Robust Plant Disease Recognition Using a Neutrosophic-Enhanced, RBF-Based Stacked Ensemble of ConvNeXt and Classical CNN Models

Emre Özbilge[1,*], **Ebru Ozbilge**[2]
[1]Department of Computer Engineering, Cyprus International University, Nicosia, North Cyprus, Turkey
[2]College of Business Administration, American University of the Middle East, Kuwait
Emails: eozbilge@ciu.edu.tr; ebru.kahveci@aum.edu.kw

Abstract

Accurate and timely recognition of plant diseases is crucial to prevent crop loss and ensure global food security. This paper presents a robust ensemble-based framework that combines six classical and state-of-the-art deep convolutional neural networks (DCNNs), including a ConvNeXt architecture, and integrates Neutrosophic Science to better handle uncertainty in leaf images. The proposed approach features three main components: (1) transfer learning with pre-trained DCNNs, (2) a model-averaging strategy to stabilise individual predictions, and (3) a stacked ensemble design that employs a radial basis function (RBF) meta-learner to refine the classification outputs. Experiments on the Plant Village dataset, comprising 54,305 segmented images of 38 plant diseases, included 10-fold cross-validation. The results show that the final stacking ensemble achieved near-perfect performance with 99.97% accuracy and an F_1 score of 99.55% on an unseen test set of 27,160 images. Compared with standalone models, the ensemble demonstrated greater robustness in distinguishing visually similar diseases, benefiting from the complementary strengths of multiple DCNN architectures. The Neutrosophic component further enhances reliability by modelling uncertainties due to noise, occlusions, and environmental variations. Although a higher computational overhead and modest misclassifications remain, especially in certain visually overlapping classes, this study demonstrates the effectiveness of an ensemble-driven, uncertainty-aware strategy. These findings hold considerable promise for real-world agricultural applications, where rapid and accurate disease diagnosis is paramount.

Keywords: Plant disease classification; Deep learning; Computer vision; Transfer learning; Ensemble learning; Neutrosophic image

1 Introduction

Computer vision research in agriculture has gained prominence because of its capacity to boost food production, reduce waste, and enhance efficiency, contributing to a more sustainable future [1]. Plant diseases significantly affect crop yields, leading to financial losses for farmers and potential food scarcity in certain areas. Various approaches have been investigated for plant disease identification, including conventional methods, image-based techniques [2], Internet of Things (IoT) solutions [3], and approaches rooted in artificial intelligence (AI) [4], and machine learning (ML) [5].

Conventional plant disease recognition methods, such as visual assessment and molecular analysis, have proven effective when performed by skilled professionals. However, these approaches often require substantial time and effort, and their precision may decrease, particularly in the early stages of disease progression [6]. Imaging techniques, including hyperspectral imaging, drone-based imaging, and mobile applications, offer noninvasive methods for capturing plant images, which can be examined using computer vision and ML

algorithms. These approaches provide both high accuracy and efficiency, offering detailed insights into plant health. Nevertheless, they typically require expensive equipment and specialised knowledge [7]. IoT-based sensors can gather real-time data on plant growth, environmental conditions, and disease occurrence. When this information is analysed using AI and ML techniques, it enables timely disease detection and management. However, IoT-driven solutions rely on robust network infrastructure and hardware that may not be readily available in all regions [8]. AI and ML technologies further enhance imaging and IoT solutions by expediting and improving disease detection accuracy [9, 10]. By utilising large datasets, AI and ML models learn to identify disease-specific patterns, enabling real-time analysis and decision-making. Moreover, many of these techniques can be employed using cost-effective equipment, thereby increasing their feasibility for farmers in resource-limited areas.

Deep Convolutional Neural Networks (DCNNs) have witnessed remarkable advancements, offering considerable potential for accurate plant disease identification [11, 12, 13]. Various pre-trained deep models can be employed on large-scale plant image datasets to classify diseases [14, 15, 16]. The application of transfer learning, which involves fine-tuning models initially trained on extensive image collections using plant-specific disease data, can enhance recognition precision in this context. However, the successful implementation of this approach is contingent on the accessibility of substantial, correctly labelled plant image datasets.

Despite their strong performance, deep learning models can be limited by uncertainties in the plant images. These uncertainties often stem from varying environmental conditions, occlusions, or noise. Neutrosophic logic tackles such ambiguity by breaking decisions into three measurable components: truth, falsity, and indeterminacy [17]. Incorporating neutrosophic principles into DCNN architectures enables a more nuanced handling of unclear data, thus increasing reliability and interpretability. Consequently, this hybrid approach remains robust even when challenging conditions are encountered in real-world applications.

A range of DCNN architectures, such as ResNet, Inception, DenseNet, VGG, and MobileNet, have demonstrated reliable performance in plant disease tasks when refined through transfer learning from a large-scale ImageNet dataset. However, many of these architectures are considered "classical" DCNNs, and their improvements have plateaued. In contrast, the more recent ConvNeXt architecture represents a state-of-the-art design that surpasses the expectations on the ImageNet benchmark. In this study, we evaluated ConvNeXt in comparison with traditional DCNN models for plant disease recognition. Additionally, we combined ConvNeXt with weaker models using deep ensemble strategies to further enhance its detection capabilities. When complemented by Neutrosophic Science, ConvNeXt not only achieves high accuracy, but also effectively manages uncertainty in classification tasks, thus improving the reliability of diagnostic decisions.

The key contributions of this study are as follows.

(1) We trained and fine-tuned the recent ConvNeXt architecture for plant disease detection, an approach not previously explored in this domain, and benchmarked it against established DCNN models.

(2) We created ensemble models by pairing ConvNeXt with weaker DCNN architectures, thereby boosting the performance of ConvNeXt.

(3) We proposed a radial basis function (RBF)-based stacked ensemble design for plant disease recognition. We then compared its outcomes with those of standalone DCNNs, model-averaging ensembles, and top-tier custom plant detection frameworks.

(4) We incorporated neutrosophic logic into the recognition framework, thus offering a robust means of handling uncertainties in plant disease image data.

The remainder of this paper is organised as follows. Section 2 reviews the relevant literature. Section 3 describes the DCNN models used in the plant disease classification experiments and details their training and transfer learning strategies. Section 4 introduces ensemble learning methodologies. This is followed by a description of the colour image to neutrosophic image conversion for the preprocessing step in Section 5. Section 6 outlines the Plant Village dataset, data preparation processes, model performance assessment methods, and the key experimental parameters. The results of our experiments and performance analysis of the proposed frameworks are discussed in Section 7, and comparisons with state-of-the-art deep networks are provided. Finally, Section 8 presents the conclusions and insights for future research.

2 Related Work

An IoT-based real-time system for detecting and classifying groundnut leaf diseases using hybrid machine learning was introduced by [1]. This approach exhibited better accuracy, precision, and F_1 score than previously reported methods. In a similar vein, [8] developed a cost-effective and highly precise technique for identifying apple leaf diseases using the MobileNet model. They evaluated its performance against ResNet152 and InceptionV3 on a dataset gathered by agricultural experts in Shaanxi Province, China, and validated their method through various experiments. Additionally, [16] created a mobile application called the "Rice Disease Detector", which identifies 12 rice diseases and nutrient deficiencies employing image segmentation techniques and optimised models for offline use. Their research showed that the application could identify multiple diseases in a single image, with MobileNetV2 demonstrating the most effective transfer learning capability.

A multitask learning approach for the concurrent prediction of plant types and diseases was proposed by [10]. Their method exploited shared representations between related tasks by integrating raw images and transferring deep features into a multi-input network. This approach enhances the training efficiency for similar detection tasks in the future. Conversely, [15] devised a technique for the automatic identification of tomato leaf diseases. Their methodology used three compact convolutional neural networks (CNNs) and transfer learning to extract deep features. The researchers assessed six classifiers, with the k-nearest neighbour and support vector machine classifiers achieving the highest accuracies of 99.92% and 99.90%, respectively, despite using a limited feature set.

The identification and classification of plant leaf diseases were addressed by [5] through the development of a specialised PDICNet model. Their approach incorporated ResNet-50 for feature extraction, a modified Red Deer optimisation algorithm for feature selection, and a DLCNN classifier for final classification. When evaluated using the Plant Village and Rice Plant datasets, this model demonstrated superior performance, surpassing state-of-the-art methods in terms of accuracy and F_1 scores. In a parallel study, [6] explored deep transfer learning within a DCNN-based system for plant disease recognition, resulting in the creation of a lightweight architecture called selective kernel mobile net (SK-MobileNet). This model achieved 99.28% accuracy on a public dataset while reducing computational demands. Alternatively, [7] introduced VGG-ICNN, a lightweight CNN designed for crop disease identification using plant leaf images. This model outperformed recent deep learning methods, achieving 99.16% accuracy on the Plant Village dataset and maintaining a consistent performance across five datasets, including Embrapa, Apple, Maize, and Rice. Focusing specifically on tomato crops, [9] designed a compact six-layer CNN to detect diseases in tomato leaves. Trained on the Plant Village dataset encompassing ten classes, this model outperformed state-of-the-art transfer learning approaches while maintaining lower computational costs. Similarly, [11] developed a deep CNN model trained on an open dataset comprising 39 plant leaf classes and background images, achieving 96.46% accuracy with data augmentation, thereby surpassing several conventional machine learning techniques. In recent research, [12] proposed an innovative deep-learning method incorporating a transformer block to capture both long-range and local features. This approach demonstrated superior accuracy compared to existing convolution- and vision transformer-based models, achieving 99.94% on the Plant Village dataset, 99.22% on Ibean, 86.89% on AI2018, and 77.54% on the PlantDoc dataset.

Significant progress in plant disease classification was achieved by [13], who devised an innovative framework utilising a tailored CenterNet model with DenseNet-77 for feature extraction. Their approach encompassed three distinct stages: region-of-interest annotation, deep keypoint extraction, and single-stage detection and classification using CenterNet. This method demonstrated superior results when evaluated on the Plant Village Kaggle database, surpassing the existing techniques. Concurrently, [14] proposed a cutting-edge plant disease recognition system that employed transfer learning to extract high-level latent features and exhibited exceptional classification performance across three separate datasets.

3 Base DCNN Models Used in Ensemble Strategies

A DCNN is a powerful pattern-recognition approach for learning large-scale data, which has been well proven in the ImageNet challenge. The DCNN contains multiple layers through which the input data pass for feature extraction and modelling the nonlinear relationship between the inputs and outputs of the corresponding task.

The advantage of using a DCNN is that this type of deep network extracts the features internally. Unlike traditional machine-learning approaches, a DCNN extracts various feature abstractions from low-level to high-level, more complex features, while processing the input data from the lower layer to the deeper layers of the network. A DCNN consists of various types of layers, including convolutional, pooling, bath normalisation, fully connected dense, recurrent layers, residual connections, and various activation functions.

3.1 Transfer Learning

Very large deep networks usually require many data and iterations to learn a good model and overcome the overfitting (high variance) problem. Instead of training a very large network, the knowledge (*i.e.* connection weights) learned from another task is transferred to a new task; this is called transfer learning. In transfer learning, deep convolutional layers of a network that are learned from another task are used as a pre-trained base model for feature extraction without retraining these layers; that is, the connection weights are not updated during the training of the new task. A new classifier layer was inserted at the top of these pre-trained layers, and the extracted features from the pre-trained base were presented to the newly added classifier layer to obtain the relationship between the extracted input deep features and target outputs. Figure 1 presents a graphical representation of transfer learning in a plant disease recognition task. The ImageNet challenge consists of more than 14 million images, and well-known deep network architectures were trained on this dataset. Thus, these networks have already learned some relevant image features of the plant leaves. Therefore, transfer learning is a suitable approach for use in Plant Village datasets where there is a lack of images of some disease types.

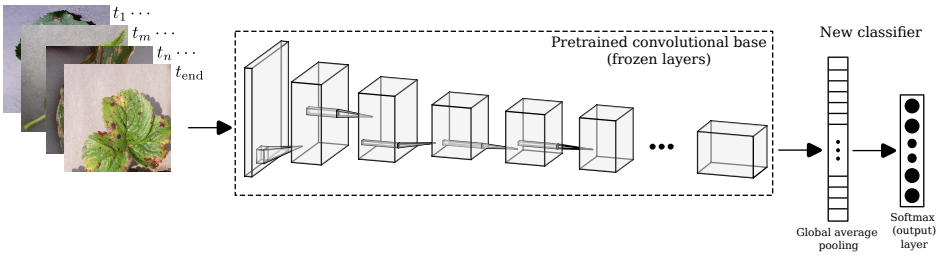

Figure 1: Transfer learning. Pre-trained base can be any ImageNet convolutional neural network architecture for feature extraction.

In this study, the following pre-trained ImageNet models were utilised: ResNet-152V2, Inception-V3, DenseNet-201, VGG-16, MobileNet-V3L, and ConvNeXt-XL. The next section explains the architecture of these networks and defines their distinctions.

3.2 MobileNet Model

The MobileNet network architecture was proposed in [24] for efficient use in mobile devices with convolutional networks as accurately as larger ones. This network introduces depthwise separable convolution instead of regular convolution. Regular convolution uses the same filter for all the input image channels. Depthwise separable convolution uses two separate operations: depthwise and pointwise convolution. Depthwise convolution uses separate filters for each input image channel. Pointwise convolution, which is equivalent to regular convolution with 1×1 kernel, combines all output channels from depthwise convolution to one output value, as shown in Figure 2. This process was repeated for each kernel to complete the convolution operations of the layers. Both methods yield similar results; however, depthwise separable convolution is computationally cheaper and faster, making it suitable for small mobile devices.

[25] introduced an inverted residual block, as shown in Figure 3. This block expands the input tensor before filtering the data in depthwise convolution, followed by a projection layer compressing the data. This process obtains lower-dimensional output features, and a residual connection is added to aid gradient flow through the network. Working with a low-dimensional tensor after compression at the end of the bottleneck block reduces the overall computations.

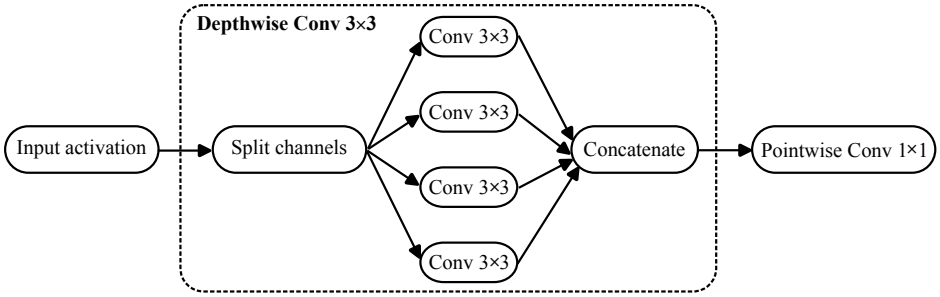

Figure 2: Depthwise convolution is performed by independent spatial convolution per channel which is followed by pointwise convolution.

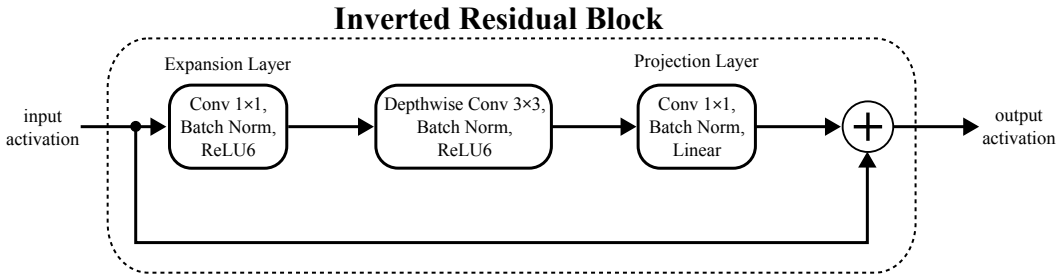

Figure 3: Sequence of convolutional layers within the inverted residual block.

3.3 VGG Model

The VGG network is a classical convolutional neural network architecture proposed by [21], using 3×3 kernel size for all convolution layers instead of the larger 11×11 in AlexNet [26]. Reducing the kernel size increases the number of convolution layers, covering the same filtering areas with fewer parameters than the larger kernels. This approach leads to a faster convergence and reduced overfitting. Additionally, after every pooling operation, the number of filters in each convolution layer was doubled to recover lost information.

3.4 Inception Model

The salient parts of the images can vary, and the network must focus on extracting relevant features to correctly classify objects. To address this issue, it is necessary to select the correct kernel size. Larger kernels lose details by smoothing the extracted features, whereas smaller kernels provide more details. Instead of using one kernel size, multiple sizes have been proposed for the same layer [19]. Thus, the deep neural network becomes wider at the layers instead of deeper. The inception network uses three different kernel sizes for convolution operations in each inception module: 1×1, 3×3 and 5×5. Max-pooling was also performed. Deep networks can be computationally expensive. To limit the input channels, 1×1 kernel was used to perform projection mapping to reduce channel dimensions. All outputs from the different kernel-sized convolutions were concatenated channel-wise. Figure 4 shows the single-inception module used in this network.

3.5 ResNet Model

To model highly nonlinear data, deeper layers are required to represent complex input-output relationships. Deeper networks enable the learning of features at many abstraction levels, from edges and corners in shallower layers to complex features in deeper layers. However, deep networks face vanishing or exploding gradients, where gradient signals approach zero quickly or grow exponentially, slowing gradient descent or causing large weight updates. Identity mapping was proposed to address this issue [27]. The idea is that, when training cannot find a solution for a layer, its output is at least as good as its input. This allows for an increase in the number of convolutional layers with less sensitivity to gradient issues. Figure 5 shows a typical residual block.

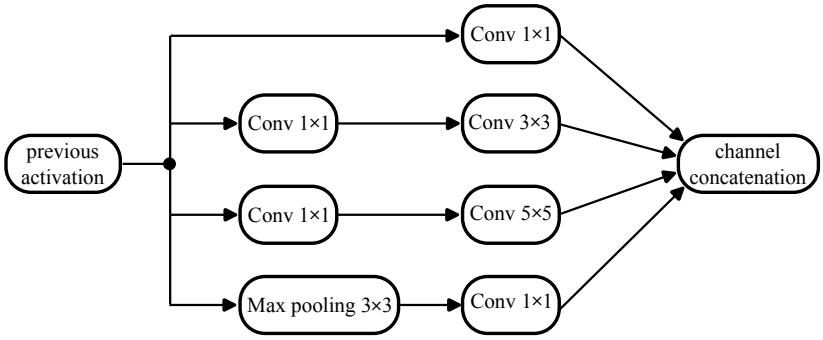

Figure 4: The inception module performs a 1×1 convolution first before presenting activations to 3×3 and 5×5 convolutions for channel dimension reduction. In the later inception network version, the 5×5 kernel was replaced with two 3×3 kernel layers to improve computational efficiency.

A copy of the input to the convolution layer propagates to the end of several layers without processing. This creates another path with a residual connection (or skip connection), and both the convolution outputs and the input are added element-wise. Thus, the gradient signals are less prone to strength loss.

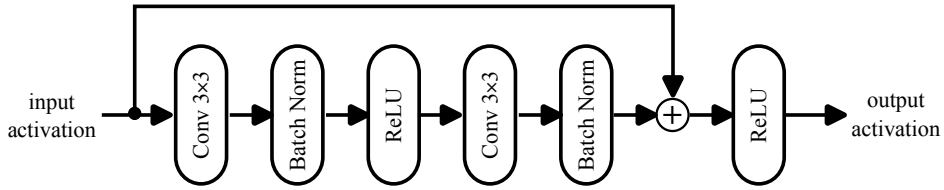

Figure 5: The input activation is skipped over two convolution layers, with both the skipped input activation and output from the last convolution having the same dimension. If necessary, 1×1 convolution can be used to reduce the channel size of the skip connection for an element-wise addition.

3.6 DenseNet Model

Previously, ResNet used a short skip connection to create a deeper network and achieve better accuracy. However, [20] proposed DenseNet, a densely connected convolutional network using longer skip connections to improve classification accuracy. In DenseNet, each convolution layer receives skip connections from all preceding layers, as shown in Figure 6. Unlike ResNet, the outputs from all preceding layers are concatenated channel-wise before computing the output of the corresponding layer. This yields richer and more diversified features, unlike ResNet which uses features correlated with the following layer. The classifier layer in DenseNet receives features from all complexity levels from the earliest to the deepest layers, producing smoother decision boundaries when the training data are insufficient. During backpropagation, owing to connections between the earlier layers and the final classification layer, the earlier layers receive direct supervision to update the connection weights.

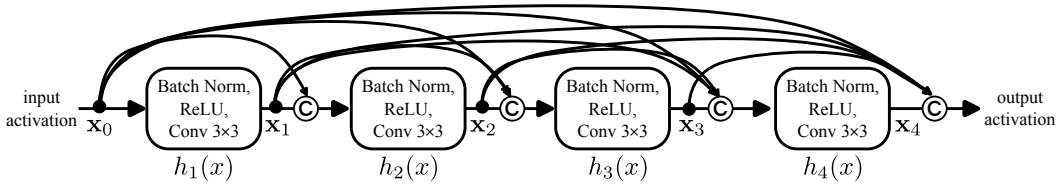

Figure 6: A 4-layer dense block, where each layer receives all the preceding layer feature maps before computing the activation output as $\mathbf{x}_l = h_l([\mathbf{x}_0, \mathbf{x}_0, \cdots, \mathbf{x}_{l-1}])$.

3.7 ConvNeXt Model

ConvNeXt, a recent architecture proposed by [23], achieves an accuracy comparable to transformer-based approaches while maintaining CNN design simplicity without incorporating *transformer encoder* or *attention* mechanisms. It starts with a standard ResNet architecture, gradually modernising it by replacing existing layers with improved approaches from other successful deep network architectures to match the Swin transformer performance.

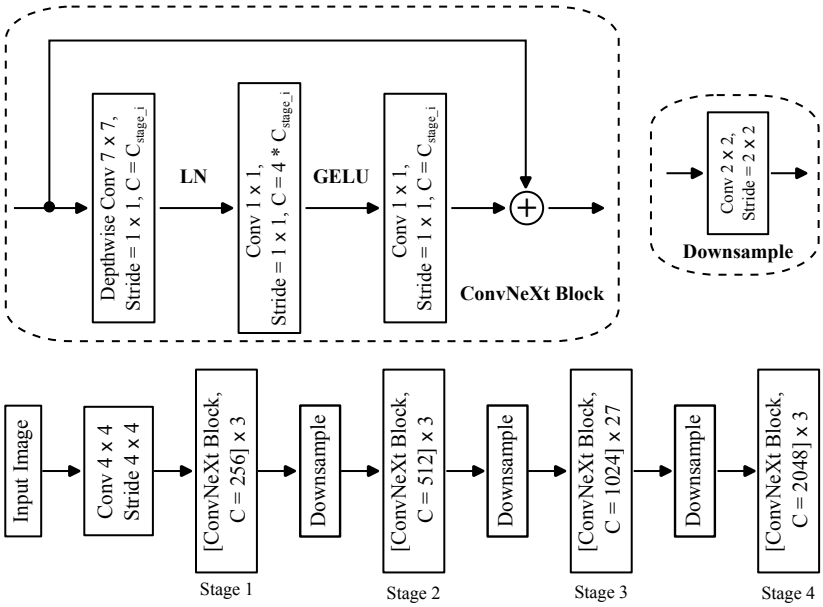

Figure 7: The architecture of the ConvNeXt network without a classifier layer consists of four stages and extracts features at different scales. LN and GELU represent layer normalisation and Gaussian error linear units, respectively.

One change was a training technique using an AdamW optimiser. New image augmentation techniques like Mixup, Cutmix, RandAugment, random erasing, stochastic depth, and label smoothing were introduced, which improved network performance without changing the ResNet architecture. The computation ratio of the ResNet blocks per stage was changed to 3:3:27:3, similar to the Swin transformer architecture. ConvNeXt has three blocks in stages one, two and four, and 27 blocks in stage three, as shown in Figure 7. The stem layer is changed to a 4×4 convolutional layer with a stride of four, similar to the Swin transformer's "patchify" layer. Standard convolution was replaced with depthwise and pointwise convolutions, and standard residual blocks were replaced with inverted residual blocks. The kernel size was increased from 3×3 to 7×7, and the depthwise convolution was moved to the first layer within the block. Rectified Linear Unit (ReLU) activation functions were replaced with a Gaussian Error Linear Unit (GELU) before the channel size decreased in the last 1×1 convolution layer within the inverted residual block. GELU activation is used only within inverted residual blocks, thereby reducing the overall activation functions. Batch normalisation is replaced with layer normalisation after depthwise convolution within the inverted residual blocks. A separate downsampling layer with layer normalisation and 2×2 convolution with a stride of two is inserted between the stages, slightly improving the performance. These changes yielded an accuracy similar to that of transformer-based networks.

3.8 Training of Pre-trained DCNN Models

DCNNs have proven highly effective for a wide range of tasks by learning hierarchical feature representations from raw input data. However, real-world datasets often suffer from imbalanced class distributions, which can hinder training and negatively impact overall performance. The error signals between the predicted and actual outputs were computed using the focal loss cross-entropy function ($\mathcal{FL}$), which is recommended for use when the distributions of the data in each class are imbalanced [28]. The focal loss function is defined as follows:

$$\mathcal{FL}(p) = \begin{cases} -\alpha \left(1 - f\left(p\right)\right)^{\gamma} \ln\left(f\left(p\right)\right), & y = 1 \\ -\left(1-\alpha\right) f\left(p\right)^{\gamma} \ln\left(1 - f\left(p\right)\right), \text{else} \end{cases} \tag{1}$$

$$f\left(z\right) = \frac{\exp\left(z\right)}{\sum_{k=1}^{c} \exp\left(z\right)} \tag{2}$$

where p is the logit output; $f(\cdot)$ is the softmax activation function; γ is the focusing parameter; α is the weighting factor; and c indicates the number of output classes.

During the training of the DCNN models, a more sophisticated version of the standard backpropagation algorithm called Adam optimisation was used. The Adam optimiser combines the momentum eq. (5), and root mean square propagation (RMSprop) eq. (6) to remove the oscillation of the gradient descent to accelerate the learning process. In addition, gradient centralisation (GC) is also integrated into the Adam optimisation, as given in eq. (4), instead of using the computed gradients in eq. (3) with respect to the weights of the corresponding layer $^{[l]}$ directly to update the weights of the corresponding layers using eq. (7) during training. Simply, GC computes the column mean of the gradient matrix (for the dense layer) or slice mean of the gradient tensor (for the convolutional layer) and subsequently centralises each column or slice to have a zero mean before applying standard Adam optimisation.

$$\mathbf{g}_t^{[l]} = \frac{\partial \mathcal{FL}}{\partial \mathbf{w}_t^{[l]}} \tag{3}$$

$$\widehat{\mathbf{g}}_t^{[l]} = \Phi_{GC}(\mathbf{g}_t^{[l]}) \tag{4}$$

$$\mathbf{m}_t^{[l]} = \beta_1 \mathbf{m}_{t-1}^{[l]} + (1-\beta_1)\widehat{\mathbf{g}}_t \tag{5}$$

$$\mathbf{v}_t^{[l]} = \beta_2 \mathbf{v}_{t-1}^{[l]} + (1-\beta_2)\widehat{\mathbf{g}}^2 \tag{6}$$

$$\mathbf{w}_{t+1}^{[l]} = \mathbf{w}_t^{[l]} - \eta_t \frac{\mathbf{m}_t^{[l]}}{\sqrt{\mathbf{v}_t^{[l]}} + \epsilon} \tag{7}$$

where $\Phi_{GC}(\mathbf{x}) = \mathbf{x}_i - \bar{\mathbf{x}}_i,\ \forall i = 1, 2, \ldots, N$, i indicates i^{th} column vector or slice matrix, β_1, β_2 are momentum and discount factor parameters respectively, ϵ is a very small constant number to prevent the denominator from becoming zero, and η_t is the current learning rate.

4 Proposed Deep Ensemble Learning Strategies

Deep ensemble learning has emerged as a robust strategy for improving the performance and reliability of deep neural networks. By integrating multiple learning models, this approach leverages diversity among individual predictors to reduce variance, mitigate overfitting, and enhance generalisation. In applications ranging from computer vision to natural language processing, combining the strengths of various models through ensemble techniques has proven crucial for achieving state-of-the-art results. This section presents two popular ensemble strategies, model-averaging and stacking, that illustrate the effectiveness of model fusion in deep learning.

4.1 Model-Averaging Ensemble

The variance problem is associated with the use of a standalone deep network model. This problem occurs when the model attempts to learn small fluctuations in the data by modelling random noise in the training dataset, which causes a high-variance (overfitting) issue. To overcome this, multiple deep models were trained individually using the same training dataset, and the predicted probabilities of these models on the unseen test data were averaged before the highest probability score among the classes was selected as the final class prediction. Taking the predictions of the other models into account results in strengthening the indecisive

prediction by using a standalone sensitive model. Furthermore, it is possible that the best-acquired model in training did not exhibit the best performance on the unseen test data. Therefore, a combination of multiple models can prevent incorrect predictions. Figure 8 presents an overview of the model-averaging ensemble architecture. First, each model predicts the current input image, and then the prediction probabilities from all the models are averaged to obtain the final probability values of the output classes. The class with the highest probability is selected as the output of the classification. Thus, the magnitude of the incorrectly predicted classes is reduced and the frequently predicted class among the acquired models dominates the final decision.

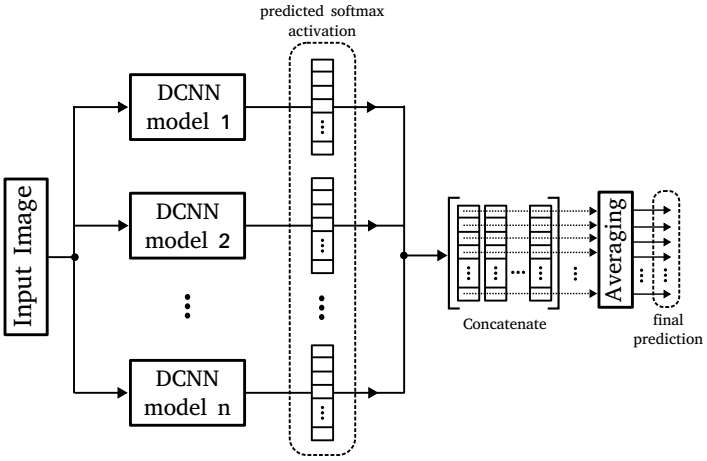

Figure 8: Overview of the model-averaging ensemble architecture using deep network models.

4.2 Stacking Ensemble

The stacking ensemble approach contains two-stage data modelling, which are base and meta-models. In the first stage, base models are used as the individual DCNN models, as used in the model-averaging ensemble, where all base models are trained on the same training dataset and model the relationship between the input and output multiclasses. Typically, the performance of base models on input data is expected to be diverse. Each base model yields a softmax activation for each class, and the activation outputs from all base models are then concatenated as new input data, which are then fed to the trained meta-model for the final softmax prediction. Thus, instead of combining each model prediction with the same weight to contribute to the final outputs, the stacking ensemble base models contribute to the final prediction based on their learned relationship with the output classes. Figure 9 summarises the overall architecture of the stacking ensemble. In this specific plant disease identification task, a radial basis function network (RBFN) was used as the meta-learning model.

The RBFN architecture comprises two primary layers: a radial basis function (RBF) layer and a softmax layer. Initially, the combined predicted logit outputs from the base models were scaled in the range of [0,1] and then fed into Gaussian functions, as described in eq. (8). Subsequently, these Gaussian outputs were normalised to ensure that their sum was equal to one, as outlined in the eq. (9). The normalised outputs were then weighted and aggregated before the final softmax output activations were calculated, as detailed in eq. (10).

$$\varphi_i(\mathbf{u}) = \exp\left(\frac{-||\mathbf{u}-\mathbf{c}_i||^2}{2\sigma_i^2}\right) \tag{8}$$

$$\delta_i = \frac{\varphi_i}{\sum_j^k \varphi_j} \tag{9}$$

$$\widehat{y}_i = f\left(\sum_{i=1}^{k} \theta_i \delta_i + b_i\right) \tag{10}$$

where $||\cdot||$ indicates the Euclidean distance; $\mathbf{c}_i$ is the centroid of i-th RBF; $\mathbf{u}$ is the concatenated logit outputs from base models; σ_i is the width of the Gaussian function; k indicates the number of the RBFs on the network; b is the bias term and $f(\cdot)$ is softmax activation function given in eq. (2).

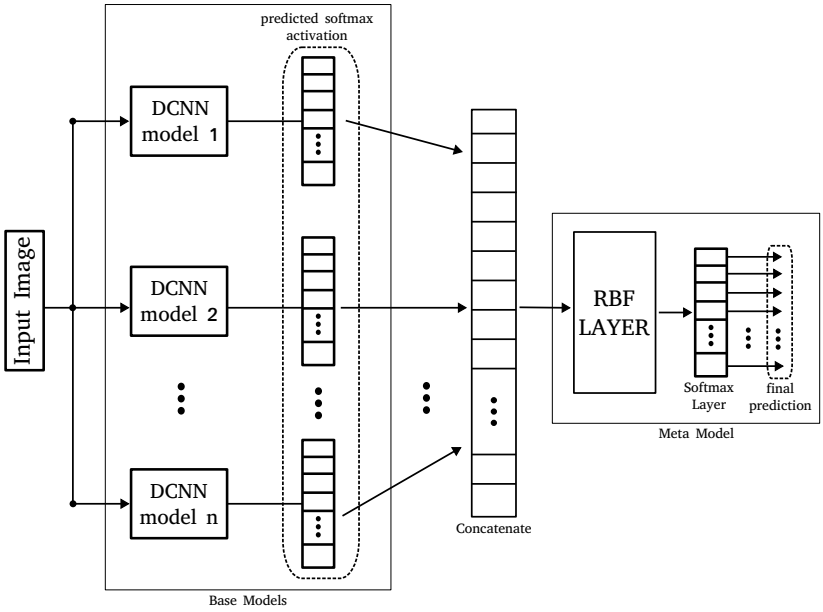

Figure 9: Overview of the stacking ensemble architecture in which the prediction of base DCNN models is combined within the RBFN-based meta model.

Meta model (RBFN) training. The RBFN has two training stages. First, the centroid of each RBF is obtained using the k-means clustering algorithm. Therefore, the cluster number indicates the number of RBFs used in a network. Once the centroid of each RBF is obtained, the p-nearest neighbour approach is used to compute the width of each RBF, as given in eq. (11). After obtaining the normalised Gaussian outputs for all the training samples, the softmax regression model between the normalised Gaussian values and the one-hot encoded target training output classes were trained in the same manner as the base DCNN models were trained using the Adam optimisation algorithm, as described in Section 3.8.

$$\sigma_i = \frac{1}{p}\left(\sum_{j=1}^{p}||\mathbf{c}_i - \mathbf{c}_j||^2\right)^{1/2}, \quad \text{where } p = 2 \tag{11}$$

Stacking ensemble training. The k-fold cross-validation approach was used to train the base and meta-model of the stacking ensemble approach. The entire training dataset was split into k equal-sized folds. Then, the base models were trained using k-1 folds, and the remaining fold was used to make predictions. This process was repeated k times, and at each repetition, different folds were used for the prediction. In addition, the parameters of the base models are re-initialised at the beginning of each repetition in order to predict that the fold is not seen during the training which results in overfitting if parameters are not re-initialised. These out-of-fold predictions formed a new dataset to train the meta-model. The complete training steps are presented in Listing 1.

5 Neutrosophic Image Conversion Phase for Colour Image Preprocessing

DCNNs encounter significant challenges when dealing with images or regions that exhibit ambiguity, such as partial occlusion, overlapping symptoms, or suboptimal lighting conditions. These factors often lead to elevated uncertainty in network predictions. To mitigate this, [17] introduced *neutrosophic logic*, which incorporates three membership components: the *truth* degree (T), representing how strongly a pixel (or instance) belongs to a given class; the *indeterminacy* degree (I), representing uncertainty; and the *falsity* degree (F), indicating the degree of non-membership to the class.

Listing 1: Python-style pseudocode of the stacked ensemble training.

```
# Load base and meta model objects
from models import DCNN,RBFN
# Load training dataset: X(input) and y(output)
X, y = load_dataset()
# Define number of base models in a form ('name',model)
base_models = [('base_1',DCNN()),('base_2',DCNN()),...]
# Define meta-model
meta_model = RBFN()
# Number of folds
k = 10
# Use stratified k-fold cross-validator for imbalanced dataset
skf = StratifiedKFold(n_splits=k, shuffle=True)
# Initialize arrays for storing out-of-fold predictions
oof_preds = np.zeros((X.shape[0], len(base_models)))
# Train base models and generate out-of-fold predictions
for train_idx, val_idx in skf.split(X):
    X_train, X_val = X[train_idx], X[val_idx]
    y_train = y[train_idx]
    for i, (name, model) in enumerate(base_models):
        cloned_model = clone(model) # get initial model
        cloned_model.fit(X_train, y_train) # train the current base model
        oof_preds[val_idx, i] = cloned_model.predict(X_val) # get logit outputs
# Train meta-model on out-of-fold predictions
meta_model.fit(oof_preds, y)
# Retrain base models on the full training dataset
final_base_models = [clone(model).fit(X, y) for _, model in base_models]
```

For plant disease images, we can convert each channel of a colour image $\mathbf{x} \in \mathbb{Z}^{h\times w}$ into a neutrosophic image $\mathbf{x}_N \in \mathbb{R}^{H\times W\times 3}$ by defining suitable membership functions T, I, and F. In the neutrosophic domain, a pixel $x(i,j)$ is mapped to

$$x_N(i,j) \;=\; \big\{\, T(i,j),\; I(i,j),\; F(i,j) \big\},$$

where these three components denote the corresponding neutrosophic set memberships. As described by [29], each colour channel is transformed independently. Specifically,

$$T(i,j) \;=\; \frac{\overline{x}(i,j) - \overline{x}_{\min}}{\overline{x}_{\max} - \overline{x}_{\min}}, \tag{12}$$

$$\overline{x}(i,j) \;=\; \frac{1}{k\times k} \sum_{m=i-\frac{k}{2}}^{i+\frac{k}{2}} \sum_{n=j-\frac{k}{2}}^{j+\frac{k}{2}} x(m,n), \quad \text{where } k=7 \tag{13}$$

$$I(i,j) \;=\; \frac{\delta(i,j) - \delta_{\min}}{\delta_{\max} - \delta_{\min}}, \tag{14}$$

$$\delta(i,j) \;=\; \big|\, x(i,j) \;-\; \overline{x}(i,j) \big|, \tag{15}$$

$$F(i,j) \;=\; 1 \;-\; T(i,j). \tag{16}$$

In these equations, $\overline{x}(i,j)$ is the local mean in a $k\times k$ neighborhood window around (i,j). The terms $\overline{x}_{\min}$ and $\overline{x}_{\max}$ denote the global minimum and maximum of the local mean across the entire image. The quantity $\delta(i,j)$ is the absolute difference between a pixel intensity $x(i,j)$ and its local mean $\overline{x}(i,j)$, whereas $\delta_{\min}$ and $\delta_{\max}$ are the minimum and maximum values of $\delta(i,j)$ across all pixels in the image.

Within the RGB colour model, each channel (R, G, B) produces its own (T_r, I_r, F_r), (T_g, I_g, F_g), and (T_b, I_b, F_b), thereby expanding the total number of channels from three to nine. Although training a custom deep network on this 9-channel input may be straightforward, using transfer learning with standard ImageNet-trained models can be problematic because these models expect only three input channels.

To address this mismatch, we computed the average neutrosophic channels:

$$T_{\text{avg}} \;=\; \frac{T_r + T_g + T_b}{3}, \quad I_{\text{avg}} \;=\; \frac{I_r + I_g + I_b}{3}, \quad F_{\text{avg}} \;=\; \frac{F_r + F_g + F_b}{3}, \tag{17}$$

thus yielding a 3-channel image $(T_{\text{avg}}, I_{\text{avg}}, F_{\text{avg}})$ that is compatible with typical pre-trained networks. In a transfer learning setting, the weights of most layers are often frozen, whereas the newly introduced classifier

layer is trained from scratch. However, because the first convolutional layer in these pre-trained models was originally optimised for standard RGB intensities, we re-initialised it to learn from the (T, I, F) representation. This approach preserves the benefits of pre-trained feature extraction in deeper layers while allowing the initial layer to adapt to the neutrosophic-based input channels. An overview of RGB-to-neutrosophic image conversion is illustrated in Figure 10.

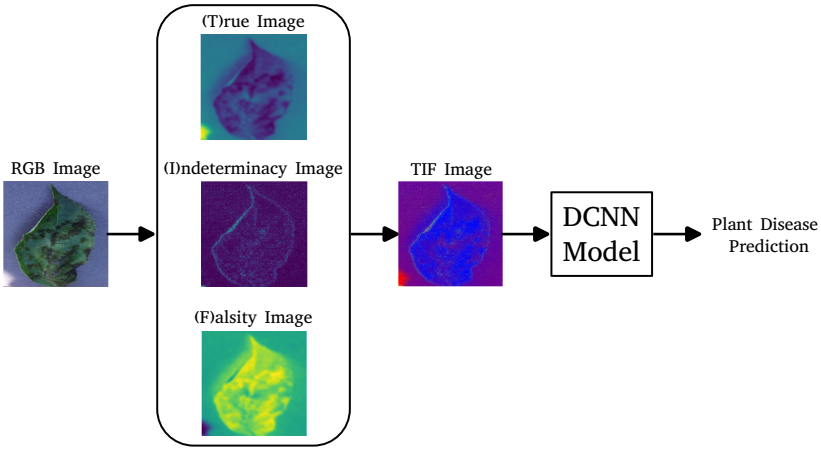

Figure 10: Overview of the conversion of RGB colour images to neutrosophic T,I, and F images as input to the DCNN model.

6 Experimental Methods

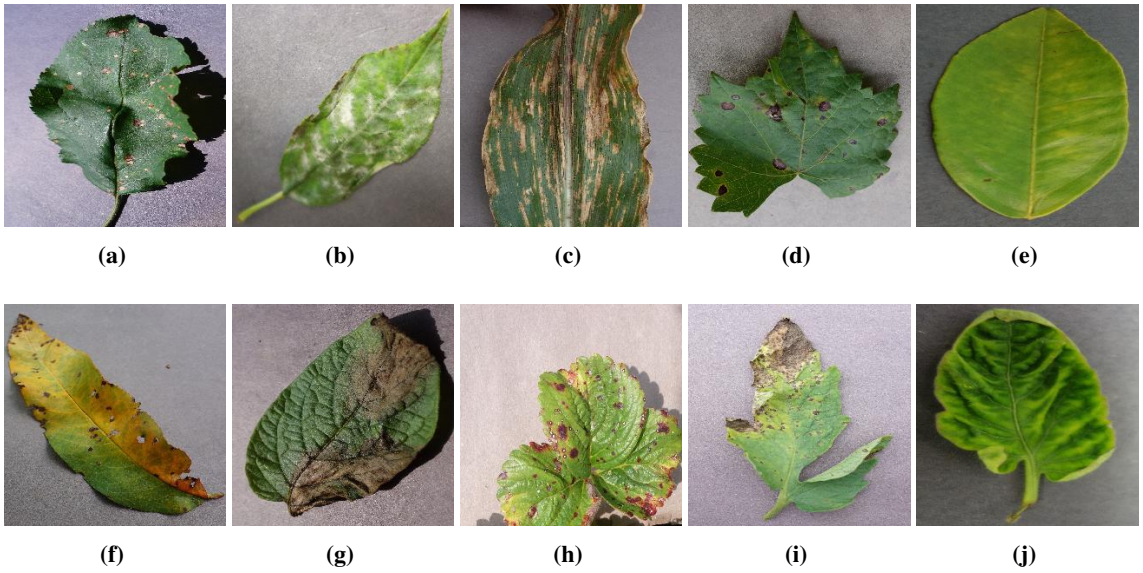

Figure 11: Sample images from the Plant Village dataset. (**a**) Apple black rot. (**b**) Cherry powdery mildew. (**c**) Corn gray leaf spot. (**d**) Grape black rot. (**e**) Orange haunglongbing. (**f**) Peach bacterial spot. (**g**) Potato late blight. (**h**) Strawberry leaf scorch. (**i**) Tomato septoria leaf spot. (**j**) Tomato yellow leaf curl virus.

6.1 Datasets

The Plant Village dataset consists of 54,305 images of 38 plant diseases with the shape of $[256 \times 256]$. Figure 11 shows 10 various sample images. As can be clearly seen, the images have already been segmented; therefore, only the leaf part of the plant appears in the image. Thus, using the segmented leaf images provides

a clear focus for the network model to learn the unique disease of these leaf images during training; otherwise, unsegmented leaf images contain much irrelevant information, which generates noise for the network training, which might result in lower accuracy. In addition, the number of sample images was not uniformly distributed in each plant disease, where the classes of orange haunglongbing, tomato yellow leaf curl virus diseases, and soybean healthy contained the highest number of sample images, with 20.28%, 18.75%, and 19.73% of the whole dataset, respectively. Approximately 40% of the image data were shared with 35 plant diseases, with a similar distribution ratio. Table 1 lists the names of the plant diseases available in Plant Village dataset and their corresponding class identifier (ID).

Table 1: Plant disease names and their corresponding indexes for the Plant Village dataset.

ID	Disease Name	ID	Disease Name	ID	Disease Name
0	Apple scab	13	Grape leaf blight	26	Strawberry leaf scorch
1	Apple black rot	14	Grape healthy	27	Strawberry healthy
2	Apple cedar rust	15	Orange haunglongbing	28	Tomato bacterial spot
3	Apple healthy	16	Peach bacterial spot	29	Tomato early blight
4	Blueberry healthy	17	Peach healthy	30	Tomato late blight
5	Cherry powdery mildew	18	Pepper bell bacterial spot	31	Tomato leaf mold
6	Cherry healthy	19	Pepper bell healthy	32	Tomato septoria leaf spot
7	Corn grey leaf spot	20	Potato early blight	33	Tomato two spotted spider mite
8	Corn common rust	21	Potato late blight	34	Tomato target spot
9	Corn northern leaf blight	22	Potato healthy	35	Tomato yellow leaf curl virus
10	Corn healthy	23	Raspberry healthy	36	Tomato mosaic virus
11	Grape black rot	24	Soybean healthy	37	Tomato healthy
12	Grape esca	25	Squash powdery mildew		

6.2 Performance Evaluation Metrics

To evaluate the performance of the DCNN models, a 38×38 confusion matrix was generated using a test dataset. Then, for each class, true positive (TP), true negative (TN), false positive (FP), and false negative (FN) were computed using the obtained confusion matrix. Subsequently, various statistical tests were performed. These were accuracy (ACC), F_1 score, Matthews correlation coefficient (MCC), true-positive rate (TPR), and true-negative rate (TNR). To address the multi-class problem, these measurements were performed for each class independently, and then the weighted average value over the classes was reported as the overall performance of each model. The most informative statistical tests are F_1 score and MCC, which are the main criteria for evaluating the efficiency of the models. The equations for the statistical tests were as follows:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{18}$$

$$\mathrm{F_1} = \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{19}$$

$$\mathrm{MCC} = \frac{\mathrm{TP} \cdot \mathrm{TN} - \mathrm{FP} \cdot \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP}) \cdot (\mathrm{TP} + \mathrm{FN}) \cdot (\mathrm{TN} + \mathrm{FP}) \cdot (\mathrm{TN} + \mathrm{FN})}} \tag{20}$$

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{21}$$

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{22}$$

6.3 Experimental Procedure

The dataset was divided into two halves: training (27,145 images) and test (27,160 images) sets. To ensure that all types of diseases were observed by the network during training, each plant disease class was split in half, where one half was used for the training set and the other half was used for the test set, instead of randomly splitting the training and test datasets without controlling the sample size in each class per set. The training dataset was used for network model training and selection, and the test dataset was used to evaluate the final unbiased performance of the selected model. It is also important to note that all images were resized to $[224 \times 224]$ before being used as the input to the pre-trained DCNN models. Figure 12 shows the number of images in each class of plant diseases after being equally split in the training and test sets.

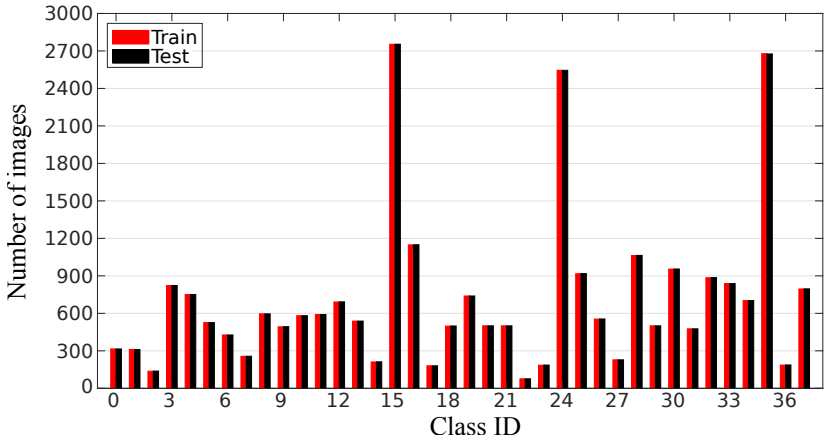

Figure 12: Distribution of the Plant Village leaves dataset in each class. Each class is split into two equal halves to be used for the training and testing processes.

A k-fold cross-validation approach was employed to statistically compare the performances of the models. Upon completion of each training process (a total of k training sessions), all the performance measures (ACC, F_1, MCC, TPR, and TNR) of the test folds were computed and recorded. Following the completion of all the training processes, the mean value of each performance measure and its corresponding standard deviation were calculated and reported for the overall performance of the model. However, a notable issue with the Plant Village dataset was the disparity in the number of images across plant disease classes. This imbalance may lead to complications when dividing the dataset into k-folds because the test fold must encompass all positive and negative classes (*i.e.* the test fold must contain images from all 38 plant disease classes). To address this issue, stratified random selection was implemented to select images from the plant disease dataset, as opposed to random selection, thereby ensuring that each fold comprised of images from each class. Notably, only the training dataset was utilised during the k-fold cross-validation process, and identical folds were initially generated for each compared model; thus, the comparison was conducted in a reliable and equitable manner. In subsequent experiments, $\text{k} = 10$ was utilised. After obtaining the optimal model, it was retrained using all training images. Subsequently, the overall performance of the model was evaluated by using an unseen test dataset. Finally, a summary of all training parameters for the models is provided in Table 2.

7 Experimental Results

Previously, neutrosophic image preprocessing within the standalone pretrained model training was proposed, and an extensive comparison can be found in [30]. The results demonstrated that the neutrosophic image exhibited superior performance compared to the grayscale image. In subsequent experiments, all comparisons between the standalone pretrained models and ensemble strategies involving various combinations of models were conducted using the neutrosophic images. The final optimal model was additionally trained using an RGB colour image for comparative analysis.

Table 2: Training parameters of the pre-trained & RBFN models.

Description	Value/Method
Pre-trained Model	
Number of Epoch	30
Batch Size	32
Image Size	[224×224]
Number of Classes	38
Learning Rate	0.001
Dropout Rate	0.5
Feature Extraction Layer	Global Average Pooling
Loss Function	Focal Loss Cross-Entropy
Optimizer	Adam with GC
Augmentation	Not Applicable
RBFN Model	
RBF Type	Normalised Gaussian Function
Number of RBFs	100
Centroid Training	K-means
Width of Kernel	P-nearest neighbours (p=2)
Neuroscopic Preprocessing	
Neighborhood window size	7×7

7.1 Performance Comparison of Transfer Learning Models

Table 3 shows the performance of each model in 10-fold cross-validation. The ConvNeXt-XL deep network model outperformed the other models, with average F_1 and MCC scores of 0.9891 and 0.9889, respectively. By contrast, the worst performance in identifying plant disease types was obtained when the InceptionNet-V3 model was used, where $F_1 = 0.8797$ and MCC $= 0.8798$. It is important to emphasise that the performance of the MobileNet-V3L model is also promising, as it achieves the second-best performance ($F_1 = 0.9641$ and MCC $= 0.9636$) during the 10-fold cross-validation. In particular, when limited processing power or memory is available on the board of any device, such as flying drones or mobile phones, which are used for running the acquired very large deep network for plant disease identification, the MobileNetV3Large model is preferable for use in these devices because ConvNeXt-XL contains 350.1 million of connection weights. However, the MobileNetV3Large model contains only 5.4 million weights, making MobileNetV3Large less expensive from a computational point of view.

7.2 Performance Comparison of Ensemble Learning Strategies

Various pre-trained models were combined within the ensemble mechanisms. Table 4 lists the combined model labels that are used for clarity. Two types of ensemble approaches were investigated, as discussed in sections 4.1 and 4.2: model-averaging and stacking ensemble approaches. Additionally, the optimal model (*i.e.* ConvNeXt-XL model) obtained during the standalone transfer learning training is included as label 'A' for comparative purposes.

The performance of the model-averaging ensemble models is presented in Table 5. Notably, the standalone ConvNeXt-XL model exhibited a slight advantage in performance over other model-averaging approaches.

Table 3: Performance of the pre-trained models on the Plant Village dataset obtained from 10-fold cross-validation.

Model	ACC	F_1	MCC	TPR	TNR
ResNet-152V2	0.9962±0.0003	0.9383±0.0053	0.9375±0.0050	0.9389±0.0053	0.9978±0.0003
Inception-V3	0.9921±0.0004	0.8797±0.0057	0.8798±0.0049	0.8828±0.0061	0.9951±0.0003
DenseNet-201	0.9978±0.0003	0.9620±0.0047	0.9619±0.0044	0.9630±0.0045	0.9986±0.0002
VGG-16	0.9949±0.0004	0.9150±0.0065	0.9155±0.0058	0.9169±0.0059	0.9969±0.0003
ConvNeXt-XL	**0.9994±0.0001**	**0.9891±0.0018**	**0.9889±0.0018**	**0.9892±0.0017**	**0.9997±0.0001**
MobileNet-V3L	0.9979±0.0002	0.9641±0.0015	0.9636±0.0016	0.9645±0.0015	0.9988±0.0001

Table 4: Combination of pre-trained models which are used for the ensemble systems.

Label	Combined Models
A	[ConvNeXt-XL]
B	[ConvNeXt-XL, MobileNet-V3L]
C	[ConvNeXt-XL, DenseNet-201]
D	[ConvNeXt-XL, MobileNet-V3L, DenseNet-201]
E	[MobileNet-V3L, DenseNet-201]

Nevertheless, Model 'B', which combines ConvNeXt-XL and MobileNet-V3L, demonstrated comparable efficacy to that of the standalone ConvNeXt-XL model. Given that ConvNeXt-XL and MobileNet-V3L were among the most effective models when employing standalone transfer learning, their averaged predictions yielded results that closely approximated those of the top-performing standalone ConvNeXt-XL model. However, inaccurate predictions made by the MobileNet-V3L model adversely affected the overall accuracy when averaged with ConvNeXt-XL. Moreover, the lack of confidence in erroneous predictions from both models precluded the mutual correction of false negative outcomes.

Table 5: Performance of the model-averaging ensemble models on the Plant Village dataset obtained from 10-fold cross-validation.

Model	ACC	F_1	MCC	TPR	TNR
A	**0.9994±0.0001**	**0.9891±0.0018**	**0.9889±0.0018**	**0.9892±0.0017**	0.9997±0.0001
B	**0.9994±0.0001**	0.9890±0.0010	0.9888±0.0010	0.9891±0.0010	**0.9997±0.0000**
C	0.9993±0.0001	0.9869±0.0013	0.9868±0.0012	0.9870±0.0012	0.9996±0.0001
D	0.9993±0.0001	0.9868±0.0014	0.9867±0.0013	0.9870±0.0013	0.9996±0.0001
E	0.9983±0.0002	0.9706±0.0032	0.9706±0.0030	0.9712±0.0030	0.9990±0.0001

On the other hand, Table 6 shows the classification performance of five stacked ensemble models (A–E) evaluated on the Plant Village dataset using 10-fold cross-validation. Model A already achieves a high baseline, with ACC near 99.94%, F_1 and MCC values of approximately 0.989. However, Models B, C, D, and E exhibit slight yet consistent improvements in all metrics. These improvements suggest that the meta-model effectively captures the relationship between the activations of the base models and the actual plant disease labels, thereby correcting misclassifications through the additional stacking layer. Notably, Model D delivered the best overall performance across ACC, F_1, MCC, and TPR, indicating that it excels in correctly classifying positive cases while maintaining minimal misclassification. Meanwhile, Model B achieved a top-tier TNR (0.9999), demonstrating a high degree of accuracy in identifying negative cases. Although the gains over

Model A may appear small, they are meaningful, given the near-ceiling performance. Moreover, the small standard deviations across these metrics underscore the stability of ensemble approaches over multiple folds. The Plant Village dataset is well segmented and thus less prone to environmental noise or distortion, which partly explains the very high baseline performance of Model A. Given the already near-perfect results, any further improvements become especially challenging, highlighting the significance of the gains achieved by Models B, C, D, and E.

Furthermore, the optimal model D was trained using RGB colour images (*i.e.* Model D+RGB) instead of neutrosophic images. The results show that training with neutrosophic images significantly outperformed training with RGB images, similar to the results of [30]. RGB-based model D and standalone ConvNeXt with neuroscopic image model A yielded similar results, with F_1 scores of 0.9885 and 0.9891, respectively.

These findings underscore the effectiveness of stacked ensemble learning, in which multiple base learners are combined to mitigate individual weaknesses and capture complementary strengths. By integrating diverse prediction patterns, the ensemble can be generalised to unseen instances, which often yields robust improvements. Consequently, even in scenarios with strong single-model baselines, stacking provides discernible benefits: elevating performance metrics and showcasing the synergy gained through ensemble strategies.

Table 6: Performance of the stacking ensemble models on the Plant Village dataset from 10-fold cross-validation.

Model	ACC	F_1	MCC	TPR	TNR
A	0.9994±0.0001	0.9891±0.0018	0.9889±0.0018	0.9892±0.0017	0.9997±0.0001
B	0.9997±0.0003	0.9905±0.0024	0.9904±0.0023	0.9902±0.0022	**0.9999±0.0001**
C	0.9996±0.0003	0.9903±0.0019	0.9902±0.0018	0.9900±0.0018	0.9998±0.0001
D	**0.9997±0.0002**	**0.9909±0.0022**	**0.9910±0.0024**	**0.9906±0.0023**	0.9999±0.0002
E	0.9996±0.0002	0.9902±0.0023	0.9901±0.0022	0.9899±0.0021	0.9998±0.0002
D+RGB	0.9992±0.0003	0.9885±0.0021	0.9884±0.0021	0.9886±0.0020	0.9996±0.0003

7.3 Acquisition of Final Model

After determining the best model for plant disease identification, which is a combination of the ConvNeXt-XL, MobileNet-V3L, and DenseNet-201 models using the RBFN-based meta-model in the stacking ensemble approach, the model was retrained using all 27,145 training images using the hybrid training algorithm described in Listing 1, and the acquired model was verified using a test dataset containing 27,160 unseen images. To prevent the base models in the stacked ensemble system from overfitting, image augmentation methods were also added during the training process, and only the training images were augmented because these images were used to optimise each base model to prevent them from becoming highly variable during inference. Consequently, random flip (flipping both horizontally and vertically), zoom ($\pm$10% vertical zooming), and translation ($\pm$10% shifting both vertically and horizontally) were used for image augmentation. It is important to note that RGB-to-neutrosophic image conversion is applied after the image is augmented. The same training parameters were used, as listed in Table 2. Once the connection weights of the newly trained classifier layer of the base models were learned, the acquired base models were fine-tuned for further improvement using additional 30 epochs. For the fine-tuning process, the last 20 layers (*i.e.* deeper layers) of each base model were enabled to be trainable (unfronzen) to update the connection weights. However, it is important to note that changing these connection weights by a large magnitude can be very dangerous, as it can cause the learned knowledge to be lost. Hence, the learning rate was set to a very low constant ($\eta = 0.00001$). In addition, while unfreezing the layers of these base models for further learning, any batch normalisation layer with the model architectures was excluded from the unfreezing process.

The overall performance of the stacked ensemble model is shown in Table 7. The model demonstrated a very robust and reliable identification of the true disease types, where TPR had a value of 0.9955, which implies

Table 7: Overall performance of the stacked ensemble model on the test dataset (27,160 images) after retraining using all training images.

Measure	Score
ACC	0.9997
F_1	0.9955
MCC	0.9954
TPR	0.9955
TNR	0.9999

that less than 0.5% of the test images were misclassified as incorrect disease types. This can also be observed in the confusion matrix shown in Figure 14, where 122 of 27,160 images (0.45%) were misclassified.

Furthermore, the performance of the acquired stacked ensemble model was the lowest for predicting class 9 (*i.e.* corn northern leaf blight), as shown in Figure 14, where 31 images were misclassified and 26 of them were incorrectly classified as class 7 (*i.e.* corn grey leaf spot). One reason for this could be that the number of images available for both classes 9 and 7 was only 1.81% and 0.94% of the entire training dataset, respectively, and more images were required to distinguish these two closely similar classes. Figure 13 shows some misclassified cases in which the acquired model was unable to detect the true disease type with a large confidence margin between the probability scores of the first- and second-predicted classes.

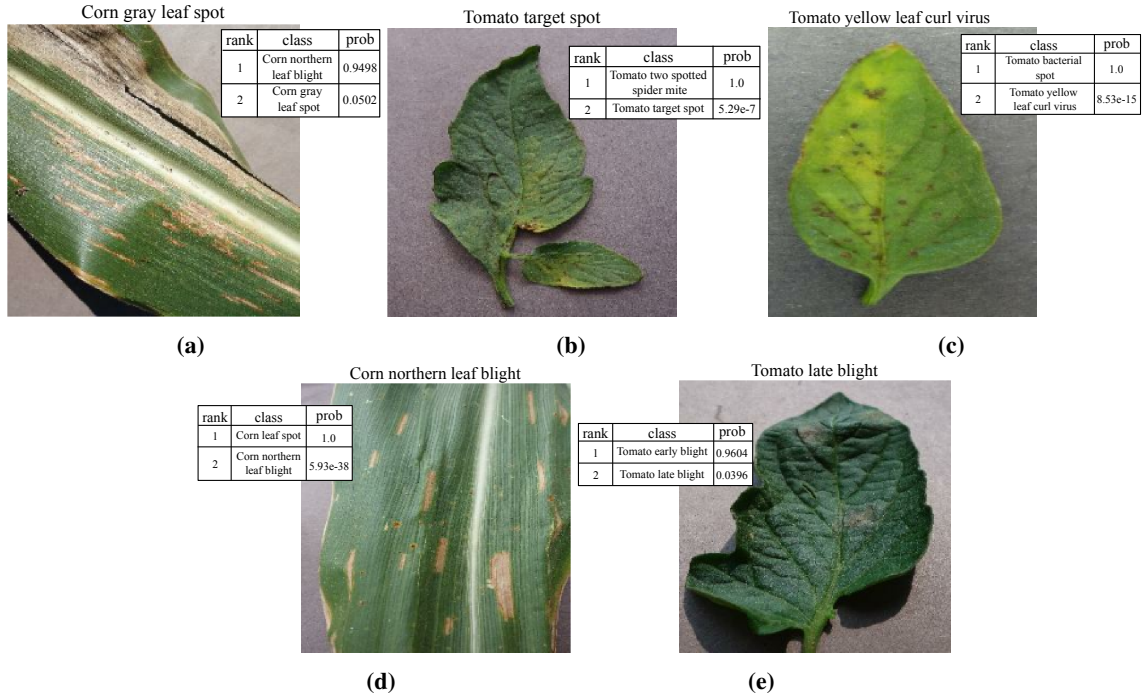

Figure 13: Examples of various misclassified images from the test dataset. The title of each image indicates the actual disease label of the plant.

The performance of the proposed stacked ensemble model for plant disease recognition was also compared with that of recent state-of-the-art systems that had the best performance on the Plant Village dataset. The results show that the proposed model achieved one of the best performances among the top, as shown in Table 8. However, the number of training images used in these models is different, and the number of training images directly affects the learning capability of such large deep network-based models. Therefore, it would not be fair to state that any one model outperformed the others.

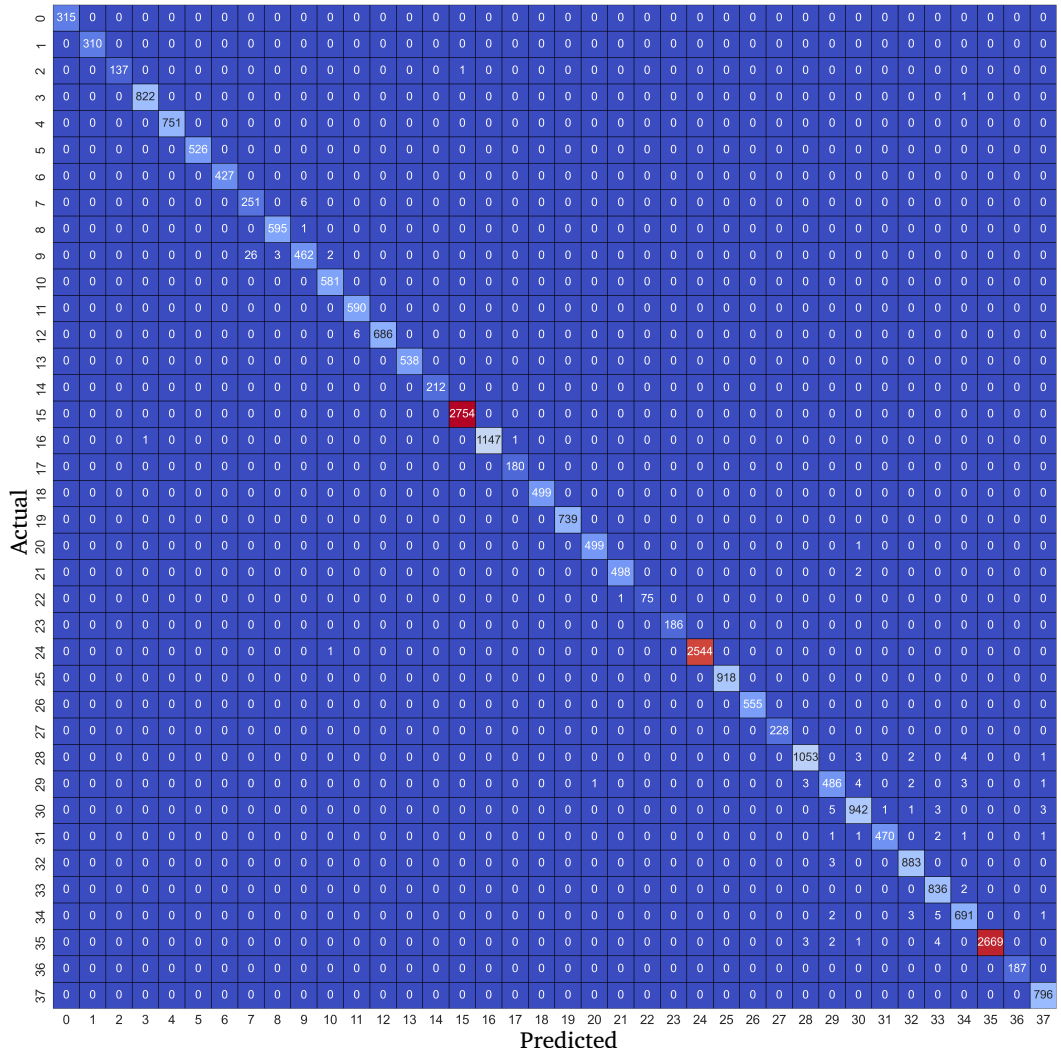

Figure 14: The confusion matrix was acquired from the test images (27,160 images), which shows the performance of the stacked ensemble model predictions against the actual classes.

Table 8: Comparison of state-of-the-art plant disease recognition models trained using the Plant Village dataset.

Method	**Train Image Size**	**Test Image Size**	**Number of Diseases**	**ACC**	**F_1**
ICVT[12]	44,371	11,077	39	99.94%	99.88%
VGG-ICNN[7]	34,755	11,061	38	99.16%	-
CenterNet[13]	38,017	5,316	38	99.98%	99.72%
Stacked Ensemble (proposed)	27,145	27,160	38	99.97%	99.55%

8 Conclusion

This study introduced a novel ensemble-based framework for plant disease recognition, leveraging both a state-of-the-art ConvNeXt architecture and classical DCNNs (DenseNet and MobileNet, among others), enhanced by Neutrosophic Science, to address uncertainties in leaf images. Experimental evaluations on the Plant Village dataset demonstrate that the proposed stacked ensemble design consistently achieves near-perfect performance, with 99.97% accuracy and an F_1 score of 99.55% on 27,160 unseen images.

A key contribution of this work is the combination of transfer learning, multimodel averaging, and a stacked ensemble with neutrosophic logic for uncertainty modelling. The RBF-based meta-learner effectively harnesses the complementary strengths of individual DCNNs, reduces misclassifications, and improves generalisation. By explicitly quantifying ambiguous regions through neutrosophic membership components, the system increases the interpretability and resilience when dealing with noisy or partially occluded leaf images.

Despite these advances, several limitations of this study remain. First, stacking multiple DCNNs increases the computational and memory requirements, making the framework more challenging to deploy in real-time or resource-constrained settings. Second, although the Plant Village dataset is large, it consists of uniformly segmented images, which may limit the adaptability of the model to unsegmented or varied real-world conditions. Finally, the performance drops slightly in certain visually similar classes (*e.g.* corn leaf blight variants), suggesting the need for more diverse training samples or targeted data augmentation strategies.

Looking ahead, expanding the datasets to include non-segmented leaves and a wider range of environmental conditions will provide a more realistic test bed. Additionally, techniques such as model compression, pruning, and quantisation can support low-power devices, such as drones and smartphones, for in-field disease diagnosis. Extending the neutrosophic framework to handle multiple concurrent diseases or new plant disease variants would further enhance real-world applicability. Overall, this study underscores that integrating modern CNN architectures with ensemble learning and explicit uncertainty modelling offers significant promise for sustainable and efficient agriculture, paving the way for early and accurate plant disease management.

References

[1] K. Seetharaman, "Real-time automatic detection and classification of groundnut leaf disease using hybrid machine learning techniques," *Multimedia Tools and Applications*, vol. 82, no. 2, pp. 1935–1963, 2023.

[2] Z. Chen, R. Wu, Y. Lin, C. Li, S. Chen, Z. Yuan, S. Chen, and X. Zou, "Plant disease recognition model based on improved yolov5," *Agronomy*, vol. 12, no. 2, p. 365, 2022.

[3] I. Nanda, S. Chadalavada, M. Swathi, and L. Khatua, "Implementation of iiot based smart crop protection and irrigation system," in *Journal of Physics: Conference Series*, vol. 1804, p. 012206, IOP Publishing, 2021.

[4] R. Thangaraj, S. Anandamurugan, P. Pandiyan, and V. K. Kaliappan, "Artificial intelligence in tomato leaf disease detection: a comprehensive review and discussion," *Journal of Plant Diseases and Protection*, vol. 129, no. 3, pp. 469–488, 2022.

[5] S. R. Reddy, G. S. Varma, and R. L. Davuluri, "Resnet-based modified red deer optimization with dlcnn classifier for plant disease identification and classification," *Computers and Electrical Engineering*, vol. 105, p. 108492, 2023.

[6] G. Liu, J. Peng, and A. A. A. El-Latif, "Sk-mobilenet: A lightweight adaptive network based on complex deep transfer learning for plant disease recognition," *Arabian Journal for Science and Engineering*, vol. 48, no. 2, pp. 1661–1675, 2023.

[7] P. S. Thakur, T. Sheorey, and A. Ojha, "Vgg-icnn: A lightweight cnn model for crop disease identification," *Multimedia Tools and Applications*, vol. 82, no. 1, pp. 497–520, 2023.

[8] C. Bi, J. Wang, Y. Duan, B. Fu, J.-R. Kang, and Y. Shi, "Mobilenet based apple leaf diseases identification," *Mobile Networks and Applications*, pp. 1–9, 2022.

[9] E. Özbilge, M. K. Ulukök, Ö. Toygar, and E. Ozbilge, "Tomato disease recognition using a compact convolutional neural network," *IEEE Access*, vol. 10, pp. 77213–77224, 2022.

[10] A. S. Keceli, A. Kaya, C. Catal, and B. Tekinerdogan, "Deep learning-based multi-task prediction system for plant disease and species detection," *Ecological Informatics*, vol. 69, p. 101679, 2022.

[11] G. Geetharamani and A. Pandian, "Identification of plant leaf diseases using a nine-layer deep convolutional neural network," *Computers & Electrical Engineering*, vol. 76, pp. 323–338, 2019.

[12] S. Yu, L. Xie, and Q. Huang, "Inception convolutional vision transformers for plant disease identification," *Internet of Things*, vol. 21, p. 100650, 2023.

[13] W. Albattah, M. Nawaz, A. Javed, M. Masood, and S. Albahli, "A novel deep learning method for detection and classification of plant diseases," *Complex & Intelligent Systems*, pp. 1–18, 2022.

[14] X. Fan, P. Luo, Y. Mu, R. Zhou, T. Tjahjadi, and Y. Ren, "Leaf image based plant disease identification using transfer learning and feature fusion," *Computers and Electronics in Agriculture*, vol. 196, p. 106892, 2022.

[15] O. Attallah, "Tomato leaf disease classification via compact convolutional neural networks with transfer learning and feature selection," *Horticulturae*, vol. 9, no. 2, p. 149, 2023.

[16] A. Nayak, S. Chakraborty, and D. K. Swain, "Application of smartphone-image processing and transfer learning for rice disease and nutrient deficiency detection," *Smart Agricultural Technology*, p. 100195, 2023.

[17] F. Smarandache, "Neutrosophy: neutrosophic probability, set, and logic: analytic synthesis synthetic analysis," *American Research Press*, 1998.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," 2016.

[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015.

[20] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2016.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[22] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," 2019.

[23] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," 2022.

[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[25] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2018.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[28] Z. Zhang, S. Qiao, C. Peng, X. Li, and J. Yan, "VarifocalNet: An IoU-aware dense object detector," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 8514-8523.

[29] A. Sengur, and Y. Guo, "Color texture image segmentation based on neutrosophic set and wavelet transformation," *Computer Vision and Image Understanding*, vol. 115, no. 8, pp. 1134-1144, 2021.

[30] N.E.M. Khalifa, F. Smarandache, G. Manogaran, and M. Loey, "A study of the neutrosophic set significance on deep transfer learning models: An experimental case on a limited covid-19 chest x-ray dataset," *Cognitive Computation*, pp. 1-10, 2021.