2 152 images of healthy and unhealthy potato leaves, an accuracy of 98.22% was obtained by the ensemble model, compared to 96.16% for VGG-16 and 95.68% for ResNet-50. The authors acknowledge that their study is based on a small number of potato leaf images, which may limit the model's robustness or generalizability.

Ajra et al. [20] introduced a technique for plant disease detection using the AlexNet and ResNet-50 architectures. The authors used a Kaggle dataset containing tomato and potato leaf images to investigate various leaf diseases. To enable early disease identification, the workflow included image preprocessing, data augmentation, feature extraction, and classification, followed by accuracy-based model evaluation. Experimental findings showed that AlexNet achieved an overall accuracy of 95.9 %, while ResNet-50 reached 97.3 %. The authors noted that the dataset's relatively small size (9,473 images) could limit the generalization capacity of the models.

In [21], the authors developed a deep learning fusion approach for classifying plant leaf diseases. Features obtained from ResNet50, VGG-16, and EfficientNetB0 are combined through concatenation and subsequently passed through a neural network to generate the final classification results. The experiment involved eight balanced classes. The proposed approach obtained an accuracy of 98.25% and a loss value of 0.0315.

In these works, the hybridization of the three models was achieved by concatenating the feature vectors produced individually by each model. Reported accuracies ranged from 95% to 99%. However, most of these studies considered only ten classes (or fewer), which restricts their ability to generalize to more diverse scenarios or novel disease types. Such a lack of diversity can result in reduced robustness.

## II.2 LIGHTWEIGHT DEEP LEARNING ARCHITECTURES FOR PLANT DISEASE RECOGNITION

The model discussed in Section 2.1 (as presented in reference [18]) leverages a hybrid architecture by fusing VGG16 and ResNet50. According to the authors, the hybrid model capitalizes on the strengths of both networks while avoiding the excessive complexity that each might present in isolation. This fusion effectively reduces redundancy and improves structural efficiency.

Nguyen and Huynh [22] introduced an architecture that integrates ResNet into a modified version of VGG-16, designed to meet the constraints of low-end hardware. They demonstrate that ResNet-50 performs poorly with small batch sizes due to ineffective BatchNorm, while the proposed model removes BatchNorm and avoids this issue. Moreover, the original VGG-16's large fully connected layers create a memory burden and may worsen vanishing gradient issues. Their model addresses both memory constraints and gradient vanishing, and achieves an overall accuracy of 85.4%, outperforming ResNet-50's 75.9%.

In [23], the authors proposed an efficient lightweight model for classifying pepper leaf diseases. The dataset included images of healthy leaves and those exhibiting symptoms, these images were captured with human palms as the background, reflecting real-world field conditions. The proposed model was based on the GGM-VGG16 architecture, enhanced with Ghost modules, global average pooling (GAP), and multi-scale convolutions. Experimental results demonstrated that the model achieved 100% accuracy on images with human palm backgrounds and 87.38% accuracy on images with other backgrounds.

Zhengjie et al. [24] developed ICS-ResNet, a lightweight version of ResNet-50 that incorporates an improved spatial-channel attention mechanism and depth-separable residual blocks to reduce parameters and computation. Using cosine annealing for better training stability, the model achieved 98.87% accuracy on the PlantVillage maize leaf disease dataset while reducing parameters by about 69% compared to ResNet-50.

In [25], the authors introduced a lightweight architecture for plant disease classification, referred to as the Lightweight Pyramidal Multi-scale Attention Residual Network (LPMA-ResNet). The framework employs a dual-level pyramid structure combined with an adaptive attention mechanism to capture informative multi-scale features while suppressing noise. When evaluated on the Soybean Disease and PlantVillage datasets, LPMA-ResNet consistently outperformed existing leading approaches. Specifically, on the soybean dataset, LPMA-ResNet ( $\times 1.0$ ) achieved 2.12% higher accuracy than ShuffleNet-v2 ( $\times 2.0$ ), while requiring only about two-thirds of its parameters.

These studies demonstrate that by combining the VGG-16 and ResNet-50 models, it is possible to create deep learning architectures that decrease network complexity and enable the model perform effectively on low-end hardware or in resource-limited environments. In these works, only a limited number of classes were considered (a maximum of eight), achieving accuracies between 85% and 98%.

## III. MATERIALS AND METHODS

The methodology employed in this study is delineated in this section, including data description, pre-processing steps, and the deep learning models. Each stage is described in detail to provide a comprehensive understanding of the approach taken to develop and evaluate our plant disease classification system.

### III.1 DATA DESCRIPTION

In this work, the leaf images were obtained from the benchmark Plant Village dataset [26], with a modified version specifically prepared for the purposes of this study. The dataset contain 67 118 images of diseased and healthy plant leaves from 9 different species. It is split into 29 classes, each class represents a type of disease. Figure 1 shows a sample of leaves belonging to different classes.

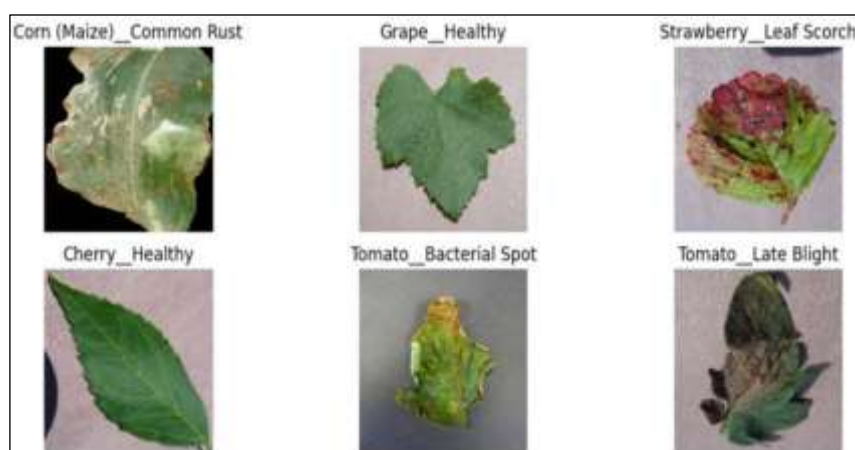


Figure 1: Samples of leaf images.  
Source: [26].

### III.2 PRE-PROCESSING

Pre-processing is very important step. Within this stage, images are subjected to resizing, normalization, and augmentation. Then, data set is partitioned into training, validation and test sets.

#### III.2.1 Resizing and normalization images

The dataset images are 256×256 pixels, but some deep learning models are built to work with a specific size (224×224). Their architecture expects inputs of that dimension. The resizing makes the images compatible with the pre-trained models. Next, the pixel values of an image (input) are adjusted to a standard range, usually between [0, 1]. This stage is the normalization, and it helps the model train faster and improves its performance [27].

#### III.2.2 Data augmentation

After collecting the images, data augmentation is utilized during the pre-processing stage. The application of data augmentation expands the training set in terms of size and variety, thereby helping to reduce overfitting and enhancing the model's generalization performance [28]. The transformations used in this pre-processing part are rotation and zooming, which allow minor variations in the images (see Figure 2).

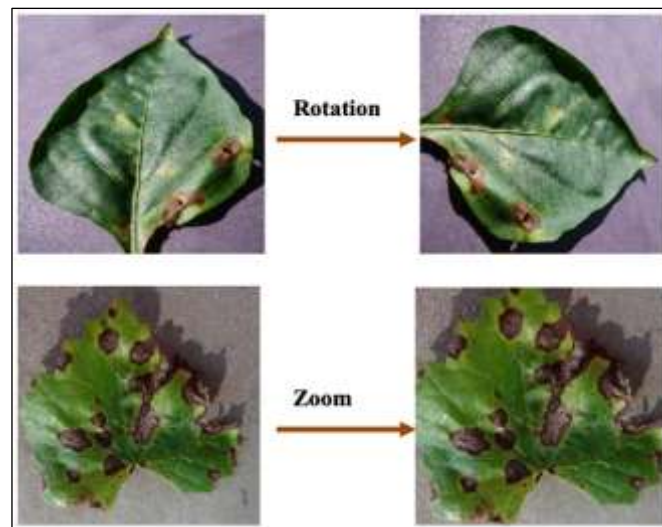


Figure 2: Data augmentation using rotation and zoom.  
Source: Authors, (2026).

#### III.2.3 Data splitting

To optimize the performance of a model, in most cases it is typically developed and evaluated on three sets of data. The Training Set (80%) represents the biggest part of the data and it is essential for the model's ability to learn and generalize. It contains 53 694 distinct images, offering a wide range of examples that enable the model to capture discriminative features across classes. The Validation Set (10%), consisting of 6 712 images, serves to assess the model's performance on unseen data during training. It helps detect overfitting and enables the fine-tuning of hyperparameters, such as batch size and learning rate. The Test Set (10%) also contains of 6 712 images and is used after training to measure the model's final effectiveness on independent data not seen during training. The histogram in Figure 3 shows the overall number of images in each category. The x-axis represents the different types of plants (29 classes): some are healthy, others affected by diseases. The y-axis shows the total number of images, combining the training, validation, and test sets.

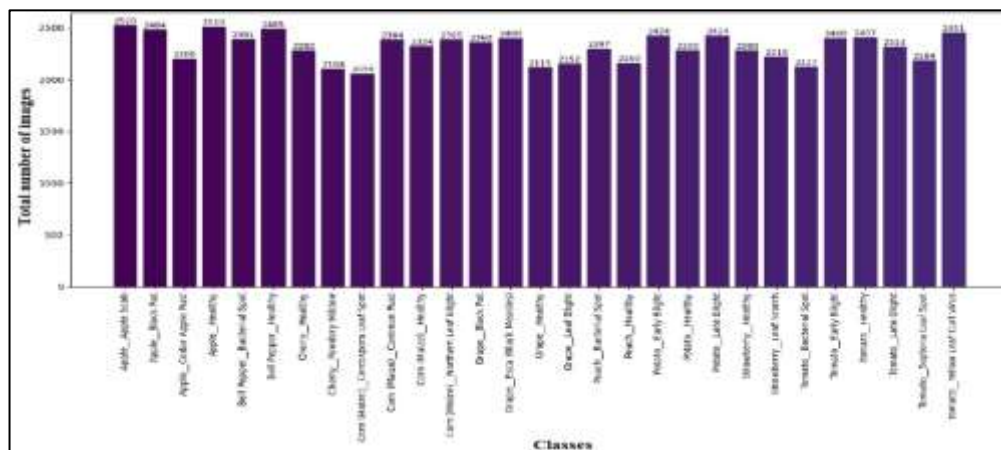


Figure 3: Histogram showing the data Split.  
Source: Authors, (2026).

The number of images differs slightly between classes (as shown in the histogram), but overall the dataset is well balanced , which is an essential condition for effective learning.

### III.3 DEEP LEARNING MODELS

In this work, three CNN architectures —AlexNet, VGG-16, and ResNet-50— were employed as baseline models. These architectures were chosen due to their demonstrated effectiveness in image classification tasks. A lightweight CNN model, named AgriNet (Agriculture Network), was developed specifically for plant disease classification.

#### III.3.1 AlexNet model

AlexNet [8] represents one of the pioneering deep convolutional neural network architectures, designed specifically for image classification tasks. It incorporates a total of eight layers: five convolutional layers, some succeeded by MaxPooling layers, and three dense layers. The Rectified Linear Unit (ReLU) is employed as the activation function. It is adopted in this network due to its superior performance over traditional functions such as tanh and sigmoid. The ReLU function is formulated as:

$$f(x) = \max(0, x) \quad (1)$$

It is a non-linear activation that leaves positive values unchanged and replaces all negative values with zero, which results in sparse activations. This sparsity, helps prevent overfitting, reduces computational cost and acts as a form of implicit regularization. Convolutional layers extract features from the image. They transform the raw image into a rich and compact representation that can be utilized by the subsequent layers for classification. By downsampling feature maps, pooling layers minimize spatial dimensions, which reduces computational cost. This also simplifies the input to the dense layers, where the combined outputs of the convolutional and pooling stages are used to perform the final classification. Figure 4 presents the structure of the AlexNet model as applied to our dataset. This modified version of AlexNet incorporates Batch Normalization to enhance the stability and efficiency of deep neural network training.

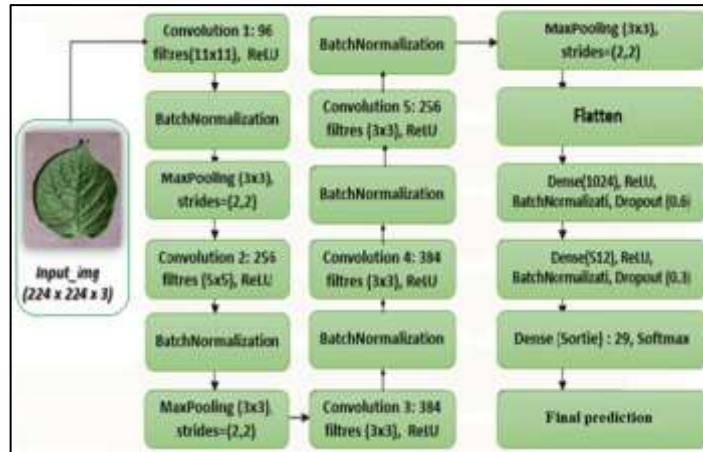


Figure 4: AlexNet model architecture.  
Source: Authors, (2026).

AlexNet model takes color images of size  $(227 \times 227 \times 3)$  pixels as input. It is composed of five convolutional layer blocks, which progressively extract hierarchical characteristics from images.

- **Block 1 :** In the first block, processing starts with a convolutional layer that applies 96 filters each of size  $(11 \times 11)$ , using a stride of 4. It's role is the extraction of low-level characteristics. The resulting feature maps are then batch normalized, which normalizes the activations of a layer within a mini-batch. After that, a max pooling with a  $(3 \times 3)$  window and stride 2 is used to reduced size of feature maps.
- **Block 2:** This block uses convolutional layer with 256 filters, each of size  $(5 \times 5)$ , using a stride of 1. Afterwards, a batch normalization and a max pooling are applied (similar to the first block).
- **Blocks 3 and 4:** These two blocks are similar and contain a convolutional layer with 384 filters, each of size  $(3 \times 3)$ , with a stride of 1. A batch normalization is applied.
- **Block 5:** 256 filters, each of size  $(3 \times 3)$ , with a stride of 1 are used in the last convolutional layer, and followed by batch normalization and max pooling. At this level, feature extraction is complete.

Next, the output produced by the final convolutional block is flattened and represented as a one-dimensional vector. This step prepares the extracted features for integration into two dense layers that produce the ultimate classification result.

- **First dense layer:** The first dense layer consists of 1024 neurons using the ReLU activation function, followed by batch normalization and a dropout of 0.6. A slightly higher dropout rate than in the original AlexNet (0.5) was used to further enhance generalization and reduce the risk of overfitting, given the model's complexity and the large number of target classes in the dataset (29 classes).

- *Second dense layer:* This dense layer comprises 512 neurons, employs ReLU activation, and is regularized with batch normalization and a 0.3 dropout rate. The lower dropout rate in this layer helps retain more learned features while still providing regularization to prevent overfitting.

In the *output layer*, 29 neurons are employed with a softmax activation to represent the 29 classification categories. The softmax activation transforms the outputs values into probabilities, approximating how likely each class is correct. The probability is determined using the following formula:

$$\text{Softmax}(s_i) = \frac{e^{s_i}}{\sum_{j=1}^K e^{s_j}} \quad (2)$$

for  $i=1, \dots, K$

The function converts the raw outputs  $s=[s_1, s_2, \dots, s_K]$  of the final dense layer into a probability distribution across the  $K$  target classes. Each term  $s_i$  in the softmax vector denotes the probability attributed to class  $i$  for the current input. These probabilities adhere to specific properties ; All probability values fall within the interval  $[0,1]$ , with their sum constrained to 1.

### Training Details

For multi-class classification, the training procedure used a categorical cross-entropy loss function in conjunction with the Adam optimizer, which had a learning rate of 0.0005.

#### III.3.2 VGG-16 model

The VGG-16 model [11] employs 13 convolutional layers that utilize small (3×3) kernels, followed by three dense layers for classification. Stacking several convolutional layers enables the network to capture progressively more detailed features from an image. AlexNet uses larger kernel sizes (11×11), while VGG-16 adopts (3×3) filters, allowing it to capture fine local details with greater accuracy. The network takes as input an RGB image of dimensions (224×224×3). This image is processed through convolutional layers with a stride of 1 pixel. Downsampling is performed using five max-pooling layers, each applied after specific convolutional layers, though not every convolution layer is followed by pooling. Max-pooling is applied on a 2×2 pixel window, with a stride of 2. In this section, the core convolutional architecture of VGG-16 was retained and initialized with weights pre-trained on ImageNet. The parameter `include_top=False` was used to exclude the original fully connected classification layers. Only the last three layers were set as trainable, while the rest of the network remained frozen. Figure 5 illustrates the VGG-16 architecture as applied to our dataset.

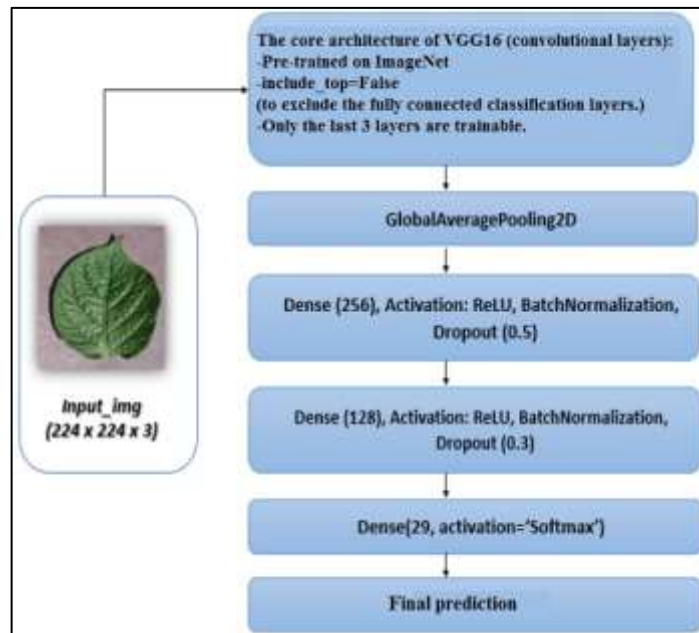


Figure 5: VGG-16 model architecture.

Source: Authors, (2026).

The added layers are:

- A *GlobalAveragePooling layer (GAP)*, which summarizes the extracted features into a vector. GAP calculates the average of all values in the feature map. The output size produced by Global Average Pooling (GAP) is minimal compared to Flatten. This decreases the trainable parameter count and gives better the model's generalization ability.
- *Two dense layers:* the first one comprises 256 neurons activated with ReLU, followed by bath normalization and a 0.5 dropout rate to limit overfitting. The second dense layer is composed of 128 neurons, also using ReLU activation, combined with bath normalization, and a 0.3 dropout rate.

The *output layer* contains 29 neurons, with each neuron representing a specific disease class, in accordance with the dataset's class count.

### Training Details

The training of the VGG-16 model employed the Adam optimizer configured with a 0.001 learning rate. The loss function selected was categorical cross-entropy, as it is widely used for multi-class classification tasks.

#### III.3.3 ResNet-50 model

ResNet-50 [14] is composed of residual blocks, each made up of several convolutional layers, along with a skip connection that directly adds the block's input to its output. The primary distinction between VGG-16 and ResNet-50 is the use of these skip (residual) connections. This design preserves gradient magnitudes during backpropagation, facilitating more effective training of deep neural networks. The structure of the ResNet-50 residual block is illustrated in Figure 6.

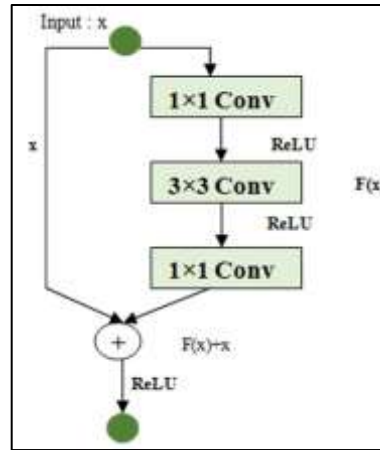


Figure 6: ResNet-50 residual block.

Source: [14].

- The first convolutional layer uses (1×1) filter to decrease the number of input channels.
- The second convolutional layer employs a (3×3) filter to capture spatial features from the input data.
- The third convolutional layer also employs a (1×1) filter to revert to the original number of channels prior to merging the output with the shortcut connection.

The *ReLU* function is applied following each convolutional layer and its ensuing batch normalization layer. *Skip connection* essentially skips convolutional layers, and allows the original input to be added directly to the convolutional layer output immediately prior to ReLU activation. A skip connection preserves important information and maintains performance. Formally, it is defined as:

$$y = F(x) + x \quad (3)$$

Here,  $x$  denotes the original input,  $F(x)$  represents the transformation performed by the convolutional layers, and  $y$  corresponds to the layer's output. In this part, we retained the core architecture of ResNet-50, training only the final ten layers while freezing the remaining ones. The architecture of ResNet-50, adapted for our dataset, is illustrated in Figure 7.

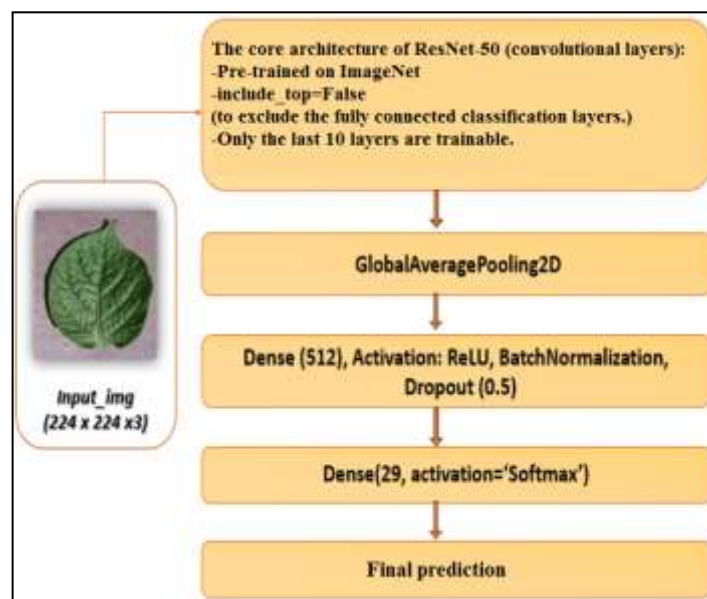


Figure 7: ResNet-50 model architecture.

Source: Authors, (2026).

Next, we added a *GlobalAveragePooling* layer to reduce the learned feature representations into a compact vector representation. Following the GAP, a dense layer with 512 neurons is applied, incorporating ReLU activation, batch normalization, and a 0.5 dropout to enhance generalization. The *output layer* comprises 29 neurons with a softmax activation, corresponding to the different diseases to be identified.

### Training Details

For training the ResNet-50 model, the Adam optimizer was used with a 0.0001 learning rate, along with categorical cross-entropy as the loss function.

#### III.3.4 AgriNet: The Proposed model

The objective of this study is to propose a lightweight and performant model, designed for mobile, IoT, or other resource-constrained devices. The proposed model AgriNet (Agriculture Network), combines: the simplicity of AlexNet, the efficiency of VGG-16, and the power of residual connection from ResNet-50. AgriNet avoids the complexity of ResNet-50. AgriNet consists of three convolutional blocks, followed by a Global Average Pooling layer (GAP) and two dense layers for the final classification, as illustrated in the figure 8. The reduced number of layers (a total of six) is inspired from the Alexnet model, which consists of only eight layers and has demonstrated high efficiency in image classification tasks. VGG-16 is known for its strong ability to extract fine-grained features through its simple convolutional layers with  $3 \times 3$  filters. In AgriNet, we preserve this strength in both the first and second blocks. The number of filters increases with network depth, allowing for more complex feature extraction. The *first block* is designed to capture initial features while incorporating batch normalization and dimensionality reduction through max pooling. Next, a dropout layer with a 0.25 rate is used.

The value is set at 0.25 to provide moderate regularization. Given the number of layers, the size of the dataset, and the number of classes (29 classes), this rate will avoid overfitting while preserving sufficient feature information for effective learning. The *second block* focuses on extracting more complex features while maintaining regularization. The convolutional layer applies 64 filters each of size  $(3 \times 3)$ . ResNet-50's strength lies in its use of residual (skip) connections, which facilitate the learning of deep representations. In AgriNet, we introduce a skip connection in the third block (mid-network), where features transition from local, detailed information to more abstract representations. Without this skip connection, this block risks over-transforming features, potentially leading to the loss of fine-grained details. By adding the skip connection at this stage, we reintroduce the original input features into the deeper, more latent representations, preserving a balance between fine details and the overall context. In a skip connection, the dimensions of  $x$  may be different from  $F(x)$  due to the convolution, which can reduce its size. A convolutional layer with 128 filters of size  $(1 \times 1)$  is applied to  $x$  to match the channel required dimensions for the addition, ensuring a direct linear path for information flow.

*Block 3* deepens the extraction of complex features while gradually reducing the dimensionality of the feature maps, ensuring that essential information is preserved for classification. The ReLU activation is applied after the addition, allowing the skip connection to transmit useful negative values, thereby improving the overall information flow.

The *GAP layer* performs dimensionality reduction, while preserving the essential information through the computation of the mean value across each feature map, producing a compact, low-dimensional vector (GAP is lighter than Flatten). The use of GAP layer followed by a dense layer greatly reduces the count of trainable parameters relative to the traditional Flatten–Dense approach. This design choice lowers memory consumption and further supports the model's suitability for lightweight deployments on mobile and embedded systems. The *first Dense layer* (512 neurons) learns abstract representations from the vectors produced by the previous layers. Dropout regularization is employed to mitigate the risk of overfitting by preventing this layer from excessively memorizing the training data. Batch normalization, applied after each convolutional layer, ensures better regularization and more stable convergence. The *final layer* is composed of 29 neurons, activated through a softmax function, corresponding to the set of diseases to be classified.

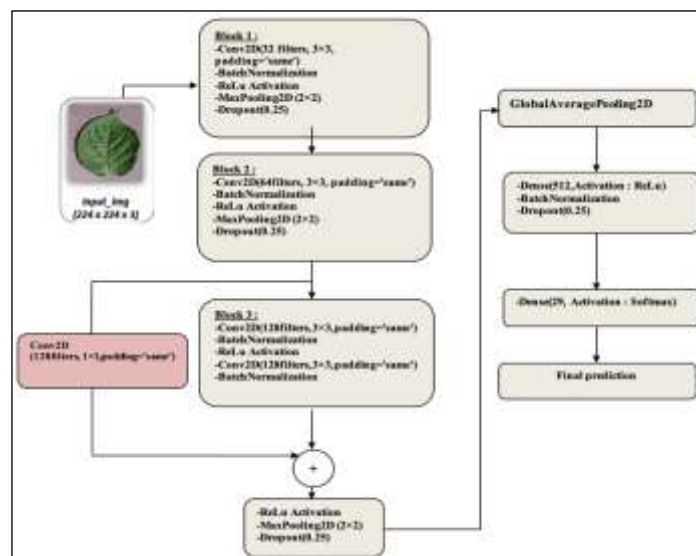


Figure 8: Proposed model architecture.  
Source: Authors, (2026).

## Training Details

The proposed model was developed using the Keras Functional API, although the architecture is relatively simple. Functional API was chosen because it allows the inclusion of a skip connection between blocks, which cannot be implemented directly with the Sequential API. Once the architecture was defined, the training process employed the Adam optimizer, initialized at a learning rate of 0.001, which was automatically reduced to 0.00001 through the ReduceLROnPlateau strategy. For this 29-class classification task, the model uses categorical cross-entropy as the loss function.

## Performance Evaluation

The assessment of the model is essential for validating its classification performance, and several evaluation metrics are employed for this purpose [29].

- *Accuracy* represents the overall proportion of predictions that were correct (whether positive or negative), as shown in equation (4):

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (4)$$

- *Precision* measures the proportion of positive predictions that are actually correct, it is mathematically defined as follows:

$$precision = \frac{TP}{TP+FP} \quad (5)$$

- *Recall* reflects the model's capacity to correctly detect all actual positive instances, as defined in equation (6):

$$Recall = \frac{TP}{TP+FN} \quad (6)$$

- *F1-score* is a metric that combines *Precision* and *Recall* into a single value, as shown in equation (7):

$$F1 - score = \frac{2 \times precision \times recall}{precision + recall} \quad (7)$$

Here, TP, TN, FP, and FN stand for True Positives, True Negatives, False Positives, and False Negatives, respectively. These metrics provide a clear and comprehensive assessment of each model, both at the global and class-specific levels.

## IV. RESULTS AND DISCUSSIONS

In deep learning, the last stage in classification is to identify the category of the input. In this work, after processing the image with a deep neural model (AlexNet, VGG-16, ResNet-50, or our proposed model 'AgriNet'), the system predicts its category. The probability for each class was calculated, and the final result is the class with the highest probability. In this way, the image is automatically categorized into one of the listed disease classes. Unlike binary models that output only 'diseased' or 'healthy', our approach provides a more detailed diagnosis.

### IV.1 PERFORMANCE ANALYSIS OF THE ALEXNET MODEL

The AlexNet model was trained for 40 epochs on our dataset. The figure below illustrates the evolution of training and validation accuracy and loss during this process.

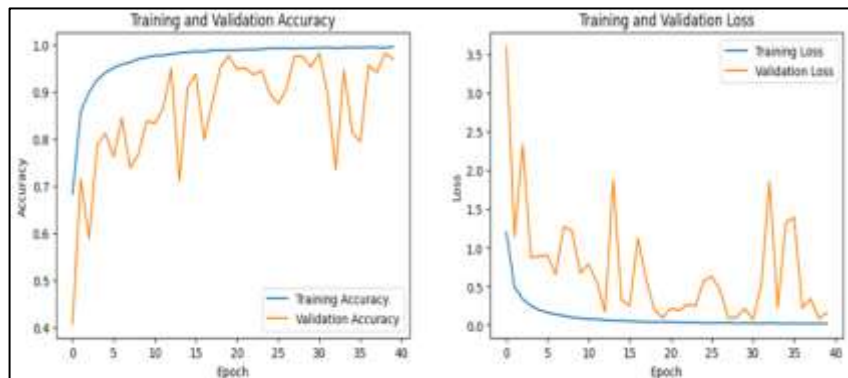


Figure 9: AlexNet model performance.

Source: Authors, (2026).

The training accuracy increases rapidly during the initial epochs and stabilizes above 95%. This indicates that the model successfully learns the training data. In parallel, the training loss significantly decreases toward zero, confirming a stable convergence process. The validation accuracy shows strong fluctuations across epochs, which reflects instability in the model's ability to generalize, despite the use of batch normalization. The validation loss is high and highly variable, and present several significant peaks, confirming a tendency toward overfitting.

The instability in the validation loss point to possible improvements, such as incorporating additional data augmentation. Overall, the model learns the training data effectively, but it has trouble generalizing to unseen data. The obtained test accuracy is 98.15%, with a loss of 0,0846.

#### IV.2 PERFORMANCE ANALYSIS OF THE VGG-16 MODEL

The VGG-16 network was trained on our dataset for 30 epochs, and its training and validation performance is illustrated in the following figure.

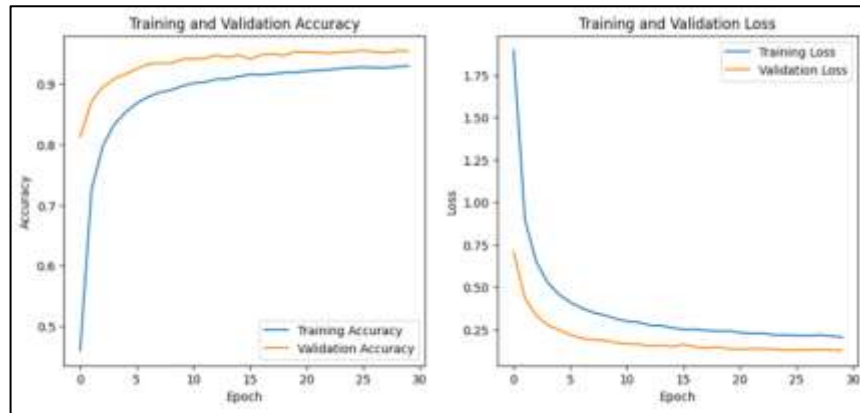


Figure 10: VGG-16 model performance.  
Source: Authors, (2026).

Training accuracy rises quickly in the first few epochs and reaches around 92%, while validation accuracy exceeds 95%, showing excellent generalization capability. This is largely due to the application of data augmentation and dropout. The training and validation loss curves gradually decrease and stabilize at low values, without any noticeable divergence between them. It means that the model effectively minimizes error on the training data. The absence of overfitting further confirms that the model converges well, with stable and reliable behavior. The results obtained on the test data are: an accuracy of 95.87% with a low loss of 0.1241.

#### IV.3 PERFORMANCE ANALYSIS OF THE RESNET-50 MODEL

The ResNet-50 architecture was trained the on our dataset for 30 epochs. Figure 11 illustrates its training and validation evolution.

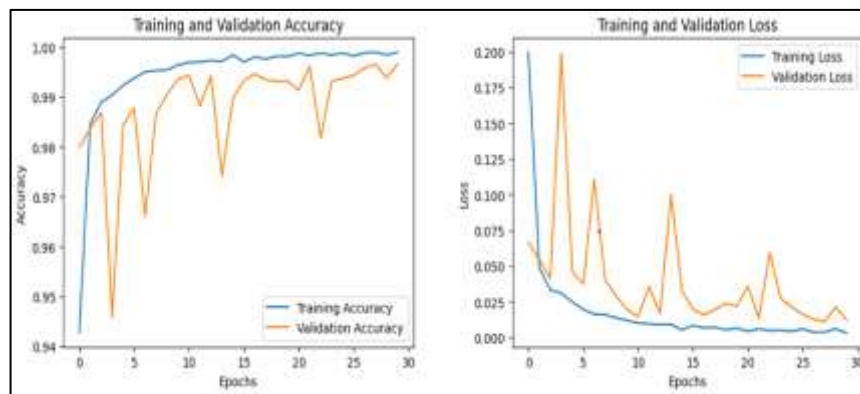


Figure 11: ResNet-50 model performance.  
Source: Authors, (2026).

During training, the accuracy curve increases quickly in the first few epochs. It show the strong learning capacity of the ResNet-50 model. In parallel, the training loss decreases steadily and converges quickly towards zero, confirming stable and efficient convergence. For validation, the accuracy curve indicates that the model experienced slight overfitting at the 4th epoch, but stabilizing around 99%, which indicates good generalization ability. The validation loss shows some initial instability with noticeable peaks, but overall it decreases and remains at low values, demonstrating a satisfactory fit of the model. Once training was completed, the model was evaluated on the test set, obtaining an accuracy of 99.73% with a corresponding low loss of 0.0118.

#### IV.4 PERFORMANCE ANALYSIS OF THE AGRINET MODEL

The proposed model 'AgriNet' was trained on our dataset for 25 epochs. The following figure shows the results obtained.

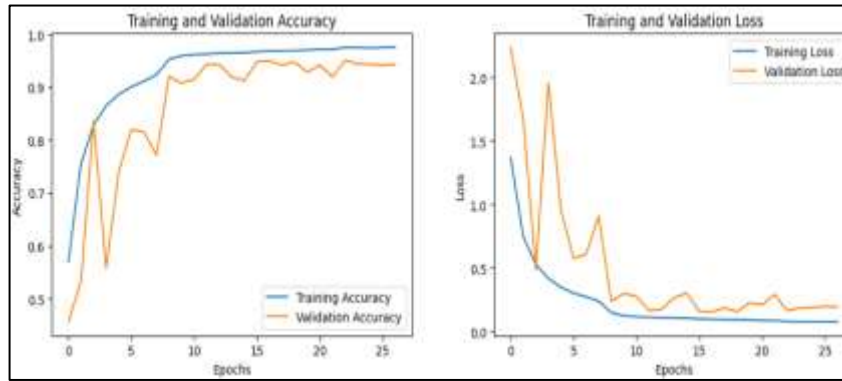


Figure 12: AgriNet model performance.

Source: Authors, (2026).

The results demonstrate strong performance and stability of AgriNet. As illustrated in the training and validation curves, the model achieves a rapid increase in training accuracy during the initial epochs and stabilizes above 98%, while the training loss steadily decreases toward very low values. It indicates a robust convergence process. Similarly, the accuracy on the validation data remains consistently elevated, and stabilizing at 97%, which confirms the model's capacity to perform well on unseen data. Although the validation loss exhibits slight fluctuations compared to the training loss, it remains relatively low and stable at 0.2, without signs of divergence (no significant overfitting). The proposed model achieves excellent convergence, and maintains a good balance between training and validation performance. The obtained test accuracy is 97.13%, with a corresponding loss of 0.1579. Figure 13 illustrates the model's performance for a subset of classes, showing clearly that the model performs very well across most categories, with recall, precision, and F1-scores generally within the 90–100% range.

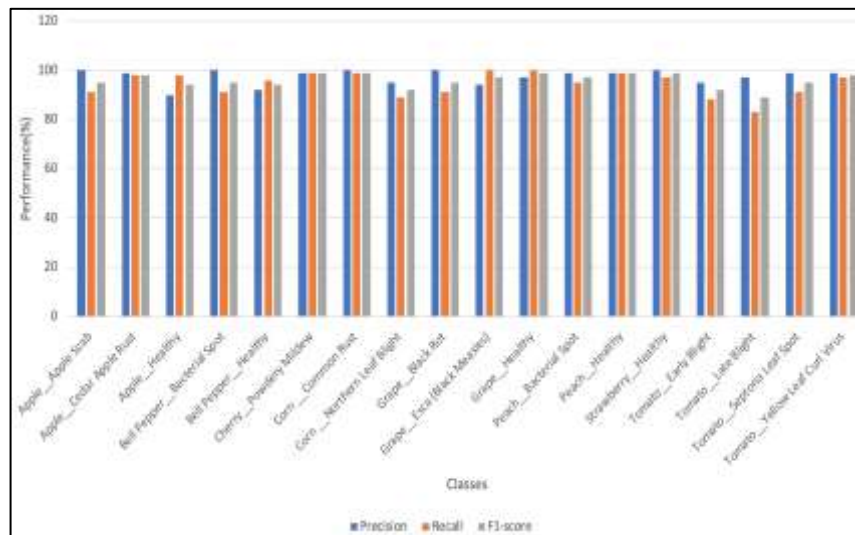


Figure 13: Precision, recall and f1-score analysis.

Source: Authors, (2026).

The number of epochs for each model was chosen empirically, based on their depth and the behavior of the learning curves. AlexNet is relatively shallow (but fairly wide), it required about 40 epochs to stabilize its performance on training set. However, the validation curves reveal persistent instability. VGG-16 and ResNet-50, being more complex, converged in roughly 30 epochs, while our proposed 6-layer model reached its best performance in just 25 epochs. According to these results, AgriNet appears to be the best suited to the size and complexity of the data.

By analyzing Table 1, it can be observed that our proposed model 'AgriNet' achieves slightly lower accuracy compared to ResNet-50. However, ResNet-50 incurs higher computational cost due to its deeper structure and greater parameter count. In contrast, each parameter represents a weight stored in memory. A model with fewer parameters requires less RAM and can be loaded and executed more efficiently. With only 323 165 parameters and a depth of six layers, AgriNet is the most compact model among those evaluated.

This compact design makes AgriNet especially useful in environments with limited resources, such as mobile or embedded devices, characterized by constrained memory and processing power. In addition, AgriNet demonstrates that reducing model complexity does not necessarily lead to a dramatic loss in predictive performance.

The baseline models AlexNet and VGG-16 use more parameters and require more epochs to converge, whereas AgriNet converges in just 25. This shows that older or deeper networks are not always optimal and highlights the need to design models that fit the computational resources, instead of relying solely on accuracy.

Table 1: Comparison of the models based on their architecture and performance.

|                             | AlexNet[8]       | VGG-16 [11]      | ResNet-50 [14] | AgriNet                 |
|-----------------------------|------------------|------------------|----------------|-------------------------|
| <b>Depth</b>                | 8 layers         | 16 layers        | 50 layers      | 6 layers                |
| <b>Number of Parameters</b> | 10,8 M           | 14.9 M           | 23.6 M         | 323 165                 |
| <b>Pretrained Weights</b>   | ImageNet         | ImageNet         | ImageNet       | None                    |
| <b>Dropout rate</b>         | 0.6, 0.3 (Dense) | 0.5, 0.3 (Dense) | 0.5 (Dense)    | 0.25(Conv), 0.5 (Dense) |
| <b>Batch size</b>           | 32               | 32               | 32             | 32                      |
| <b>Optimizer</b>            | Adam             | Adam             | Adam           | Adam                    |
| <b>Learning rate</b>        | 0.0005           | 0.001            | 0.0001         | 0.001,0.0001, 0.00001   |
| <b>Epochs</b>               | 40               | 30               | 30             | 25                      |
| <b>Accuracy (%)</b>         | 98.15%           | 95.87%           | 99.73%         | 97.13%                  |

Source: Authors, (2026).

Importantly, these results are consistent with recent advances in lightweight architectures [22], [23], [24], [25], as previously discussed in the second section, which also emphasize balancing accuracy with efficiency (see Table 2). Notably, our model addresses 29 classes, in contrast to these studies where the number was limited to a maximum of eight. Moreover, AgriNet is the most compact, with only eight layers.

Table 2: Comparison of the proposed model with recent works on lightweight deep learning architectures.

| Ref.No                       | Nbr. Of classes | Dataset                                          | Methodologys used             | Layers                                                                   | Accuracy %                                              |
|------------------------------|-----------------|--------------------------------------------------|-------------------------------|--------------------------------------------------------------------------|---------------------------------------------------------|
| <b>Nguyen and Huynh [22]</b> | 5               | Flowers dataset                                  | VGG-16 + ResNet-50            | 5 blocks, each containing 2 to 3 convolutional layers, plus 2 FC layers. | -% 75,9 for ResNet-50<br>-85.4%, for VGG-16 + ResNet-50 |
| <b>Fu and Huang [23]</b>     | 4               | Images of pepper leaves collected by the authors | GGM+VGG-16                    | 16                                                                       | 87.38%                                                  |
| <b>Ji et al. [24]</b>        | 5               | PlantVillage                                     | ResNet-50+ISA                 | 50                                                                       | 98,87 %                                                 |
| <b>Liu et al. [25]</b>       | 8               | Soybean diseases, PlantVillage                   | ResNet+ Dual pyramidal design | 50                                                                       | 98,10 %                                                 |
| <b>AgriNet</b>               | 29              | PlantVillage                                     | Alexnet+VGG-16 + ResNet-50    | 6                                                                        | 97.13%                                                  |

Source: Authors, (2026).

**Abbreviations:** FC, fully connected; GGM, Ghost modules, Global average pooling, Multi-scale convolution; ISA: Improved Spatial Attention.

To evaluate the generalization of the model under real-world conditions, an additional validation was carried out on the PlantSeg dataset [30]. This dataset contains 115 disease classes across 34 plant species. Since PlantSeg does not exactly cover the 29 diseases of PlantVillage, the evaluation was performed only on the classes common to both datasets (partial validation). The results show an accuracy of 95,87%, this slight decrease reflects the increased complexity of field images. Nevertheless, these results confirm that the model maintains robustness and significant generalization capability under real-world conditions.

## V. CONCLUSIONS

In this paper, we have proposed a novel model, AgriNet, which fuses the strengths of three Deep Learning architectures : Alexnet, VGG-16, and ResNet. The design of AgriNet was motivated by the need for accurate yet computationally efficient models for agricultural applications, especially in the domain of plant disease detection. The proposed network is lightweight, consisting of only six layers : three convolutional blocks, one Global Average Pooling layer, and two dense layers. Experimental evaluation is conducted on the Plant Village dataset (29 classes), and it demonstrates that AgriNet achieves a high recognition accuracy of 97.13% with a corresponding loss of 0.1579. These results confirm that the model can accurately detect plant diseases while also generalizing well to unseen data. Furthermore, the model maintains low complexity, with fewer layers and fewer parameters than the other three state-of-the-art models. In the future, we aim to evaluate AgriNet on additional datasets to provide stronger evidence of its robustness. We also plan to explore its practical deployment on IoT and mobile devices. These intelligent agricultural systems will provide farmers with accessible and cost-effective tool to monitor plant health, reduce crop losses, and ultimately improve food security.

## VI. REFERENCES

- [1] J. Sheradha, "A Comprehensive Review on Various Plant Diseases and Impact on Crop Yield and Quality," Journal of Information Systems Engineering & Management, vol. 10, no. 38s, pp. 430–439, 2025.
- [2] G. Yashodha, B. Rajakumar, and M. Krishnamoorthy, "Efficient plant disease detection using K-means clustering and DenseNet-based classification," in Proc. Int. Conf. Electron. Renewable Syst. (ICEARS), 2025, pp. 1197–1204.

- [3] Y. Resti, C. Irsan, M. T. Putri, et al., "Identification of corn plant diseases and pests based on digital images using multinomial naïve Bayes and k-nearest neighbor," *Science and Technology Indonesia*, vol. 7, no. 1, pp. 29–35, 2022, doi: 10.26554/sti.2022.7.1.29-35
- [4] S. Krushna, A. Anushri, S. Sakshi, H. Pratiksha, C. Shreni, P.S.N.Wasankar, "Plant Disease Identification Using SVM," *International Journal Of Creative Research Thoughts*, vol 10, pp.2320-2882, 2023.
- [5] J. Patra, M. Chakraborty, and S. Gupta, "Random Forest Algorithm for Plant Disease Prediction," In *AI to Improve e-Governance and Eminence of Life: Kalyanathon*, pp. 89-101, Springer Nature Singapore, 2023.
- [6] S. Mehta and A. Aneja, "AI in agriculture: Hybrid convolutional neural networks and random forest for sunflower disease identification," in *Proc. Int. Conf. Interdiscip. Approaches Technol. Manag. Soc. Innov. (IATMSI)*, 2025, pp. 1–5.
- [7] R. Sujatha, J. M. Chatterjee, N. Z. Jhanjhi, and S.N. Brohi, "Performance of deep learning vs machine learning in plant leaf disease detection," *Microprocessors and Microsystems*, 80, 103615, 2021.
- [8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol 25, pp. 1097–1105, 2012.
- [9] H. C. Chen, A. M. Widodo, A. Wisnujati, M. Rahaman, J. C. W. Lin, L. Chen, and C. E. Weng, "AlexNet convolutional neural network for disease detection and classification of tomato leaf," *Electronics*, vol. 11, no. 6, Art. no. 951, 2022, doi: 10.3390/electronics11060951
- [10] S. K. Kannaiah et al., "Automatic plant leaf disease detection using AlexNet and image processing techniques," in *Proc. Int. Conf. Cybern. Comput. (CYBERCOM)*, 2024, pp. 441–446.
- [11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [12] B. Jangid and R. S. Sharma, "Rice disease detection using deep learning VGG-16 model and Flask," *International Journal of Scientific Research in Engineering and Management (IJSREM)*, vol. 7, no. 3, 2023.
- [13] K. Ashok et al., "Detecting plant pathogens by using deep learning with VGG-16 CNN," in *Proc. 4th Int. Conf. Data Eng. Commun. Syst. (ICDECS)*, 2024.
- [14] K. He, X. Zhang, S. Ren, et al., "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 770–778, 2016.
- [15] S. Khan, A. K. Yadav, S. S. P. Shukla, et al., "ResNet-50: A breakthrough in plant disease detection accuracy for agricultural empowerment," in *Computational Methods in Science and Technology*, CRC Press, 2024, pp. 364–371.
- [16] I. Adhikari, B. Koirala, and M. R. Panda, "Decoding plant diseases: Unleashing the power of ResNet-50 for superior disease detection," in *Proc. Int. Conf. Adv. Smart, Secure, Intell. Comput. (ASSIC)*, 2024, pp. 1–6.
- [17] E. Cengil and A. Çinar, "Hybrid convolutional neural network based classification of bacterial, viral, and fungal diseases on tomato leaf images," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 4, pp. e6617, 2022.
- [18] S. Farhat and A. Chokka, "A lightweight and erudite VGG-16 + ResNet-50 based deep learning architecture model for enhanced plant yield production," *Trait. Signal*, vol. 42, no. 1, pp. 253, 2023.
- [19] A. K. Trivedi, T. Mahajan, T. Maheshwari, et al., "Leveraging feature fusion ensemble of VGG16 and ResNet-50 for automated potato leaf abnormality detection in precision agriculture," *Soft Comput.*, vol. 29, no. 4, pp. 2263–2277, 2025.
- [20] H. Ajra, M. A. Majid, M. Shohidul Islam, et al., "Leaf disease detection in plant care using CNN architecture: AlexNet and ResNet-50 models," *International Journal on Advanced Science, Engineering & Information Technology*, vol. 15, no. 1, 2025.
- [21] M. Ishaq and C. Direkoglu, "A deep learning based fusion model for plant leaf disease classification," in *Proc. 7th Int. Congr. Hum.-Comput. Interact., Optim. Robot. Appl. (ICHORA)*, 2025, pp. 1–6.
- [22] H. L. Nguyen and K. T. Huynh, "New model for low-end computers: ResNet and VGG-16," *VNUHCM Journal of Science and Technology Development*, vol. 26, no. 1, pp. 2672–2680, 2023.
- [23] Y. Fu, L. Guo, and F. Huang, "A lightweight CNN model for pepper leaf disease recognition in a human palm background," *Heliyon*, vol. 10, no. 12, 2024.
- [24] Z. Ji, S. Bao, M. Chen, et al., "ICS-ResNet: a lightweight network for maize leaf disease classification," *Agronomy*, vol. 14, no. 7, p. 1587, 2024.
- [25] W. Liu, G. Wu, H. Wang, et al., "Lightweight pyramidal multi-scale attention residual network for plant disease recognition," *IEEE Access*, vol. 13, pp. 61526–61536, 2025.
- [26] Available on Kaggle : <https://www.kaggle.com/datasets/tushar5harma/plant-village-dataset-updated>
- [27] L. Huang, J. Qin, Y. Zhou *et al.*, "Normalization techniques in training dnns: Methodology, analysis and application," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no 8, pp. 10173-10196, 2023.
- [28] S. Yang, W. Xiao, M. Zhang, *et al.*, "Image data augmentation for deep learning: A survey," *arXiv preprint arXiv:2204.08610*, 2022.
- [29] G. Naidu, T. Zuva, and E.M. Sibanda, "A review of evaluation metrics in machine learning algorithms," In : *Computer science on-line conference*, Cham : Springer International Publishing, p. 15-25, 2023.
- [30] Available on : <https://zenodo.org/records/13762907>